Digital Electronics and Logic Design

(Code : 210245)

Semester III – Computer Engineering

(Savitribai Phule Pune University)

Strictly as per the New Choice Based Credit System Syllabus (2019 Course)

Savitribai Phule Pune University w.e.f. academic year 2020-2021

J. S. Katre

M.E. (Electronics and Telecommunication)
Formerly, Assistant Professor
Department of Electronics Engineering
Vishwakarma Institute of Technology (V.I.T.), Pune.
Maharashtra, India

Harish G. Narula

Formerly Assistant Professor (Senior)
Department of Computer Engineering
D. J. Sanghvi College of Engineering, Mumbai.
Maharashtra, India.

TechKnowledge Publications

TechKnowledge™ Publications

PO123A Price ₹ 385/-

(Book Code : PO123A)

Digital Electronics and Logic Design : (Code : 210245)

(Semester III, Computer Engineering, Savitribai Phule Pune University)

J. S. Katre, Harish G. Narula

Copyright © by Authors. All rights reserved. No part of this publication may be reproduced, copied, or stored in a retrieval system, distributed or transmitted in any form or by any means, including photocopy, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

This book is sold subject to the condition that it shall not, by the way of trade or otherwise, be lent, resold, hired out, or otherwise circulated without the publisher's prior written consent in any form of binding or cover other than which it is published and without a similar condition including this condition being imposed on the subsequent purchaser and without limiting the rights under copyright reserved above.

First Printed in India : January 2002

First Edition : August 2020 **(TechKnowledge Publications)**

This edition is for sale in India, Bangladesh, Bhutan, Maldives, Nepal, Pakistan, Sri Lanka and designated countries in South-East Asia. Sale and purchase of this book outside of these countries is unauthorized by the publisher.

ISBN : 978-93-89889-47-5

Published by :

TechKnowledge Publications

Head Office : B/5, First floor, Maniratna Complex, Taware Colony, Aranyeshwar Corner,
Pune - 411 009. Maharashtra State, India
Ph : 91-20-24221234, 91-20-24225678.
Email : info@techknowledgebooks.com,
Website : www.techknowledgebooks.com

[210245] (FID : PO123) (Book Code : PO123A)

TechKnowledge Publications

We dedicate this Publication soulfully and wholeheartedly,

in loving memory of our beloved founder director,

Late Shri. Pradeepji Lalchandji Lunawat,

who will always be an inspiration, a positive force and strong support behind us.

"My work is my prayer to God"

- Lt. Shri. Pradeepji L. Lunawat

Soulful Tribute and Gratitude for all Your Sacrifices, Hardwork and 40 years of Strong Vision…

(Book Code : PO123A)

Syllabus…

Digital Electronics and Logic Design : Sem. III, Computer Engineering (SPPU)

Teaching Scheme :	Credit Scheme	Examination Scheme and Marks
Lecture : 03 Hrs. / Week	03	Mid-Sem. (TH) : 30 Marks End-Sem. (TH) : 70 Marks

Prerequisite Courses : 104010 : Basic Electronics Engineering

Companion Course : 210249 : Digital Electronics Lab

Course Objectives :

The goal of this course is to impart the fundamentals of digital logic design; starting from learning the basic concepts of the different base number systems, to basic logic elements and deriving logical expressions to further optimize a circuit diagram. Objective is to see that learners are not only able to evaluate different combinational logic designs, but also design their own digital circuits given different parameters.

1. To study number systems and develop skills for design and implementation of combinational logic circuits and sequential circuits.
2. To understand the functionalities, properties and applicability of logic families.
3. To introduce programmable logic devices and ASM chart and synchronous state machines.
4. To introduce students to basics of microprocessor.

Course Outcomes :

On completionof the course, learner will be able to :

CO1 : Simplify Boolean expressions using KMap

CO2 : **Design and implement** combinational circuits.

CO3 : **Design and implement** sequential circuits.

CO4 : **Develop** simple real-world application using ASM and PLD.

CO5 : Differentiate and Choose appropriate logic families IC packages as per the given design specifications.

CO6 : Explain organization and architecture of computer system.

Course Contents

Unit I

Minimization Technique :

Logic design minimization technique : Minimization of Boolean function using K-Maps (upto 4 variables) and Quine - McCluskey method, Representation of signed number - Sign magnitude representation, 1's complement and 2's complement form, Sum of product and product of sum form, Minimization of SOP and POS using K-Maps. Exemplar / Case studies : Digital locks using logic gates. **(Refer chapter 1)**

Unit II

Combination Logic Design :

Code converter - BCD, Excess-3, Gray code, Binary code, Half-adder, Full adder, Half subtractor, Full subtractor, Binary adder (IC 7483), BCD adder, Look ahead carry generator, Multiplexers (MUX) : MUX (IC 74153, 74151), Cascading multiplexers, Demultiplexers (DEMUX) - Decoder (IC 74138, IC 74154), Implementation of SOP and POS using MUX, DMUX, Comparators (2 bit), Parity generators and checker. Exemplar / Case studies : Combinational logic design of BCD to 7 segment display controller. **(Refer chapter 2)**

Technknowledge Publications

Unit III

Sequential Logic Design :

Flip flop : SR, JK, D, T, Preset and clear, Master slave JK flip flops, Truth tables and excitation tables, Conversion from one type to another type of flip flop. Registers : SISO, SIPO, PISO, PIPO, Shift registers, Bidirectional shift register, Ring counter, Universal shift register. Counters : Asynchronous counter, Synchronous counter, BCD counter, Johnson's counter, Modulus of the counter (IC 7490). Synchronous Sequential circuit Design : Models – Moore and Mealy, State diagram and state table, Design procedure, Sequence generator and detector. Exemplar / Case studies : Electronic Voting Machine (EVM). **(Refer chapters 3, 4, 5 and 6)**

Unit IV

Algorithmic State machines and Programmable Logic Devices :

Algorithmic State Machines : Finite State Machines (FSM) and ASM, ASM charts, Notations, Construction of ASM chart and realization for sequential circuits. PLD's : PLD, ROM as PLD, Programmable Logic Array (PLA), Programmable Array Logic (PAL), Designing combinational circuits using PLDs. Exemplar / Case studies : Waveform generator using MUX controller method.

(Refer chapters 7 and 8)

Unit V

Logic Families :

Classification of logic families : Unipolar and bipolar logic families, Characteristics of digital ICs : Fan-in, Fan-out, Current and voltage parameters, Noise immunity, Propagation delay, Power dissipation, Figure of Merits, Operating temperature range, Power supply requirements. **Transistor** - Transistor logic : Operation of TTL NAND gate (Two input), TTL with active pull up, TTL with open collector output, Wired AND connection, Tristate TTL devices, TTL characteristics. CMOS : CMOS inverter, CMOS characteristics, CMOS configurations - Wired logic, Open drain outputs. Exemplar / Case studies : To study the various basic gate design using TTL/CMOS logic family. **(Refer chapter 9)**

Unit VI

Introduction to Computer Architecture :

Introduction to ideal microprocessor – Data bus, Address bus, Control bus, Microprocessor based systems – Basic operation, Microprocessor operation, Block diagram of microprocessor, Functional units of microprocessor – ALU using IC 74181, Basic arithmetic operations using ALU IC 74181, 4-bit multiplier circuit using ALU and shift registers, Memory organization and operations, Digital circuit using decoder and registers for memory operations. Exemplar / Case Studies : Microprocessor based system in Communication / Instrumentation Control. **(Refer chapter 10)**

Exemplar / Case Studies may be assigned as self-study by students and to be excluded from theory examinations

❑❑❑

Technical Knowledge Publications

(Book Code : PO123A)

Unit I

Chapter 1 : Minimization Techniques 1-1 to 1-46

Syllabus : Logic design minimization technique : Minimization of Boolean function using K-Maps (upto 4 variables) and Quine - McCluskey method, Representation of signed number - Sign magnitude representation, 1's complement and 2's complement form, Sum of product and product of sum form, Minimization of SOP and POS using K-Maps.

Exemplar / Case study : Digital locks using logic gates.

1.1 System or Circuit 1-2

1.1.1 Digital Systems 1-2

1.1.2 Types of Digital Systems 1-2

1.1.3 Combinational Circuit Design 1-3

1.2 Standard Representations for Logical Functions .. 1-3

1.2.1 Sum-of-Products (SOP) Form 1-4

1.2.2 Product of the Sums Form (POS) 1-4

1.2.3 Standard or Canonical SOP and POS Forms 1-4

1.2.4 Conversion of a Logic Expression to Standard SOP or POS Form 1-5

1.3 Concepts of Minterm and Maxterm 1-6

1.3.1 Representation of Logical Expressions using Minterms and Maxterms 1-7

1.3.2 Writing SOP and POS Forms for a Given Truth Table 1-7

1.3.3 Representation of Truth Table using Standard SOP Expression 1-7

1.3.4 Representation of Truth Table using Standard POS Expression 1-8

1.3.5 Conversion from SOP to POS and Vice Versa 1-8

1.4 Methods to Simplify the Boolean Functions 1-9

1.4.1 Algebraic Simplification 1-9

1.4.2 Disadvantages of Algebraic Method of Simplification 1-11

1.5 Karnaugh-Map Simplification 1-11

1.5.1 K-map Structure 1-11

1.5.2 K-map Boxes and Associated Product Terms 1-12

1.5.3 Alternative Way to Label the K-map 1-12

1.5.4 Truth Table to K-map 1-12

1.5.5 Representation of Standard SOP Form on K-map 1-13

1.6 Minimization of Boolean Expressions using K-map 1-14

1.6.1 How does Simplification Takes Place ? 1-14

1.6.2 Way of Grouping (Pairs, Quads and Octets) 1-14

1.6.3 Grouping Two Adjacent One's (Pairs) 1-15

1.6.4 Grouping Four Adjacent Ones (Quad) 1-16

1.6.5 Grouping Eight Adjacent Ones (Octet) 1-17

1.6.6 Summary of Rules Followed for K-Map Simplification 1-18

1.7 Minimization of SOP Expressions (K Map Simplification) 1-18

1.7.1 Elimination of a Redundant Group 1-20

1.7.2 Minimization of Logic Functions not Specified in Standard SOP Form 1-21

1.7.3 Don't Care Conditions 1-22

1.8 Product of Sum (POS) Simplification 1-25

1.8.1 K-map Representation of POS Form ... 1-25

1.8.2 Representation of Standard POS Form on K-map 1-26

TechKnowledge Publications

1.8.3 Simplification of Standard POS Form using K-map 1-26

1.9 Quine Mc-Cluskey Minimization Technique (Tabular Method) 1-30

1.9.1 Important Definitions 1-31

1.10 Unsigned Binary Numbers 1-40

1.10.1 Important Features of Unsigned Numbers 1-40

1.11 Sign-Magnitude Numbers 1-40

1.11.1 Range of Sign-Magnitude Numbers 1-40

1.12 1's and 2's Complements of a Binary Number ... 1-41

1.12.1 1's Complement 1-41

1.12.2 Representation of Positive and Negative Numbers using 1's Complement 1-41

1.12.3 2's Complement 1-42

1.12.4 Representation of Positive and Negative Numbers using 2's Complement 1-42

1.12.5 Signed Complement Numbers 1-43

1.12.6 Addition of Signed Magnitude Numbers 1-45

1.12.7 Arithmetic Subtraction (Subtraction of Signed Numbers) 1-46

- **Review Questions 1-46**

Unit II

Chapter 2 : Combinational Logic Design 2-1 to 2-70

Syllabus : Code converter - BCD, Excess-3, Gray code, Binary code, Half-adder, Full adder, Half subtractor, Full subtractor, Binary adder (IC 7483), BCD adder, Look ahead carry generator, Multiplexers (MUX) : MUX (IC 74153, 74151), Cascading multiplexers, Demultiplexers (DEMUX) - Decoder (IC 74138, IC 74154), Implementation of SOP and POS using MUX, DMUX, Comparators (2 bit), Parity generators and checker.

Exemplar / Case study : Combinational logic design of BCD to 7 segment display controller.

2.1 Introduction to Combinational Circuits 2-2

2.1.1 Analysis of a Combinational Circuit 2-2

2.1.2 Design of Combinational Logic using Statements 2-3

2.2 Code Converters 2-4

2.2.1 BCD to Excess 3 Converter 2-4

2.2.2 BCD to Gray Code Converter 2-5

2.2.3 Binary to Gray Code Converter 2-6

2.2.4 Gray to BCD Converter 2-7

2.2.5 Excess 3 to BCD Converter 2-10

2.3 Binary Adders and Subtractors 2-10

2.3.1 Types of Binary Adders 2-10

2.3.2 Half Adder 2-11

2.3.3 Full Adder 2-12

2.3.4 Full Adder using Half Adder 2-13

2.3.5 Applications of Full Adder 2-14

2.3.6 Binary Subtractors 2-14

2.3.7 Half Subtractor 2-14

2.3.8 Full Subtractor 2-15

2.3.9 Full Subtractor using Half Subtractors 2-16

2.4 The n-Bit Parallel Adder 2-17

2.4.1 A Four Bit Parallel Adder Using Full Adders 2-17

2.4.2 Propagation Delay in Parallel Adder 2-17

2.4.3 Look Ahead – Carry Adder 2-18

2.4.4 Four Bit Fast Adder with Look-Ahead Carry 2-19

2.4.5 Binary Adder IC 74 LS 83 / 74 LS 283 . 2-19

2.4.6 Four Bit Binary Adder using IC 7483 2-20

2.4.7 Cascading of Adders 2-20

TechKnowledge Publications

2.5 n-bit Parallel Subtractor (Use of Adder as Subtractor) 2-20
2.5.1 4 Bit Parallel Subtractor using 2's Complement 2-21
2.5.2 4-Bit Binary Parallel Adder/Subtractor ..2-21
2.6 BCD Addition 2-22
2.6.1 BCD Adder 2-23
2.7 Magnitude Comparators 2-25
2.7.1 1-Bit Binary Comparator 2-25
2.7.2 A 2-Bit Comparator 2-26
2.8 Multiplexer (Data Selector) 2-27
2.8.1 Necessity of Multiplexers 2-28
2.8.2 Advantages of Multiplexers 2-28
2.9 Types of Multiplexers 2-28
2.9.1 2 : 1 Multiplexer 2-28
2.9.2 A 4 : 1 Multiplexer 2-29
2.9.3 8 : 1 Multiplexer 2-30
2.9.4 Applications of a Multiplexer 2-30
2.10 Study of Different Multiplexer ICs 2-31
2.10.1 8 : 1 Multiplexer (74151) 2-31
2.10.2 54LS 153/DM 54LS 153/DM 74LS 153 (Dual 4 : 1 Multiplexer) 2-32
2.11 Multiplexer Tree / Cascading of Multiplexer 2-32
2.12 Use of Multiplexers in Combinational Logic Design 2-36
2.12.1 Implementation of a Logical Expression in the Standard SOP Form 2-36
2.12.2 Use of 8 : 1 MUX to Realize a 4 Variable Function 2-39
2.12.3 Implementation of a Logical Expression in the Non-standard SOP Form 2-41
2.12.4 Implementing a Standard POS Expression using Multiplexer 2-42
2.12.5 Implementation of Boolean SOP Expression with Don't Care Conditions.. 2-43
2.13 Demultiplexers 2-44
2.13.1 Demultiplexer Principle 2-44
2.14 Types of Demultiplexers 2-45
2.14.1 1 : 2 Demultiplexer 2-45
2.14.2 1 : 4 Demultiplexer 2-46
2.14.3 1 : 8 Demultiplexer 2-47
2.14.4 IC 74138 as 1 : 8 DE-MUX 2-47
2.14.5 1 : 16 Demultiplexer 2-48
2.14.6 DM 54 LS 154 / DM 74 LS 154 4 Line to 16 Line Decoder / Demux 2-48
2.15 Demultiplexer Tree 2-49
2.15.1 Comparison of Multiplexer and Demultiplexer 2-50
2.15.2 Use of DEMUX in Combinational Logic Design 2-50
2.16 Encoders 2-52
2.16.1 Types of Encoders 2-52
2.17 Priority Encoder 2-53
2.17.1 Priority Encoders in the IC Form 2-53
2.18 Decoder 2-54
2.18.1 2 to 4 Line Decoder 2-54
2.18.2 Difference between Decoder and Demultiplexer 2-54
2.18.3 Demultiplexer as Decoder 2-55
2.18.4 3 to 8 Line Decoder 2-55
2.18.5 1 : 8 DEMUX as 3 : 8 Decoder 2-55
2.18.6 IC 74138 as 3 : 8 Decoder 2-56
2.18.7 4 Line to 16 Line Decoder using 3 : 8 Decoder 2-57

Tech Knowledge Publications

2.18.8 Combinational Logic Design Using Decoders 2-57
2.18.9 Advantage of Decoder Realization 2-60
2.19 Parity Generators / Checkers 2-61
2.19.1 Parity Generator 2-61
2.19.2 Parity Checker 2-62
2.19.3 IC 74180 A Parity Generator / Checker 2-63
2.19.4 74180 as Parity Generator 2-64
2.19.5 Cascading of 74180 2-65
2.20 Examples for Practice 2-65
2.21 Case Study Combinational Logic Design of BCD to 7 Segment Display Controller 2-66
2.21.1 Seven Segment LED Display 2-66
2.21.2 Types of Seven Segment Displays 2-66
2.21.3 Common Anode Display 2-66
2.21.4 Common Cathode Display 2-66
2.21.5 Use of a Decoder for Driving the Seven Segment Display 2-67
2.21.6 BCD to Seven Segment Display Driver (Common Anode Display) 2-67
- **Review Questions 2-69**

Unit III

Chapter 3 : Flip Flops 3-1 to 3-36

Syllabus : SR, JK, D, T, Preset and clear, Master slave JK flip flops, Truth tables and excitation tables, Conversion from one type to another type of flip flop.

3.1 Introduction 3-2
3.1.1 Clock Signal 3-2
3.1.2 Comparison of Combinational and Sequential Circuits 3-2
3.1.3 1-Bit Memory Cell (Basic Bistable Element) 3-3
3.1.4 Latch 3-3
3.2 S-R Latch using NOR Gates 3-4
3.2.1 Operation of S-R Latch 3-4
3.2.2 Symbol and Truth Table of S-R Latch 3-5
3.2.3 Characteristic Equation 3-6
3.2.4 S-R Latch using NAND Gates 3-6
3.3 Triggering Methods 3-7
3.3.1 Concept of Level Triggering 3-7
3.3.2 Types of Level Triggered Flip-flops 3-7
3.3.3 Concept of Edge Triggering 3-8
3.3.4 Types of Edge Triggered Flip Flops 3-8
3.4 Gated Latches 3-8
3.4.1 Types of Level Triggered (Clocked) Flip Flops 3-8
3.5 The Gated S-R Latch (Level Triggered S-R Flip Flop) 3-8
3.5.1 Positive Level Triggered SR Flip-flop 3-8
3.5.2 Negative Level Triggered SR Flip Flop 3-9
3.5.3 Disadvantage of S-R latch 3-10
3.5.4 Application of S-R latch 3-10
3.6 The Gated D Latch (Clocked D Flip Flop) 3-10
3.7 Gated JK Latch (Level Triggered JK Flip Flop) 3-11
3.7.1 Race Around Condition in JK Latch 3-11
3.7.2 Difference between Latch and Flip-flop 3-12
3.8 Edge Triggered Flip Flops 3-12
3.8.1 Positive Edge Triggered S-R Flip Flop 3-13
3.8.2 Negative Edge Triggered S - R Flip Flop 3-14

TechKnowledge Publications

3.9 Edge Triggered D Flip Flop 3-14
3.9.1 Positive Edge Triggered D Flip Flop 3-14
3.9.2 Negative Edge Triggered D Flip Flop 3-15
3.9.3 Applications of D Flipflop 3-16
3.10 Edge Triggered J-K Flip Flop 3-16
3.10.1 Positive Edge Triggered JK Flip Flop 3-16
3.10.2 Characteristic Equation of JK Flip Flop 3-17
3.10.3 How does an Edge Triggered JK FF Avoid Race Around Condition ?.....3-18
3.10.4 Negative Edge Triggered JK Flip-Flop 3-18
3.11 Toggle Flip Flop (T Flip Flop) 3-18
3.11.1 Positive Edge Triggered T-FF 3-18
3.11.2 Negative Edge Triggered T Flip Flop 3-19
3.11.3 Application of T F/F 3-20
3.12 Master Slave (MS) JK Flip Flop 3-20
3.13 Preset and Clear Inputs 3-21
3.13.1 S-R Flip-Flop with Preset and Clear Inputs 3-22
3.13.2 Synchronous Preset and Clear Inputs 3-22
3.13.3 JK Flip Flop with Preset and Clear Inputs 3-23
3.13.4 Applications of JK Flip Flop 3-23
3.14 Various Representations of Flip Flops 3-23
3.14.1 Characteristic Equations 3-23
3.15 Excitation Table of Flip-Flop 3-23
3.15.1 Excitation Table of SR Flip Flops 3-23
3.15.2 Excitation Table of D Flip Flop 3-24
3.15.3 Excitation Table of JK Flip Flop 3-24
3.15.4 Excitation Table of T Flip Flop 3-25
3.16 Conversion of Flip Flops 3-25
3.16.1 Conversion from S-R Flip Flop to D Flip Flop 3-25
3.16.2 Conversion of JK FF to T FF 3-26
3.16.3 SR Flip Flop to T Flip Flop 3-26
3.16.4 SR Flip Flop to JK Flip Flop 3-27
3.16.5 Conversion of D Flip Flop to T Flip Flop 3-27
3.16.6 T Flip Flop to D Flip Flop Conversion 3-28
3.16.7 JK Flip Flop to D Flip Flop Conversion 3-28
3.16.8 JK Flip Flop to SR Flip Flop Conversion 3-29
3.16.9 D FF to SR FF Conversion 3-29
3.16.10 T FF to SR FF Conversion 3-30
3.16.11 Conversion from D FF to JK FF 3-30
3.17 Applications of Flip Flops 3-30
3.17.1 Application of SR Latch for Elimination of Keyboard Debounce 3-30
3.18 Analysis of Clocked Sequential Circuits 3-31
3.18.1 State Table 3-31
3.18.2 State Diagram 3-32
3.18.3 State Equation 3-32
3.18.4 Flip Flop Input Equations 3-34
- **Review Questions 3-35**

Unit III

Chapter 4 : Registers 4-1 to 4-14

Syllabus : SISO, SIPO, PISO, PIPO, Shift registers, Bidirectional shift register, Universal shift register.

4.1 Introduction 4-2

TechKnowledge Publications

4.2 Data Formats 4-2
4.3 Classification of Registers 4-2
4.4 Buffer Registers 4-2
4.5 Shift Registers 4-3
4.5.1 Serial Input Serial Output (Shift Left Mode) 4-4
4.5.2 Serial In Serial Out (Shift Right Mode) ...4-5
4.5.3 Applications of Serial Operation 4-6
4.6 Serial In Parallel Out (SIPO) 4-7
4.7 Parallel In Serial Out Mode (PISO) 4-7
4.8 Parallel In Parallel Out (PIPO) 4-8
4.9 Bidirectional Shift Register 4-8
4.9.1 A 3-bit Bidirectional Register using the JK Flip Flops 4-9
4.10 Universal Shift Register 4-9
4.10.1 Universal Shift Register using Multiplexers and D-flip flops 4-10
4.11 Applications of Shift Registers 4-12
4.11.1 Serial to Parallel Converter 4-12
4.11.2 Parallel to Serial Converter 4-12
4.11.3 Ring Counter 4-12
4.11.4 Johnson's Counter (Twisted / Switch Tail Ring Counter) 4-12
4.11.5 Sequence Generator 4-12
4.11.6 Pseudo Random Binary Sequence (PRBS) Generator 4-12
- **Review Questions 4-14**

Unit III

Chapter 5 : Counters 5-1 to 5-40

Syllabus : Asynchronous counter, Synchronous counter, Ring counter, BCD counter, Johnson's counter, Modulus of the counter (IC 7490).

5.1 Introduction 5-2
5.1.1 Types of Counters 5-2
5.1.2 Classification of Counters 5-2
5.2 Asynchronous / Ripple Up Counters 5-2
5.2.1 Two Bit Asynchronous Up Counter using JK Flip-Flops 5-4
5.2.2 3 Bit Asynchronous Up Counter 5-4
5.2.3 4 Bit Asynchronous up Counter 5-5
5.2.4 State Diagram of a Counter 5-6
5.3 Asynchronous Down Counters 5-6
5.3.1 3- Bit Asynchronous Down Counter 5-6
5.4 UP / DOWN Counters 5-7
5.4.1 UP/DOWN Ripple Counters 5-8
5.4.2 3-bit Up Down Ripple Counters 5-8
5.5 Modulus of the Counter (MOD-N Counter) 5-9
5.5.1 Design of Asynchronous MOD Counters 5-10
5.5.2 Frequency Division Taking Place in Asynchronous Counters 5-12
5.5.3 Decade (BCD) Ripple Counter 5-12
5.5.4 Disadvantages of Ripple Counters 5-13
5.6 Ripple Counter IC 7490 (Decade Counter) 5-13
5.6.1 The Internal Diagram of IC 7490 5-14
5.6.2 Other Applications of IC 7490 5-15
5.6.3 Symmetrical Bi-quinary Divide by Ten Counter 5-15
5.7 Problems Faced by Ripple Counters 5-19
5.8 Synchronous Counters 5-20
5.8.1 2-Bit Synchronous up Counter 5-20
5.8.2 3-Bit Synchronous Binary up Counter ..5-21

TechKnowledge Publications

5.8.3 Design of the 3 Bit Synchronous Counter 5-22

5.8.4 Four Bit Synchronous Up Counter 5-24

5.9 Modulo – N Synchronous Counters 5-25

5.9.1 Synchronous Decade Counter 5-25

5.10 UP / DOWN Synchronous Counter 5-28

5.10.1 3-bit Up/Down Synchronous Counter 5-29

5.10.2 Advantages of Synchronous Counter 5-29

5.10.3 Comparison of Synchronous and Asynchronous Counters 5-30

5.11 Lock Out Condition 5-30

5.11.1 Bushless Circuit 5-30

5.12 Bush Diagram 5-33

5.13 Applications of Counters 5-35

5.14 Ring Counter 5-35

5.15 Johnson's Counter (Twisted / Switch Tail Ring Counter) 5-37

5.15.1 Johnson's Counter using D Flip-flops 5-38

- **Review Questions 5-39**

Unit III

Chapter 6 : Synchronous Sequential Circuit Design 6-1 to 6-32

Syllabus : Models – Moore and Mealy, State diagram and State table, Design procedure, Sequence generator and detector.

Exemplar / Case study : Electronic Voting Machine (EVM).

6.1 General Model of Sequential Circuits 6-2

6.1.1 Classification of Sequential Circuits 6-2

6.2 Models for Synchronous Sequential Circuits 6-2

6.3 Moore Circuit 6-2

6.3.1 Example of Moore Circuit 6-3

6.4 Mealy Circuit 6-3

6.4.1 Example of Mealy Circuit 6-4

6.4.2 Comparison of Moore and Mealy Models 6-4

6.5 Finite State Machine (FSM) 6-4

6.5.1 Advantages of FSM 6-4

6.5.2 Disadvantages of FSM 6-5

6.5.3 State Machine Notations 6-5

6.6 Analysis of Clocked Sequential Circuits 6-5

6.7 State Reduction and Assignments 6-8

6.7.1 Row Elimination Method 6-8

6.7.2 Example on Row Elimination Method 6-8

6.7.3 Reduction in Number of FFs due to State Reduction 6-10

6.7.4 Implication Table Method 6-10

6.8 State Assignment 6-15

6.9 Design of Clocked Synchronous State Machine using State Diagram 6-16

6.10 Design using Unused States 6-18

6.10.1 Synthesis Table 6-18

6.11 Sequence Generator 6-19

6.11.1 Sequence Generator using Shift Register 6-19

6.11.2 Sequence Generator using Counters ...6-22

6.12 Sequence Detector 6-27

- **Review Questions 6-32**

Unit IV

Chapter 7 : Algorithmic State Machines 7-1 to 7-30

Syllabus : Finite State Machines (FSM) and ASM, ASM charts, Notations, Construction of ASM chart and realization for sequential circuits.

Exemplar / Case study : Waveform generator using MUX controller method.

TechKnowledge Publications

7.1 Introduction to ASM (FSM and ASM) 7-2

7.2 ASM Chart Notations 7-2

7.2.1 The State Box 7-2

7.2.2 Decision Box 7-3

7.2.3 Conditional Box 7-3

7.2.4 ASM Block 7-3

7.3 How to use the ASM Chart for Sequential Circuit Design ? 7-8

7.3.1 Realization of the Logic Circuit 7-8

7.4 Case Study Waveform Generator using MUX Controller Method 7-16

- **Review Questions 7-30**

Unit IV

Chapter 8 : Programmable Logic Devices 8-1 to 8-30

Syllabus : PLD, ROM as PLD, Programmable Logic Array (PLA), Programmable Array Logic (PAL), Designing combinational circuits using PLDs.

8.1 Introduction to PLDs 8-2

8.1.1 Advantages of PLDs.....8-2

8.1.2 Types of PLDs.....8-2

8.2 ROM used as PLD 8-3

8.2.1 Types of ROMs 8-3

8.2.2 Internal Logic of a ROM 8-3

8.2.3 Implementation of a Combinational Circuit (Generating the Boolean Function) 8-5

8.2.4 Advantages of ROM as PLD 8-6

8.2.5 Disadvantages of ROM as PLD.....8-6

8.3 Programmable Logic Array (PLA).....8-6

8.3.1 The Programming Procedure for PLA 8-8

8.3.2 Expansion of PLA Capacity.....8-8

8.3.3 Application Areas of PLA.....8-8

8.3.4 Designing of Combinational Circuit using PLA 8-8

8.3.5 How to Specify the Size of a PLA ?.....8-9

8.4 Programmable Array Logic (PAL) 8-18

8.4.1 Solved Examples on PAL.....8-19

8.4.2 Types of PAL Devices.....8-29

8.4.3 Registered PALs 8-29

8.4.4 Configurable PAL.....8-29

8.5 Comparison of PROM, PLA and PAL.....8-30

- **Review Questions 8-30**

Unit V

Chapter 9 : Logic Families 9-1 to 9-34

Syllabus : Classification of logic families : Unipolar and bipolar logic families, Characteristics of digital ICs : Fan-in, Fan-out, Current and voltage parameters, Noise immunity, Propagation delay, Power dissipation, Figure of Merits, Operating temperature range, Power supply requirements.

Transistor - Transistor logic : Operation of TTL NAND gate (Two input), TTL with active pull up, TTL with open collector output, Wired AND connection, Tristate TTL devices, TTL characteristics.

CMOS : CMOS inverter, CMOS characteristics, CMOS configurations - Wired logic, Open drain outputs.

Exemplar / Case study : To study the various basic gate design using TTL/CMOS logic family.

9.1 Introduction 9-2

9.1.1 Classification Based on Circuit Complexity 9-2

9.2 Classification of Logic Families 9-2

9.2.1 Classification Based on Devices Used 9-2

9.3 Characteristics of Digital ICs 9-3

9.3.1 Voltage and Current Parameters 9-3

9.3.2 Fan-in and Fan-out 9-4

Tech Knowledge Publications

9.3.3 Noise Margin 9-5
9.3.4 Propagation Delay (Speed of Operation) 9-5
9.3.5 Power Dissipation 9-6
9.3.6 Operating Temperature 9-6
9.3.7 Figure of Merit (Speed Power Product (SPP)) 9-7
9.3.8 Invalid Voltage Levels 9-7
9.3.9 Current Sourcing and Current Sinking 9-7
9.3.10 Power Supply Requirements 9-7
9.4 Transistor-Transistor Logic (TTL) 9-7
9.4.1 The Multiple Emitter Transistor 9-8
9.4.2 Two Input TTL-NAND Gate (Totempole Output) 9-8
9.4.3 Totem-pole (Active Pull up) Output Stage 9-10
9.4.4 Unconnected Inputs 9-11
9.4.5 Clamping Diodes 9-12
9.4.6 5400 Series 9-12
9.5 Sourcing and Sinking in TTL 9-12
9.5.1 Current Sinking Action 9-12
9.5.2 Current Sourcing Action 9-13
9.5.3 Three Input TTL NAND Gate 9-13
9.6 Wired AND Connection (TTL) 9-14
9.7 Open Collector Outputs (TTL) 9-15
9.7.1 Disadvantages of Open Collector Output 9-16
9.7.2 Advantage of Open Collector Output 9-16
9.7.3 Wired ANDing 9-16
9.7.4 Comparison of Totem-pole and Open Collector Outputs 9-17
9.8 Tristate (Three state) TTL Devices 9-17
9.8.1 Advantages of Tristate 9-18
9.8.2 Tristate Buffers 9-18
9.8.3 Applications of Tristate Buffers (Bus Organization) 9-19
9.8.4 A TRI-STATE Inverter 9-19
9.9 Standard TTL Characteristics 9-20
9.10 Advantages and Disadvantages of TTL 9-21
9.10.1 Advantages of TTL 9-21
9.10.2 Disadvantages of TTL 9-21
9.11 MOS - Logic Family 9-21
9.12 CMOS Logic 9-21
9.12.1 CMOS Inverter 9-22
9.12.2 CMOS NOR Gate 9-22
9.12.3 CMOS NAND Gate 9-24
9.12.4 CMOS Series 9-25
9.13 CMOS Characteristics 9-25
9.13.1 Power Supply Voltage 9-25
9.13.2 Logic Voltage Levels 9-26
9.13.3 Noise Margins 9-26
9.13.4 Power Dissipation 9-26
9.13.5 Fan Out 9-26
9.13.6 Switching Speed 9-27
9.13.7 Unconnected Inputs 9-27
9.14 Open Drain Outputs (CMOS) 9-27
9.14.1 Applications of Open Drain Inverter 9-28
9.15 Wired Logic (CMOS) 9-28
9.15.1 Advantages of CMOS 9-29
9.15.2 Disadvantages of CMOS 9-29
9.15.3 Handling Precautions for CMOS Devices 9-29

TechKnowledge Publications

9.15.4 Tristate Logic Output 9-29

9.16 Interfacing 9-30

9.16.1 TTL to CMOS Interfacing 9-30

9.16.2 TTL Driving High Voltage CMOS 9-31

9.16.3 Interfacing using Level-Shifter (TTL to High Voltage CMOS) 9-31

9.16.4 CMOS to TTL Interface 9-31

9.16.5 CMOS Driving TTL in the HIGH State ..9-32

9.16.6 CMOS Driving TTL in the LOW State ...9-32

9.16.7 High Voltage CMOS Driving TTL 9-33

9.17 Comparison of CMOS and TTL 9-33

- **Review Questions 9-34**

Unit VI

Chapter 10 : Introduction to Computer Architecture 10-1 to 10-28

Syllabus : Introduction to ideal microprocessor – Data bus, Address bus, Control bus, Microprocessor based systems – Basic operation, Microprocessor operation, Block diagram of microprocessor, Functional units of microprocessor – ALU using IC 74181, Basic arithmetic operations using ALU IC 74181, 4-bit multiplier circuit using ALU and shift registers, Memory organization and operations, Digital circuit using decoder and registers for memory operations.

Exemplar / Case Study : Microprocessor based system in Communication / Instrumentation Control.

10.1 Introduction to Ideal Microprocessor – Data Bus, Address Bus, Control Bus 10-2

10.1.1 Some Basic Terms used in Microprocessors 10-2

10.1.2 Microprocessor Characteristics and Functions.....10-2

10.1.2.1 Functions of a Microprocessor 10-3

10.1.3 Buses and Memory Accessing..... 10-3

10.1.3.1 Address Bus..... 10-3

10.1.3.2 Data Bus 10-4

10.1.3.3 Control Bus 10-4

10.1.4 ALU (Arithmetic and Logic Unit)..... 10-4

10.1.5 Some More Terms 10-5

10.2 Microprocessor Based Systems – Basic Operation..... 10-5

10.3 Block Diagram of Microprocessor 10-6

10.4 Functional Units of Microprocessor – ALU..... 10-8

10.5 ALU using IC 74181 10-8

10.6 Basic Arithmetic Operations using ALU IC 74181..... 10-10

10.6.1 Cascading of Two 74181 ALUs..... 10-10

10.6.2 2-bit Multiplier Circuit using ALU and Shift Registers..... 10-11

10.7 Memory Organization and Operations..... 10-13

10.7.1 Characteristics of Memory System 10-13

10.7.2 Memory Hierarchy..... 10-14

10.7.3 Basic Memory Cell 10-14

10.7.3.1 SRAM and DRAM 10-14

10.7.4 ROM (Read Only Memory)..... 10-15

10.7.4.1 Types of ROM 10-16

10.8 Digital Circuit Using Decoder and Registers for Memory Operations..... 10-17

10.8.1 Types of Memory..... 10-17

10.8.1.1 Memory Map, Structure and its Requirements..... 10-17

10.8.1.2 Memory Chip Size and Numbers 10-17

10.9 Exemplar / Case Studies : Microprocessor Based System in Communication / Instrumentation Control..... 10-24

TechKnowledge Publications

10.9.1 Requirement Analysis 10-24

10.9.2 Hardware Block Diagram..... 10-25

10.9.3 Software Model for Car Cruise Control..... 10-25

10.9.4 Component / Hardware Selection 10-26

- **Review Questions 10-26**

❑❑❑

TechKnowledge Publications

Unit I

Chapter 1

Minimization Techniques

Technical Knowledge Publications

Syllabus

Logic design minimization technique : Minimization of Boolean function using K-Maps (upto 4 variables) and Quine - McCluskey method, Representation of signed number - Sign magnitude representation, 1's complement and 2's complement form, Sum of product and product of sum form, Minimization of SOP and POS using K-Maps.

Exemplar / Case study : Digital locks using logic gates.

Chapter Contents

1.1	System or Circuit	1.7	Minimization of SOP Expressions (K Map Simplification)
1.2	Standard Representations for Logical Functions	1.8	Product of Sum (POS) Simplification
1.3	Concepts of Minterm and Maxterm	1.9	Quine Mc-Cluskey Minimization Technique (Tabular Method)
1.4	Methods to Simplify the Boolean Functions	1.10	Unsigned Binary Numbers
1.5	Karnaugh-Map Simplification	1.11	Sign-Magnitude Numbers
1.6	Minimization of Boolean Expressions using K-map	1.12	1's and 2's Complements of a Binary Number

1.1 System or Circuit :

- A system or circuit is defined as the physical device or group of devices or algorithm which performs the required operations on the signal applied at its input which can be either analog or digital.
- Depending on the type of input signal (analog or digital) the systems or circuits can be classified into two types :
  1. Analog systems
  2. Digital systems.

1.1.1 Digital Systems :

- We may define the digital system as the system which processes or works upon the digital signals to produce another digital signal.

(C-215) **Fig. 1.1.1 : A digital system**

- Thus the input signal to a digital system is digital and its output signal is also digital.

Input output relation :

- In Fig. 1.1.1, A, B, C ... are the inputs whereas Y or f (A, B ...) represents the output of the system. The system may have a single output or it can have multiple outputs.
- The system output is a function of inputs. Hence it is denoted by f (A, B, C, ...).
- The relation between input and output can be represented in various different ways as given below :
  1. Truth table.
  2. Switching equations.
  3. Logic diagram.

1. Truth table :

- Here the relation between the input variables (A, B, C ...) with the output Y is presented in a tabular form as shown in Table 1.1.1.
- The state of output (0 or 1) is written for all the possible combinations of inputs.

(C-8064) **Table 1.1.1**

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	1
	⋮		⋮
1	1	1	1

2. Switching equations :

- The relation between inputs and output(s) can be presented in the form of equation(s) called as switching equations.
- The input variables are called as switching variables. The output (Y) is written on the left hand side of the equation whereas the terms containing the input variables are written on the right hand side of the switching equation as shown below :

$$Y = ABC + A\bar{B}C + \bar{A}\bar{B}\bar{C} \quad ...(1.1.1)$$

$$\text{Or } f(A, B, C) = A\bar{B}C + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} \quad ...(1.1.2)$$

$$\text{Or } f(A, B, C) = (A + \bar{B} + C) \cdot (A + B + C) \quad ...(1.1.3)$$

- The switching equations are also called as the Boolean equations or the **system equations.**
- The system equations or switching equations can be of two different types :
  1. Sum of Products (SOP) format.
  2. Product of Sums (POS) format.

3. Logic diagram :

This is a diagrammatic way of expressing the input-output relationship of a digital circuit.

1.1.2 Types of Digital Systems :

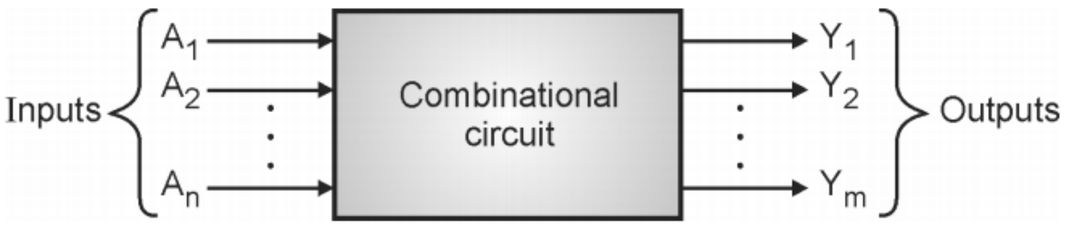
- The digital systems in general are classified into two categories namely :
  1. Combinational logic circuits.
  2. Sequential logic circuits.

Combinational circuits :

- It is a logic circuit, whose output at any instant of time, depends only on the levels present at input terminals.
- The combinational circuits do not use any memory. Hence the previous state of input does not have any effect on the present state of the circuit.
- The sequence in which the inputs are being applied also has no effect on the output of a combinational circuit.

Tech Knowledge Publications

– The block diagram of a combinational circuit is shown in Fig. 1.1.2.



(C-332) **Fig. 1.1.2 : Block diagram of a combinational circuit**

– A combinational circuit can have a number of inputs and a number of outputs.

– The circuit of Fig. 1.1.2 has "n" inputs and "m" outputs.

– Between the inputs and outputs, logic gates are connected so combinational circuit basically consists of logic gates.

– A combinational circuit operates in three steps :

1. It accepts n-different inputs.
2. The combination of gates operates on the inputs.
3. "m" different outputs are produced as per requirement.

Examples of combinational circuits :

– Following are the examples of some combinational circuits :

1. Adders, subtractors
2. Comparator
3. Code converters
4. Encoders, decoders
5. Multiplexers, demultiplexers

1.1.3 Combinational Circuit Design :

– In this chapter we are going to discuss the design of combinational circuits.

– To design a combinational circuit we have been given the specifications or the requirements of combinational circuits. Such specifications or requirements can be specified in the following ways :

1. A list of statements
2. Boolean expressions
3. Truth table.

– From the specified requirements we have to design a combinational circuit using a combination of gates which will fulfill all the requirements.

– We can adopt one of the following two approaches to the combinational circuit design :

1. Traditional methods.
2. Use of Medium Scale Integration (MSI).

– In this chapter we will discuss the traditional method.

Steps followed in traditional circuit design method :

1. A truth table is given.
2. Obtain the Boolean expression from the truth table.
3. Simplify the Boolean equation using standard methods.
4. Realize the simplified equation using gates i.e. build the logic circuit.

Methods to simplify the boolean equations :

– The methods used for simplifying the Boolean functions are as follows :

1. Algebraic method.
2. Karnaugh-map (K-map) simplification.
3. Quine-Mc Cluskey method and
4. Variable Entered Mapping (VEM) technique.

– The Boolean theorems and De-Morgan's theorems are useful in simplifying the given Boolean expressions. We can then realize the logical expressions using either the conventional gates or universal gates.

– We should use the minimum number of logic gates for the realization of a logical expression.

– This is possible if we can simplify the logical expressions. In this chapter we will discuss **Karnaugh map or K-map** and **Quine Mc-Cluskey method**.

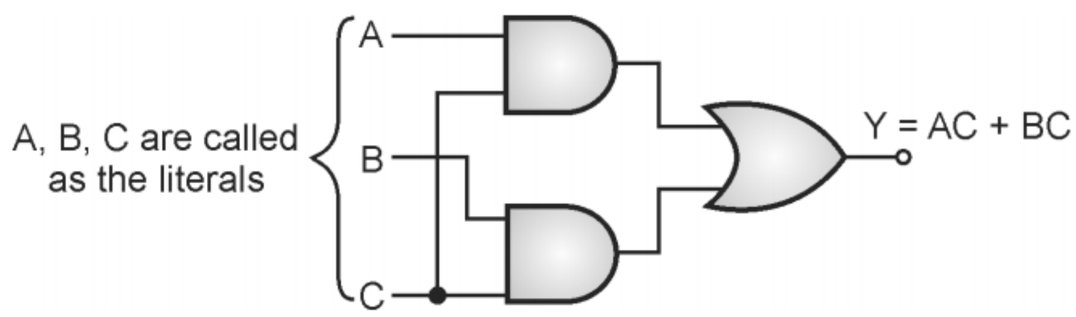
1.2 Standard Representations for Logical Functions :

– Consider that the logic expression given to us is as follows :

$$Y = AC + BC$$

– Then it can be realized using basic gates as shown in Fig. 1.2.1.

– In this Boolean expression, Y is the result or output and A, B, C are called as **literals.**



(C-1217) **Fig. 1.2.1 : Realization of given logic expression**

– When we realize the Boolean equation by using gates, each literal acts as an input as shown in Fig. 1.2.1.

TechKnowledge Publications

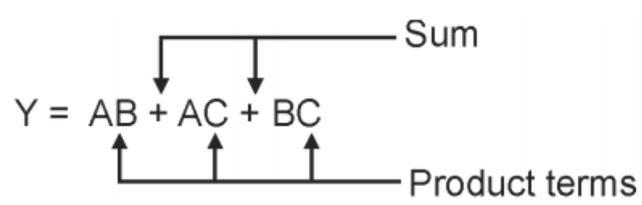
– Any logic expression can be expressed in the following two standard forms :

1. Sum-of-Products form (SOP) and
2. Product-of-Sums form (POS).

– These two forms are suitable for reducing the given logic expression to its simplest form.

1.2.1 Sum-of-Products (SOP) Form :

– Refer to logic expression shown in Fig. 1.2.2. It is in the form of sum of three terms AB, AC and BC with each individual term is product of two variables. Say A·B or A·C etc.



(C-1217(a)) Fig. 1.2.2 : Sum-of-Products (SOP) form

– Therefore such expressions are known as expression in **SOP** form.

– The sums and products in the SOP form are not the actual additions or multiplications. In fact they are the OR and AND functions.

– A, B and C are the **literals** or the inputs of the combinational circuit.

– Some more examples of the SOP expressions are as follows :

$$Y = ABC + BCD + AB\overline{D},$$

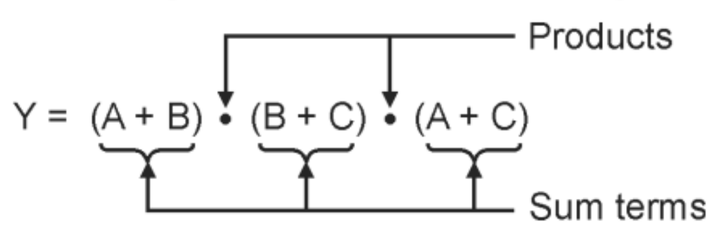
$$A = XY + \overline{X}Y + X\overline{Y},$$

$$Y = \overline{P}\,\overline{Q} + PQR + PQ\overline{R}$$

– Thus in each product term there can be one or more than one literals ANDed together. The literals can be in their complemented or uncomplemented form.

1.2.2 Product of the Sums Form (POS) :

– Refer to the logic expression shown in Fig. 1.2.3.



(C-1218) Fig. 1.2.3 : Product of Sums (POS) form

– It is in the form of product of three terms (A + B), (B + C) and (A + C), with each term is in the form of sum of two variables.

– Such expressions are said to be in the Product of Sums **(POS)** form.

– Some other examples of POS expressions form are as follows :

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

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

$$Y = (P + R) \cdot (P + \overline{Q}) \cdot (\overline{P} + R)$$

– The literals are ORed together to form the sum terms and the sum terms are ANDed to get the expression in the POS form.

1.2.3 Standard or Canonical SOP and POS Forms :

Canonical or standard forms :

– The word standard is used in order to describe a condition of switching equation. The meaning of the word standard is conforming to a general rule.

– This rule states that each term used in a switching equation must contain all the available input variables.

– The two formats of a switching equation in the standard form are :

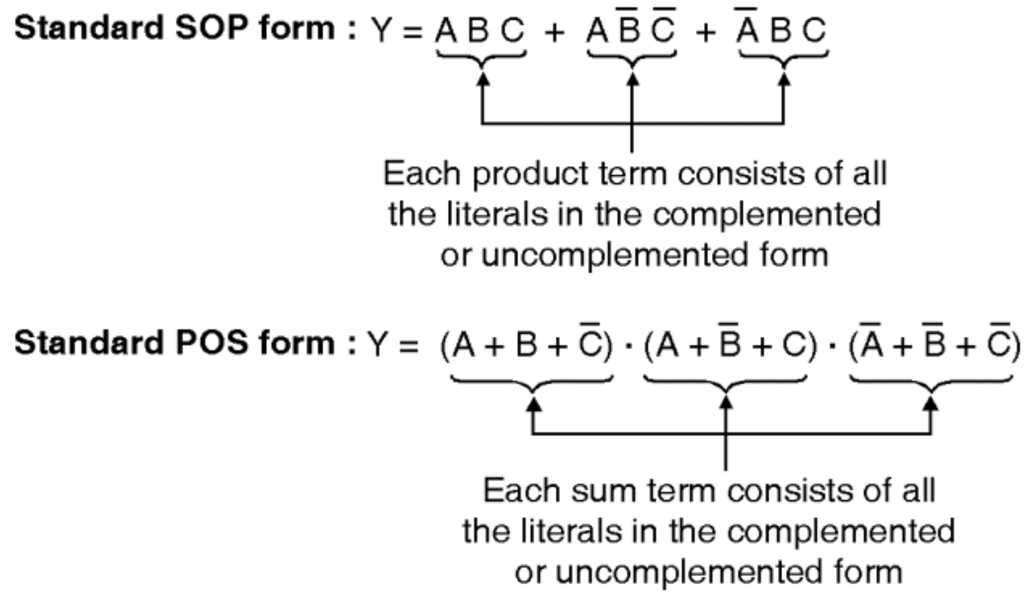
1. Sum of Product (SOP) format.
2. Product of Sum (POS) format.

– When we simplify a Boolean equation, sometimes an input variable is eliminated to simplify the equation.

– But standard expressions are not simplified. It is said that the standard expression is the opposite of simplification. So it contains redundancies.

– Many times, switching equations written in the SOP or POS form are not standard. That means each term may not contain all the input variables.

– Consider the SOP and POS expressions shown in Fig. 1.2.4.



(C-1219(a)) Fig. 1.2.4 : Standard SOP and POS forms

TechKnowledge Publications

– Referring to Fig. 1.2.4 we can say that a logic expression is said to be in the **standard** SOP or POS form if each product term (for SOP); and sum term (for POS) consists of all the literals in their complemented or uncomplemented form.

Non-standard forms :

– The two standard forms discussed earlier are the basic forms that are obtained from the truth table.

– However these forms are not used often because these equations are not in the minimized form because each term contains all the literals.

– There is another way to express the Boolean functions. It is called as the **non-standard form.**

– In this form each term may contain one, two or any number of literals. It is not necessary that each term should contain all the literals.

– There can be two types of non-standard forms :

1. Non-standard SOP form.
2. Non-standard POS form.

– The examples of standard and non-standard SOP and POS expressions are given in Table 1.2.1.

(C-8065) Table 1.2.1 : Non-standard and standard SOP and POS forms

Sr. No.	Expression	Type
1.	$Y = AB + AB\overline{C} + \overline{A}BC$	Non-standard SOP
2.	$Y = AB + A\overline{B} + \overline{A}\,\overline{B}$	Standard SOP
3.	$Y = (\overline{A} + B) \cdot (A + B) \cdot (A + \overline{B})$	Standard POS
4.	$Y = (\overline{A} + B) \cdot (A + B + C)$	Non-standard POS

1.2.4 Conversion of a Logic Expression to Standard SOP or POS Form :

– Let us see how to convert the given non-standard SOP and POS expressions into the corresponding non-standard SOP and POS forms.

Conversion from non-standard SOP to standard SOP form :

– The procedure to be followed for converting a non-standard SOP expression into a standard SOP expression is as follows :

Steps to be followed :

Step 1 : For each term in the given non-standard SOP expression find the missing literal.

Step 2 : Then AND this term with the term formed by ORing the missing literal and its complement.

Step 3 : Simplify the expression to get the standard SOP expression.

Ex. 1.2.1 : Convert the expression $Y = AB + A\overline{C} + BC$ into the standard SOP form.

Soln. :

– The given expression has 3-input variables A, B and C.

Step 1 : Find the missing literal for each term : **(C-2382)**

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

Missing literal → C B A

Step 2 : AND each term with (Missing literal + Its complement) : **(C-2383)**

$$Y = \underbrace{AB}_{\text{Original product term}} \overset{\text{ANDING}}{\cdot} \underbrace{(C + \overline{C})}_{\text{(Missing literal + Its complement)}} + A\overline{C}\,(B + \overline{B}) + BC\,(A + \overline{A})$$

Step 3 : Simplify the expression to get the standard SOP :

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

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

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

But A + A = A

$$\therefore (ABC + ABC) = ABC \text{ and } (AB\overline{C} + AB\,\overline{C}) = AB\overline{C}$$

$$\therefore Y = \underbrace{ABC + AB\overline{C} + A\overline{B}\,\overline{C} + \overline{A}BC}_{\text{Each term contains all the literals}} \qquad \text{Standard SOP form}$$

(C-6152)

Conversion from non-standard POS to standard POS form :

– The conversion of non-standard POS expression into standard POS form can be obtained by following the steps given below :

Steps to be followed :

Step 1 : For each term in the given non-standard POS expression, find the missing literal.

Step 2 : Then OR each such term with the term formed by ANDing the missing literal in that term with its complement.

Step 3 : Simplify the expression to get the standard POS.

Ex. 1.2.2 : Convert the expression $Y = (A + B)(A + C)(B + \overline{C})$ into standard POS form.

Soln. :

Step 1 : Find the missing literal for each term :

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

Missing literal → C B A

(C-2384)

Step 2 : OR each term with (Missing literal. Its complement) :

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

→ Missing literal ANDed with its complement

→ This term is ORed with the term formed by ANDing the missing literal with its complement

(C-6153)

Step 3 : Simplify the expression to get standard POS :

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

But $A + BC = (A + B)(A + C)$

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

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

But $A \cdot A = A$

$$\therefore (A + B + C)(A + C + B) = (A + B + C)$$

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

(C-6154)

$$\therefore Y = \underbrace{(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+B+\overline{C})}_{\text{Each term contains all the literals}} \quad \text{Standard POS form}$$

Ex. 1.2.3 : Convert the following expressions into their standard SOP or POS forms :

1. $Y = AB + AC + BC$
2. $Y = (A + B)(\overline{B} + C)$
3. $Y = A + BC + ABC$

Soln. : Solve it yourself.

Ans. :

1. $Y = ABC + AB\overline{C} + A\overline{B}C + \overline{A}BC$
2. $Y = (A + B + C)(A + B + \overline{C})(\overline{B} + C + A)(\overline{B} + C + \overline{A})$
3. $Y = ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + BC\overline{A}$

1.3 Concepts of Minterm and Maxterm :

Minterm :

– Each individual term in the standard SOP form is called as **minterm.** This is shown in Fig. 1.3.1.

Maxterm :

– Each individual term in the standard POS form is called as **maxterm.** This is shown in Fig. 1.3.1.

Standard SOP $Y = \underbrace{ABC}_{} + \underbrace{A\overline{B}\,\overline{C}}_{} + \underbrace{\overline{A}B\overline{C}}_{}$

Each individual term is called minterm

Standard POS $Y = \underbrace{(A + B)}_{} \cdot \underbrace{(A + \overline{B})}_{}$

Each individual term is called maxterm

(C-1220(a)) Fig. 1.3.1 : Concept of maxterm and minterm

– Table 1.3.1 gives the minterms and maxterms for a three variable/literal logic function. Let Y be the output and A, B, C be the inputs.

(C-8067) Table 1.3.1 : Minterms and maxterms for three variables

Variables			Minterms	Maxterms
A	B	C	m_i	M_i
0	0	0	$\overline{A}\,\overline{B}\,\overline{C} = m_0$	$A + B + C = M_0$
0	0	1	$\overline{A}\,\overline{B}\,C = m_1$	$A + B + \overline{C} = M_1$
0	1	0	$\overline{A}\,B\,\overline{C} = m_2$	$A + \overline{B} + C = M_2$
0	1	1	$\overline{A}\,B\,C = m_3$	$A + \overline{B} + \overline{C} = M_3$
1	0	0	$A\,\overline{B}\,\overline{C} = m_4$	$\overline{A} + B + C = M_4$
1	0	1	$A\,\overline{B}\,C = m_5$	$\overline{A} + B + \overline{C} = M_5$
1	1	0	$A\,B\,\overline{C} = m_6$	$\overline{A} + \overline{B} + C = M_6$
1	1	1	$A\,B\,C = m_7$	$\overline{A} + \overline{B} + \overline{C} = M_7$

– The number of minterms and maxterms is $2^3 = 8$. In general for "n" number of variables the number of minterms or maxterms will be 2^n.

– Each minterm is represented by m_i where i = 0, 1, ..., $2^n - 1$ and each maxterm is represented by M_i.

Writing minterm for a particular combination of ABC :

– Consider ABC = 011. Assume that A, B, C are input to an AND gate.

– We want the output of the AND gate to be 1. For that all its inputs should be 1.

– So take the complement of that input which is 0 i.e. A in this case and write the minterm.

$\therefore$ Minterm = $\overline{A}\,B\,C$ corresponding to ABC = 0 1 1

Similarly other minterms in Table 1.3.1 can be obtained.

Writing maxterm for a particular combination of ABC :

– Let ABC = 011.

– Assume that ABC are inputs to an OR gate.

TechKnowledge Publications

– We want the output of the OR gate to be 0. So all the inputs to the OR gate should be 0s.

– So invert the inputs which are 1s. i.e. B and C in this case and write the maxterm.

∴ Maxterm = $(A + \overline{B} + \overline{C})$ corresponding to ABC = 0 1 1

Similarly other maxterms in Table 1.3.1 can be obtained.

Ex. 1.3.1 : For the truth table of two variables write the minterms and maxterms.

Soln. :

Refer Table P. 1.3.1 for solution.

(C-8068) Table P. 1.3.1 : Solution to Ex. 1.3.1

Variables/literals		Minterms	Maxterms
A	B	m_i	M_i
0	0	$m_0 = \overline{A}\ \overline{B}$	$M_0 = A + B$
0	1	$m_1 = \overline{A}\ B$	$M_1 = A + \overline{B}$
1	0	$m_2 = A\overline{B}$	$M_2 = \overline{A} + B$
1	1	$m_3 = A\ B$	$M_3 = \overline{A} + \overline{B}$

1.3.1 Representation of Logical Expressions using Minterms and Maxterms :

– We can represent the logical expression using the minterms and maxterms as follows :

1. $Y = \underbrace{ABC}_{m_7} + \underbrace{\overline{A}BC}_{m_3} + \underbrace{A\overline{B}\overline{C}}_{m_4}$ ← Given logic expression ← Corresponding minterms

$\therefore\ Y = m_7 + m_3 + m_4 = \Sigma m\ (3, 4, 7)$ ← Other way of representation

(C-6155)

where Σ denotes sum of products.

2. $Y = \underbrace{(A+\overline{B}+C)}_{M_2}\ \underbrace{(A+B+C)}_{M_0}\ \underbrace{(\overline{A}+\overline{B}+C)}_{M_6}$ ← Given expression ← Corresponding maxterms

$\therefore\ Y = M_2\, M_0\, M_6$

$\therefore\ Y = \Pi M\ (0, 2, 6)$ ← Other way of representation

where Π denotes product of sums.

(C-6156)

1.3.2 Writing SOP and POS Forms for a Given Truth Table :

– We know that a logic expression can be represented in the truth table form. For example, the expression $Y = AB + \overline{A}\,\overline{B}$ which is the Boolean expression for an EX-NOR gate can also be represented using a truth table.

– Hence it is possible to obtain the logic expression in the standard SOP or POS form if a truth table is given to us.

1.3.3 Representation of Truth Table using Standard SOP Expression :

– The procedure to be followed for writing the standard SOP expression from a given truth table is as follows :

Step 1 : From the given truth table, consider only those combinations of inputs which produce an output Y = 1.

Step 2 : Write down a product term interms of input variables for each such combination.

Step 3 : OR all these product terms produced in step 2 to get the standard SOP.

Ex. 1.3.2 : From the truth table P. 1.3.2, obtain the logical expression in the standard SOP form.

(C-8114) Table P. 1.3.2 : Given truth table

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

Soln. :

Step 1 : Consider only those combinations of inputs which correspond to Y = 1.

Steps 2 and 3 :

– For the second and the third entries in Table P. 1.3.2(a) write the product terms :

(C-6157) Table P. 1.3.2(a)

	A	B	Y
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

$Y_1 = \overline{A}B$, $Y_2 = A\overline{B}$ — Boolean expressions in the product forms

– Now OR (Add) all the product terms to write the final expression in standard SOP form as follows :

$$\therefore\ Y = Y_1 + Y_2$$

$$= \overline{A}\,B + A\overline{B} = m_1 + m_2$$

$$= \Sigma\, m\ (1, 2)$$

Ex. 1.3.3 : For the truth table shown in Table P. 1.3.3 write the logic expression in the standard SOP form.

TechKnowledge Publications

(C-6412) Table P. 1.3.3 : Given truth table

A	B	C	Y	
0	0	0	0	
0	0	1	1	$\bar{A}\bar{B}C$ (m_1)
0	1	0	0	
0	1	1	0	
1	0	0	1	$A\bar{B}\bar{C}$ (m_4)
1	0	1	0	
1	1	0	0	
1	1	1	1	ABC (m_7)

Soln. :

Step 1 : Product terms corresponding to combinations of inputs for which Y = 1.

Step 2 : OR (Add) all the product terms :

– ORing all the product terms we get,

$$Y = \bar{A}\,\bar{B}C + A\bar{B}\bar{C} + A\,B\,C \qquad \textbf{...Ans.}$$

$$\therefore Y = m_1 + m_4 + m_7 = \Sigma\, m\,(1, 4, 7)$$

1.3.4 Representation of Truth Table using Standard POS Expression :

– Follow the procedure given below to get the expression in standard POS expression from a given truth table.

Step 1 : From the given truth table, consider only those combinations of inputs which produce a low output (Y = 0).

Step 2 : Write the maxterms only for such combinations.

Step 3 : AND these maxterms to obtain the expression in standard POS form.

Ex. 1.3.4 : Write the logic expression in standard POS form for the truth table shown in Table P. 1.3.4.

(C-6413) Table P. 1.3.4 : Given truth table

A	B	C	Y	
0	0	0	0	$A + B + C$ (M_0)
0	0	1	1	
0	1	0	1	
0	1	1	0	$A + \bar{B} + \bar{C}$ (M_3)
1	0	0	1	
1	0	1	0	$\bar{A} + B + \bar{C}$ (M_5)
1	1	0	0	$\bar{A} + \bar{B} + C$ (M_6)
1	1	1	1	

Soln. :

Step 1 : Write maxterms for the combinations of input which produce Y = 0.

Step 2 : AND (take product of) all the maxterms to get standard POS form :

– ANDing (taking product of) all the maxterms written in step 1 we get,

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

$$\therefore Y = M_0 \cdot M_3 \cdot M_5 \cdot M_6 = \Pi\, M\,(0, 3, 5, 6)$$

1.3.5 Conversion from SOP to POS and Vice Versa :

– It is important to note that the SOP and POS forms written for the same truth table are always logically equivalent.

– This point can be proved by solving the following example.

Ex. 1.3.5 : For the given truth table write the logical expressions in the standard SOP and POS forms and prove their equivalence.

(C-6414) Table P. 1.3.5 : Given truth table

Minterms	A	B	C	Y	Maxterms
	0	0	0	0	$A + B + C$
$\bar{A}\,\bar{B}\,C$	0	0	1	1	
	0	1	0	0	$A + \bar{B} + C$
	0	1	1	0	$A + \bar{B} + \bar{C}$
$A\,\bar{B}\,\bar{C}$	1	0	0	1	
	1	0	1	0	$\bar{A} + B + \bar{C}$
	1	1	0	0	$\bar{A} + \bar{B} + C$
$A\,B\,C$	1	1	1	1	

Soln. :

Step 1 : Minterms and maxterms for the combinations of inputs producing Y = 1 and 0 respectively.

Step 2 : Write the standard SOP and POS expressions :

Standard SOP form :

$$Y = \bar{A}\,\bar{B}\,C + A\bar{B}\bar{C} + ABC$$

Standard POS form :

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

Step 3 : To prove the equivalence between SOP and POS forms :

– Consider the standard POS form.

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

TechKnowledge Publications

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

$= (A + C)$

$\therefore\ Y = \underset{(1)}{(A + C)}\ \underset{(2)}{(A + \bar{B} + \bar{C})}\ \underset{(3)}{(\bar{A} + B + \bar{C})}\ \underset{(4)}{(\bar{A} + \bar{B} + C)}$ **(C-6354)**

– Multiply the terms (1 × 2) and (3 × 4) to get,

$$Y = [A A + A\bar{B} + A\bar{C} + AC + \bar{B}C + C\bar{C}]$$

$$[\bar{A}\bar{A} + \bar{A}\bar{B} + \bar{A}C + \bar{A}B + B\bar{B} + BC + \bar{A}\bar{C} + \bar{B}\bar{C} + C\bar{C}]$$

But $AA = A$, $C\bar{C} = 0$, $\bar{A}\bar{A} = \bar{A}$, $B\bar{B} = 0$, $C\bar{C} = 0$

$$\therefore Y = [A + A\bar{B} + A\bar{C} + AC + \bar{B}C]$$

$$[\bar{A} + \bar{A}\bar{B} + \bar{A}C + \bar{A}B + BC + \bar{A}\bar{C} + \bar{B}\bar{C}]$$

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

$$[(1 + \bar{B})\bar{A} + \bar{A}(C + \bar{C}) + \bar{A}B + BC + \bar{B}\bar{C}]$$

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

$$\therefore Y = [A + A + \bar{B}C]\,[\bar{A} + \bar{A} + \bar{A}B + BC + \bar{B}\bar{C}]$$

But $A + A = A$

And $\bar{A} + \bar{A} = \bar{A}$

$$\therefore Y = (A + \bar{B}C)\,[\bar{A} + \bar{A}B + BC + \bar{B}\bar{C}]$$

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

But $(1 + B) = 1$

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

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

But $A\bar{A} = 0$, $(\bar{B}C \cdot BC) = 0$ and $(\bar{B}C \cdot \bar{B}\bar{C}) = 0$

$\therefore Y = ABC + A\bar{B}\bar{C} + \bar{A}\bar{B}C$ ← This is same as standard SOP form. Thus the equivalence SOP and POS forms is proved.

(C-6363)

Step 4 : Express output in terms of minterms and maxterms :

– The output can be expressed in terms of minterms and maxterms as follows :

$Y = m_1 + m_4 + m_7$ ...In terms of minterms.

$= \sum m\,(1, 4, 7)$

and $Y = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$ ...In terms of maxterms.

$= \Pi\, M\,(0, 2, 3, 5, 6)$

– Due to equivalence between SOP and POS forms we can write,

$$\sum m\,(1, 4, 7) = \Pi\, M\,(0, 2, 3, 5, 6) \quad ...(1)$$

Complementary relationship between minterms and maxterms :

– From Equation (1) we can conclude that the relationship between the expressions expressed using minterms and maxterms is complementary relationship.

– We can exploit this complementary relationship to write the expression in terms of maxterms if the expression in terms of minterms is known and vice versa.

– For example, if a SOP expression for 4-variables is given by,

$$Y = \sum m\,(0, 1, 3, 5, 6, 7, 11, 12, 15)$$

– Then we can get the equivalent POS expression using the complementary relationship as follows :

$$Y = \Pi\, M\,(2, 4, 8, 9, 10, 13, 14)$$

1.4 Methods to Simplify the Boolean Functions :

– The methods used for simplifying the Boolean expressions are as follows :

1. Algebraic method.
2. Karnaugh-map simplification.
3. Quine Mc-Cluskey method and
4. Variable Entered Mapping (VEM) technique.

– The Boolean theorems and De-Morgan's theorems are useful in simplifying the given Boolean expressions.

– We can then realize the logical expressions using either the standard gates or universal gates.

– We should use the minimum possible number of logic gates for the realization of a logical expression.

– This is possible if we can simplify the logical expressions.

– In this chapter we will discuss one of the simplification techniques called **Karnaugh map or K-map** and the **Quine Mc-Cluskey Method** or the **Tabular Method.**

1.4.1 Algebraic Simplification :

– We have studied the Boolean laws and De-Morgan's theorems.

– We can use them to simplify the given Boolean expression in the following way.

– The most important thing is to convert the given expression into SOP form.

TechKnowledge Publications

Standard procedure for algebraic simplification :

Step 1 :	Bring the given expression into the SOP form by using the Boolean laws and De-Morgan's theorems.
Step 2 :	Simplify this SOP expression by checking the product terms for common factors.

Ex. 1.4.1 : Simplify the expression given below :

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

Soln. :

Step 1 : Bring the given expression in SOP form :

Given expression :

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

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

Step 2 : Search for common factors and simplify :

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

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

But $AB + AB = AB$, $BB = B$ and $A\bar{A} = 0$

$$\therefore Y = AB + B + \bar{A}B = B(A + 1) + \bar{A}B$$

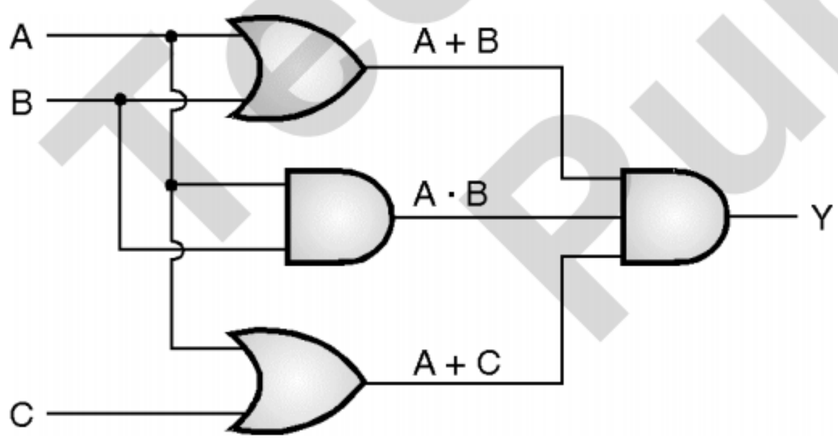
But $(A + 1) = 1$

$$\therefore Y = B + \bar{A}B = B(1 + \bar{A}) = B \text{ ...since } (1 + \bar{A}) = 1$$

$$\therefore Y = B$$ **....Ans.**

– This is simplified expression.

Ex. 1.4.2 : For the logic circuit shown in Fig. P. 1.4.2 write the Boolean expression and simplify it.



(C-1221) **Fig. P. 1.4.2 : Given logic circuit**

Soln. :

Step 1 : Write the Boolean expression :

– The expression for output of the given logic circuit is,

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

Step 2 : Bring this expression in SOP form :

– Multiply the terms to get the expression into SOP form.

$$\therefore Y = (A + B)(AAB + ABC)$$

But $AA = A$

$$\therefore Y = (A + B)(AB + ABC)$$

$$= AAB + AABC + BAB + BABC$$

$$= AB + ABC + AB + ABC$$

$$= AB + AB + ABC + ABC$$

But $AB + AB = AB$ and $ABC + ABC = ABC$

$$\therefore Y = AB + ABC = AB(1 + C)$$

$$Y = AB \quad \text{...since } 1 + C = 1.$$

– This is the simplified expression.

Ex. 1.4.3 : Simplify the following three variable logic expression, $Y = \Pi M(1, 3, 5)$.

Soln. :

– The given expression can be expressed in terms of maxterms as,

$$Y = M_1 \cdot M_3 \cdot M_5$$

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

Step 1 : Bring the expression into SOP form :

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

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

Step 2 : Simplify :

But $A\bar{A} = 0$, $\bar{B}B = 0$ and $\bar{C}\bar{C} = \bar{C}$

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

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

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

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

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

But $\bar{C} + \bar{C} = \bar{C}$

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

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

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

$$= AAB + A\bar{A}\bar{B} + A\bar{C} + ABB + \bar{A}\bar{B}B + B\bar{C} + AB\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{C}\bar{C}$$

But $AAB = AB$, $A\bar{A}\bar{B} = 0$, $ABB = AB$, $\bar{A}\bar{B}B = 0$ and $\bar{C}\bar{C} = \bar{C}$

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

But $AB + AB = AB$

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

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

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

$= AB + A\overline{C} + B\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + \overline{C}$ ...since $(1 + \overline{C}) = 1$

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

$\therefore\ Y = AB + \overline{C}$ ...since $(A + B + \overline{A}\,\overline{B} + 1) = 1$

$\therefore\ Y = AB + \overline{C}$ **...Ans.**

– This is simplified expression.

Ex. 1.4.4 : Simplify the following three variable Boolean expression.

$Y = \sum m\,(2, 4, 6)$

Soln. : Solve it yourself.

Ans. : $Y = AB + \overline{C}$

Ex. 1.4.5 : Convert the expression $Y = \Pi\ M\,(1, 3, 5)$ into minterms and simplify to get the same answer.

Soln. : Solve it yourself.

Ans. : $Y = \overline{C} + AB$

1.4.2 Disadvantages of Algebraic Method of Simplification :

1. In the algebraic method of simplification, we need to write lengthy equations, find the common terms, and use various laws and theorems to minimize the expressions. So it is a time consuming work.
2. On top of that, sometimes we are not sure whether the further simplification is really possible or not. That means whether we have really obtained the minimized expression or not.

1.5 Karnaugh-Map Simplification :

– This is another simplification technique to reduce the Boolean equation.

– It overcomes all the disadvantages of the algebraic simplification technique.

– K-map (short form of Karnaugh map) is a graphical method of simplifying a Boolean equation.

– K-map is a graphical chart made up of rectangular boxes.

– The information contained in a truth table or available in the SOP or POS form can be represented on a K-map.

– The K-map can be used for systematic simplification of Boolean expression.

– K-maps can be written for 2, 3, 4 ... upto 6 variables. Beyond that the K-map technique becomes very cumbersome.

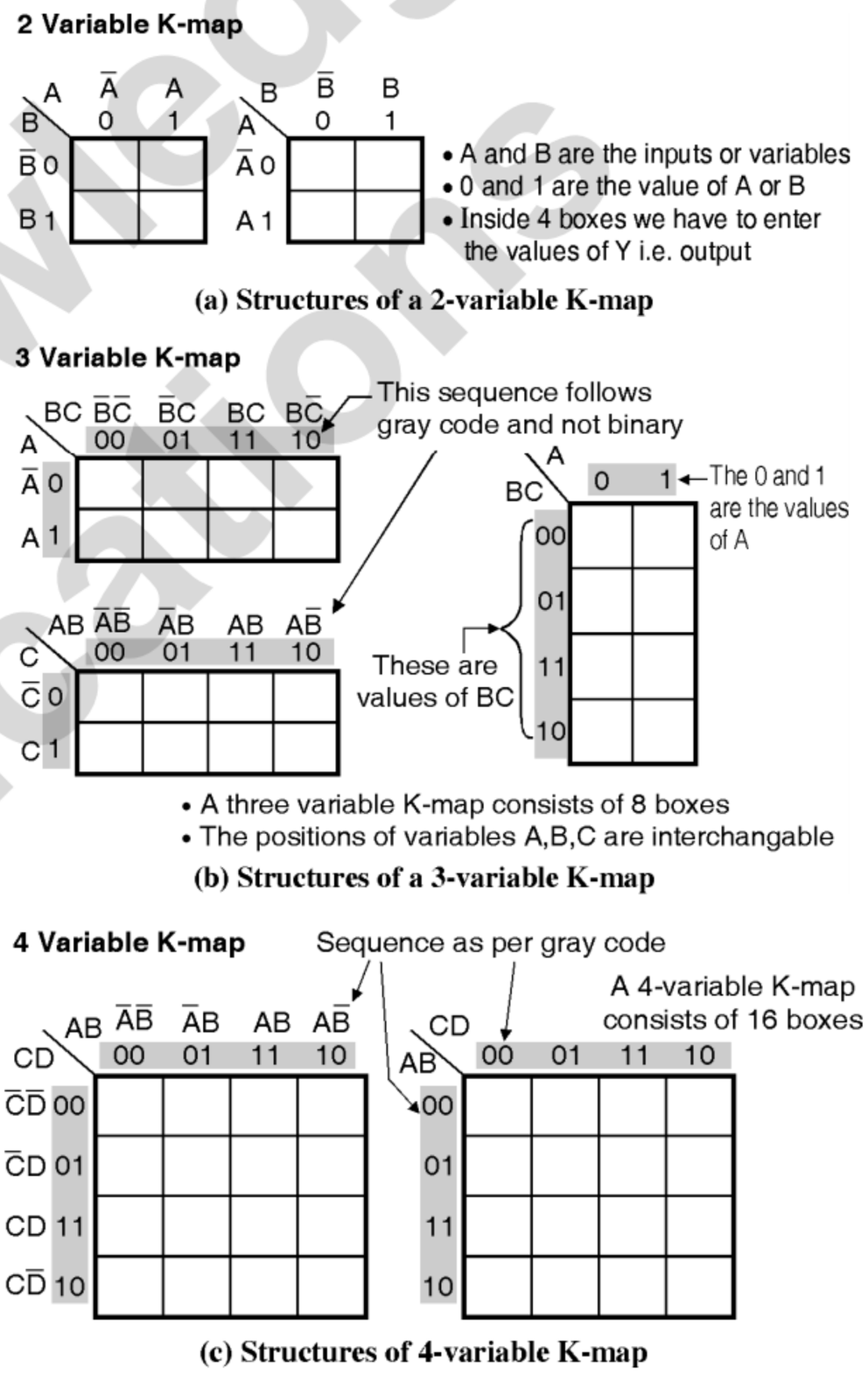
K-map is ideally suitable for designing the combinational logic circuits using either a SOP method or a POS method.

– The K-map is drawn for output Y and the input variables (A, B, C... etc.) are used for making the entries in the boxes.

1.5.1 K-map Structure :

– The structure of a 2 input (variable) Karnaugh is shown in Fig. 1.5.1(a).

– This K-map is drawn for output Y of any two input combinational circuit such as a logic gate with inputs A and B. i.e. AB = 00, 01, 10 and 11.



(C-217) **Fig. 1.5.1 : Structures of 2, 3, 4 variable K-map**

– A 2 variable K-map consists of $2^2 = 4$ rectangular boxes. Inside these boxes we have to enter the values of output Y for different combinations of inputs A and B.

– The K-map comprises a box for every line in the truth table. For a 2-input combinational circuit there are 4-lines in the truth table, so there will be 4-boxes in the 2 variable K-map.

TechKnowledge Publications

– For a 3-variable K-map there will be 8 boxes, for 4-variable map there will be 16 boxes and so on.

– The 0's and 1's written on top or sides of the boxes represent the values of the corresponding variables.

The sequence is funny (It is gray code) :

– In truth tables, the values of inputs follow a standard binary sequence (00, 01, 10, 11).

– But in K-maps the input values are ordered in a different sequence (00, 01, 11, 10).

– This is as per the **gray code** and not binary code. So that as we move along a row or column only one variable will change its value at a time.

– If the labeling is done as per the gray code, only then the elimination of variables and therefore the simplification will take place.

– When pairs, quads or octets are identified with the normal labeling, this elimination will not take place.

– Hence gray code is used in labeling the cells of a K-map.

1.5.2 K-map Boxes and Associated Product Terms :

– The rectangular boxes in a K-map are to be filled with the values of output Y corresponding to different combinations of inputs, as shown in Fig. 1.5.2.

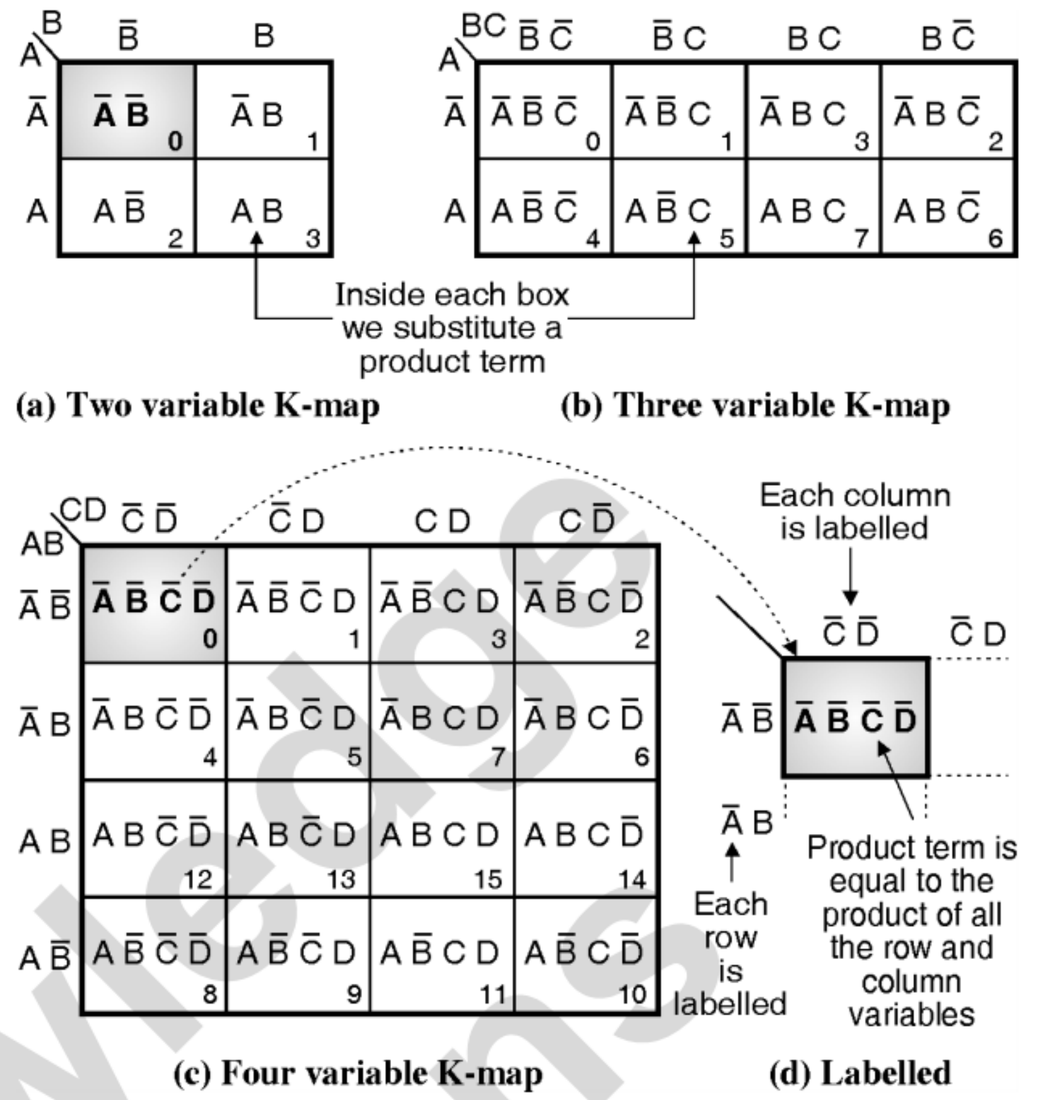
– Each row and column of a K-map is labelled with a variable, or a group of variables or their complements. For example, see the shaded box in Fig. 1.5.2(a).

– This box corresponds to the first row which is labelled by $\bar{A}$ and first column labelled by $\bar{B}$.

– Hence the product term written inside this box is $\bar{A}\,\bar{B}$ as shown in Fig. 1.5.2(a).

– Another example is the shaded box shown in Fig. 1.5.2(c). The first row is labelled with $\bar{A}\,\bar{B}$ and the first column with $\bar{C}\,\bar{D}$. Hence the product term written in this box is $\bar{A}\,\bar{B}\,\bar{C}\,\bar{D}$.

– Similarly the other product terms have been inserted, in the remaining boxes.



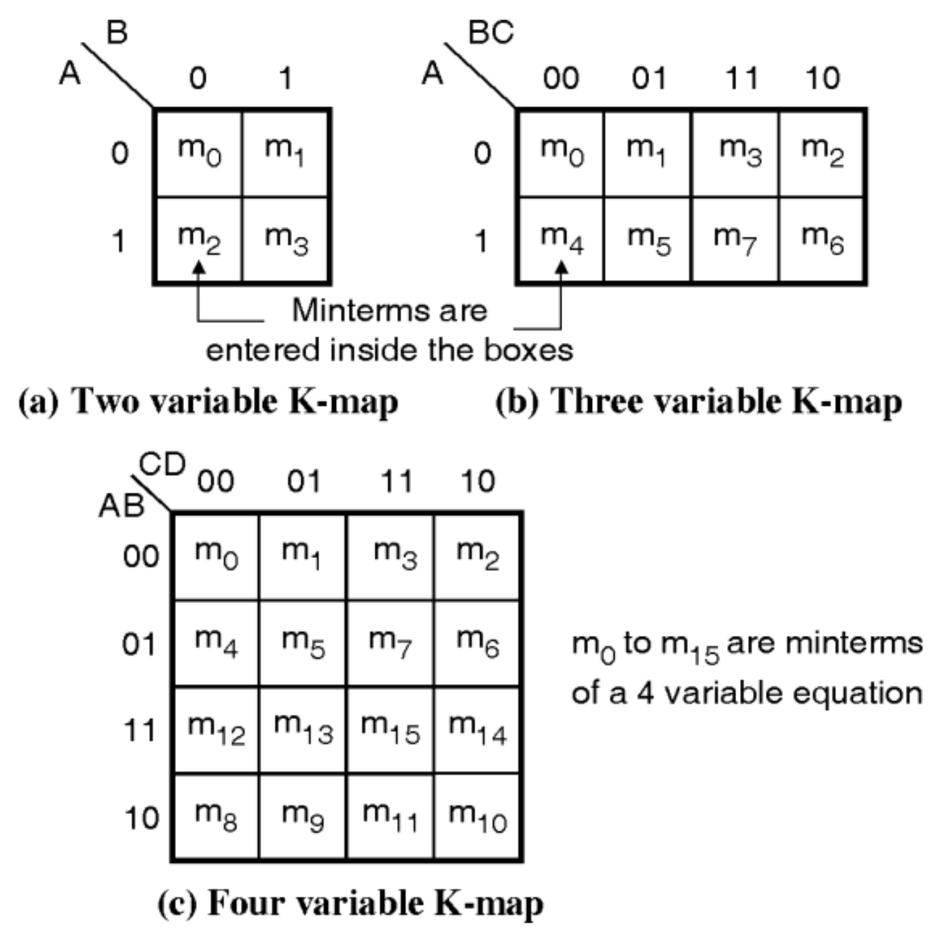
(C-218) **Fig. 1.5.2 : K-map boxes and associated product terms**

1.5.3 Alternative Way to Label the K-map :

– We can label the rows and columns of a K-map in a different way as shown in Fig. 1.5.3.

– Instead of labelling the rows and columns with the inputs and their complements (A, $\bar{A}$, $A\bar{B}$ etc.), their values in terms of 0s and 1s is used for labelling.

– And inside the boxes, instead of writing the actual product term, the corresponding shorthand minterm notations m_0, m_1 ..... are entered.



(C-219) **Fig. 1.5.3 : Alternative way to label K-map**

1.5.4 Truth Table to K-map :

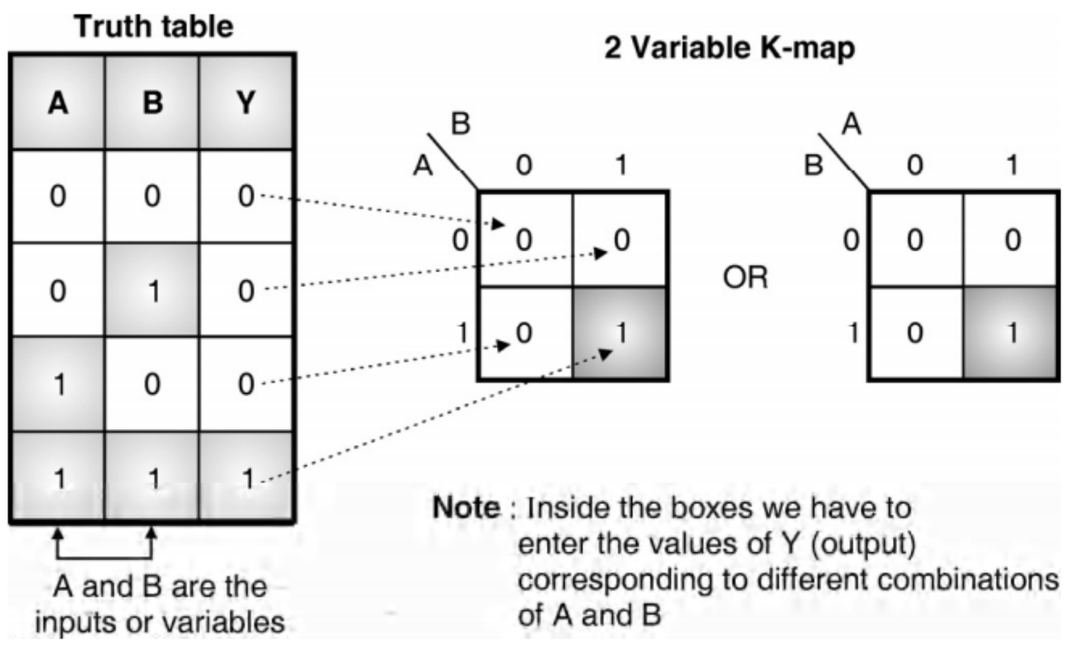
– Whenever the K-map is to be practically used for simplification, the entries inside the boxes are to be written by referring to the given truth table.

TechKnowledge Publications

– In this section we are going to learn how to represent the given truth table on a Karnaugh map.

Relation between a truth table and K-map entries :

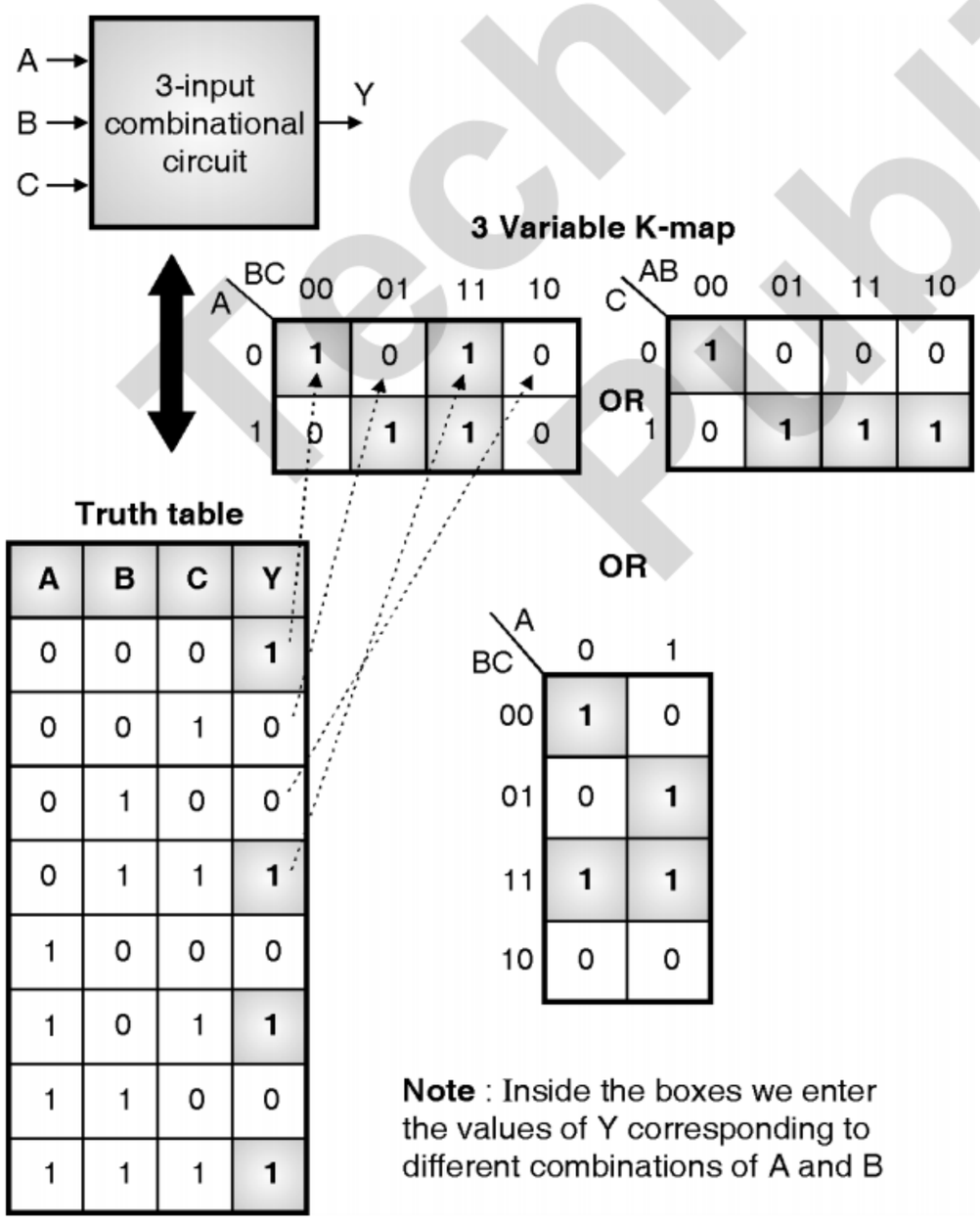
– Fig. 1.5.4(a) illustrates the mapping of a truth table on the K-map.



(C-220) Fig. 1.5.4(a) : Relation between a truth table and K-maps for 2 variables

– Referring to Fig. 1.5.4(a) we conclude that inside the boxes of the K-map we have to enter the values of output Y corresponding to various combinations of A and B.

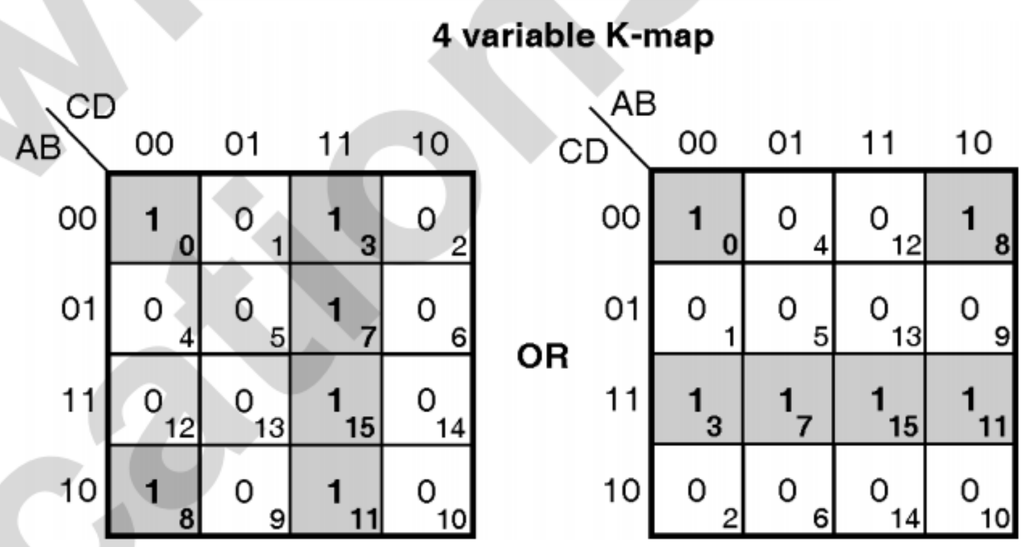
– Fig. 1.5.4(b) shows the representation of a truth table using a 3-variable K-map and Fig. 1.5.4(c) shows the representation of a truth table using a 4-variable K-map.



(C-221) Fig. 1.5.4(b) : Relation between a truth table and K-map for 3-variables

(C-6693) Truth table

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



(C-222) Fig. 1.5.4(c) : Relation between the truth table and 4-variable K-map

1.5.5 Representation of Standard SOP Form on K-map :

– The logical expression in standard SOP form can be represented with the help of a K-map by simply entering 1's in the cells (boxes) of the K-map corresponding to each minterm present in the equation.

– The remaining cells (boxes) are filled with zeros.

– Ex. 1.5.1 illustrates the concept of transferring a standard SOP expression on K-map.

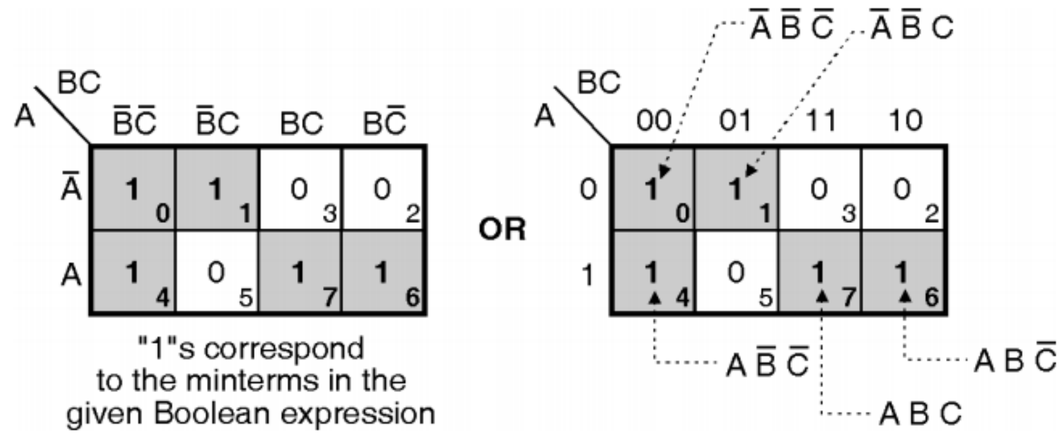
Ex. 1.5.1 : Represent the equation given below on Karnaugh map.

$$Y = \bar{A}\,\bar{B}\bar{C} + \bar{A}\,\bar{B}\,C + A\bar{B}\bar{C} + AB\bar{C} + ABC.$$

Soln. :

– The given expression is in the standard SOP form. Each term represents a minterm.

– We have to enter 1's in the boxes corresponding to each minterm, as shown in Fig. P. 1.5.1.

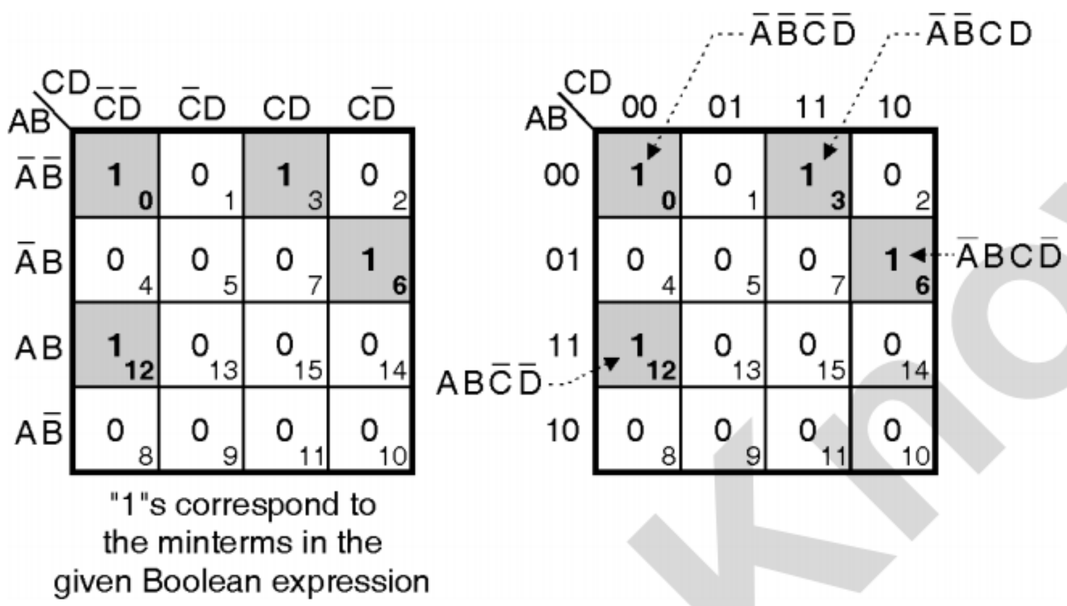


(C-223) Fig. P. 1.5.1 : Representation of standard SOP on K-map

Ex. 1.5.2 : Plot the following Boolean expression on K-map.

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

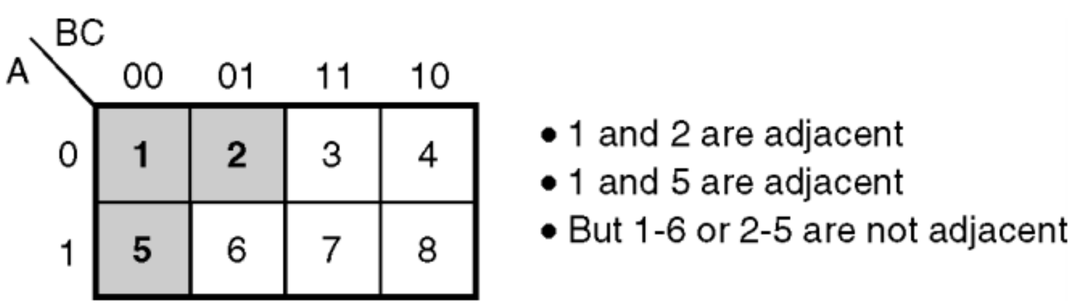
Soln. : Refer Fig. P. 1.5.2.



(C-224) Fig. P. 1.5.2 : Representation of canonical SOP on Karnaugh map

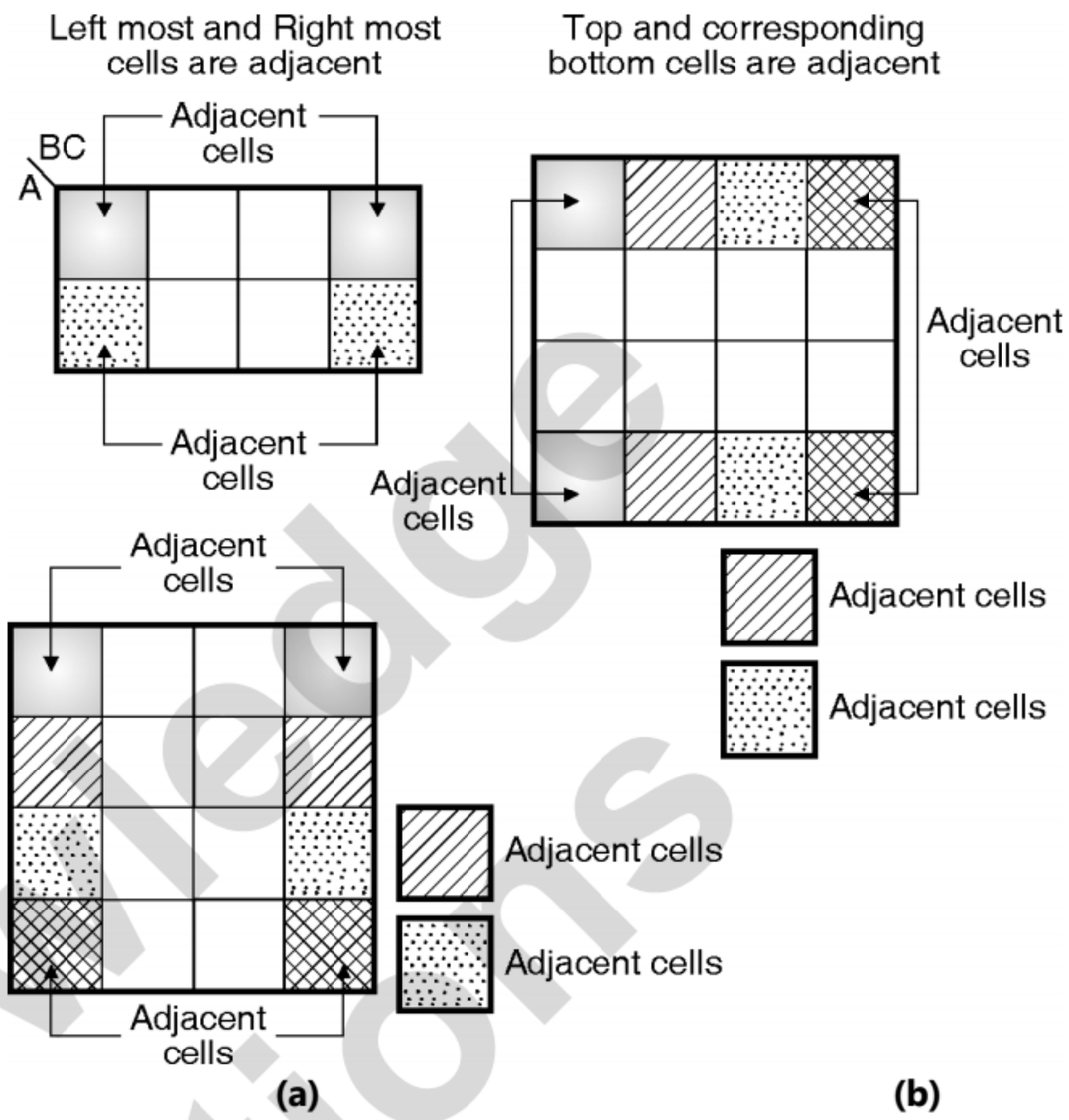
1.6 Minimization of Boolean Expressions using K-map :

- Simplification of Boolean expressions using K-map is based on combining or grouping the terms in the adjacent cells (or boxes) of a K-map.
- Two cells of a K-map are said to be adjacent if they differ in only one variable as shown in Fig. 1.6.1.



(C-225) Fig. 1.6.1 : Adjacent or non-adjacent cells

- Note the cells on left or right side or at the bottom and top of cells are adjacent cells. But the cells connected diagonally are not the adjacent ones.
- The left most cells are adjacent to their corresponding right most cells as shown in Fig. 1.6.2(a).
- And the top cells are adjacent to their corresponding bottom cells.



(C-226) Fig. 1.6.2 : Illustration of adjacent cells

1.6.1 How does Simplification Takes Place ?

- Once we transfer the logic function or truth table on a Karnaugh map, we have to use the grouping technique for simplifying the logic function.
- Grouping means combining the terms in the adjacent cells.
- The grouping of either adjacent 1's or adjacent 0's results in the simplification of Boolean expression in the SOP or POS forms respectively.
- If we group the adjacent 1's then the result of simplification is in SOP (Sum Of Products) form.
- And if the adjacent 0's are grouped, then the result of simplification is in the POS (Product Of Sums) form.

1.6.2 Way of Grouping (Pairs, Quads and Octets) :

- While grouping, we should group most number of 1's (or 0's).
- The grouping follows the binary rule i.e. we can group 1, 2, 4, 8, 16, 32 ..... number of 1's or 0's. We cannot group 3, 5, 7, ... number of 1's or 0's.

1. **Pairs :** A group of two adjacent 1's or 0's is called as a pair.

Tech Knowledge Publications

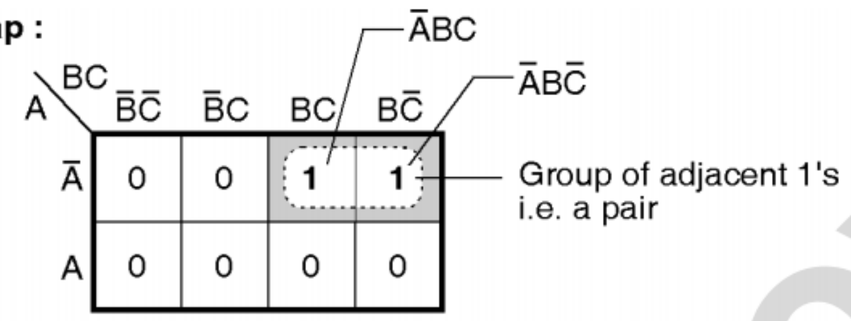
2. **Quad :** A group of four adjacent 1's or 0's is called as a quad.
3. **Octet :** A group of eight adjacent 1's or 0's is called as octet.

1.6.3 Grouping Two Adjacent One's (Pairs) :

> If we group two adjacent 1's on a K-map, to form a pair, then the resulting term will have one less literal (variable) than the original term. **That means by pairing two adjacent 1's we can eliminate one variable.**

– The grouping of two adjacent 1's and elimination of one variable due to pairing is illustrated in Figs. 1.6.3(a) to (d).

Given K-map :



(C-228) **Fig. 1.6.3(a)**

Simplification :

$$Y = \bar{A}BC + \bar{A}B\bar{C}$$
$$= \bar{A}B(C + \bar{C})$$
$$= \bar{A}B \quad \text{....since } C + \bar{C} = 1$$

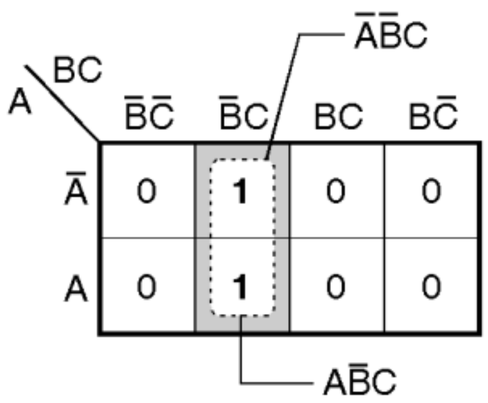
– Thus C is eliminated.

Conclusion :

– Due to formation of pair the variable C is eliminated. So henceforth just by looking at the pair we should be able to identify the variable that will be eliminated.

– The other types of pairs and corresponding simplifications are as shown in Fig. 1.6.4(a) and Fig. 1.6.4(b).

Given K-map :



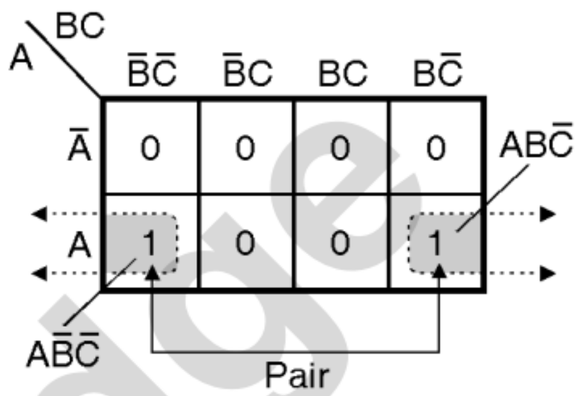
(C-228) **Fig. 1.6.3(b)**

Simplification :

$$Y = \bar{A}\bar{B}C + A\bar{B}C$$
$$= \bar{B}C(\bar{A} + A)$$
$$Y = \bar{B}C \quad \text{....since } (\bar{A} + A) = 1$$

– Thus A is eliminated.

Given K-map :



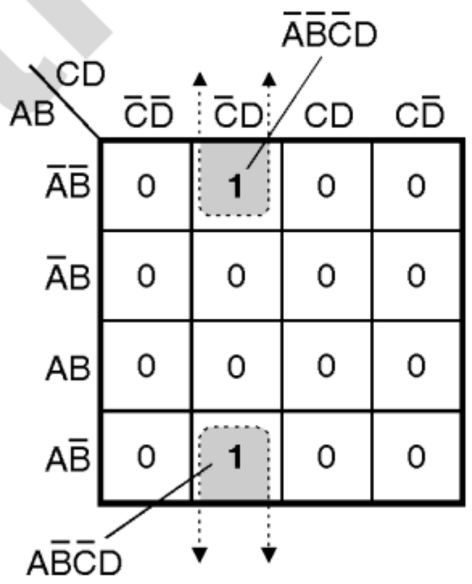
(C-228) **Fig. 1.6.3(c)**

Simplification :

$$Y = AB\bar{C} + A\bar{B}\bar{C} = A\bar{C}(B + \bar{B})$$
$$Y = A\bar{C} \quad \text{...since } (B + \bar{B}) = 1$$

– Thus B is eliminated.

Given K-map :



(C-228) **Fig. 1.6.3(d)**

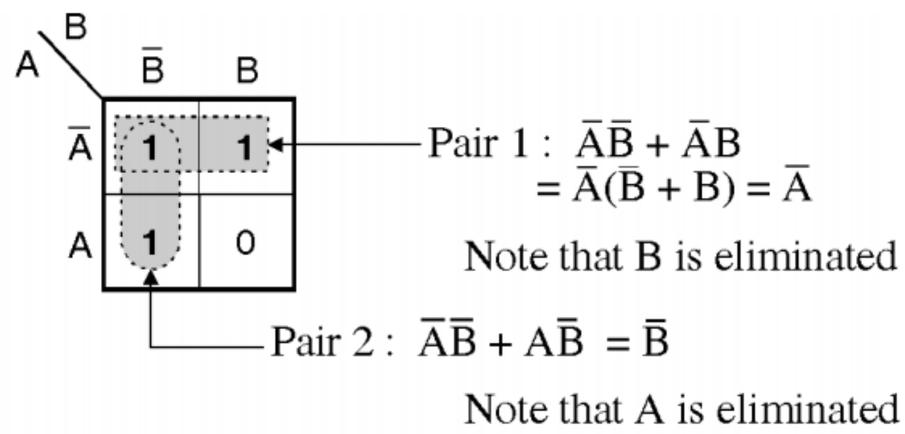
Simplification :

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

– Thus A is eliminated.

Example of overlap :

Given K-map :



(C-229) **Fig. 1.6.4(a)**

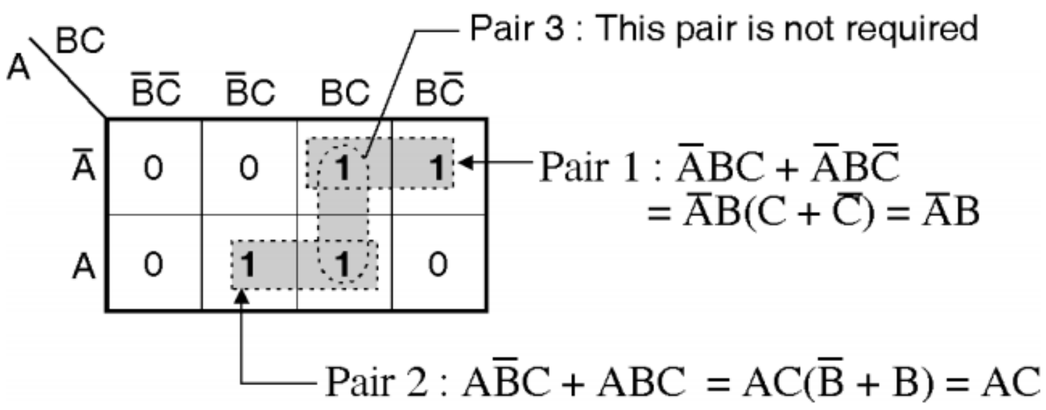
TechKnowledge Publications

∴ Final expression

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

– Note that in order to cover all the 1's, we have to overlap two pairs as shown.

Given K-map :



(C-229) Fig. 1.6.4(b) : Different types of pairs and the corresponding simplifications

– Add the two product terms to get, $Y = \bar{A}B + AC$

Note : In this K-map three pairs were possible to be formed. However only two pairs are sufficient to include all the 1's present in the K-map. Then the third pair is not required.

1.6.4 Grouping Four Adjacent Ones (Quad) :

– If we group four 1's from the adjacent cells of a K-map, then the group is called as a **Quad.**

– In such a quad two variables associated with the minterms are same and the other two are not the same.

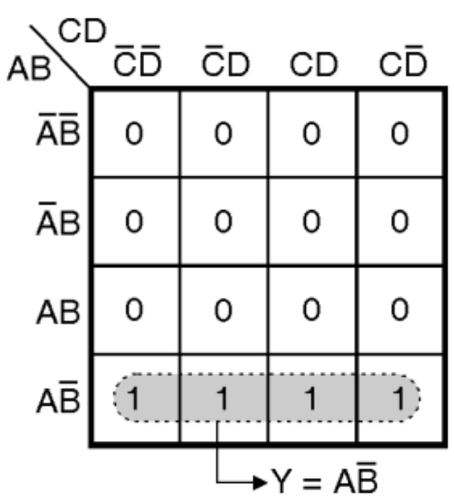
– After forming a Quad, the simplification takes place in such a way that the two variables which are not same will be eliminated.

– **Thus a quad eliminates two variables.**

– Fig. 1.6.5(a) to (f) shows various types of Quads and the corresponding mathematical simplification.

– Note that overlapping is possible in Quads.

Given K-map :



(C-230) Fig. 1.6.5(a)

Simplification :

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

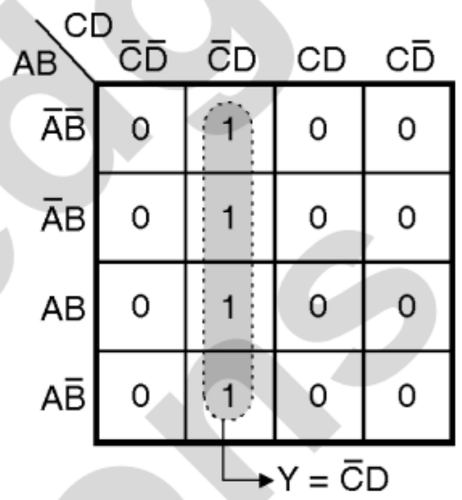
$$= A\bar{B}[\bar{C}\bar{D} + \bar{C}D + CD + C\bar{D}]$$

$$= A\bar{B}[\bar{C}(\bar{D} + D) + C(D + \bar{D})]$$

$$\therefore Y = A\bar{B}[\bar{C} + C] = A\bar{B}$$

Thus C and D are eliminated.

Given K-map :



(C-231) Fig. 1.6.5(b)

Simplification :

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

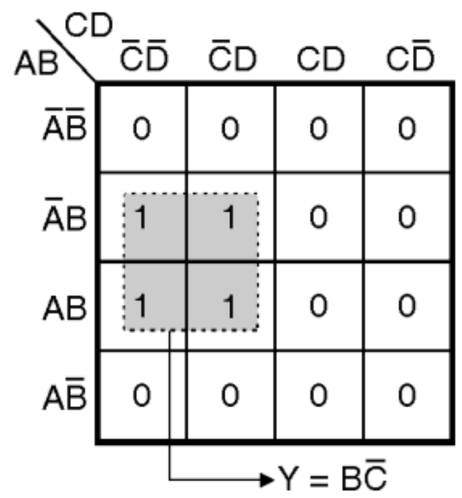
$$= \bar{C}D[\bar{A}\bar{B} + \bar{A}B + AB + A\bar{B}]$$

$$= \bar{C}D[\bar{A}(\bar{B} + B) + A(B + \bar{B})]$$

$$= \bar{C}D[\bar{A} + A] = \bar{C}D$$

Thus A and B are eliminated.

Four adjacent ones forming a square :



(C-231(a)) Fig. 1.6.5(c)

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

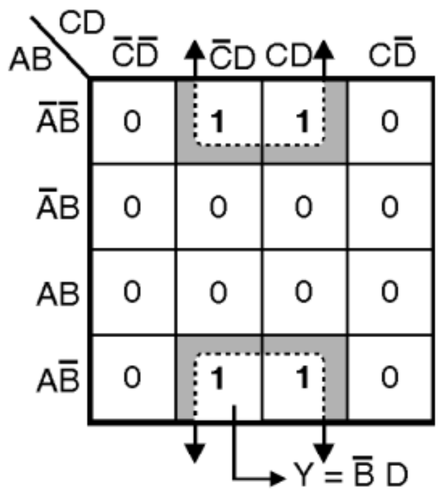
$$= B\bar{C}\bar{D}(\bar{A} + A) + B\bar{C}D(\bar{A} + A)$$

$$= B\bar{C}\bar{D} + B\bar{C}D = B\bar{C}(\bar{D} + D)$$

$\therefore Y = B\bar{C}$ Thus A and D are eliminated.

TechKnowledge Publications

Top and bottom 1's forming a Quad :



(C-231(b)) **Fig. 1.6.5(d)**

$$Y = \bar{A}\,\bar{B}\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}CD$$

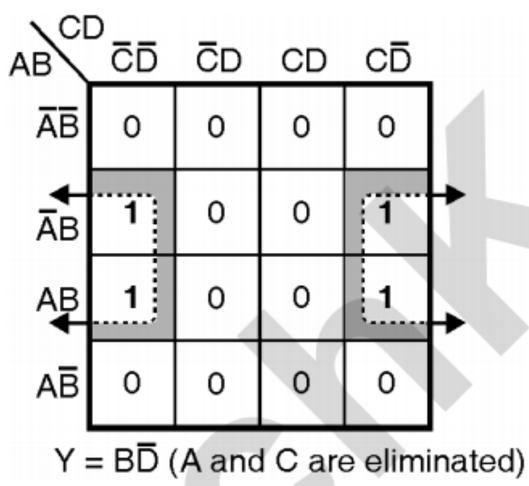
$$= \bar{A}\,\bar{B}D\,(\bar{C} + C) + A\bar{B}D\,(\bar{C} + C)$$

$$= \bar{A}\,\bar{B}D + A\,\bar{B}D$$

$$\therefore Y = \bar{B}D\,(\bar{A} + A) = \bar{B}D$$

Thus A and C are eliminated.

Leftmost and rightmost 1's forming a Quad :



(C-231(c)) **Fig. 1.6.5(e)**

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

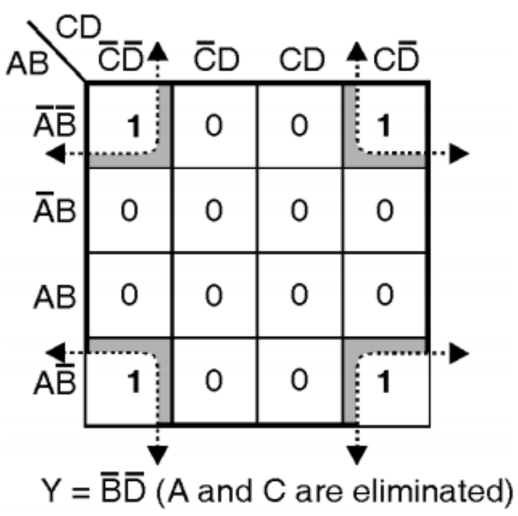
$$= B\bar{C}\,\bar{D}\,(\bar{A} + A) + B\,C\,\bar{D}\,(\bar{A} + A)$$

$$= B\,\bar{C}\,\bar{D} + B\,C\,\bar{D}$$

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

Thus A and C are eliminated.

1's corresponding to corners forming a Quad :



(C-232) **Fig. 1.6.5(f)**

$$Y = \bar{A}\,\bar{B}\bar{C}\,\bar{D} + \bar{A}\,\bar{B}CD + A\bar{B}\bar{C}\,\bar{D} + A\bar{B}CD$$

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

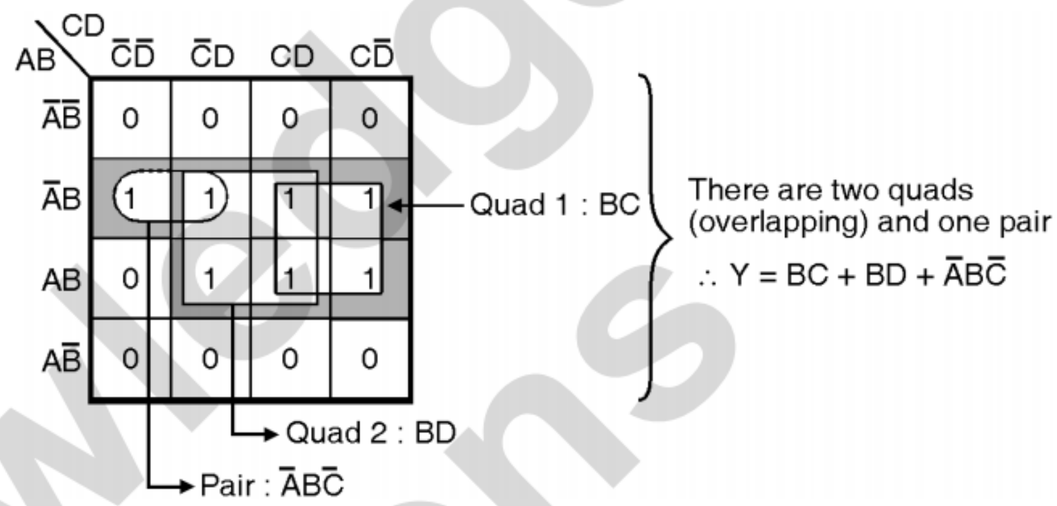
$$= \bar{B}\bar{C}\,\bar{D} + \bar{B}C\bar{D}$$

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

$$\therefore Y = \bar{B}\bar{D}$$

Thus A and C are eliminated.

Overlapping of Quads and pairs :

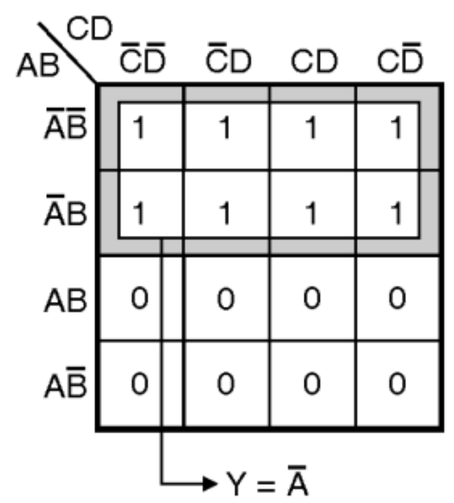


(C-233) **Fig. 1.6.5(g)**

1.6.5 Grouping Eight Adjacent Ones (Octet) :

- It is possible to form a group of eight adjacent ones. Such a group is called as octet.
- When an octet is formed three variables will change and only one variable will remain same in all the minterms.
- The three variables that change will be eliminated and the variable which does not change will appear as output.
- Thus octet eliminates three variables.
- Fig. 1.6.6(a) to (d) shows various types of octets and the corresponding output.

Given K-map :



(C-234) **Fig. 1.6.6(a)**

TechKnowledge Publications

Simplification :

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

$$+ \bar{A}\,B\bar{C}\,\bar{D} + \bar{A}\,B\bar{C}\,D + \bar{A}\,B\,C\,D + \bar{A}\,B\,C\,\bar{D}$$

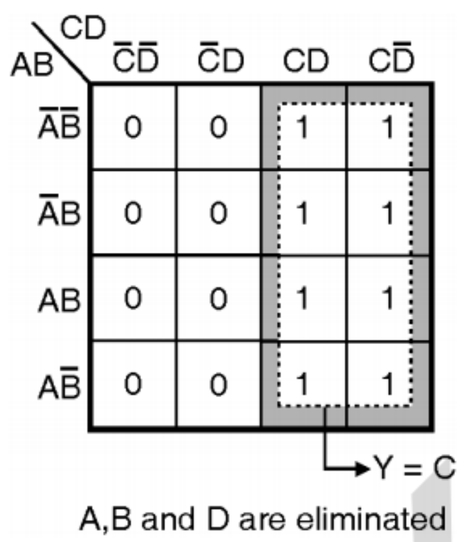
$$\therefore \quad Y = \bar{A}\,\overline{BC}\,(\bar{D} + D) + \bar{A}\,\bar{B}C\,(D + \bar{D})$$

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

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

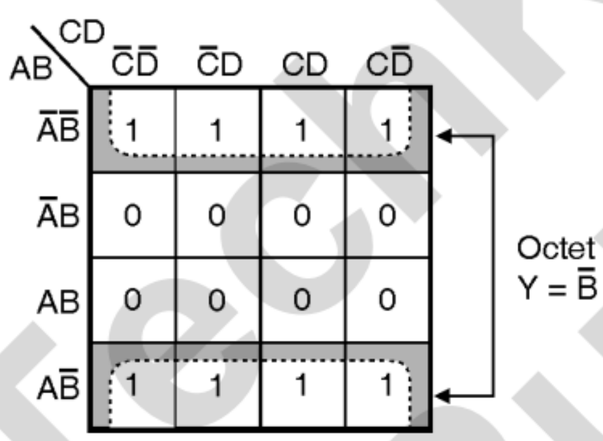
$$\therefore \quad Y = \bar{A}\,(\bar{B} + B) = \bar{A}$$

– The only variable that remains same is $\bar{A}$. So it appears as output.



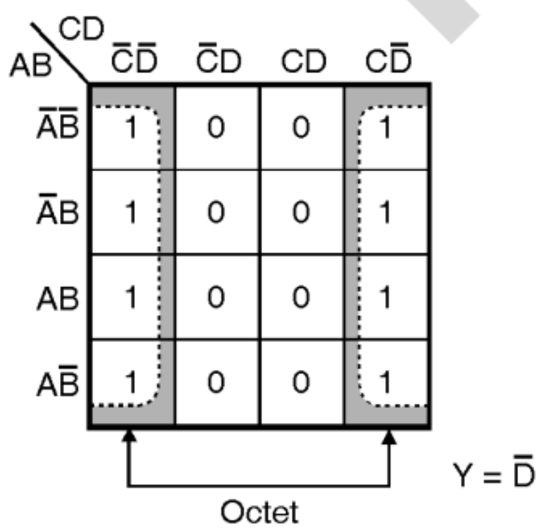
A,B and D are eliminated

(C-234(a)) (b)



A,C and D are eliminated

(C-234(b)) (c)



A,B and C are eliminated

(C-235) (d)

Fig. 1.6.6

1.6.6 Summary of Rules Followed for K-Map Simplification :

Summary :

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

1.7 Minimization of SOP Expressions (K Map Simplification) :

– The K-map can be used to simplify the logical expression to a level beyond which it cannot be further simplified.

– After such a simplification, it will require minimum number of gates with minimum number of inputs to the gates.

– Such an expression is called as a **minimized** expression.

– For minimizing the logical expression, follow the procedure given below.

Minimization procedure :

Step 1 : Prepare the K-map and place 1's according to the given truth table or logical expression. Fill the remaining cells by 0's.

Step 2 : Locate the isolated 1's i.e. the 1's which cannot be combined with any other 1. Encircle such 1's.

Step 3 : Identify the 1's which can be combined to form a pair in only one way and encircle them.

Step 4 : Identify the 1's which can form a quad in only one way and encircle them.

Step 5 : Identify the 1's which can form an octet in only one way and encircle them.

Step 6 : After identifying the pairs, quads and octets, check if any 1 is yet to be encircled. If yes then encircle them with each other or with the already encircled 1's (by means of overlapping).

– Note that the number of groups should be minimum.

– Also note that any 1 can be included any number of times without affecting the expression.

– Let us solve some examples on this to make the concept clear.

TechKnowledge Publications

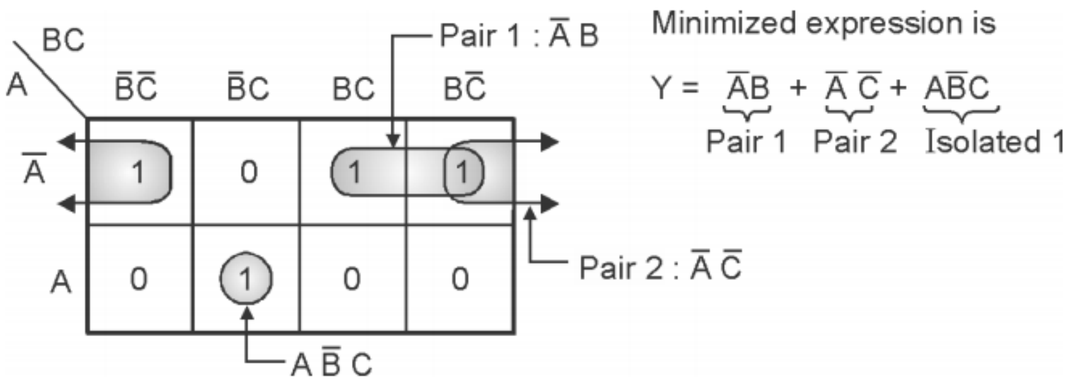
Ex. 1.7.1 : A logical expression in the standard SOP form is as follows :

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

Minimize it using the K-map technique.

Soln. : $Y = \Sigma m\,(0, 2, 3, 5)$

– The required K-map is as shown in Fig. P. 1.7.1.



(C-250) Fig. P. 1.7.1

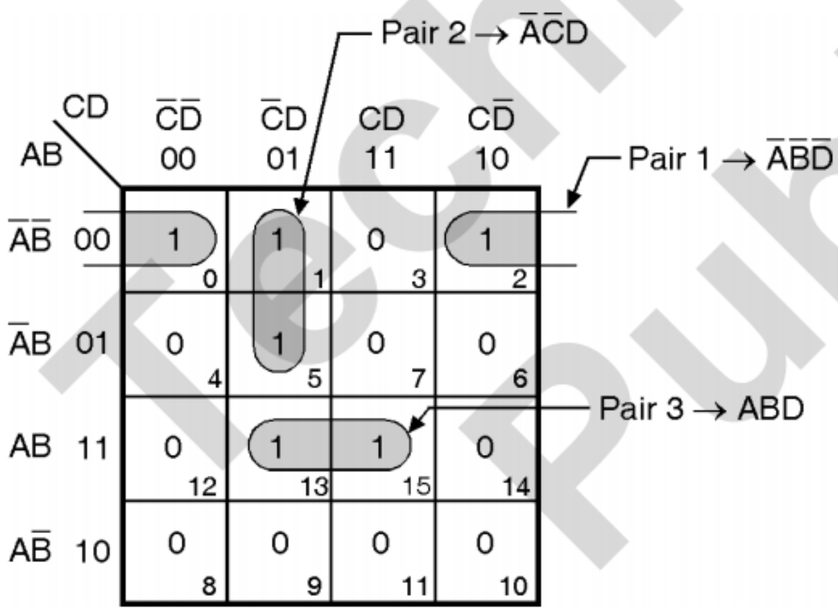
Ex. 1.7.2 : The logical expression representing a logic circuit is $Y = \Sigma m\,(0, 1, 2, 5, 13, 15)$. Draw the K-map and find the minimized logical expression.

Soln. :

– From the given expression, it is clear that the number of variables is 4.

$$Y = m_0 + m_1 + m_2 + m_5 + m_{13} + m_{15}$$

– The required K-map is as shown in Fig. P. 1.7.2.



(C-251) Fig. P. 1.7.2

– Minimized expression is,

(C-6158)

$$Y = \underbrace{\bar{A}\bar{B}\bar{D}}_{\text{Pair 1}} + \underbrace{\bar{A}\bar{C}D}_{\text{Pair 2}} + \underbrace{ABD}_{\text{Pair 3}}$$

Ex. 1.7.3 : For the logical expression given below draw the K-map and obtain the simplified logical expression. $Y = \Sigma\, m\,(1, 5, 7, 9, 11, 13, 15)$.

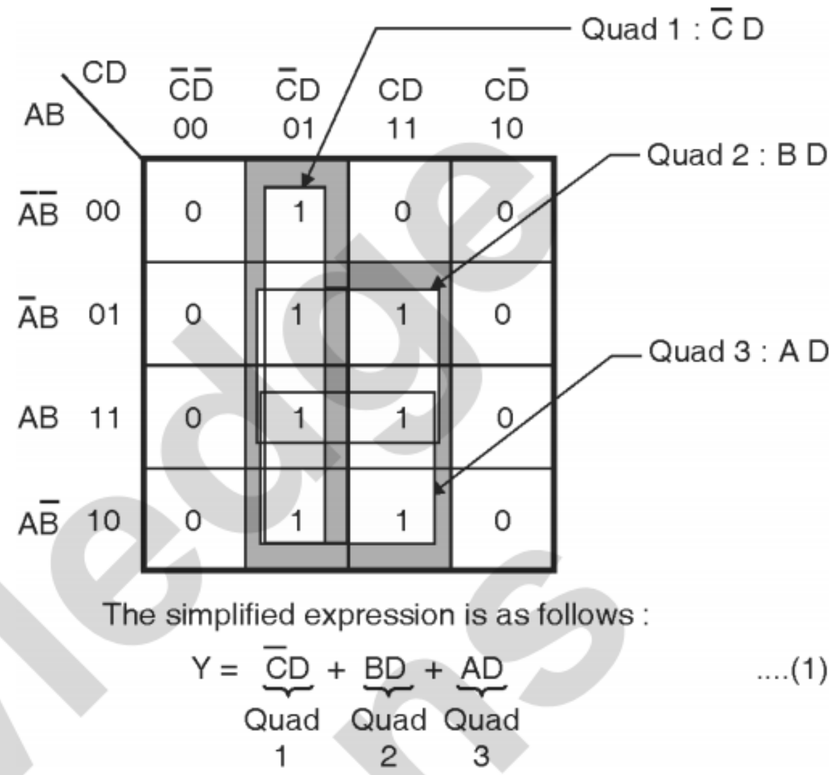
Realize the minimized expression using the basic gates.

Soln. :

– The given expression is,

$$Y = m_1 + m_5 + m_7 + m_9 + m_{11} + m_{13} + m_{15}$$

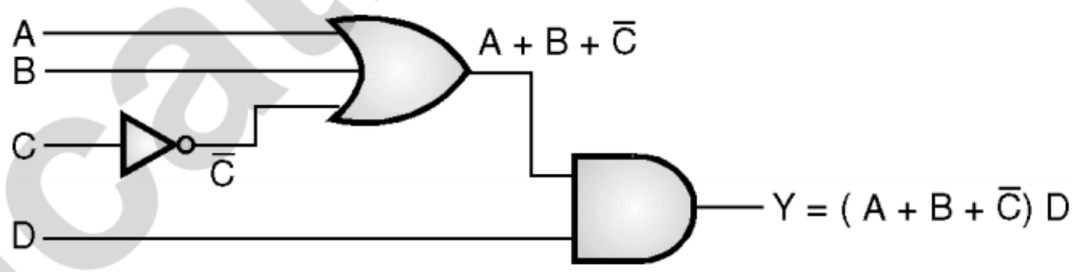
– It can be expressed on K-map as shown in Fig. P. 1.7.3(a).



(C-252) Fig. P. 1.7.3(a)

Realization :

– Equation (1) can be realized as shown in Fig. P. 1.7.3(b).



(C-253) Fig. P. 1.7.3(b) : Realization with minimum number of gates

Ex. 1.7.4 : Minimize the following Boolean expression using K-map and realize it using the basic gates. $Y = \Sigma m\,(1, 3, 5, 9, 11, 13)$

Soln. :

– The given expression can be expressed in terms of the minterms as,

$$Y = m_1 + m_3 + m_5 + m_9 + m_{11} + m_{13}$$

– The corresponding K-map is shown in Fig. P. 1.7.4(a).

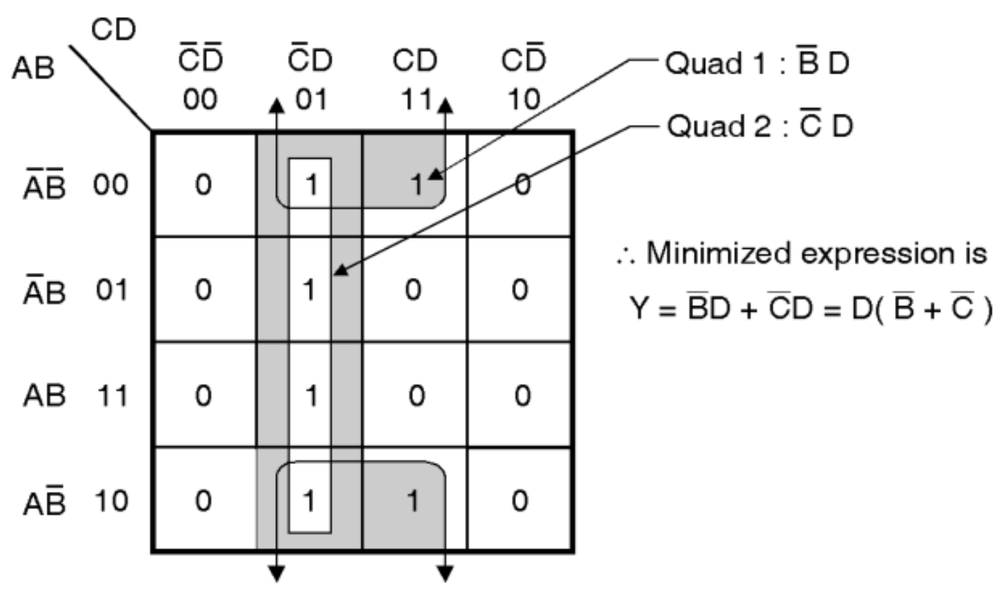
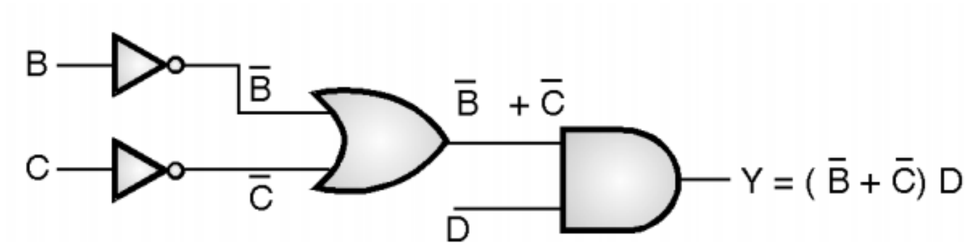


Fig. P. 1.7.4(a)

Realization :

– Fig. P. 1.7.4(b) shows realization using gates.



(C-254) **Fig. P. 1.7.4(b)**

Ex. 1.7.5 : Using K-map realize the following expression using minimum number of gates.

$Y = \Sigma\ m\ (1, 3, 4, 5, 7, 9, 11, 13, 15)$

Soln. : Solve it yourself.

Ans. : $Y = A'BC' + D$

Ex. 1.7.6 : Minimize the following expression using K-map and realize using the basic gates.

$Y = \Sigma m\ (1, 2, 9, 10, 11, 14, 15)$

Soln. : Solve it yourself.

Ans. : $Y = \overline{B}\ (C \oplus D) + AC$

Ex. 1.7.7 : Minimize the following expression and realize using basic gates.

$Y = \Sigma m\ (0, 2, 5, 6, 7, 8, 10, 13, 15)$

Soln. : Solve it yourself.

Ans. : $Y = \overline{B}\ \overline{D} + BD + \overline{A}BC$

1.7.1 Elimination of a Redundant Group :

– If all the 1's in a group are already being used in some other groups, then that group is called as a redundant group.

– A redundant group has to be eliminated, because it increases the number of gates required.

– Effect of a redundant group and its elimination is illustrated in the Ex. 1.7.8.

Ex. 1.7.8 : Minimize the following expression using the K-map.

$Y = \Sigma\ m\ (1, 5, 6, 7, 11, 12, 13, 15)$

Soln. :

1. The given expression can be expressed in the standard SOP form as follows :

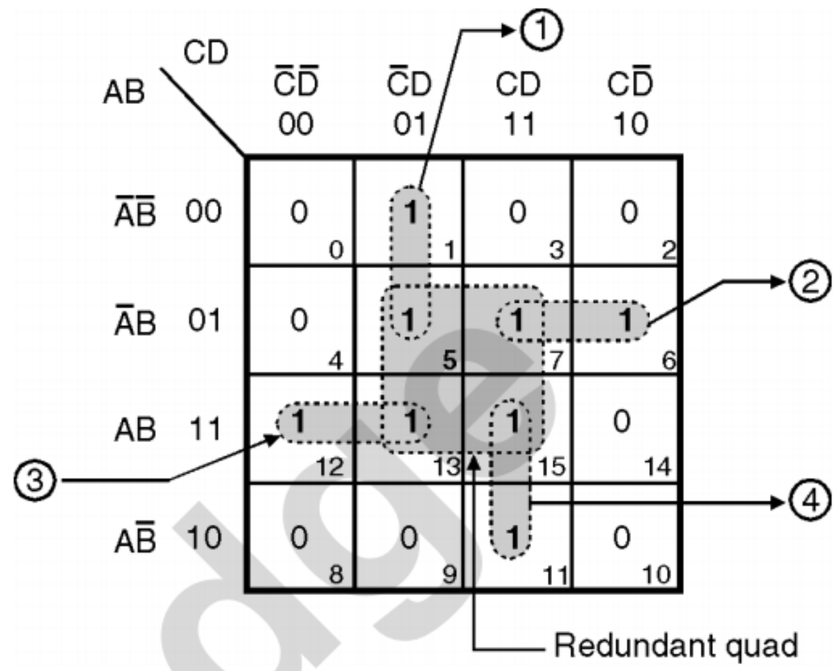
   $Y = m_1 + m_5 + m_6 + m_7 + m_{11} + m_{12} + m_{13} + m_{15}$

   where m_1, m_5 ..... m_{15} are the minterms.

2. The required K-map is shown in Fig. P. 1.7.8(a).

3. There are no isolated 1's. So encircle the separate pairs as shown in Fig. P. 1.7.8(a).

4. We would visualize a quad shown by dotted lines in Fig. P. 1.7.8(a).



(C-258) **Fig. P. 1.7.8(a)**

– The question is should we encircle this quad ?

– The answer will be obtained by writing the expression of Y without and with this quad as follows :

Expression without quad : $Y = \underset{(1)}{\overline{A}\overline{C}D} + \underset{(2)}{\overline{A}BC} + \underset{(3)}{AB\overline{C}} + \underset{(4)}{ACD}$

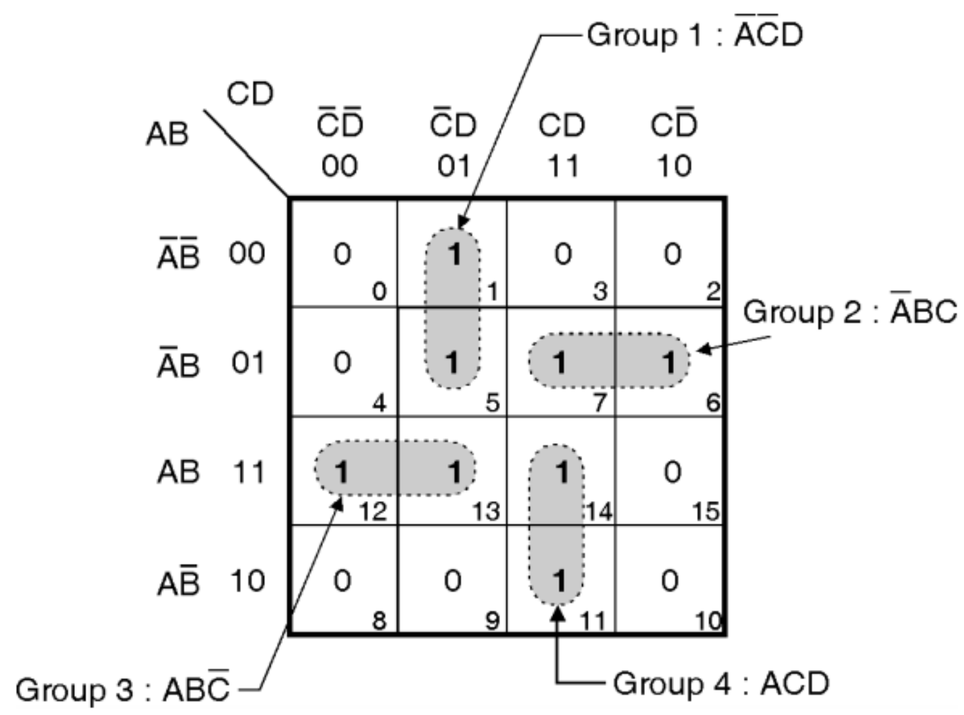
Expression with quad : $Y = \underset{(1)}{\overline{A}\overline{C}D} + \underset{(2)}{\overline{A}BC} + \underset{(3)}{AB\overline{C}} + \underset{(4)}{ACD} + \underbrace{BD}_{\text{Redundant quad}}$

(C-6159)

Conclusion :

– If we encircle the quad, then the expression for output consists of an additional term. So quad should not be encircled. It is called as the **redundant group.**

– Hence the correct K-map is shown in Fig. P. 1.7.8(b).



(C-259) **Fig. P. 1.7.8(b)**

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

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

TechKnowledge Publications

$$= D\,\underbrace{(\bar{A}\bar{C} + AC)}_{\text{EX-NOR}} + B\,\underbrace{(\bar{A}C + A\bar{C})}_{\text{EX-OR}}$$ **(C-6160)**

$$\therefore\ Y = D\,(\overline{A \oplus C}) + B\,(A \oplus C)$$

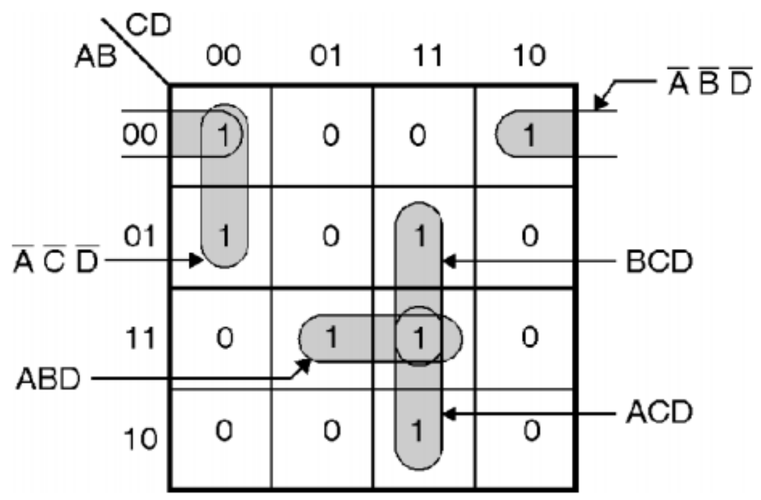
Ex. 1.7.9 : Solve the following using minimization technique :

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

Dec. 09, 10 Marks

Soln. :

Simplification using K-map :



(C-1692) Fig. P. 1.7.9(a)

$$z = \bar{A}\,\bar{B}\,\bar{D} + \bar{A}\,\bar{C}\,\bar{D} + ABD + BCD + ACD$$

Ex. 1.7.10 : Solve the following equations using corresponding minimization techniques, also draw MSI design for the minimized output equation :

$Z = f(A, B, C, D) = \Sigma\,(0, 3, 4, 9, 10, 12, 14).$

May 10, 12 Marks

Soln. : Solve it yourself.

Ans. :

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

1.7.2 Minimization of Logic Functions not Specified in Standard SOP Form :

– Till now we have seen how to use the K-map to minimize an expression which is given in standard SOP form.

– Now let us see the use of K-map for minimization when the given expression is not in the standard form. The procedure to be followed under such conditions is as follows :

– The given expression,

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

Procedure :

Step 1 : Enter 1 for minterms (i.e. $\bar{A}\,\bar{B}CD$ and $\bar{A}BC\bar{D}$) present in the given expression.

Step 2 : Enter a pair of 1's for each term with one less variable than total (i.e. for $\bar{A}\,\bar{B}\,\bar{C}$, $\bar{A}\,\bar{B}\,\bar{D}$ because these terms have one less variable)

Step 3 : Enter four adjacent 1's for each term with two less variables than total (i.e. $A\bar{C}$).

Step 4 : Repeat for the other terms in a similar way.

Step 5 : Once the K-map is obtained, the minimization procedure is same as the one discussed earlier.

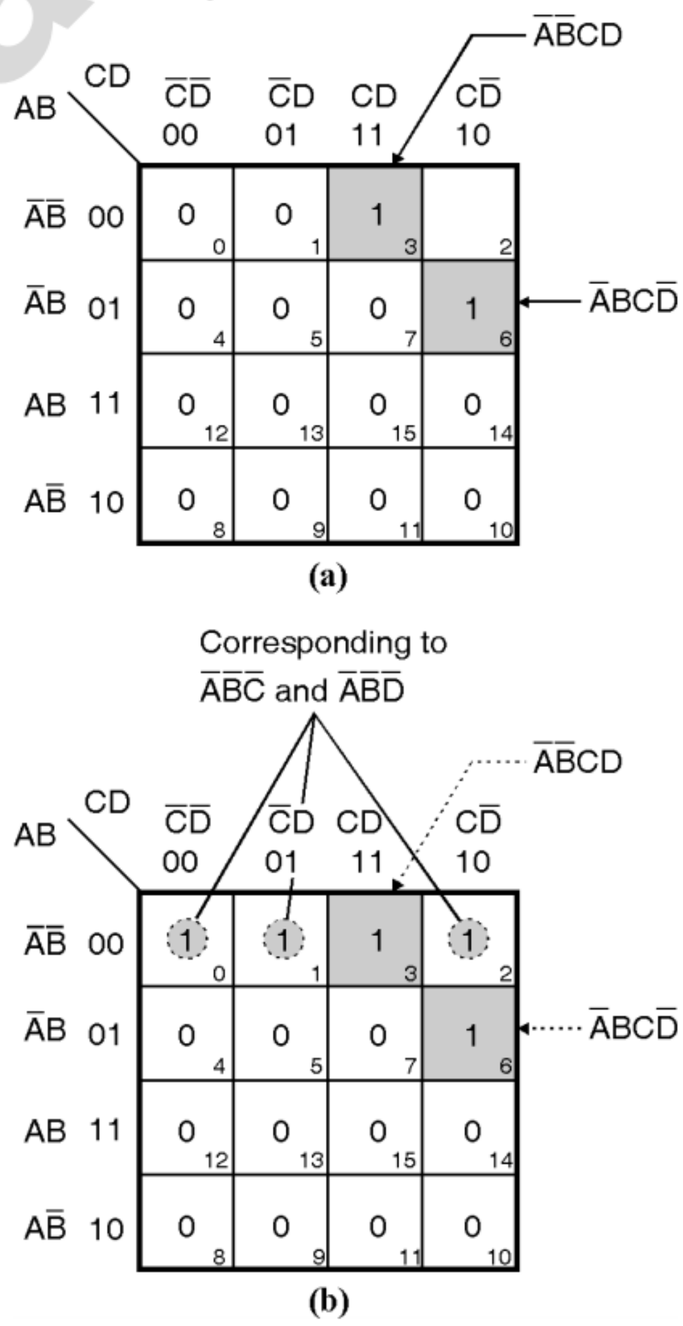
– This procedure will be clear as you solve the example given below.

Ex. 1.7.11 : Minimize the logic equation given below :

$$\therefore\ Y = \underset{(1)}{\bar{A}\bar{B}CD} + \underset{(2)}{\bar{A}BC\bar{D}} + \underset{(3)}{\bar{A}\bar{B}\bar{C}} + \underset{(4)}{\bar{A}\bar{B}\bar{D}} + \underset{(5)}{A\bar{C}} + \underset{(6)}{\bar{B}}$$ **(C-6357)**

Soln. :

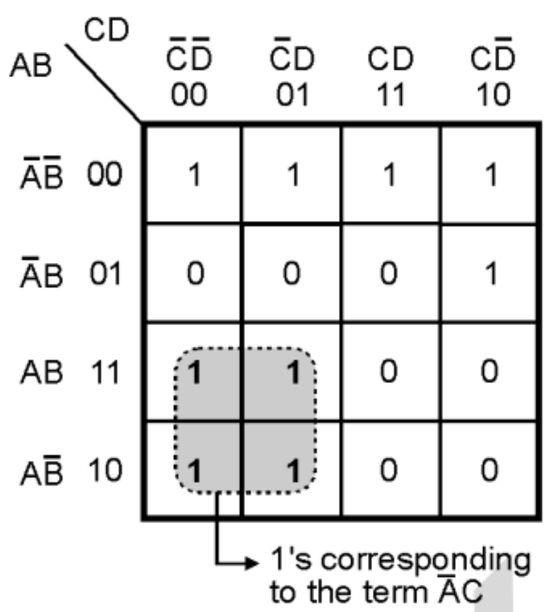
Step 1 : Enter 1 in the cell with A = 0, B = 0, C = 1, D = 1 for the first term $\bar{A}\,\bar{B}CD$ and in the cell with A = 0, B = 1, C = 1, D = 0 for the second term $\bar{A}BC\bar{D}$ as shown in Fig. P. 1.7.11(a).



(C-261) Fig. P. 1.7.11

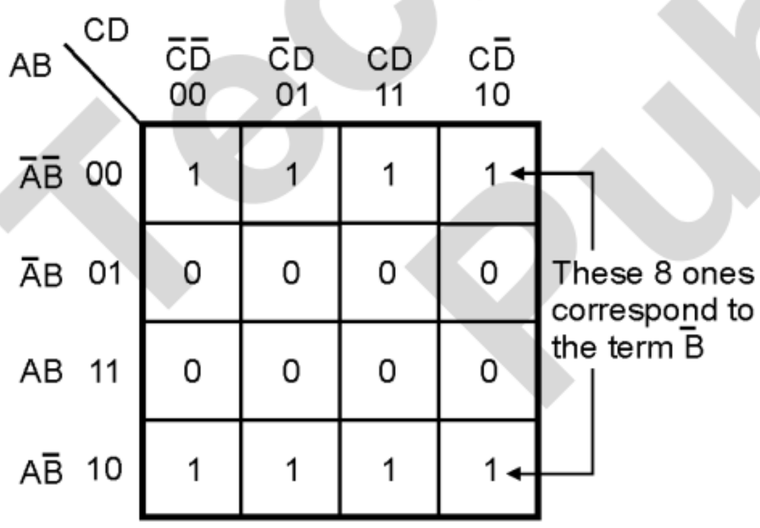
Step 2 : Consider the third term $\bar{A}\,\bar{B}\,\bar{C}$. Enter 1 in two cells which correspond to A = 0, B = 0 and C = 0, as shown in Fig. P. 1.7.11(b). Now consider the fourth term $\bar{A}\,\bar{B}\,\bar{D}$ and enter 1 in two cells which correspond to A = 0, B = 0 and D = 0 as shown in Fig. P. 1.7.11(b).

Step 3 : Now consider the fifth term $A\bar{C}$. Enter 1 in four cells corresponding to A = 1 and C = 0, as shown in Fig. P. 1.7.11(c).



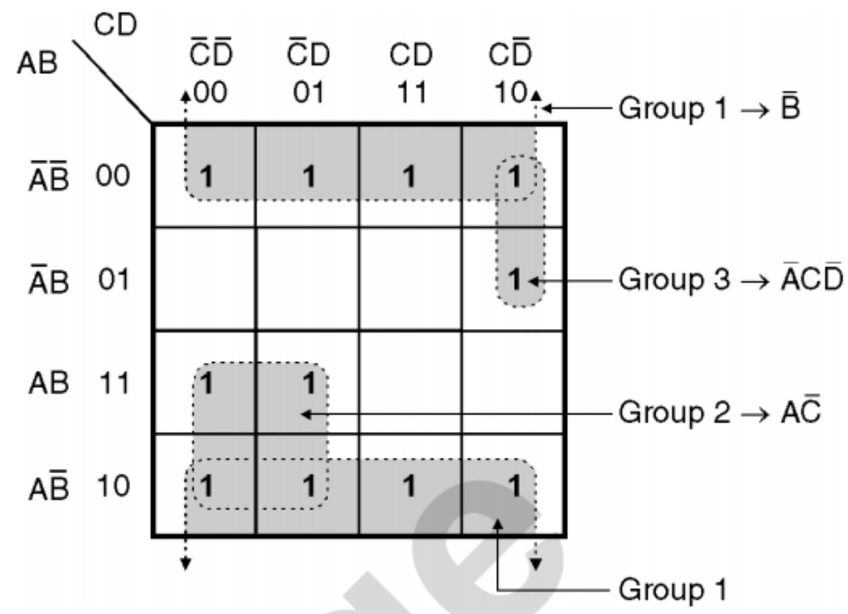
(C-261) Fig. P. 1.7.11(c)

Step 4 : Finally corresponding to the sixth term, enter eight 1's corresponding to B = 0 as shown in Fig. P. 1.7.11(d).



(C-262) Fig. P. 1.7.11(d)

– The final K-map is as shown in Fig. P. 1.7.11(e). Use the normal simplification techniques discussed earlier to simplify this K-map and get the minimized expression.



(C-263) Fig. P. 1.7.11(e) : Final K-map

– The simplified equation is,

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

1.7.3 Don't Care Conditions :

– For SOP form, we enter 1's corresponding to the combinations of input variables which produce a high output. And we enter 0's in the remaining cells of the K-map.

– For the POS form we enter 0's corresponding to the combinations of inputs which produce a low output and enter 1's in the remaining cells of the K-map.

– But it is not always true that the cells not containing 1's (in SOP) will contain 0's, because some combinations of input variable do not occur.

– Take the example of a 4 bit BCD counter. It will have valid outputs from 0000 to 1001 only.

– Also for some functions the outputs corresponding to certain combinations of input variables **do not matter.**

– That means for such input combinations it does not matter whether the value of output is 0 or 1.

– In such situation we are free to assume a 0 or 1 as output for each of such input combinations.

– These conditions are known as the **"Don't care conditions"** and in the K-map it is represented as × (cross) mark in the corresponding cell.

Important :

– The don't care condition (×) may be assumed to be 0 or 1 as per the need for simplification.

Ex. 1.7.12 : Simplify the expression given below using K-map. The don't care conditions are indicated by d ( ).

$Y = \Sigma m\,(1, 3, 7, 11, 15) + d\,(0, 2, 5)$.

Dec. 12, 5 Marks, Dec. 13, 4 Marks, Dec. 19, 6 Marks

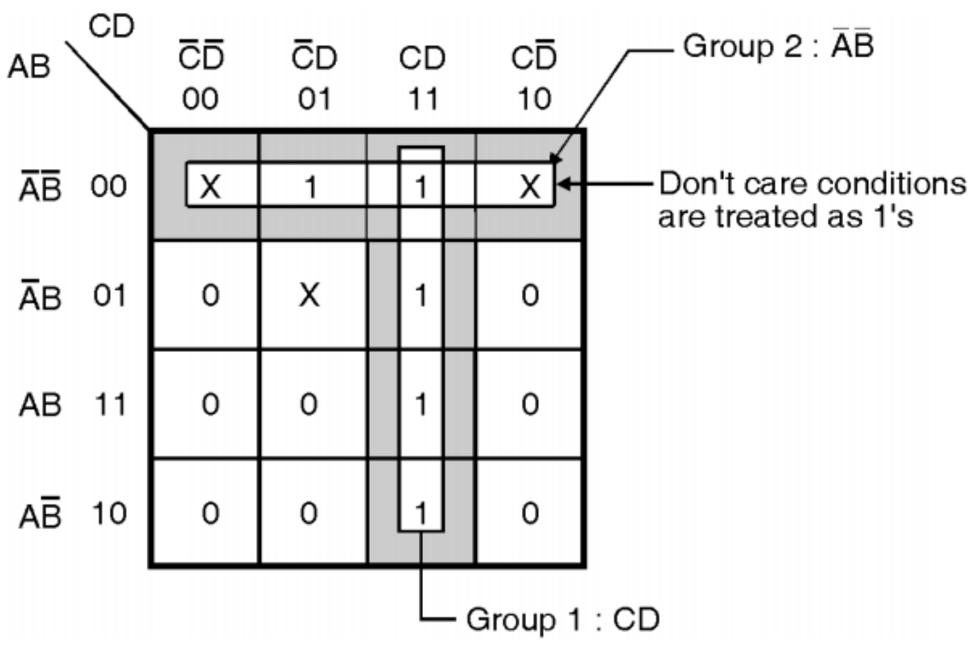
TechKnowledge Publications

Soln. : The given equation is,

$$Y = \underbrace{m_1 + m_3 + m_7 + m_{11} + m_{15}}_{\text{Regular minterms so enter 1's}} + \underbrace{d(0, 2, 5)}_{\text{Don't care conditions so enter x marks}}$$

(C-6161)

– The required K-map is shown in Fig. P. 1.7.12.



(C-264) Fig. P. 1.7.12

– Simplified expression $Y = CD + \bar{A}\,\bar{B}$

Note : Every don't care mark need not be considered while grouping.

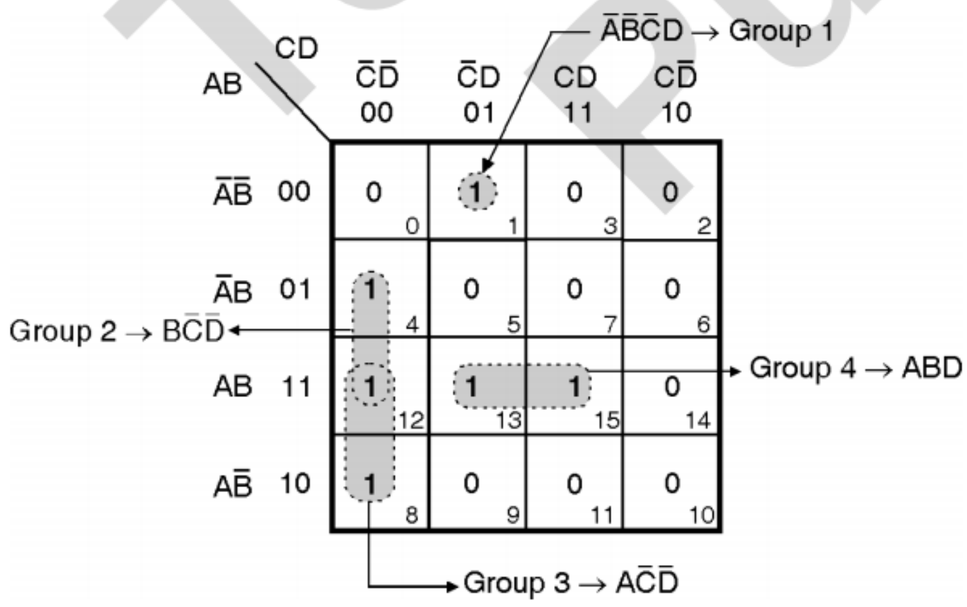
Ex. 1.7.13 : Simplify the following expression once by considering the don't care conditions and once by ignoring the don't care condition.

$Y = \Sigma m\,(1, 4, 8, 12, 13, 15) + d\,(3, 14)$

Soln. :

Part I : Simplification without don't care condition

– The K-map by neglecting the don't care condition is shown in Fig. P. 1.7.13(a).



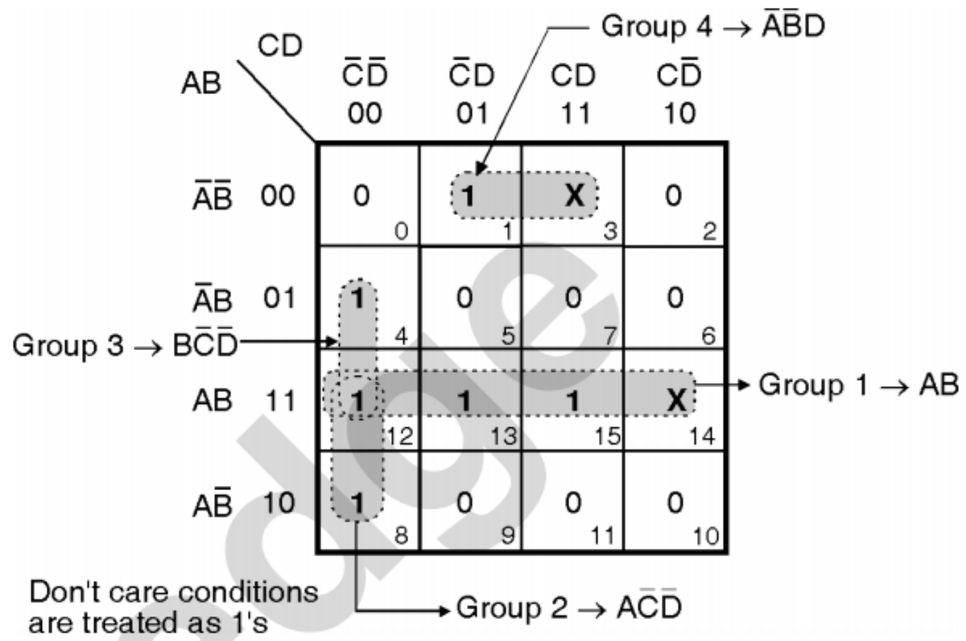
(C-265) Fig. P. 1.7.13(a) : Simplification neglecting don't care condition

$\therefore$ Simplified equation is $Y = \underset{(1)}{\bar{A}\bar{B}\bar{C}D} + \underset{(2)}{B\bar{C}\bar{D}} + \underset{(3)}{A\bar{C}\bar{D}} + \underset{(4)}{ABD}$

(C-6385)

Part II : Simplification with don't care conditions

– The K-map including the don't care conditions is shown in Fig. P. 1.7.13(b).



(C-266) Fig. P. 1.7.13(b) : Simplification using don't care conditions

$\therefore$ Simplified equation is $Y = \underset{(1)}{AB} + \underset{(2)}{A\bar{C}\bar{D}} + \underset{(3)}{B\bar{C}\bar{D}} + \underset{(4)}{\bar{A}\bar{B}D}$

(C-6355)

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

Note : The don't care condition reduces the number of gates required for implementation.

Ex. 1.7.14 : Minimize the following function using K-map and realize using logic gates :

F (A, B, C, D)

$= \Sigma\, m\,(1, 3, 7, 11, 15) + d\,(0, 2, 5)$

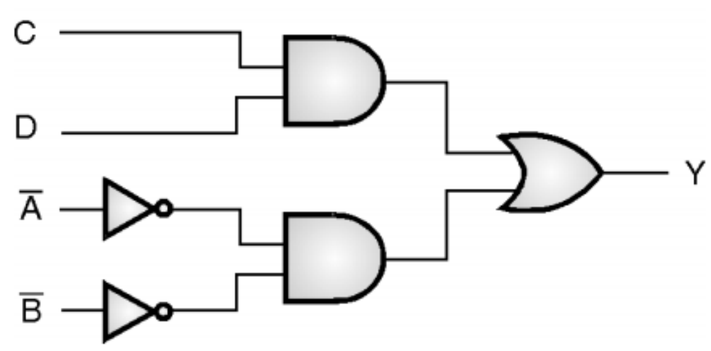
Dec. 13, 4 Marks

Soln. :

– For K-map refer Ex. 1.7.12.

– Implementation using logic gates is as shown in Fig. P. 1.7.14.

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



(C-4185) Fig. P. 1.7.14 : Implementation using logic gates

Ex. 1.7.15 : Minimize the following function using K-map and realize using logic gates :

F (A, B, C, D)

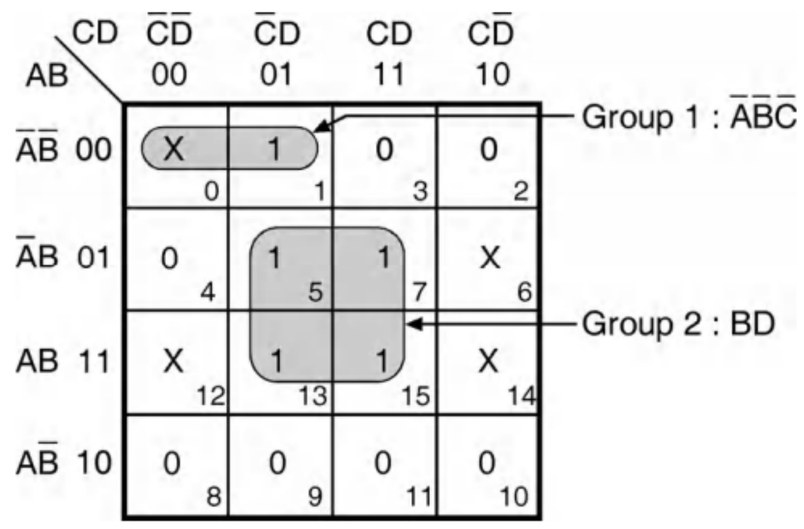
$= \Sigma\, m\,(1, 5, 7, 13, 15) + d(0, 6, 12, 14)$

May 15, May 18, 4 Marks

TechKnowledge Publications

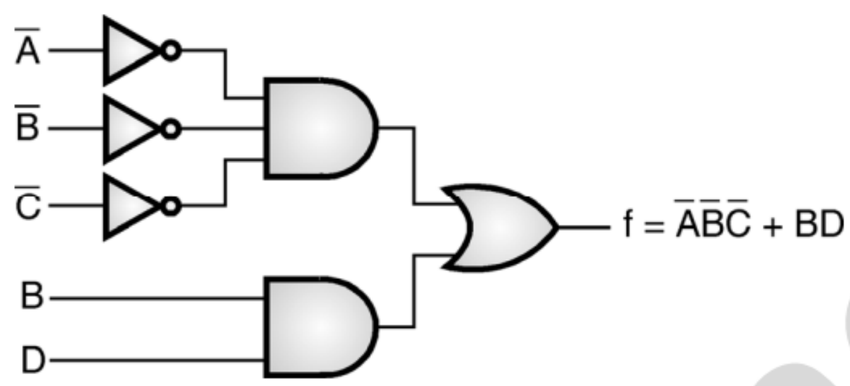
Soln. :

– Simplification using K-map is as shown in Fig. P. 1.7.15.



(C-5101) **Fig. P. 1.7.15**

– Implementation using logic gates is as shown in Fig. P. 1.7.15(a).



(C-5102) **Fig. P. 1.7.15(a)**

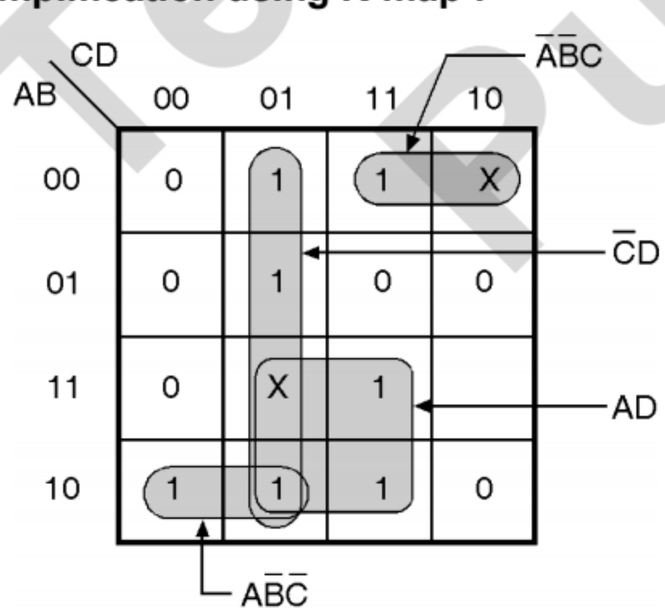
$\therefore$ $f(A, B, C, D) = \overline{A}\overline{B}\,\overline{C} + BD$

Ex. 1.7.16 : Minimize the following logic function and realize using NAND gates :

$F(A, B, C, D) = \sum m\,(1, 3, 5, 8, 9, 11, 15) + d\,(2, 13)$ **Dec. 16, 4 Marks**

Soln. :

Step 1 : Simplification using K-map :



$\therefore$ $F(A,B,C,D) = \overline{A}\overline{B}C + \overline{C}D + AD + A\overline{B}\overline{C}$

(C-5698) **Fig. P. 1.7.16(a) : K-map**

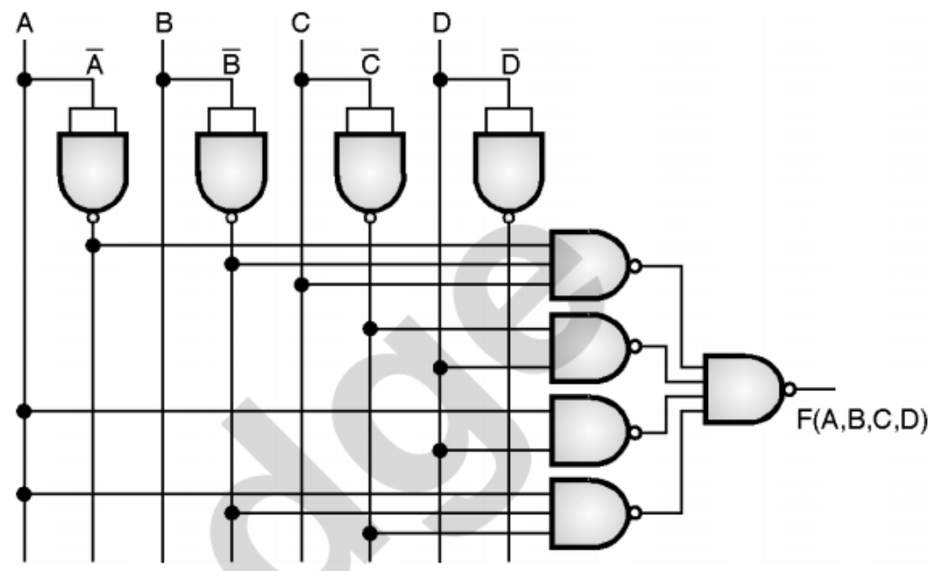
Step 2 : Implementation using NAND gates :

– Taking double inversion of R.H.S.,

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

$$= \overline{\overline{\overline{A}\overline{B}C} \cdot \overline{\overline{C}D} \cdot \overline{AD} \cdot \overline{A\overline{B}\overline{C}}}$$ ...(Using De-Morgan's theorem)

– Implementation using NAND gates is as shown in Fig. P. 1.7.16(b).



(C-5699) **Fig. P. 1.7.16(b) : Implementation using NAND gates**

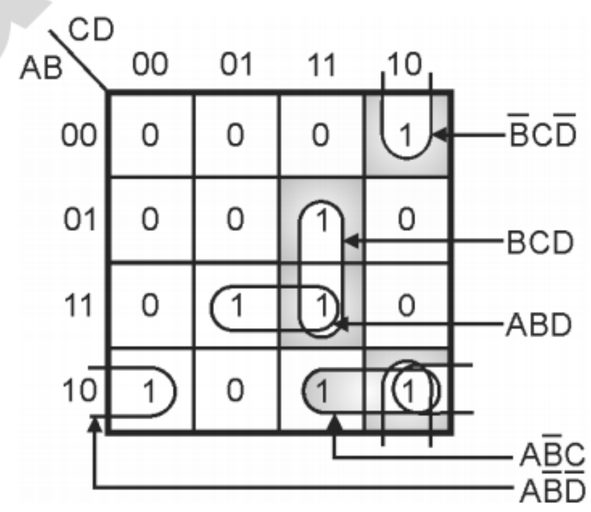
Ex. 1.7.17 : Solve the following equations using corresponding minimization techniques :

1. $z = f(A, B, C, D) = \sum m\,(2, 7, 8, 10, 11, 13, 15)$
2. $z = f(A, B, C, D) = \sum m\,(0, 3, 4, 9, 10, 12, 14)$.

May 19, 6 Marks

Soln. :

1. $z = f(A, B, C, D) = \sum m\,(2, 7, 8, 10, 11, 13, 15)$

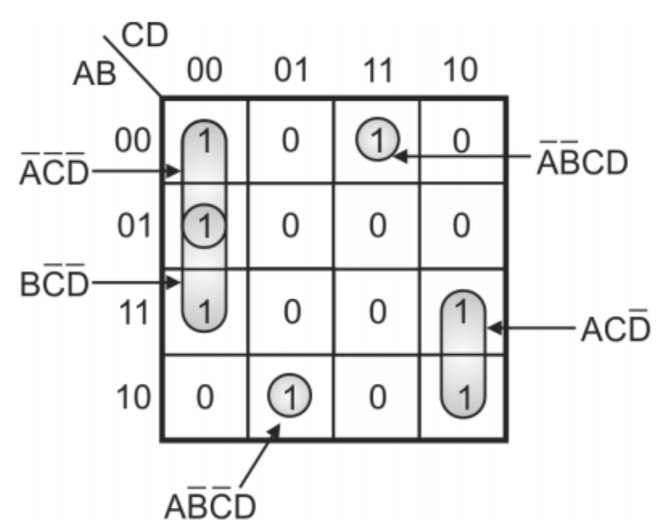


(C-7929) **Fig. 1.7.17(a) : K-map**

$\therefore$ $z = \overline{B}C\overline{D} + BCD + ABD + A\overline{B}C + A\overline{B}\overline{D}$

2. $z = f(A, B, C, D) = \sum m\,(0, 3, 4, 9, 10, 12, 14)$.

Simplification using K-map :



(C-1696) **Fig. 1.7.17(b)**

$\therefore z = \overline{A}\,\overline{C}\,\overline{D} + B\,\overline{C}\,\overline{D} + A\,C\,\overline{D} + \overline{A}\,\overline{B}\,C\,D + A\,\overline{B}\,\overline{C}\,D$

TechKnowledge Publications

Ex. 1.7.18 : Solve the following reduction using K-map, also draw MSI circuit for the output :

1. $Z = f(A, B, C, D)$
   $= \Sigma (1, 2, 7, 8, 10, 12, 15) + d (0, 5, 6)$
2. $Z = f(A, B, C, D)$
   $= \Sigma (1, 3, 4, 6, 8, 11, 15) + d (0, 5, 7)$

May 11, 12 Marks

Soln. : Solve it yourself.

Ans. :

1. $Z = f(A, B, C, D) = \bar{A}\,\bar{C}\,D + A\,\bar{C}\,\bar{D} + \bar{B}\,\bar{D} + BCD$
2. $Z = f(A, B, C, D) = \bar{A}\,D + \bar{A}\,B + CD + \bar{B}\,\bar{C}\,\bar{D}$

Ex. 1.7.19 : Solve the following equation using corresponding minimization technique. Draw the diagram for the output :

$Z = f(A, B, C, D)$
$= \Sigma m (2, 4, 6, 11, 12, 14) + d (3, 10).$

Dec. 11, 6 Marks

Soln. : Solve it yourself.

Ans. :

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

Ex. 1.7.20 : Solve the following equation using K map minimization technique. Draw the MSI design for the minimized output :

$Z = f(A, B, C, D)$
$= \Sigma m (1, 3, 6, 7, 12, 13) + d (0, 2, 8, 9)$

May 12, 6 Marks

Soln. : Solve it yourself.

Ans. :

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

1.8 Product of Sum (POS) Simplification :

1.8.1 K-map Representation of POS Form :

- The POS form equations consist of maxterms.
- In the K-map corresponding to POS form equation we have to enter a 0 corresponding to each maxterm.
- The K-maps corresponding to the POS form are shown in Fig. 1.8.1.

A \ B	B 0	$\bar{B}$ 1
A 0	$A + B$	$A + \bar{B}$
$\bar{A}$ 1	$\bar{A} + B$	$\bar{A} + \bar{B}$

OR

B \ A	A 0	$\bar{A}$ 1
B 0	$A + B$	$\bar{A} + B$
$\bar{B}$ 1	$A + \bar{B}$	$\bar{A} + \bar{B}$

(C-286) Fig. 1.8.1(a) : Two variable K-map for POS form

A \ BC	00	01	11	10
0	$A + B + C$	$A + B + \bar{C}$	$A + \bar{B} + \bar{C}$	$A + \bar{B} + C$
1	$\bar{A} + B + C$	$\bar{A} + B + \bar{C}$	$\bar{A} + \bar{B} + \bar{C}$	$\bar{A} + \bar{B} + C$

OR

C \ AB	00	01	11	10
0	$A + B + C$	$A + \bar{B} + C$	$\bar{A} + \bar{B} + C$	$\bar{A} + B + C$
1	$A + B + \bar{C}$	$A + \bar{B} + \bar{C}$	$\bar{A} + \bar{B} + \bar{C}$	$\bar{A} + B + \bar{C}$

(C-286) (b) Three variable K-map for POS form

AB \ CD	00	01	11	10
00	$A + B + C + D$	$A + B + C + \bar{D}$	$A + B + \bar{C} + \bar{D}$	$A + B + \bar{C} + D$
01	$A + \bar{B} + C + D$	$A + \bar{B} + C + \bar{D}$	$A + \bar{B} + \bar{C} + \bar{D}$	$A + \bar{B} + \bar{C} + D$
11	$\bar{A} + \bar{B} + C + D$	$\bar{A} + \bar{B} + C + \bar{D}$	$\bar{A} + \bar{B} + \bar{C} + \bar{D}$	$\bar{A} + \bar{B} + \bar{C} + D$
10	$\bar{A} + B + C + D$	$\bar{A} + B + C + \bar{D}$	$\bar{A} + B + \bar{C} + \bar{D}$	$\bar{A} + B + \bar{C} + D$

(c) Four variable K-map for POS form

(C-286) Fig. 1.8.1

- The entries in the K-map can be shown in terms of the maxterms M_0, M_1 .... etc. as shown in Fig. 1.8.2.

A \ B	0	1
0	M_0	M_1
1	M_2	M_3

(a)

A \ BC	00	01	11	10
0	M_0	M_1	M_3	M_2
1	M_4	M_5	M_7	M_6

(b)

Fig. 1.8.2(Contd...)

TechKnowledge Publications

AB \ CD	00	01	11	10
00	M_0	M_1	M_3	M_2
01	M_4	M_5	M_7	M_6
11	M_{12}	M_{13}	M_{15}	M_{14}
10	M_8	M_9	M_{11}	M_{10}

(c)

(C-287) Fig. 1.8.2 : K-map in terms of maxterms

1.8.2 Representation of Standard POS Form on K-map :

- Logical expressions in the standard POS form can be represented on K-map by entering 0's in the cells of K-map corresponding to each maxterm present in the given equation.
- The remaining cells are filled with 1's.
- This technique is illustrated in Ex. 1.8.1.

Ex. 1.8.1 : Represent the following standard POS expression on Karnaugh map.

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

Soln. :

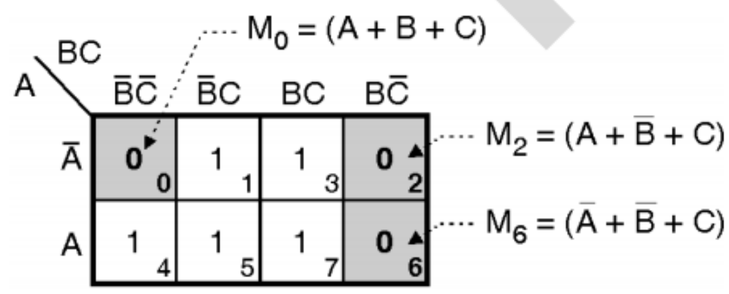
- Each term in the given logical equation is a maxterm.
- Enter a 0 corresponding to each maxterm as shown in Fig. P. 1.8.1.
- The given expression has three maxterms as follows,

$$(A + B + C) = M_0,$$

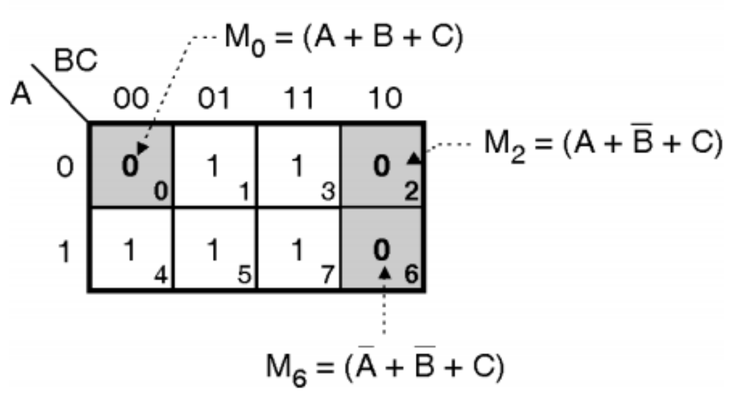
$$(A + \bar{B} + C) = M_2,$$

$$(\bar{A} + \bar{B} + C) = M_6$$

- Hence we have to write the structure of 3 variable K-map as usual and enter 0's at M_0, M_2 and M_6 as shown in Fig. P. 1.8.1.



OR



(C-288) Fig. P. 1.8.1 : Representation of standard POS on K-map

Ex. 1.8.2 : Represent the following standard POS equation on the Karnaugh map.

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

Soln. :

- The given expression contains four maxterms as follows :

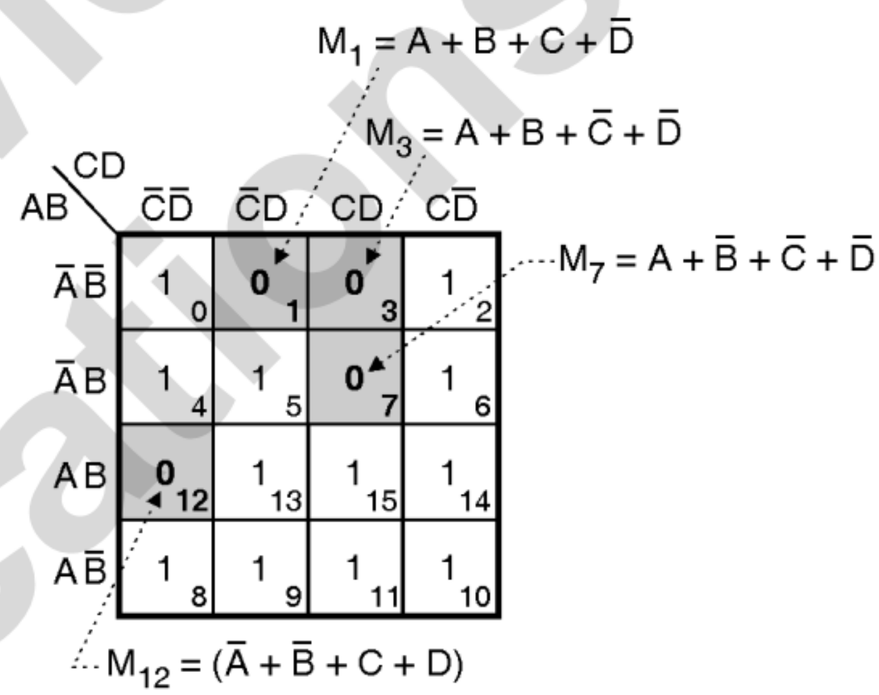
$$(A + B + C + \bar{D}) = M_1$$

$$(A + \bar{B} + \bar{C} + \bar{D}) = M_7$$

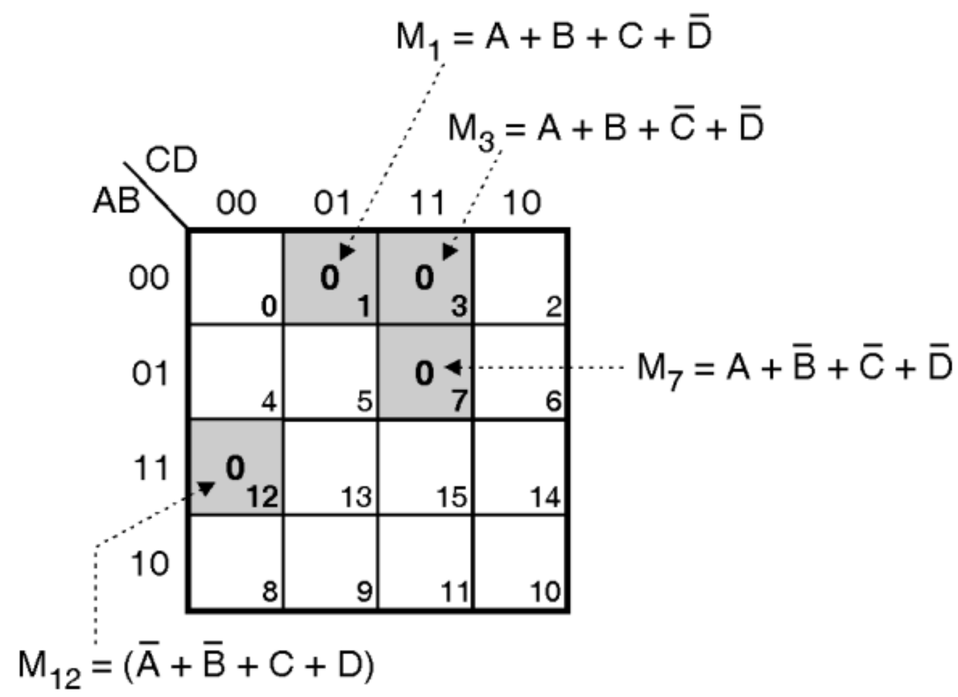
$$(A + B + \bar{C} + \bar{D}) = M_3$$

$$(\bar{A} + \bar{B} + C + D) = M_{12}$$

- Enter a 0 corresponding to each maxterm as shown in Fig. P. 1.8.2.



OR



(C-289) Fig. P. 1.8.2 : Representation of standard POS on K-map

1.8.3 Simplification of Standard POS Form using K-map :

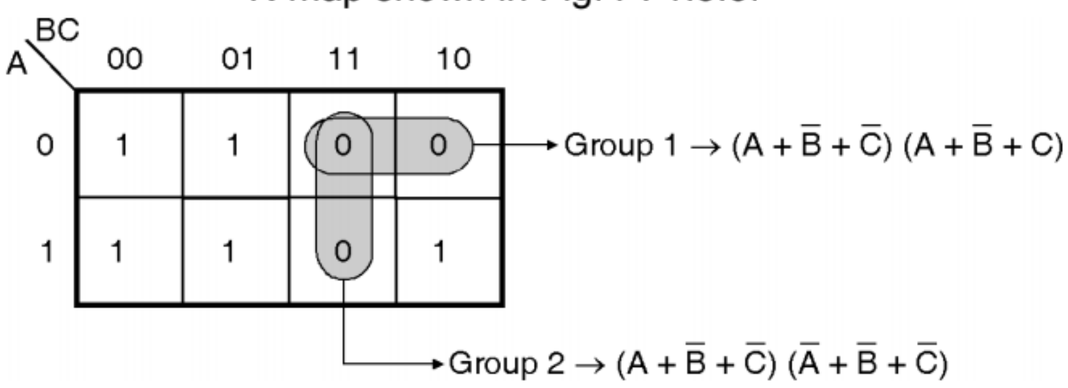
Minimization procedure :

1. The given POS expression consists of maxterms.
2. Corresponding to every maxterm enter a 0 in the K-map.

TechKnowledge Publications

3. Enter 1's in the remaining cells of K-map.
4. Encircle/group 0's instead of 1's for carrying out the simplification.
5. Rules of simplification are exactly same as those used for the SOP form.

Ex. 1.8.3 : Find the expression in the POS form for the K-map shown in Fig. P. 1.8.3.



(C-290) Fig. P. 1.8.3 : Given K-map

Soln. :

Simplification :

Group 1 → $(A + \bar{B} + \bar{C})(A + \bar{B} + C)$

$= AA + A\bar{B} + AC + A\bar{B} + \bar{B}\,\bar{B} + \bar{B}C + A\bar{C} + \bar{B}\,\bar{C} + C\bar{C}$

But $AA = A$, $\bar{B}\bar{B} = \bar{B}$, $C\bar{C} = 0$, $A\bar{B} + A\bar{B} = A\bar{B}$

$\therefore$ Group 1 $= A + A\bar{B} + AC + \bar{B} + \bar{B}C + A\bar{C} + \bar{B}\,\bar{C}$

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

$= A + A + \underbrace{\bar{B} + \bar{B}\bar{C}}$ **(C-6358)**

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

Group 1 $= A + \bar{B}$

Conclusion :

– When a pair of 0s is formed, the variable which changes will get eliminated e.g. C changes for the pair of 0's in group 1. Hence it is eliminated.

$\therefore$ Group 2 → $\bar{B} + \bar{C}$

$\therefore$ Final minimized equation is as follows :

$Y = \underset{(1)}{(A + \bar{B})}\ \underset{(2)}{(\bar{B} + \bar{C})}$

Group 2 → $(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + \bar{C})$

Remain unchanged

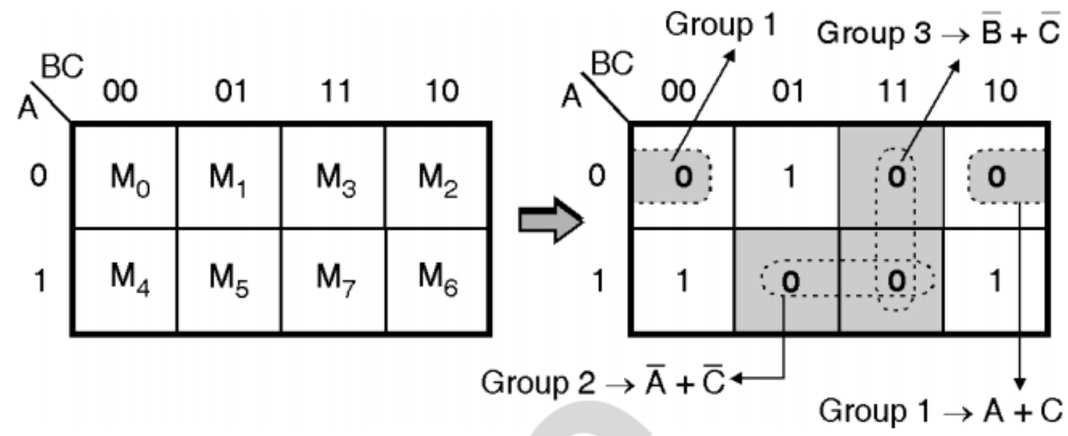
Variable that changes its value hence it gets eliminated

(C-291)

Ex. 1.8.4 : Minimize the following standard POS expression using K-map :

$Y = \Pi M(0, 2, 3, 5, 7)$

Soln. :

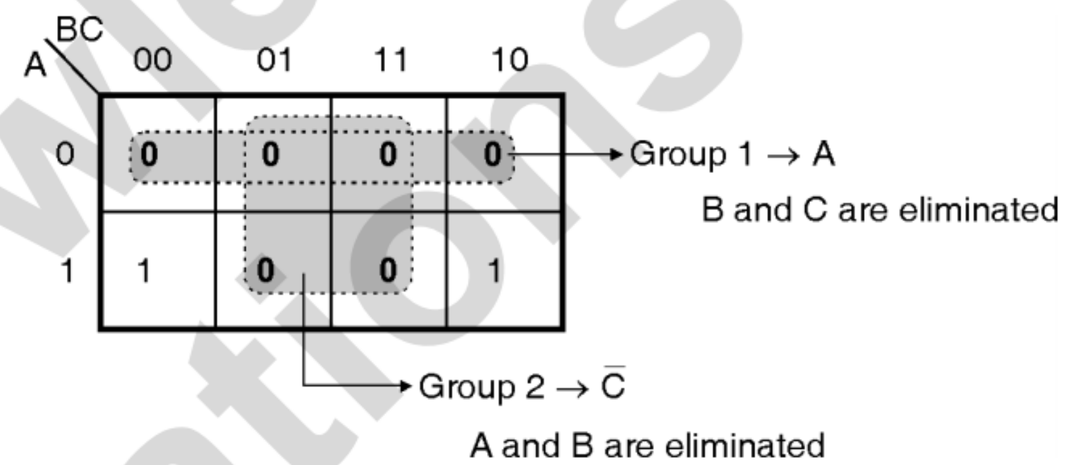


(C-292) Fig. P. 1.8.4

$\therefore$ Minimized expression is as follows :

$Y = (A + C)(\bar{A} + \bar{C})(\bar{B} + \bar{C})$ **...Ans.**

Ex. 1.8.5 : Find the expression for the output in the POS form for the K-map shown in Fig. P. 1.8.5.



(C-293) Fig. P. 1.8.5 : Given K-map

Soln. :

– Minimized expression is as follows :

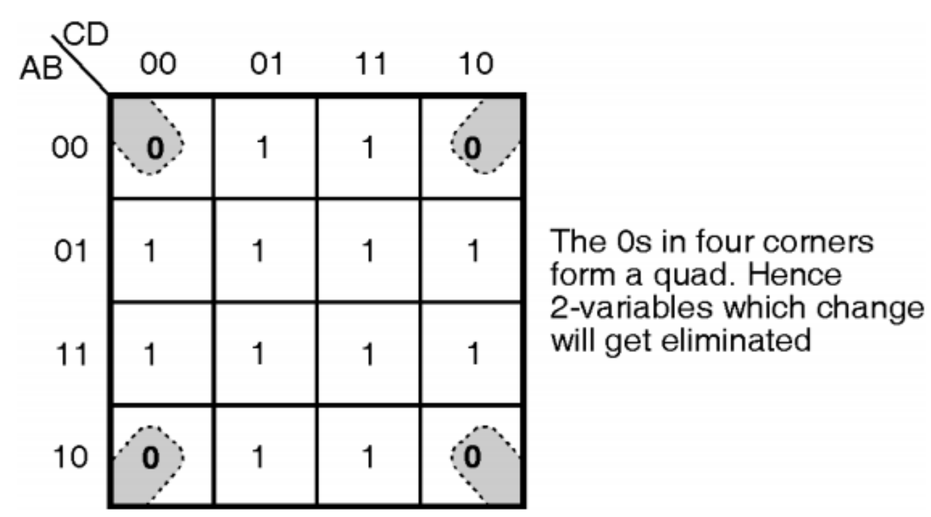
$Y = A \cdot \bar{C}$

Group 2

Group 1

(C-293(a))

Ex. 1.8.6 : Write the expression for output in the POS form for the K-map shown in Fig. P. 1.8.6.

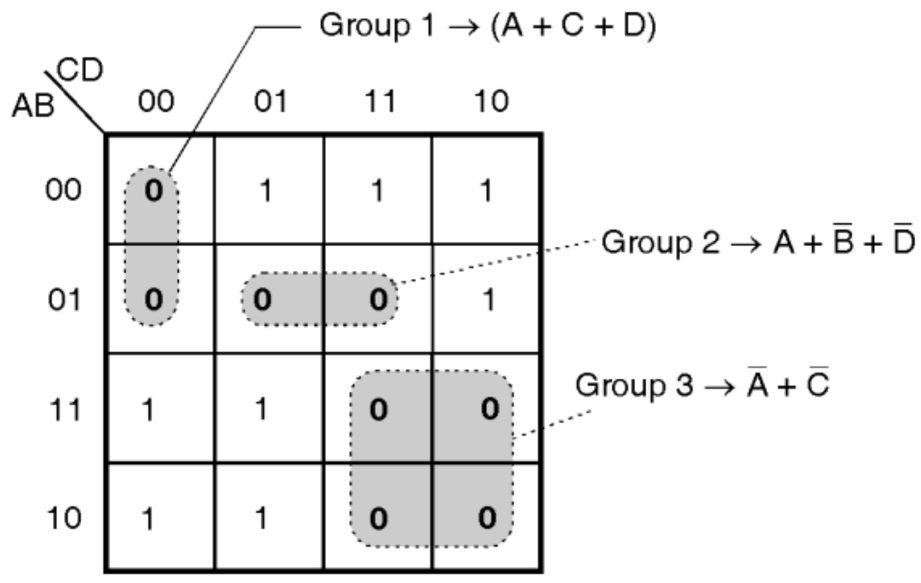


(C-295) Fig. P. 1.8.6

Soln. :

Expression for output Y = B + D

Ex. 1.8.7 : For the K-map shown in Fig. P. 1.8.7 write the expression for output in the POS form.



(C-296) Fig. P. 1.8.7

Soln. :

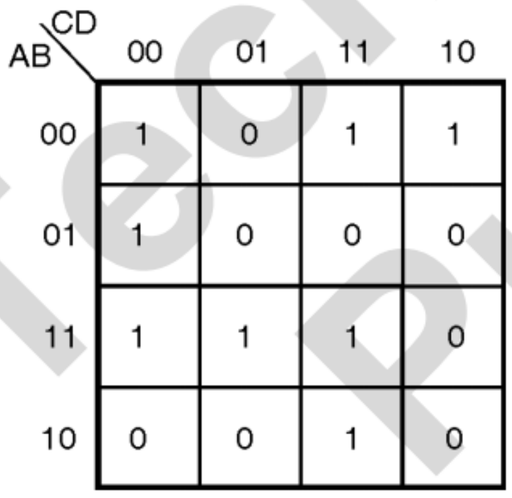
– The expression for output is,

$$Y = \underset{(1)}{(A + C + D)}\ \underset{(2)}{(A + \bar{B} + \bar{D})}\ \underset{(3)}{(\bar{A} + \bar{C})}$$

(C-6361)

Note : Sometimes there are more than one correct ways of encircling the 0's, and the answers obtained as a result of all of them are correct. This is illustrated in Ex. 1.8.8.

Ex. 1.8.8 : Group the 0s given in the K-map of the Fig. P. 1.8.8(a) in different ways to obtain the expression for output in the POS form.



(C-297) Fig. P. 1.8.8(a) : Given K-map

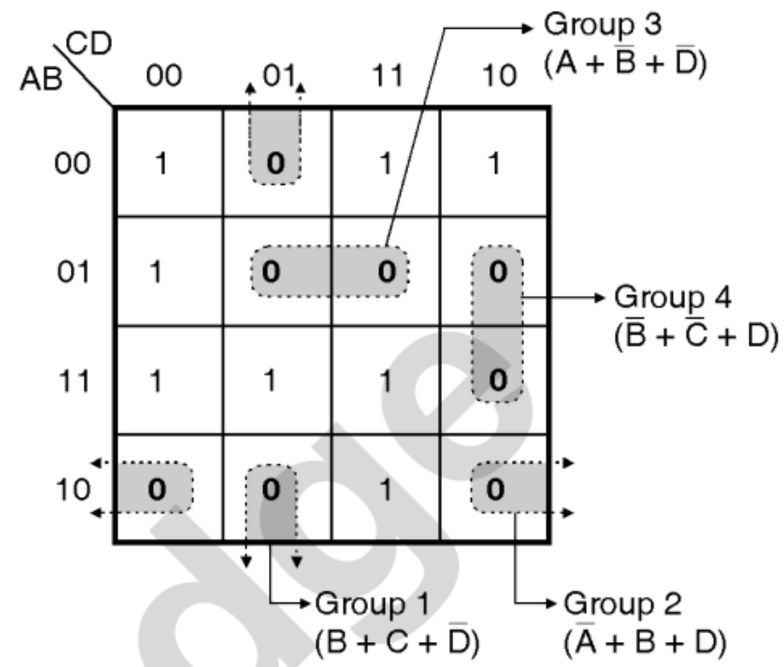
Soln. :

1. Figs. P. 1.8.8(b) and (c) shows two different but valid ways of grouping 0's.
2. The corresponding equations for the output in the POS form are obtained after that.
3. Referring to the grouping shown in Fig. P. 1.8.8(b) we get,

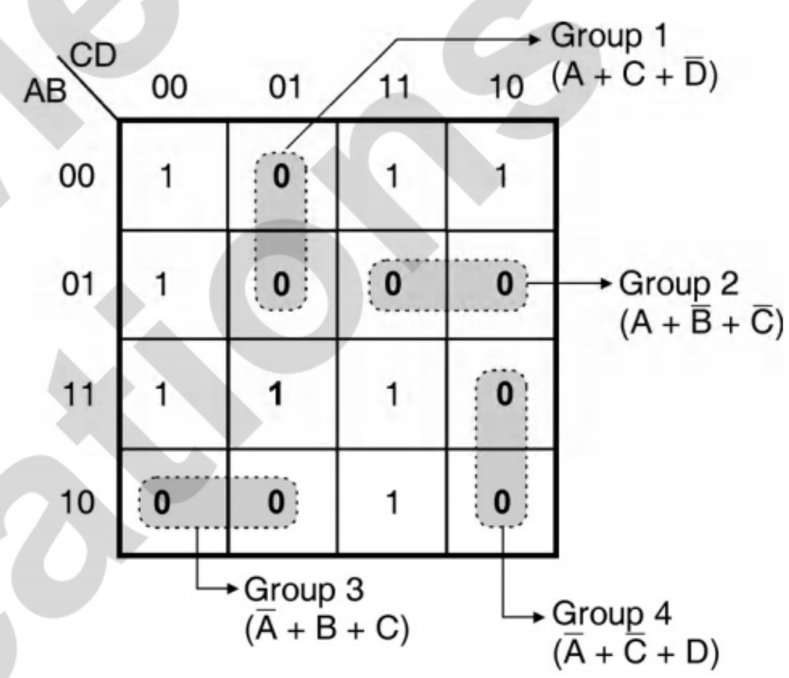
$$Y = \underset{(1)}{(B + C + \bar{D})}\ \underset{(2)}{(\bar{A} + B + D)}\ \underset{(3)}{(A + \bar{B} + \bar{D})}\ \underset{(4)}{(\bar{B} + \bar{C} + D)}$$

(C-6359)

– This shows that for the same K-map we can get completely different answers and all of them are correct.



(b) One way of grouping



(c) Another way of grouping

(C-298) Fig. P. 1.8.8

Ex. 1.8.9 : Solve the following equation using K map minimization technique. Draw the diagram for the output :

$Z = f(A, B, C, D) = \Pi M\ (0, 1, 6, 7, 8, 9)$

Dec. 11, 6 Marks

Soln. :

Simplification using K-map :

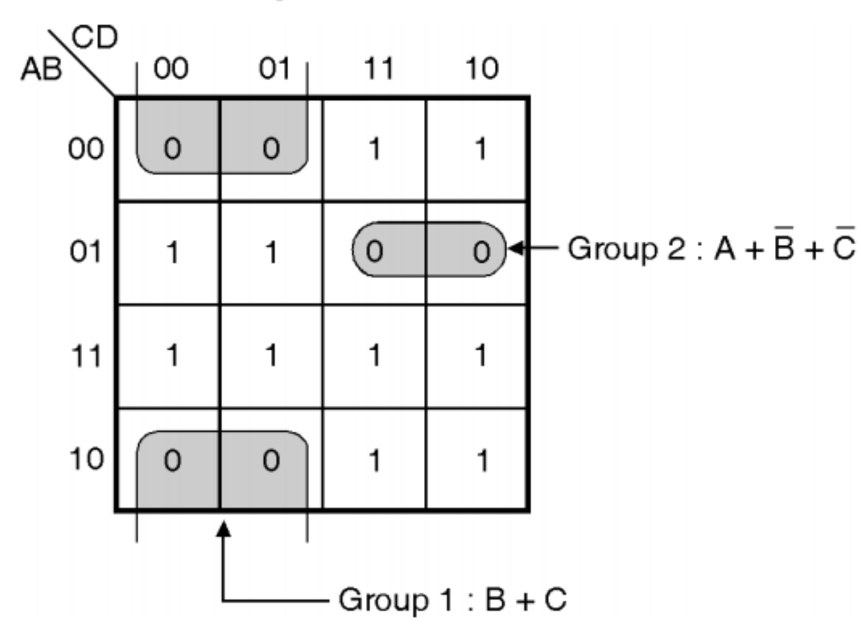
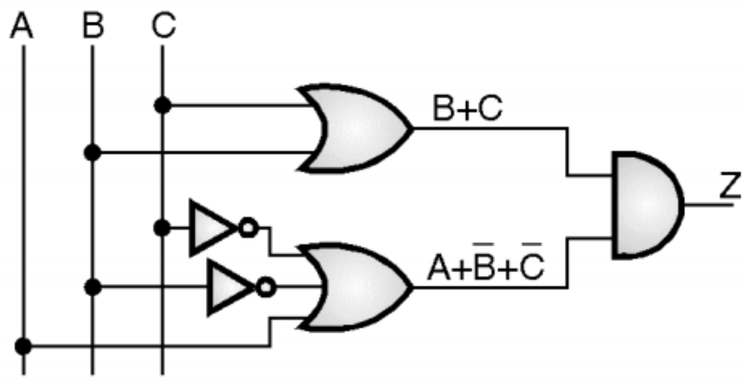


Fig. P. 1.8.9(a) (Contd...)

Realization using gates :



(C-3595) **Fig. P. 1.8.9(b)**

$\therefore \quad Z = (B + C)(A + \bar{B} + \bar{C})$ **...Ans.**

Ex. 1.8.10 : Solve the following equation using corresponding minimization technique. Draw the MSI design for the minimized output.

Z = f(A, B, C, D)

= ΠM (1, 3, 5, 6, 7, 10, 11) + d(2, 4)

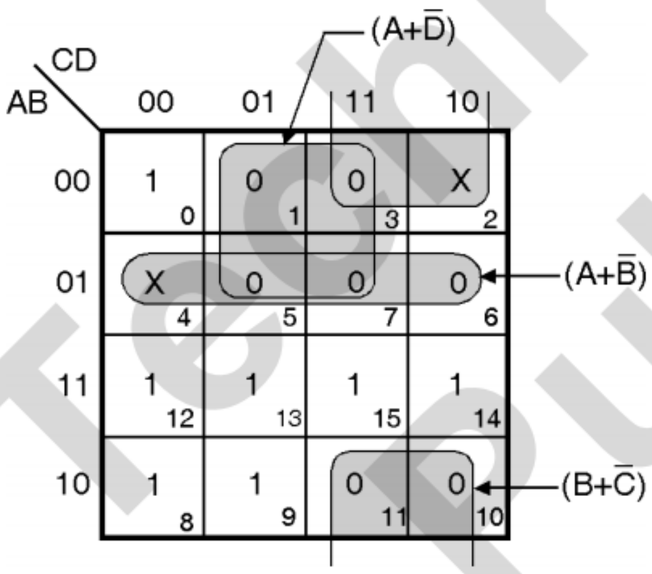
May 12, 6 Marks

Soln. :

$$Z = \pi M(1, 3, 5, 6, 7, 10, 11) + d(2, 4)$$

Step 1 : Simplification using K-map :

– Simplification using K-map is as shown in P. 1.8.10(a).



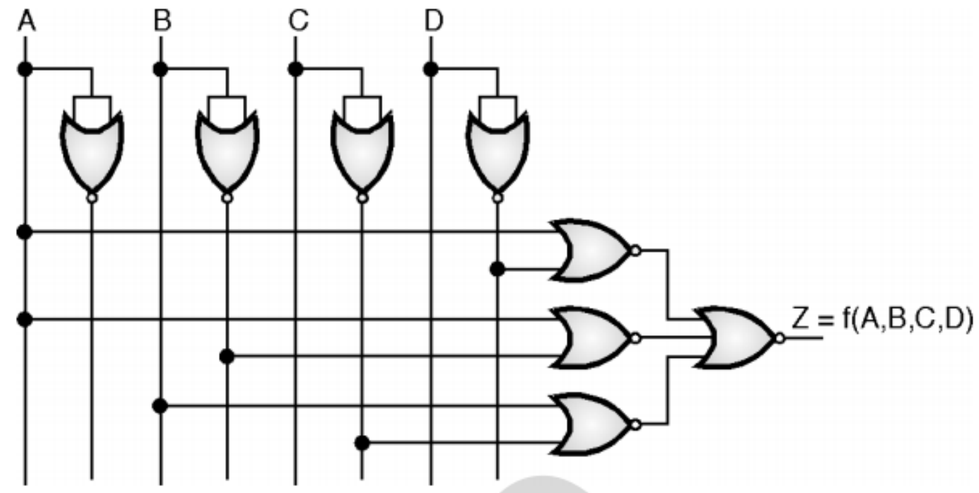
(C-3226) **Fig. P. 1.8.10(a)**

$$\therefore \quad Z = (A + \bar{D}) \cdot (A + \bar{B}) \cdot (B + \bar{C})$$

Step 2 : Realization using NOR gates :

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

$$Z = \overline{\overline{(A + \bar{D})} + \overline{(A + \bar{B})} + \overline{(B + \bar{C})}}$$



(C-3227) **Fig P. 1.8.10(b) : Implementation using NOR**

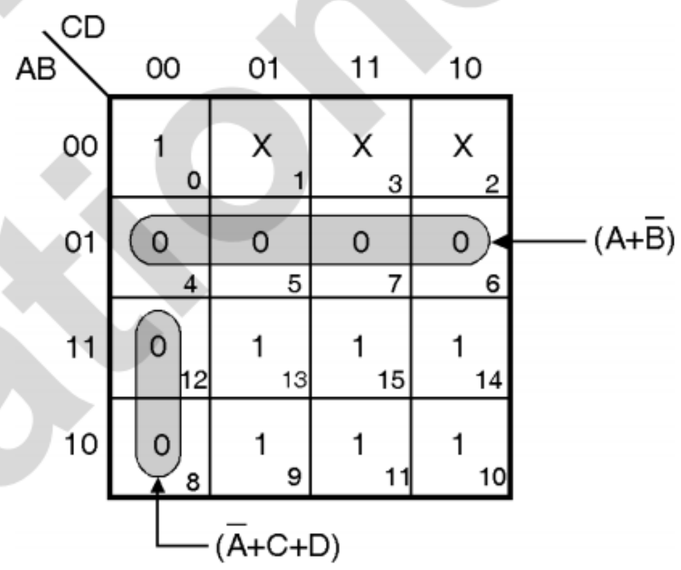
Ex. 1.8.11 : Minimize the following four variable functions using K-map. 'd' represents don't care conditions :

f(A, B, C, D) = ΠM (4, 5, 6, 7, 8, 12) · d(1, 2, 3).

Dec. 12, 5 Marks

Soln. :

– Required K-map is as follows :



(C-3228) **Fig. P. 1.8.11**

$\therefore f(A, B, C, D) = (A + \bar{B})(\bar{A} + C + D)$ **...Ans.**

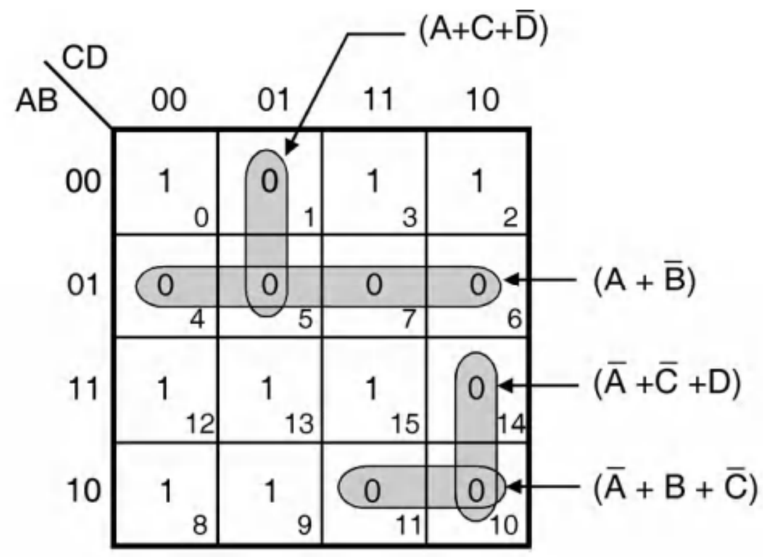
Ex. 1.8.12 : Reduce the following using K-map techniques :

f (A, B, C, D) = π (0, 2, 3, 8, 9, 12, 13, 15).

Dec. 14, 4 Marks

Soln. :

– Simplification using K-map is as shown in Fig. P. 1.8.12.



(C-4915) **Fig. P. 1.8.12**

$\therefore f(A, B, C, D) = (A + C + \bar{D})(A + \bar{B})(\bar{A} + \bar{C} + D)(\bar{A} + B + \bar{C})$ **...Ans.**

TechKnowledge Publications

Ex. 1.8.13 : Using K-map convert the following standard POS expression into a minimum POS expression, a standard SOP expression and minimum SOP expression :

$(A' + B' + C + D)\,(A + B' + C + D)$

$(A + B + C + D')\,(A + B + C' + D')$

$(A' + B + C + D')\,(A + B + C' + D)$

Dec. 15, 6 Marks

Soln. :

Step 1 : Convert the given standard POS into minimum POS :

$\bar{A} + \bar{B} + C + D = M_{12}$

$A + \bar{B} + C + D = M_4$

$A + B + C + \bar{D} = M_1$

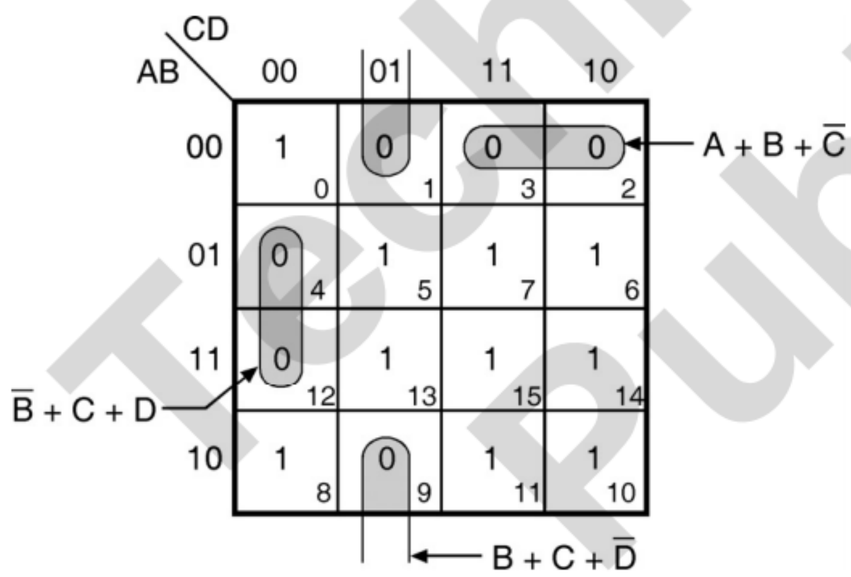
$A + B + \bar{C} + \bar{D} = M_3$

$\bar{A} + B + C + \bar{D} = M_9$

$A + B + \bar{C} + D = M_2$

$\therefore Y = \pi M (1, 2, 3, 4, 9, 12)$

– Simplification using K-map is as shown in Fig. P. 1.8.13(a).



(C-5160) Fig. P. 1.8.13(a) : K-map

$\therefore Y = (A + B + \bar{C})\,(B + C + \bar{D})\,(\bar{B} + C + D)$ **...Ans.**

Step 2 : Convert standard POS into standard SOP :

From Fig. P. 1.8.13(a),

Standard SOP expression,

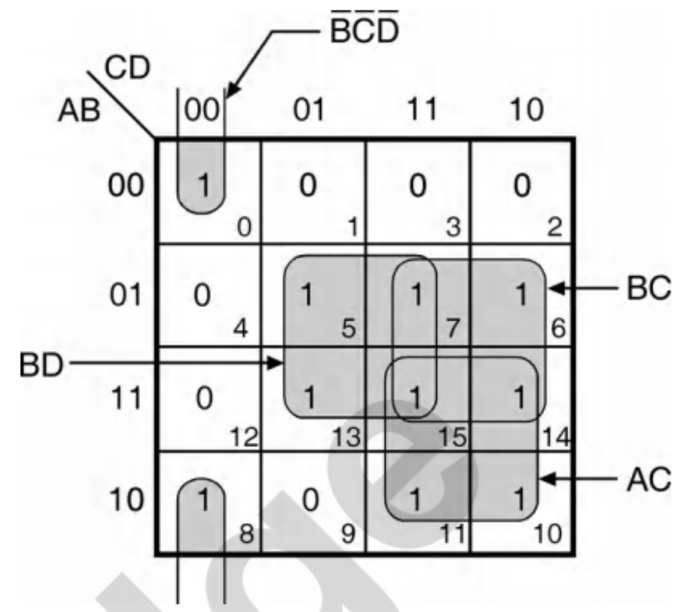
$Y = \sum m\,(0, 5, 6, 7, 8, 10, 11, 13, 14, 15)$

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

$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}D$

$+ ABC\bar{D} + ABCD$ **...Ans.**

Step 3 : Convert standard SOP into minimum SOP :



(C-5161) Fig. P. 1.8.13(b)

$\therefore Y = \bar{B}\bar{C}\bar{D} + BC + AC + BD$ **...Ans.**

Ex. 1.8.14 : Implement the following POS expression using NOR - NOR logic, $F = \Pi M\,(0, 1, 2, 4, 6)$

Soln. : Solve it yourself.

Ans. :

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

Ex. 1.8.15 : Solve the following using minimization technique :

$z = f\,(A, B, C, D) = \Pi(1, 2, 3, 6, 8, 11, 14, 15)$.

Dec. 09, 10 Marks

Soln. : Solve it yourself.

Ans. :

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

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

Ex. 1.8.16 : Solve the following equation using corresponding minimization techniques, also draw MSI design for the minimized output equation :

$Z = f\,(A, B, C, D) = \Pi(2, 7, 8, 10, 11, 13, 15)$

May 10, 12 Marks

Soln. : Solve it yourself.

Ans. :

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

1.9 Quine Mc-Cluskey Minimization Technique (Tabular Method) :

SPPU : Dec. 11

University Questions

Q. 1 What are the advantages of Quine Mc-Cluskey minimization technique over K-map ?

(Dec. 11, 4 Marks)

TechKnowledge Publications

– There are some combinational circuits with a large number of inputs.
– So the simplification using K-maps becomes too difficult. This happens when the number of variables is greater than 6.
– So some other techniques for realizing the combinational circuits are used. Some of them are as follows :

1. Use of programmable logic array.
2. Programmable logic devices.
3. Quine Mc-Cluskey minimization technique.

– Quine Mc-Cluskey technique is based on automatic or computer driven simplification routine.
– Therefore if we use Quine-Mc-Cluskey method then it is possible to use computer for simplifying the logic equations.
– A logic minimization technique should have the following characteristics :

1. It should be capable of handling large number of variables.
2. The technique should not be dependent on the abilities of the user otherwise the success of using that technique varies on the basis of user's skills.
3. It should be compatible with computers so that we can use computers to obtain the solution.
4. It should give us the minimized expression.

– The Quine Mc-Cluskey technique satisfies all these requirements.
– We can use this technique for designing various complex digital systems. Q.M. technique is also known as the **Tabular method.**

1.9.1 Important Definitions :

– **Prime Implicant (PI) :** The prime implicant is a group of minterms which cannot be combined with any other minterm or groups.
– **Essential Prime Implicant (EPI) :** The essential prime implicant is a prime implicant in which one or more minterms are unique i.e. it contains at least one minterm which is not contained in any other prime implicant.
– The Quine Mc-Cluskey technique consists of two parts :

1. First is to find all the prime implicants by an exhaustive search.
2. To identify the essential prime implicants obtained in part 1 and select from them the remaining prime implicants which can give the perfectly minimized expression.

– The Quine Mc-Cluskey method can be better understood by solving the following examples.

Ex. 1.9.1 : Simplify the following Boolean expression using K-map and verify it using the Quine-Mc Cluskey method.

$Y (A, B, C, D) = \sum m (0, 1, 3, 7, 8, 9, 11, 15)$.

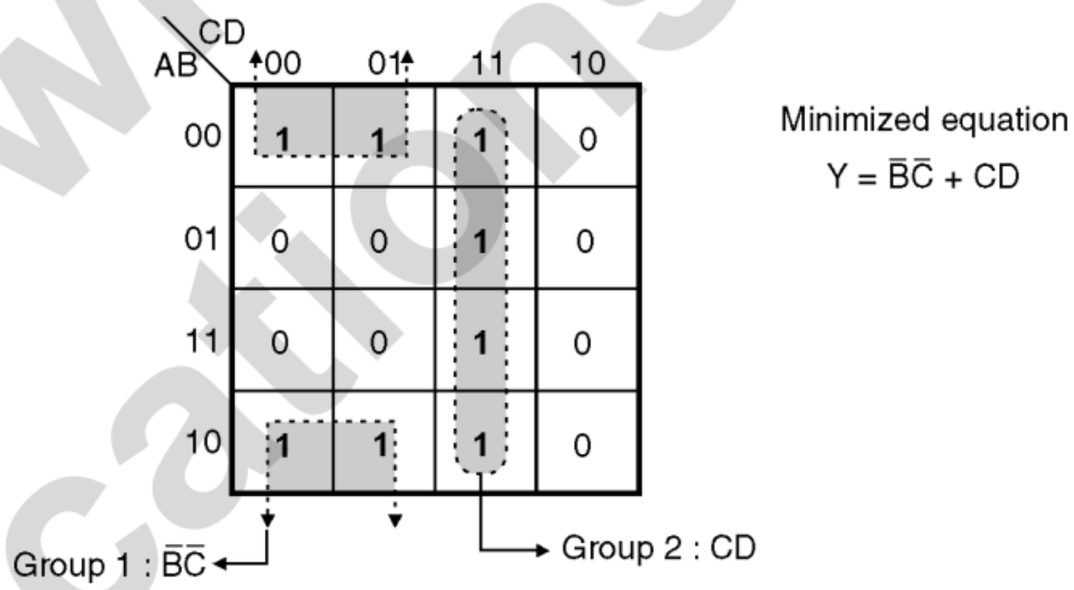
May 17, 6 Marks

Soln. :

Simplification using K-map :

Step 1 :

– The K-map for the given Boolean function is shown in Fig. P. 1.9.1(a).



(C-306) **Fig. P. 1.9.1(a) : Simplification using K-map**

Simplification using Quine-Mc Cluskey Method

Step 2 : Arrange all the minterms according to number of 1's contained and form the groups having no ones, one 1, two 1's, three 1's and so on, as shown in Table P. 1.9.1(a).

(C-6426)**Table P. 1.9.1(a)**

Group	Minterm	Representation in binary form					
		A	B	C	D		
1	m_0	0	0	0	0	✓	No 1's
2	m_1	0	0	0	1	✓	One 1
	m_8	1	0	0	0	✓	
3	m_3	0	0	1	1	✓	Two 1's
	m_9	1	0	0	1	✓	
4	m_7	0	1	1	1	✓	Three 1's
	m_{11}	1	0	1	1	✓	
5	m_{15}	1	1	1	1	✓	Four 1's

Step 3 : Now compare each minterm in group "n" with each minterm in group "(n + 1)", and identify the matched pairs. Put a check (✓) mark on each matched pair as shown in Table P. 1.9.1(a).

– A matched pair is a pair of minterms which differ only in one variable e.g. m_0 and m_4 is a matched pair because they differ only in the third position.

(C-6163)Table P. 1.9.1(b) : Combination of minterms into groups of two

Group	Minterms (Matched pairs)	Binary Representation A	B	C	D	
0 (First entry)	$m_0 - m_1$	0	0	0	–	✓
	$m_0 - m_8$	–	0	0	0	✓
1	$m_1 - m_3$	0	0	–	1	✓
	$m_1 - m_9$	–	0	0	1	✓
	$m_8 - m_9$	1	0	0	–	✓
2	$m_3 - m_7$	0	–	1	1	✓
	$m_3 - m_{11}$	–	0	1	1	✓
	$m_9 - m_{11}$	1	0	–	1	✓
3	$m_7 - m_{15}$	–	1	1	1	✓
	$m_{11} - m_{15}$	1	–	1	1	✓

Matched pairs of minterms obtained from above table — New terms generated from the matched pairs of minterms

– After comparing each minterm in the nth group with every term in the $(n + 1)^{th}$ group, another table i.e. Table P. 1.9.1(b) is prepared.

– This table lists the results of all the matchings (all the matched terms) alongwith new terms generated as a result of matching of the corresponding terms.

– For example consider the first entry in Table P. 1.9.1(b). The matched pair is $m_0 - m_1$.

		A	B	C	D
m_0	:	0	0	0	0
m_1	:	0	0	0	1

Only these bits are different

(C-307)

– Here A, B and C are equal and m_0 and m_1 differ in only "D". So the new term is written in front of $m_0 - m_1$ as 0 0 0 –.

– Note that a (–) is put in place of D which indicates that "D" gets eliminated. Similarly the other entries are made.

– Note that pairs cannot be formed between minterms 8 and 3 or 9 and 7 in Table P. 1.9.1(a).

Step 4 :

– Now compare all the pairs of minterms in Table P. 1.9.1(b) with those in the adjacent groups, to find if we can form groups of four minterms.

– For example we compare pair $(m_0 - m_1)$ of group 0 with each pair in group 1. The pair $(m_0 - m_8)$ of group 0 is compared with each pair in group 1.

– The matching between two pairs of minterms in the adjacent groups is obtained if the "dashes" are in the same position and only one variable differs.

– Such matched pairs are tickmarked (✓) in Table P. 1.9.1(b). The matched pairs are also shown by dotted lines, in Table P. 1.9.1(b).

– For example, $(m_0 - m_1)$ of group 0 matches with $(m_8 - m_9)$ of group 1 or $(m_0 - m_8)$ of group 0 matches with $(m_1 - m_9)$ of group 1 and so on.

– Prepare a new table as shown in Table P. 1.9.1(c) which contains the pairs of pairs. For example $m_0 - m_1 - m_8 - m_9$ i.e. groups of four minterms.

– In front of each group of four minterms, the new terms are written with dashes (–).

– By comparing the minterm pairs of Table P. 1.9.1(b), bit by bit we put the dashes (–) where the bits do not match.

(C-6164) Table P. 1.9.1(c) : Combination of minterms into groups of four

Group	Minterms	Binary representation A	B	C	D	
0	$m_0 - m_1 - m_8 - m_9$	–	0	0	–	$\bar{B}\bar{C}$
	$m_0 - m_8 - m_1 - m_9$	–	0	0	–	
1	$m_1 - m_3 - m_9 - m_{11}$	–	0	–	1	$\bar{B}D$
	$m_1 - m_9 - m_3 - m_{11}$	–	0	–	1	
2	$m_3 - m_7 - m_{11} - m_{15}$	–	–	1	1	CD
	$m_3 - m_{11} - m_7 - m_{15}$	–	–	1	1	

Step 5 :

– Repeat the procedure of grouping. So see if we can group the "Quads" of minterms in the adjacent groups of Table P. 1.9.1(c) to obtain groups of eight minterms.

– There are no such matchings in Table P. 1.9.1(c). So the process of grouping will end here.

Step 6 :

– Collect all the nonchecked (non ✓ marked) terms from Tables P. 1.9.1(a), (b) and (c).

– These are the Prime Implicants (PI) and they will appear in the simplified Boolean expression of Y.

– There are no such terms in Tables P. 1.9.1(a) and (b). But there are three such terms inTable P. 1.9.1(c) as $\bar{B}\bar{C}$, $\bar{B}D$ and CD marked on that table.

$$\therefore \; Y(A, B, C, D) = \bar{B}\bar{C} + \bar{B}D + CD \qquad ...(1)$$

Step 7 :

– Now prepare the prime-implicant table as shown in Table P. 1.9.1(d).

(C-308) Table P. 1.9.1(d) : Prime implicant table

PI	Decimal numbers corresponding to PI	Given minterms							
		0	1	3	7	8	9	11	15
$\overline{B}\overline{C}$ ✓	0, 1, 8, 9	⊗	×			⊗	×		
$\overline{B}D$	1, 3, 9, 11		×	×			×	×	
CD ✓	3, 7, 11, 15			×	⊗			×	⊗

Obtained from above table

– This table consists of the minterms contained in the given function, the PI terms included in Equation (1) and the decimal numbers of the minterms corresponding to these prime implicants.

– A (X) is put in each row under the minterms contained in a particular PI.

– Find the minterms that contain only one (X) in its column and encircle these cross marks as shown in Table P. 1.9.1(d) and put a tick mark (✓) infront of the corresponding PIs i.e $\overline{B}\,\overline{C}$ and CD.

– The encircled minterms 0 and 8 are contained in only one PI i.e. $\overline{B}\,\overline{C}$ whereas the minterms 7 and 15 are contained by only PI "CD".

– Hence the prime implicants $\overline{B}\,\overline{C}$ and CD are called as the **Essential Prime Implicants (EPI).**

– Now consider the other minterms (those which are not circle). The minterms 1 and 9 are contained in $\overline{B}\,\overline{C}$ (row 1 of Table P. 1.9.1(d)) and minterms 3 and 11 are contained in CD.

– Thus all the minterms in the given function are included in the two EPIs. So the minimized expression is given by,

$$Y(A, B, C, D) = \overline{B}\,\overline{C} + CD \quad \textbf{...Ans.}$$

Note : The answer obtained by the Quine Mc Cluskey method is exactly same as that obtained by the K–map method.

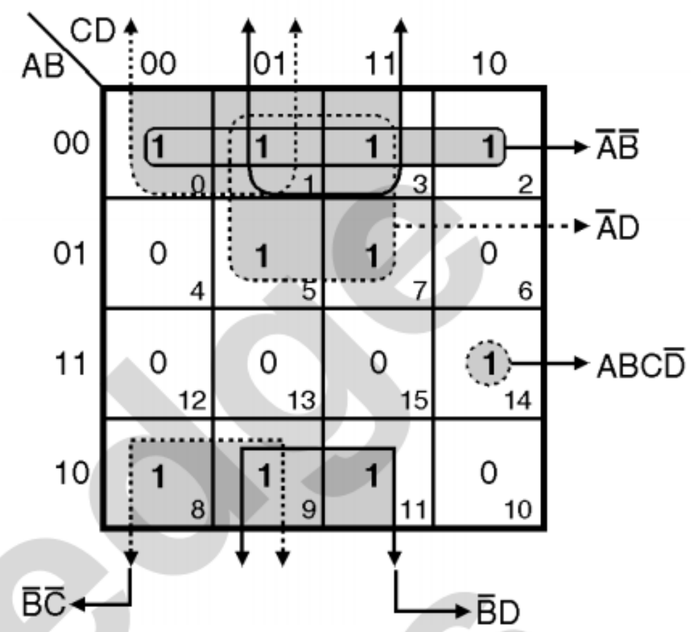
Ex. 1.9.2 : Minimize the following logic function using K-map and verify the answer using the Quine-Mc Cluskey method. Y (A, B, C, D) = ∑ m (0, 1, 2, 3, 5, 7, 8, 9, 11, 14).

May 18, Dec. 18, 6 Marks

Soln. :

Part I : Minimization using K-map

– The K-map for the given logic function is shown in Fig. P. 1.9.2.



(C-309) Fig. P. 1.9.2 : K-map

∴ Simplified expression :

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

Part II : Quine – Mc Cluskey Method

Step 1 : Arrange all minterms according to the number of 1's :

– The minterms are arranged according to the number of 1's and groups are formed as no ones, one 1, two 1s etc as shown in Table P. 1.9.2(a).

Minterm 0 forms a pair with 1, 2 and 8

	Group	Minterms	Representation in binary form				
			A	B	C	D	
No 1's	0	0	0	0	0	0	✓
One 1	1	1	0	0	0	1	✓
		2	0	0	1	0	✓
		8	1	0	0	0	✓
Two 1's	2	3	0	0	1	1	✓
		5	0	1	0	1	✓
		9	1	0	0	1	✓
Three 1's	3	7	0	1	1	1	✓
		11	1	0	1	1	✓
		14	1	1	1	0	

Minterm 14 does not appear in any pair so it is not ✓ marked

(C-6165) Table P. 1.9.2(a)

Step 2 : Combine the minterms into a group of two :

– Table P. 1.9.2(b) shows the matched pairs of minterms in the adjacent groups of Table P. 1.9.2(a) which differ at only one location (bit position) with respect to each other.

TechKnowledge Publications

– Place (✓) mark on the matched pairs in Table P. 1.9.2(a).

(C-6166) Table P. 1.9.2(b) : Combination of minterms into groups of two

Matched quads of minterms

Group	Minterms (Matched pairs)	Binary Representation A	B	C	D	
0	0 – 1	0	0	0	–	✓
	0 – 2	0	0	–	0	✓
	0 – 8	–	0	0	0	✓
1	1 – 3	0	0	–	1	✓
	1 – 5	0	–	0	1	✓
	1 – 9	–	0	0	1	✓
	2 – 3	0	0	1	–	✓
	8 – 9	1	0	0	–	✓
2	3 – 7	0	–	1	1	✓
	3 – 11	–	0	1	1	✓
	5 – 7	0	1	–	1	✓
	9 – 11	1	0	–	1	✓

Matched pairs of minterms obtained from above table

New terms generated from the matched pairs of minterms

– The bit position where the minterms differ are represented by dashes (–) in the new terms written in front of matched pairs.

Step 3 : Combine the minterm pairs into groups of four (Quad) :

– Group the minterm pairs of Table P. 1.9.2(b) (adjacent groups) to form minterm quads as shown in Table P. 1.9.2(c) and place a (✓) mark on the matched pairs in Table P. 1.9.2(b).

(C-6167) Table P. 1.9.2(c) : Combination of minterms into groups of four

Group	Minterms (Group of four)	Binary Representation A	B	C	D	Prime implicants
0	0 – 1 – 2 – 3	0	0	–	–	$\bar{A}\bar{B}$
	0 – 1 – 8 – 9	–	0	0	–	$\bar{B}\bar{C}$
	0 – 2 – 1 – 3	0	0	–	–	
	0 – 8 – 1 – 9	–	0	0	–	
1	1 – 5 – 3 – 7	0	–	–	1	$\bar{A}D$
	1 – 9 – 3 – 11	–	0	–	1	$\bar{B}D$
	1 – 3 – 5 – 7	0	–	–	1	
	1 – 3 – 9 – 11	–	0	–	1	

– No further grouping is possible after this. So the process of grouping stops here.

Step 4 : Collect all nonchecked terms :

– Collect all the nonchecked (non tickmarked) terms from Tables P. 1.9.2(a), (b) and (c), because they are the Prime-Implicants (PI).

– In Table P. 1.9.2(a), the minterm 14 is nontickmarked so $A\,B\,C\,\bar{D}$ is a PI. In Table P. 1.9.2(b) we do not have any such term and in Table P. 1.9.2(c) we have four such terms i.e. $\bar{A}\,\bar{B}$, $\bar{B}\,\bar{C}$, $\bar{A}\,D$ and $\bar{B}\,D$.

$$\therefore Y(A, B, C, D) = A\,B\,C\bar{D} + \bar{A}\,\bar{B} + \bar{B}\,\bar{C} + \bar{A}\,D + \bar{B}\,D \quad ...(1)$$

Step 5 : Prepare the PI table and obtain the EPIs :

– The PI table is shown in Table P. 1.9.2(d).

(C-310) Table P. 1.9.2(d) : Prime Implicants (PI) table

PI	Decimal number corresponding to PI	Given minterms 0	1	2	3	5	7	8	9	11	14
$ABC\bar{D}$ ✓	14										⊗
$\bar{A}\bar{B}$ ✓	0, 1, 2, 3	×	×	⊗	×						
$\bar{B}\bar{C}$ ✓	0, 1, 8, 9	×	×					⊗	×		
$\bar{A}D$ ✓	1, 3, 5, 7		×		×	⊗	⊗				
$\bar{B}D$ ✓	1, 3, 9, 11		×		×				×	⊗	

– In the PI table find the columns containing only 1 cross (×) and encircle those (×) marks. Put (✓) mark in front of the corresponding PIs.

– These (✓) marked prime implicants in Table P. 1.9.2(d) are the Essential Prime Implicants (EPIs).

– Hence the simplified expression for Y is,

$$\therefore Y(A, B, C, D) = A\,B\,C\bar{D} + \bar{A}\,\bar{B} + \bar{B}\,\bar{C} + \bar{A}\,D + \bar{B}\,D. \quad \textbf{...Ans.}$$

– Note that this expression is same as the one obtained with K-map simplification.

Ex. 1.9.3 : Obtain the minimal expression using Quine-Mc Cluskey method.

$f(A, B, C, D) = \sum m(1, 5, 6, 12, 13, 14) + d(2, 4)$ **May 14, 6 Marks**

TechKnowledge Publications

Soln. :

Step 1 : Group the minterms according to number of 1's :

(C-6367) Table P. 1.9.3(a)

Group	Minterms	Binary representation				
		A	B	C	D	
0	1	0	0	0	1	✓
	2*	0	0	1	0	✓
	4*	0	1	0	0	✓
1	5	0	1	0	1	✓
	6	0	1	1	0	✓
	12	1	1	0	0	✓
2	13	1	1	0	1	✓
	14	1	1	1	0	✓

* Represents the don't care terms

Step 2 : Group the minterms to form pairs :

(C-6168) Table P. 1.9.3(b)

Group	Minterm pair	Binary representation				
		A	B	C	D	
0	1, 5	0	–	0	1	→ $\bar{A}\bar{C}D$
	2*, 6	0	–	1	0	→ $\bar{A}C\bar{D}$
	4*, 5	0	1	0	–	✓
	4*, 6	0	1	–	0	✓
	4*, 12	–	1	0	0	✓
1	5, 13	–	1	0	1	✓
	6, 14	–	1	1	0	✓
	12, 13	1	1	0	–	✓
	12, 14	1	1	–	0	✓

Prime implicants (non ✓ marked terms)

Step 3 : Group the minterms to form groups of four :

(C-6169) Table P. 1.9.3(c)

Group	Minterm quads	Binary representation				
		A	B	C	D	
0	4, 5, 12, 13	–	1	0	–	$B\bar{C}$
	4, 6, 12, 14	–	1	–	0	$B\bar{D}$
	4, 12, 5, 13	–	1	0	–	$B\bar{C}$
	4, 12, 6, 14	–	1	–	0	$B\bar{D}$

Prime implicants (non ✓ marked terms)

Step 4 : Group the prime implicants :

– The non tick (✓) marked terms from Tables P. 1.9.3(b) and (c) are the Prime Implicants (PI). They are likely to appear in the simplified expression for output as follows :

$$f(A, B, C, D) = \bar{A}\bar{C}D + \bar{A}C\bar{D} + B\bar{C} + B\bar{D} \quad ...(1)$$

Step 5 : Prepare the table of prime implicants :

(C-311) Table P. 1.9.3(d) : Table of PI

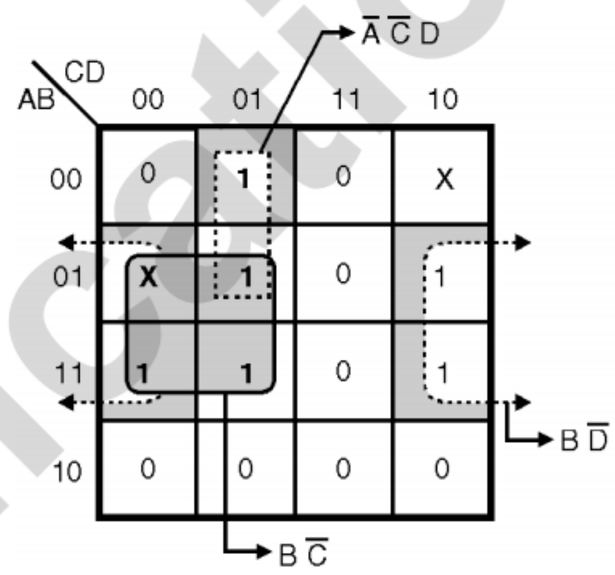
Don't care terms are not listed in this table

Prime implicants	Decimal numbers (minterms)	Minterms given					
		1	5	6	12	13	14
$\bar{A}\bar{C}D$ ✓	1, 5	⊗	X				
$\bar{A}C\bar{D}$	2, 6			X			
$B\bar{C}$ ✓	4, 5, 12, 13		X		X	⊗	
$B\bar{D}$ ✓	4, 6, 12, 14			X	X		⊗
		✓				✓	✓

– The encircled crosses (×) represent the Essential Prime Implicants (EPIs). They are $\bar{A}\bar{C}D$, $B\bar{C}$, $B\bar{D}$ and they cover the minterms 1, 5, 6, 12, 13 and 14 i.e. all the given minterms (Do not worry about 2 and 4 which are don't cares).

$$\therefore \quad f(A, B, C, D) = \bar{A}\bar{C}D + B\bar{C} + B\bar{D} \quad \textbf{...Ans.}$$

Crosscheck using K-map :



$\therefore f(A\,B\,C\,D) = \bar{A}\bar{C}D + B\bar{C} + B\bar{D}$
Thus the answers using both the methods match

(C-312) Fig. P. 1.9.3

Ex. 1.9.4 : Realize the following logic function in SOP form using Quine–Mc Cluskey method.

$f(A, B, C, D) = \Pi M(2, 7, 8, 9, 10, 12)$

Soln. :

– Note that the given function is in the POS form or maxterm form and we want the solution in the SOP form.

– Hence we have to first convert the given expression into SOP form using the complementary relationship between the POS and SOP form.

$$f(A, B, C, D) = \Pi M(2, 7, 8, 9, 10, 12)$$
$$= \Sigma m(0, 1, 3, 4, 5, 6, 11, 13, 14, 15).$$

TechKnowledge Publications

Step 1 : Group the minterms according to number of 1's :

(C-6170) Table P. 1.9.4(a)

Group	Minterms	Binary representation A	B	C	D	See Step 2	
0	0	0	0	0	0	✓	No ones
1	1	0	0	0	1	✓	One 1
	4	0	1	0	0	✓	
2	3	0	0	1	1	✓	Two 1s
	5	0	1	0	1	✓	
	6	0	1	1	0	✓	
3	11	1	0	1	1	✓	Three 1s
	13	1	1	0	1	✓	
	14	1	1	1	0	✓	
4	15	1	1	1	1	✓	Four 1s

Step 2 : Group the minterms to form pairs :

(C-6171) Table P. 1.9.4(b)

Group	Minterm pairs	Binary representation A	B	C	D	See Step 3	
0	0 – 1	0	0	0	–	✓	
	0 – 4	0	–	0	0	✓	
1	1 – 3	0	0	–	1		← $\bar{A}\bar{B}D$
	1 – 5	0	–	0	1	✓	
	4 – 5	0	1	0	–	✓	
	4 – 6	0	1	–	0		← $\bar{A}B\bar{D}$
2	3 – 11	–	0	1	1		→ $\bar{B}CD$
	5 – 13	–	1	0	1		→ $B\bar{C}D$
	6 – 14	–	1	1	0		→ $BC\bar{D}$
3	11 – 15	1	–	1	1		→ ACD
	13 – 15	1	1	–	1		→ ABD
	14 – 15	1	1	1	–		→ ABC

Prime implicants (non ✓ marked terms)

Step 3 : Group the minterms to form quads (groups of 4) :

(C-6172) Table P. 1.9.4(c)

Group	Minterm quads	Binary representation A	B	C	D	
0	0–4–1–5	0	–	0	–	$\bar{A}\bar{C}$ - Prime implicant
	0–1–4–5	0	–	0	–	

Step 4 : Collect all the Prime Implicants (PI) :

– All the nontick marked terms in Tables P. 1.9.4(b) and (c) are the Prime Implicants (PI) so they will appear in the expression for output, as follows.

$$f(A, B, C, D) = \bar{A}\bar{B}D + \bar{A}B\bar{D} + \bar{B}CD + B\bar{C}D + BC\bar{D}$$
$$+ ACD + ABD + ABC + \bar{A}\bar{C} \quad ...(1)$$

Step 5 : Prepare the PI table :

– The **encircled** term in Table P. 1.9.4(d) is called as the Essential Prime Implicant (EPI) which covers the min terms 0, 1, 4 and 5.

– This term will appear in the minimized expression.

– The remaining minterms are 3, 6, 11, 13, 14, 15 so find the terms which cover these minterms.

(C-313) Table P. 1.9.4(d)

P I term	Decimal equivalent	Given minterms 0	1	3	4	5	6	11	13	14	15
$\bar{A}\bar{B}D$	1, 3		X	X							
$\bar{A}B\bar{D}$	4, 6				X		X				
$\bar{B}CD$ ✓	3, 11			X				X			
$B\bar{C}D$	5, 13					X			X		
$BC\bar{D}$ ✓	6, 14						X			X	
ACD	11, 15							X			X
ABD ✓	13, 15								X		X
ABC	14, 15									X	X
$\bar{A}\bar{C}$ ✓	0, 1, 4, 5	X ✓	X		X	X					

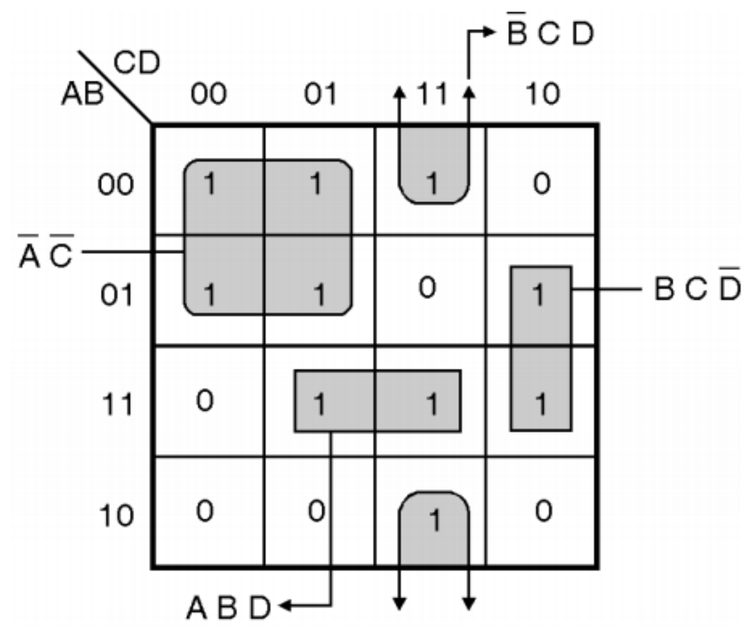
– From Table P. 1.9.4(d) such terms are $\bar{B}CD$ which covers the minterms 3 and 11, $BC\bar{D}$ which covers 6, 14 and ABD which covers 13 and 15.

– So these terms also will appear in the minimized expression.

– Hence the minimized expression in SOP form is given by

$$f(A, B, C, D) = \bar{A}\bar{C} + \bar{B}CD + BC\bar{D} + ABD \quad \textbf{...Ans.}$$

Cross check using K-map :



$\therefore f(A, B, C, D) = \bar{A}\bar{C} + \bar{B}CD + BC\bar{D} + ABD$

(C-314) Fig. P. 1.9.4

– Thus the results obtained by the two methods are exactly identical.

TechKnowledge Publications

Ex. 1.9.5 : Find the minimal expression for the following function using Quine – Mc Cluskey method.

f (A, B, C, D) = ∑ m (0, 1, 4, 6, 8, 9, 10, 12) + d (5, 7, 14).

Soln. :

Step 1 : Arrange the minterms according to number of 1's :

(C-6173) Table P. 1.9.5(a)

*** don't care terms**

Group	Minterms	Binary representation A	B	C	D	See Step 2	
0	0	0	0	0	0	✓	No ones
1	1	0	0	0	1	✓	One 1
	4	0	1	0	0	✓	
	8	1	0	0	0	✓	
2	6	0	1	1	0	✓	Two 1's
	9	1	0	0	1	✓	
	10	1	0	1	0	✓	
	12	1	1	0	0	✓	
	5*	0	1	0	1	✓	
3	7*	0	1	1	1	✓	Three 1's
	14*	1	1	1	0	✓	

Step 2 : Group the minterms into groups of two :

(C-6377) Table P. 1.9.5(b)

Group	Pair of minterm	Binary representation A	B	C	D	See step 2
0	0 – 1	0	0	0	–	✓
	0 – 4	0	–	0	0	✓
	0 – 8	–	0	0	0	✓
1	1 – 9	–	0	0	1	✓
	1 – 5*	0	–	0	1	✓
	4 – 6	0	1	–	0	✓
	4 – 12	–	1	0	0	✓
	4 – 5*	0	1	0	–	✓
	8 – 9	1	0	0	–	✓
	8 – 10	1	0	–	0	✓
	8 – 12	1	–	0	0	✓
2	6 – 7*	0	1	1	–	✓
	6 – 14*	–	1	1	0	✓
	10 – 14*	1	–	1	0	✓
	12 – 14*	1	1	–	0	✓
	5 – 7*	0	1	–	1	✓

* don't care terms

Step 3 : Group the minterms into groups of four :

(C-6174) Table P. 1.9.5(c)

Group	Quad of minterms	Binary representation A	B	C	D	Prime implicants
0	0, 1, 4, 5*	0	–	0	–	$\bar{A}\bar{C}$
	0, 1, 8, 9	–	0	0	–	
	0, 4, 1, 5*	0	–	0	–	$\bar{B}\bar{C}$
	0, 4, 8, 12	–	–	0	0	
	0, 8, 1, 9	–	0	0	–	$\bar{C}\bar{D}$
	0, 8, 4, 12	–	–	0	0	
1	4, 6, 12, 14*	–	1	–	0	
	4, 6, 5*, 7*	0	1	–	–	$B\bar{D}$
	4, 12, 6, 14*	–	1	–	0	
	4, 5*, 6, 7*	0	1	–	–	$\bar{A}B$
	8, 10, 12, 14*	1	–	–	0	
	8, 12, 10, 14*	1	–	–	0	$A\bar{D}$

* don't care terms

– No further grouping is possible. So from the Tables P. 1.9.5(a), (b) and (c) list the terms which are not (✓) marked.

– They are Prime Implicants (PI) and hence will appear in the expression for Y as follows,

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

Step 4 : Prepare the table of prime implicants :

– Table P. 1.9.5(d) shows the table of prime implicants.

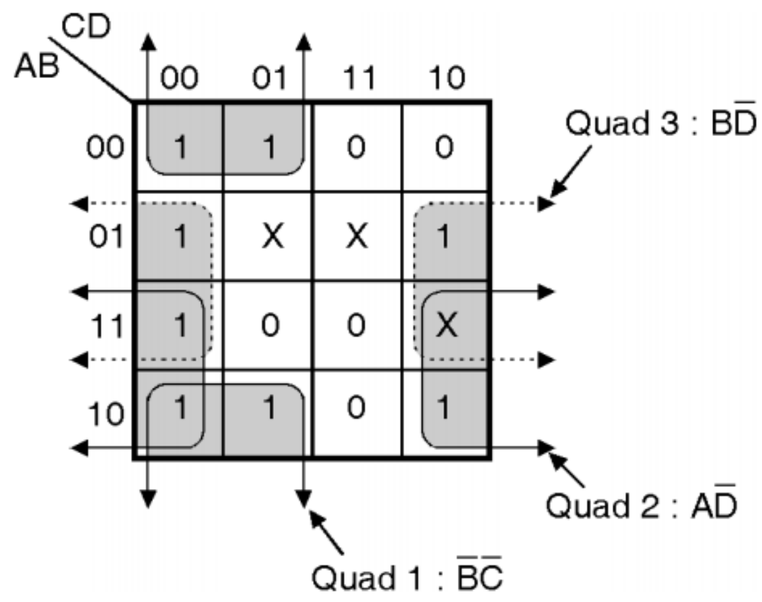
(C-315) Table P. 1.9.5(d) : Table of PI

Prime implicant	Decimal numbers	Given minterms 0	1	4	6	8	9	10	12
$\bar{A}\bar{C}$	0, 1, 4, 5*	X	X	X					
$\bar{B}\bar{C}$ ✓	0, 1, 8, 9	X	X			X	(X)		
$\bar{C}\bar{D}$	0, 4, 8, 12	X		X		X			X
$B\bar{D}$ ✓	4, 6, 12, 14*			X	X				X
$\bar{A}B$	4, 6, 5*, 7*			X	X				
$A\bar{D}$ ✓	8, 10, 12, 14*					X		(X)	X

– The encircled crosses represent the EPIs. So the terms $\bar{B}\bar{C}$ and $A\bar{D}$ are the EPIs. They cover the terms 0, 1, 8, 9 and 10, 12.

– Now check the remaining minterms i.e. 4 and 6. Both these are covered by $B\bar{D}$ so we have to include $B\bar{D}$.

Cross check using K-map :



(C-3398)

$$\therefore Y = \bar{B}\,\bar{C} + A\bar{D} + B\,\bar{D}$$ **...Ans.**

– This is same as the obtained with Quine Mc-Cluskey method.

Ex. 1.9.6 : Use the Quine Mc Cluskey method of minimization and find the expression for the following function :

f (A, B, C, D) = ∑ m (2, 4, 8, 11, 15) + d (1, 10, 12, 13).

Soln. : Solve it yourself.

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

Ex. 1.9.7 : With the help of Quine Mc-Cluskey technique determine the PI for the following equation :

z = f (A, B, C, D) = Σ (0, 1, 3, 4, 6, 8, 10, 12, 14)

Dec. 09, May 19, 10 Marks

Soln. :

z = f (A, B, C, D) = Σ (0, 1, 3, 4, 6, 8, 10, 12, 14)

Step 1 : Arrange the minterms according to number of 1's : **(C-7986)**

Group	Minterms	Binary representation				
		A	B	C	D	
0	0	0	0	0	0	✓
1	1	0	0	0	1	✓
	4	0	1	0	0	✓
	8	1	0	0	0	✓
2	3	0	0	1	1	✓
	6	0	1	1	0	✓
	10	1	0	1	0	✓
	12	1	1	0	0	✓
3	14	1	1	1	0	✓

Step 2 : Group the minterms to form pairs : **(C-7987)**

Group	Minterm pairs	Binary representation				Prime Implicants
		A	B	C	D	
0	0 , 1	0	0	0	–	$\bar{A}\bar{B}\bar{C}$
	0 , 4	0	–	0	0	✓
	0 , 8	–	0	0	0	✓
1	1 , 3	0	0	–	1	$\bar{A}\bar{B}D$
	4 , 6	0	1	–	0	✓
	4 , 12	–	1	0	0	✓
	8 , 10	1	0	–	0	✓
	8 , 12	1	–	0	0	✓
2	6 , 14	1	1	1	0	✓
	10 , 14	–	–	1	0	✓
	12 , 14	1	1	–	0	✓

Step 3 : Group the minterms to form groups of four :

(C-7988)

Group	Minterm quads	Binary representation				Prime Implicants
		A	B	C	D	
0	0 – 4 – 8 – 12	–	–	0	0	$\bar{C}\,\bar{D}$
	0 – 8 – 4 – 12	–	–	0	0	
1	8 – 10 – 12 – 14	1	–	–	0	$A\,\bar{D}$
	8 – 12 – 10 – 14	1	–	–	0	
	4 – 6 – 12 – 14	–	1	–	0	$B\,\bar{D}$
	4 – 12 – 6 – 14	–	1	–	0	

Step 4 : Collect all the Prime Implicants (PI) :

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

Step 5 : Prepare the table of prime implicants :

(C-1955)

PI term	Decimal numbers	Given minterms								
		0	1	3	4	6	8	10	12	14
$\bar{A}\,\bar{B}\,\bar{C}$	0, 1	X	X							
$\bar{A}\,\bar{B}\,D$	1, 3		X	X						
$\bar{C}\,\bar{D}$	0, 4, 8, 12	X			X		X		X	
$A\,\bar{D}$	8, 10, 12, 14						X	X	X	X
$B\,\bar{D}$	4, 6, 12, 14				X	X			X	X

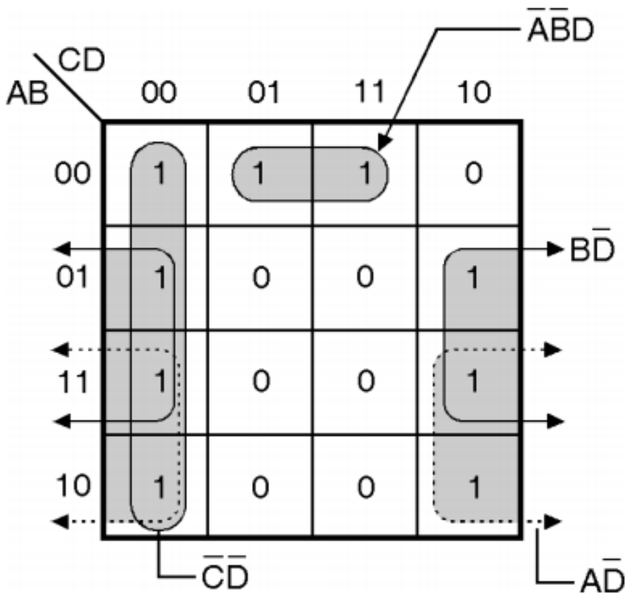
– The encircled crosses represent the EPIs. So the terms $\bar{A}\,\bar{B}\,D$, $A\,\bar{D}$ and $B\,\bar{D}$ are the EPIs.

TechKnowledge Publications

– They cover the terms 1, 3, 4, 6, 8, 10, 12, 14. The minterm 0 is not included so consider $\bar{C}\,\bar{D}$

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

Cross check :



(C-3596) **Fig. P. 1.9.7**

$$\therefore Z = \bar{A}\,\bar{B}\,D + A\bar{D} + B\,\bar{D} + \bar{C}\,\bar{D}$$

– This is same as the simplified expression obtained with Tabular method.

Ex. 1.9.8 : Minimize the following expression using Quine-McCluskey

f (A, B, C, D) = Σm (0, 2, 3, 6, 7, 8, 10, 12, 13).

Dec. 14, 6 Marks

Soln. :

$$f(A, B, C, D) = \Sigma m(0, 2, 3, 6, 7, 8, 10, 12, 13).$$

Step 1 : Arrange the minterms according to number of 1's :

Group	Minterm	Binary representation A	B	C	D	
0	0	0	0	0	0	✓
1	2	0	0	1	0	✓
	8	1	0	0	0	✓
2	3	0	0	1	1	✓
	6	0	1	1	0	✓
	10	1	0	1	0	✓
	12	1	1	0	0	✓
3	7	0	1	1	1	✓
	13	1	1	0	1	✓

(C-6437)

Step 2 : Group the minterms to form pairs :

Group	Minterm	Binary representation A	B	C	D	Prime implicants
0	0, 2	0	0	–	0	✓
	0, 8	–	0	0	0	✓
1	2, 3	0	0	1	–	✓
	2, 6	0	–	1	0	✓
	2, 10	–	0	1	0	✓
	8, 10	1	0	–	0	✓
	8, 12	1	–	0	0	$A\bar{C}\bar{D}$
2	3, 7	0	–	1	1	✓
	6, 7	0	1	1	–	✓
	12, 13	1	1	0	–	$AB\bar{C}$

(C-6438)

Step 3 : Group the minterms to form group of four :

Group	Minterms quad	Binary representation A	B	C	D	Prime implicant
0	0–2–8–10	–	0	–	0	$\bar{B}\bar{D}$
	0–8–2–10	–	0	–	0	
1	2–3–6–7	0	–	1	–	$\bar{A}C$
	2–6–3–7	0	–	1	–	

(C-6262)

Step 4 : Collect all the prime implicants :

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

Step 5 : Prepare the table of prime implicants :

PI term	Decimal numbers	Given minterms 0	2	3	6	7	8	10	12	13
$A\bar{C}\bar{D}$	8, 12						×		×	
$AB\bar{C}$	12, 13								×	⊗
$\bar{B}\bar{D}$	0, 2, 8, 10	⊗	×				×	⊗		
$\bar{A}C$	2, 3, 6, 7		×	⊗	⊗	⊗				

(C-4921)

– The encircled crosses represent the EPIs. So the terms $AB\bar{C}$, $\bar{B}\bar{D}$ and $\bar{A}C$ are EPIs. They cover all the minterms 0, 2, 3, 6, 7, 8, 10, 12, 13.

$$\therefore f(A, B, C, D) = AB\bar{C} + \bar{B}\bar{D} + \bar{A}C \quad \textbf{...Ans.}$$

Ex. 1.9.9 : With the help of Quine-Mc Cluskey technique determine the PI, EPI for the following equation : Z = f (A, B, C, D) = Σ (0, 3, 8, 9, 10, 12, 15) **May 10, 10 Marks**

Soln. : Solve it yourself.

TechKnowledge Publications

Ans. :

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

1.10 Unsigned Binary Numbers :

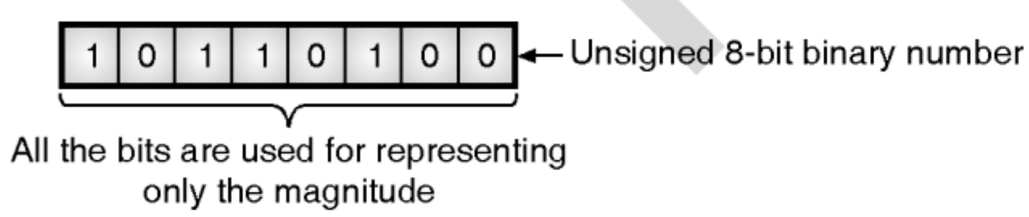
- In some applications, all the data is either positive or negative.
- Then we can just forget about the (+) or (–) signs, and concentrate only on the magnitude (absolute value) of the data.
- For example, the smallest 8 bit binary number is 0000 0000 i.e. all zeros, and the largest 8 bit binary number is 1111 1111.
- Hence the complete range of unsigned 8 bit binary numbers extends from $(00)_H$ to $(FF)_H$ or from $(00)_{10}$ to $(255)_{10}$.

It is important to note that we have not included (+) or (–) signs with these numbers.

- Similarly for 16-bit numbers the complete range is given by,

Smallest :	0000	0000	0000	0000 =	$(0000)_H$
Largest :	1111	1111	1111	1111 =	$(FFFF)_H$

- This represents the magnitude of 0 to 65,535 decimal.
- Data of this type is called as **unsigned binary** numbers because all of the bits in a binary number are used to represent the magnitude of the corresponding decimal number, as shown in Fig. 1.10.1.



(C-6109) Fig. 1.10.1 : Unsigned 8 bit binary number

1.10.1 Important Features of Unsigned Numbers :

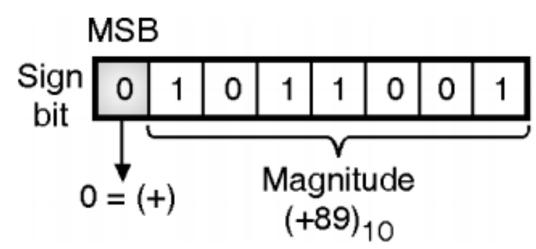
1. We can add or subtract the unsigned binary numbers if certain conditions are satisfied.
2. The microcomputers of first generation were able to process only 8 bits at a time. Therefore with 8-bit unsigned arithmetic all the magnitudes were restricted between 0 and $(255)_{10}$ or $(00)_H$ to $(FF)_H$.
3. That means all the numbers being added or subtracted must be in the range 0 to 255. More important is that the answer also should be in the range 0 to 255.
4. For the magnitudes greater than 255 we have to use 16-bit arithmetic.

Overflow :

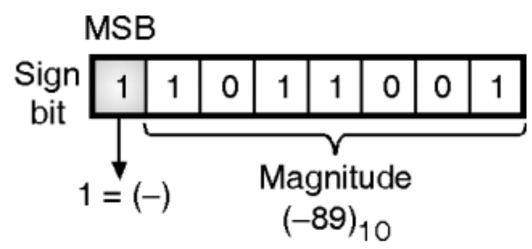
- If the addition or multiplication of two 8-bit numbers results in generation of a number, greater than $(255)_{10}$, then it is said that overflow has taken place.

1.11 Sign-Magnitude Numbers :

- If the data has positive as well as negative numbers then the signed binary numbers should be used, for their representation.
- For a sign-magnitude representation, the + or – signs are also represented in the binary form i.e. by using 0 or 1. So a 0 is used to represent the (+) sign and 1 is used to represent the (–) sign.
- The MSB of a binary number is used to represent the sign and the remaining bits are used for representing the magnitude. 8-bit signed binary numbers are shown in Fig. 1.11.1.



(a) A positive binary number



(b) A negative binary number 8-bit signed binary numbers

(C-6110) Fig. 1.11.1

Definition of signed numbers :

- The numbers shown in Fig. 1.11.1 contain a sign bit followed by magnitude bits.
- Numbers represented in this form are called as **sign-magnitude numbers** or only sign-numbers.

1.11.1 Range of Sign-Magnitude Numbers :

- The unsigned 8-bit numbers cover the decimal range of 0 to 255.
- But in the sign magnitude numbers, the MSB is utilized for representing the sign.

TechKnowledge Publications

– Therefore the range gets modified. (as there are only 7 bits left to represent the magnitude).

– For an 8-bit sign-magnitude number, the largest negative number is (– 127) given by,

1	1	1	1	1	1	1	1

$= (-127)_{10}$

(C-6369)

– And the largest positive number is (+ 127) given by,

0	1	1	1	1	1	1	1

$= (+127)_{10}$ **(C-6369)**

In this way the range of sign-magnitude, 8-bit binary numbers is modified to decimal (– 127) to (+127) from 0 to 255.

– The largest magnitude is 127, which is approximately half of the largest magnitude obtained for unsigned binary numbers.

– With the sign-magnitude numbers, we can use the 8-bit arithmetic as long as the input data range falls in decimal – 127 to + 127.

– It is still necessary to check all the sums for overflow.

– If the magnitude of data is greater than 127 then 16 bit arithmetic should be used.

– With 16-bit numbers the range of sign-magnitude numbers extends from decimal (– 32,767) to (+ 32,767).

Advantages of sign-magnitude numbers :

1. The main advantage is their simplicity.
2. We can easily find the magnitude by deleting the sign bit.

Disadvantage :

– The sign-magnitude numbers have a limited use because they require complicated circuits.

– These numbers are often used in analog-to-digital (A to D) converters.

1.12 1's and 2's Complements of a Binary Number :

– The 1's and 2's complement of a binary number are important because we can use them for representation of negative numbers.

1.12.1 1's Complement :

Definition :

– The 1's complement of a number is found by inverting all the bits in that number. This is called as taking complement.

– This complemented value represents the negative of the original number. The 1's complement system is very easy to implement merely using inverters.

Ex. 1.12.1 : Obtain the 1's complement of the following numbers : $(1010)_2$, $(11010101)_2$.

Soln. :

Given number :	1010	1101 0101
1's complement :	0101	0010 1010

(C-6279(a))

Ex. 1.12.2 : Convert following binary numbers to 1's complement :

(a) $(1101)_2$ (b) $(1011)_2$

Soln. :

(a) Given number : 1 1 0 1

1's complement : 0 0 1 0 **...Ans.**

(b) Given number : 1 0 1 1

1's complement : 0 1 0 0 **...Ans.**

1.12.2 Representation of Positive and Negative Numbers using 1's Complement :

– Positive number in 1's complement form are represented in the same way as the positive sign magnitude numbers. That means $(+5)_{10}$ can be represented as 0101.

– But the negative numbers are the 1's complement of the corresponding positive numbers.

– For example $(-5)_{10}$ is represented as follows :

$N = (-5)_{10}$ ..... Given number

$(+5)_{10} = \underbrace{0}_{\text{Sign}}\ \underbrace{1\ 0\ 1}_{\text{Magnitude}}$ ..... In sign magnitude form

(C-69) $\therefore N = 1\ 0\ 1\ 0$ ..... 1's complement form

– Thus $(-5)_{10}$ is represented as 1010 in the 1's complement form.

Range of 1's complement number system :

1. From the discussion done till now, the maximum positive number in the 1's complement form for a 4 bit number is 0111.

∴ For n = 4 the maximum positive number is 7. i.e. $[2^{n-1} - 1]$

2. The maximum negative number is represented as 1 1 1 1 .

∴ For n = 4 the maximum negative number is – 7.

i.e. $[2^{n-1} - 1]$.

> ∴ Range of 1's complement number system is given by,
>
> Maximum positive number ⇒ $(2^{n-1} - 1)$
>
> Maximum negative number ⇒ $-(2^{n-1} - 1)$

– Table 1.12.1 shows the full range of 4 bit numbers in the 1's complement form.

(C-6111) Table 1.12.1

Decimal number	1's Complement	
+ 7	0 1 1 1	Positive numbers in their normal form.
+ 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	
– 0	1 1 1 1	Negative numbers in their 1's complement form.
– 1	1 1 1 0	
– 2	1 1 0 1	
– 3	1 1 0 0	
– 4	1 0 1 1	
– 5	1 0 1 0	
– 6	1 0 0 1	
– 7	1 0 0 0	

– One of the difficulties in using 1's complement is its representation of zero.

– As seen from Table 1.12.1 both 0 0 0 0 and 1 1 1 1 represent zero. The [0 0 0 0] is called as positive zero and [1 1 1 1] is called as the negative zero.

1.12.3 2's Complement :

Definition :

– The 2's complement of a binary number is obtained by adding 1 to the LSB of 1's complement of that number.

∴ 2's complement = 1's complement + 1

– The method of 2's complement arithmetic is commonly used in computers to handle negative numbers.

Ex. 1.12.3 : Obtain the 2's complement of $(1011\ 0010)_2$.

Soln. :

1. Given number	:	1 0 1 1 0 0 1 0
2. 1's complement	:	+ 0 1 0 0 1 1 0 1
3. Add 1 to 1's complement	:	1
4. 2's complement	:	0 1 0 0 1 1 1 0

(C-6279)

– Hence the 2's complement of $(1011\ 0010)_2$ is $(0100\ 1110)_2$.

Ex. 1.12.4 : Find out 2's complement of following numbers :

1. $(110110)_2$ 2. $(010101)_2$

Soln. :

1. 2's complement of $(110110)_2$:

Given number	:	1 1 0 1 1 0
1's complement	:	0 0 1 0 0 1
Add 1	+	1
Carry :		1
Result	:	0 0 1 0 1 0

(C-6572)

∴ 2's complement of $(110110)_2$ is $(001010)_2$ **...Ans.**

2. 2's complement of $(010101)_2$:

Given number	:	0 1 0 1 0 1
1's complement	:	1 0 1 0 1 0
Add 1	+	1
Result	:	1 0 1 0 1 1

(C-6573)

∴ 2's complement of $(010101)_2$ is $(101011)_2$ **...Ans.**

1.12.4 Representation of Positive and Negative Numbers using 2's Complement :

– Positive numbers in 2's complement form are represented the same way as in the sign-magnitude and 1's complement form.

– For example $(+\ 6)_{10}$ is represented as 0110 in the 2's complement form.

– Negative numbers are the 2's complements of the corresponding positive numbers.

– For example $(-\ 6)_{10}$ is represented in the 2's complement form as follows :

TechKnowledge Publications

$N = (-6)_{10}$ ...given

$(+6)_{10} = \underbrace{0\ 1\ 1\ 0}_{\text{Invert all bits and add 1}}$ **(C-6112)**

$1\ 0\ 1\ 0$ ... 2's complement of $(-6)_{10}$

– 2's complement is another method of storing negative values. In a microcomputer the positive and negative numbers are represented with the help of 2's complement.

– The representation of four bit (+) and (–) numbers using 2's complement is shown in Table 1.12.2.

– The range of a 4 - bit sign-magnitude number extends from decimal (– 7) to (+ 7). The $(0)_{10}$ and positive numbers are represented by normal binary numbers in Table 1.12.2.

(C-6114) Table 1.12.2 : Two's complement numbers

Decimal	Signed binary	2's complement	
+7	0111	0111	Positive numbers with normal representation
+6	0110	0110	
+5	0101	0101	
+4	0100	0100	
+3	0011	0011	
+2	0010	0010	
+1	0001	0001	
0	0000	0000	Unique zero
–1	1001	1111	Negative numbers in the 2's complement notation
–2	1010	1110	
–3	1011	1101	
–4	1100	1100	
–5	1101	1011	
–6	1110	1010	
–7	1111	1001	
–8	1000	1000	

– But the negative numbers are represented in the 2's complement form.

– However note that the MSB (sign bit) is not changed while obtaining the 2's complement. For example 2's complement of $(-6)_{10}$ is obtained as follows :

$(-6)_{10} = (1\ 1\ 1\ 0)_2$

Its 2's complement = $\boxed{1}\ 0\ 1\ 0$ **(C-6113)**

No change

– Similarly the 2's complement of other negative numbers are obtained.

Important observations from Table 1.12.2 :

Two's complement numbers have some interesting characteristics. They are as follows :

1. There is one unique 0.
2. The two's complement of 0 is 0.
3. The MSB of a sign-magnitude number cannot be used to express quantity. It can only be used as a sign bit.
4. There are eight negative integers, seven positive integers and one zero, making a total of 16 unique numbers.

∴ In an n bit 2's complement system, there will always be,

- $2^{n-1} - 1$ positive integers
- 2^{n-1} negative integers
- and one 0

Total 2^n unique states

(C-6115)

5. To convert a negative number to a positive number, find its 2's complement.
6. 2's complement of a number results in the original number itself. For example if the given number is $(4)_{10} = (0\ 1\ 0\ 0)_2$ then its 2's complement is (1 1 0 0) and the 2's complement of (1 1 0 0) is (0 1 0 0) i.e. $(4)_{10}$.

1.12.5 Signed Complement Numbers :

– When arithmetic operations are carried out in a computer, it is more convenient to use a different system for representing the negative numbers.

– This system is called as the signed – complement system.

– In this system a negative number is represented by its complement.

– In the signed – magnitude system a number is negated by changing its sign, but the signed complement system will negate the number by taking its complement.

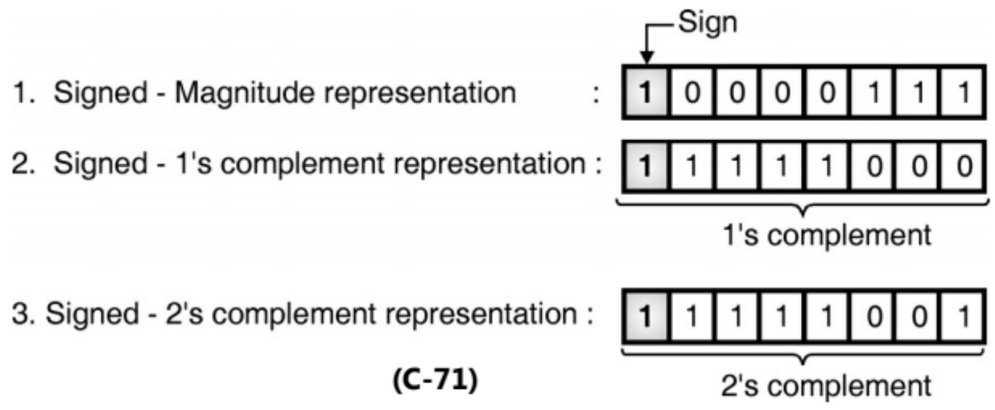
– We know that a positive number always starts with a zero (plus) in its MSB position, therefore the complement will always start with a 1, indicating a negative number.

– The signed – complement system can use either 1's complement or 2's complement, but the 2's complement is most commonly used.

– As an example let us see how to represent – $(7)_{10}$ using different techniques using 8 binary bits.

– In the signed-magnitude representation, the left most 1 represents (–) sign and the remaining seven bits represent the magnitude 7.

Tech Knowledge Publications



(C-71)

- In the signed 1's complement representation, – 7 is obtained by complementing all the bits of + 7 including the sign bit.
- In the signed 2's complement, – 7 is obtained by obtaining the 2's complement of + 7, including the sign bit.
- Table 1.12.3 shows all the possible 4 bit signed binary numbers in the three representations discussed now.

(C-8115) Table 1.12.3 : Signed binary numbers

Decimal	Signed 2's complement	Signed 1's complement	Signed magnitude
+7	0 1 1 1	0 1 1 1	0 1 1 1
+6	0 1 1 0	0 1 1 0	0 1 1 0
+5	0 1 0 1	0 1 0 1	0 1 0 1
+4	0 1 0 0	0 1 0 0	0 1 0 0
+3	0 0 1 1	0 0 1 1	0 0 1 1
+2	0 0 1 0	0 0 1 0	0 0 1 0
+1	0 0 0 1	0 0 0 1	0 0 0 1
+0	0 0 0 0	0 0 0 0	0 0 0 0
–0	–	1 1 1 1	1 0 0 0
–1	1 1 1 1	1 1 1 0	1 0 0 1
–2	1 1 1 0	1 1 0 1	1 0 1 0
–3	1 1 0 1	1 1 0 0	1 0 1 1
–4	1 1 0 0	1 0 1 1	1 1 0 0
–5	1 0 1 1	1 0 1 0	1 1 0 1
–6	1 0 1 0	1 0 0 1	1 1 1 0
–7	1 0 0 1	1 0 0 0	1 1 1 1
–8	1 0 0 0	–	–

Observations :

1. Table 1.12.3 shows that the positive numbers with all the three representations are identical and they have a 0 in their left most position.
2. The signed 2's complement system has only one representation for 0 which is always positive but the other two systems have either a positive zero or a negative zero.
3. All the negative numbers have a 1 in the leftmost bit position. This is how we can distinguish the negative numbers from the positive ones.
4. With four bits we can represent 16 binary numbers. But in the signed magnitude and 1's complement systems there are eight positive and eight negative numbers including two zeros. In 2's complement system there are eight positive numbers including one zero and eight negative numbers.

- The signed magnitude system is awkward when used in computer arithmetic because the sign and magnitude need to be handled separately.
- Therefore the signed complement system is preferred in computer arithmetic.
- The 1's complement creates some difficulties and is generally not used for the arithmetic operations.
- But it is useful as a logical operation because the change of 0 to 1 and 1 to 0 is equivalent to logical complement operation.
- The signed 2's complement system is commonly used in the computer arithmetic.

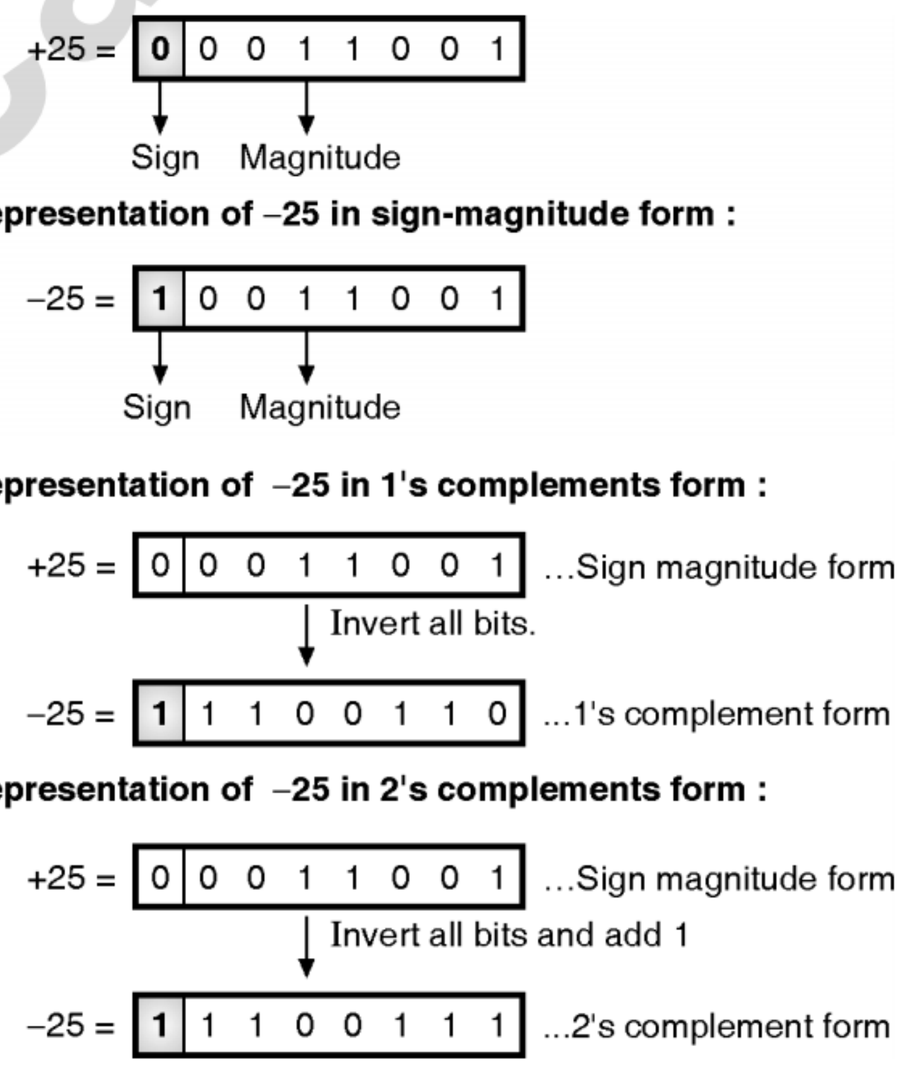
Ex. 1.12.5 : Express the decimal numbers +25 and – 25 in the 8 bit sign magnitude, 1's complement and 2's complement forms.

Soln. :

Representation of +25 :

+ 25 is represented in the sign magnitude, 1's complement and 2's complement forms as follows :

(C-6346)



Ex. 1.12.6 : Represent +40 and –40 decimal numbers using 2's complement method.

Soln. :

Representation of + 40 :

+ 40 is represented in sign magnitude and 2's complement form as follows :

$(+ 40)_{10}$ = 00101000 **...Ans.**

Representation of – 40 in 2's complement form :

(C-3223)

+ 40 = | 0 | 0 1 0 1 0 0 0 | ..sign magnitude form

↓ Invert all bits and add 1

– 40 = | 1 | 1 0 1 1 0 0 0 | ..2's complement form

1.12.6 Addition of Signed Magnitude Numbers :

- In the previous section we have learnt how to represent signed numbers in three different forms. In this section, we will discuss the signed numbers.
- The 2's complement form of representation is the most widely used form in computers and microprocessor – based systems.
- The addition of two numbers in signed-magnitude system should be carried out as per the rules of ordinary arithmetic.
- That means if the signs of numbers A and B are same, then we add their magnitudes and give the sum their common sign.
- If the signs are different, then we subtract the smaller number from the bigger one and give the sign of the larger number to the result.
- The same procedure applies to the binary numbers in the signed magnitude representation.
- For example (+ 16) + (– 25) = – (25 – 16) = – 9 and this subtraction is performed by subtracting the smaller magnitude of 16 from the larger magnitude of 25 and giving the sign of larger number to the subtraction.
- This subtraction requires the comparison of signs and magnitudes of the two numbers.
- The same procedure is applicable to the binary numbers in signed magnitude representation.

Addition in the signed complement form :

- The rule for adding numbers in signed complement form does not need comparison or subtraction but it requires only the addition.
- This procedure is very simple and can be stated as follows :

Procedure :

- The addition of two signed binary numbers with negative numbers represented in signed 2's complement form is obtained from the addition of two numbers, including the sign bits.
- A carry out of the sign bit position is discarded. This procedure can be explained with the help of following examples :

Case 1 : Addition of both positive numbers :

A = + 7 0 0 0 0 0 1 1 1

B = + 12 + 0 0 0 0 1 1 0 0

19 0 0 0 1 0 0 1 1

Sign bit — Answer

(C-6116)

The answer is positive (sign bit is 0) and equal to 00010011 i.e. 19.

Case 2 : Smaller number is negative :

- Let A = – 7 and B = + 12 and their addition is to be performed.
- Express (– 7) in the signed 2's complement form as follows :

+ 7 = 0 0 0 0 0 1 1 1 ← signed magnitude form

– 7 = 1 1 1 1 1 0 0 1 ← 2's complement of + 7

- Perform the addition

(C-6117)

Carry 1 1 1 1

A = – 7 1 1 1 1 1 0 0 1

B = + 12 + 0 0 0 0 1 1 0 0

+ 5 ~~1~~ 0 0 0 0 0 1 0 1 ← Answer

Discard final carry

- The final carry is discarded to obtain the correct answer.

Sign (+) | 0 | 0 0 0 0 1 0 1 | = +5

Magnitude 5

(C-6118)

Case 3 : Bigger number negative :

- Let A = + 7 and B = – 12

Carry

+ 7 0 0 0 0 0 1 1 1

– 12 + 1 1 1 1 0 1 0 0 ← 2's complement of +12

– 5 1 1 1 1 1 0 1 1 ← Answer negative and in 2's complement form

(C-6347)

- Therefore bring the answer in it's true form by subtracting 1 from the answer and then inverting all the bits to get the final answer as follows :
- The answer is

sign (–) | 1 | 0 0 0 0 1 0 1 | = –5 **(C-7862)**

Case 4 : Both numbers negative :

(C-6119)

```
Carry     1 1 1
– 7       1 1 1 1 1 0 0 1
– 12    + 1 1 1 1 0 1 0 0
– 19    ~~1~~ 1 1 1 0 1 1 0 1  ← Answer
Discard carry ↑  2's complement of
                 19 i.e. –19
```

1.12.7 Arithmetic Subtraction (Subtraction of Signed Numbers) :

– Subtraction of two signed binary numbers when the negative numbers are in the 2's complement form can be performed as follows :

> Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including the sign bit). A carry out of the sign bit position should be discarded.

– We can use this procedure because a subtraction is equivalent to an addition if the sign of the subtrahend is changed. This is demonstrated as follows :

Minuend → ± A

Subtrahend → – (+ B) i.e. (± A) – (+ B) = (± A) + (– B)

(± A) + (– B)

And (± A) – (– B) = (± A) + (+ B)

– Consider the subtraction (– 7) – (– 12) = + 5 in binary with eight bits. The numbers (–7) and (– 12) are expressed in the 2's complement form as (11111001) and (1111 0100) respectively.

– The subtraction of these numbers is changed to addition by taking the 2's complement of subtrahend (– 12) to (+ 12).

– The 2's complement of – 12 = (0000 1100). Then the addition is performed as follows :

```
–7      1 1 1 1 1 0 0 1
+12     0 0 0 0 1 1 0 0
+5    | 1 | 0 0 0 0 0 1 0 1
Discard ↑   ↑ Answer is + 5
```

(C-7863)

Discarding the final carry we get the answer of + 5.

Conclusion :

– The binary numbers in the signed complement system are added and subtracted by the same basic addition and subtraction rules as unsigned numbers.

– Hence the computers need only one common hardware circuit to handle the addition and subtraction.

– The user must interprete the results of such addition or subtraction differently depending on whether it is assumed that the numbers are signed or unsigned.

Review Questions

Q. 1 Explain the following terms :

1. Product term 2. Sum term

Q. 2 Explain the term SOP and POS related to Boolean function.

Q. 3 Convert the equation into standard POS form $Y = (A + B)(A + C)(B + \overline{C})$.

Q. 4 State the disadvantages of algebraic method of simplification.

Q. 5 Write the standard SOP equation for the truth table shown in Table 1.

Table 1

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

Q. 6 For the same truth table write the standard POS expression.

Q. 7 Explain the K-map reduction technique.

Q. 8 Draw the structure of four input K-map.

Q. 9 Why is gray code followed for K-map ?

Q. 10 Define a redundant group.

Q. 11 State the importance of don't care terms in K-map.

Q. 12 Draw the structure of four variable K-map to represent the standard POS form.

Q. 13 Solve the following with K-maps :

1. $f(A, B, C) = \Sigma m(0, 1, 3, 4, 5)$
2. $f(A, B, C) = \Sigma m(0, 1, 2, 3, 6, 7)$

Q. 14 Explain the different methods used to simplify the Boolean function.

Q. 15 Define 1's complement.

Q. 16 Define 2's complement.

Q. 17 How to represent the positive and negative numbers using 2's complement .

Q. 18 How to represent the positive and negative numbers using 1's complement.

Q. 19 State the range of sign-magnitude numbers.

Q. 20 State features of unsigned numbers.

Q. 21 Write a short note on : Unsigned binary numbers.

Q. 22 Write a short note on : Sign-magnitude numbers.

❑❑❑

Tech Knowledge Publications

Unit II

Chapter 2

Combinational Logic Design

Technical Knowledge Publications

Syllabus

Code converter - BCD, Excess-3, Gray code, Binary code, Half-adder, Full adder, Half subtractor, Full subtractor, Binary adder (IC 7483), BCD adder, Look ahead carry generator, Multiplexers (MUX) : MUX (IC 74153, 74151), Cascading multiplexers, Demultiplexers (DEMUX) - Decoder (IC 74138, IC 74154), Implementation of SOP and POS using MUX, DMUX, Comparators (2 bit), Parity generators and checker. Exemplar / Case study : Combinational logic design of BCD to 7 segment display controller.

Chapter Contents

2.1	Introduction to Combinational Circuits	2.12	Use of Multiplexers in Combinational Logic Design
2.2	Code Converters	2.13	Demultiplexers
2.3	Binary Adders and Subtractors	2.14	Types of Demultiplexers
2.4	The n-Bit Parallel Adder	2.15	Demultiplexer Tree
2.5	n-bit Parallel Subtractor (Use of Adder as Subtractor)	2.16	Encoders
2.6	BCD Addition	2.17	Priority Encoder
2.7	Magnitude Comparators	2.18	Decoder
2.8	Multiplexer (Data Selector)	2.19	Parity Generators / Checkers
2.9	Types of Multiplexers	2.20	Examples for Practice
2.10	Study of Different Multiplexer ICs	2.21	Case Study : Combinational Logic Design of BCD to 7 Segment Display Controller
2.11	Multiplexer Tree/Cascading of Multiplexer		

2.1 Introduction to Combinational Circuits :

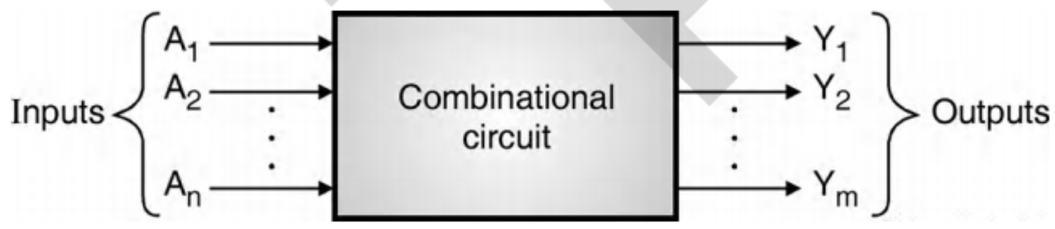
Types of digital circuits :

The digital systems in general are classified into two categories namely :

1. Combinational logic circuits.
2. Sequential logic circuits.

Combinational circuits :

- The combinational circuit is a digital system the output of which at any instant of time, depends only on the levels present at its input terminals.
- The combinational circuits do not use any memory. Hence the previous state of input does not have any effect on the present state of the circuit.
- The sequence in which the inputs are being applied also does not have any effect on the output of a combinational circuit.
- A combinational circuit is a logic circuit the output of which depends only on the combination of the inputs.
- The output does not depend on the past value of inputs or outputs.
- Hence combinational circuits do not require any memory (to store the past values of inputs or outputs).
- The block diagram of a combinational circuit is shown in Fig. 2.1.1.
- A combinational circuit can have a number of inputs and a number of outputs. The circuit of Fig. 2.1.1 has "n" inputs and "m" outputs.



(C-332) Fig. 2.1.1 : Block diagram of a combinational circuit

- A combinational circuit is made up of logic gates. These gates are connected suitably between the inputs and outputs of the combinational circuit.
- A combinational circuit operates in three steps :

1. It accepts n-different inputs.
2. The combination of gates operates on the inputs.
3. "m" different outputs are produced as per requirement.

Examples of combinational circuits :

- Following are some of the examples of combinational circuits :

1. Adders, subtractors.
2. Comparator.
3. Code converters.
4. Encoders, decoders.
5. Multiplexers, demultiplexers.

2.1.1 Analysis of a Combinational Circuit :

- In the analysis problem, the logic diagram of a combinational circuit is given to us.
- We have to obtain the Boolean expression for its output or write a truth table or explain the operation of circuit.

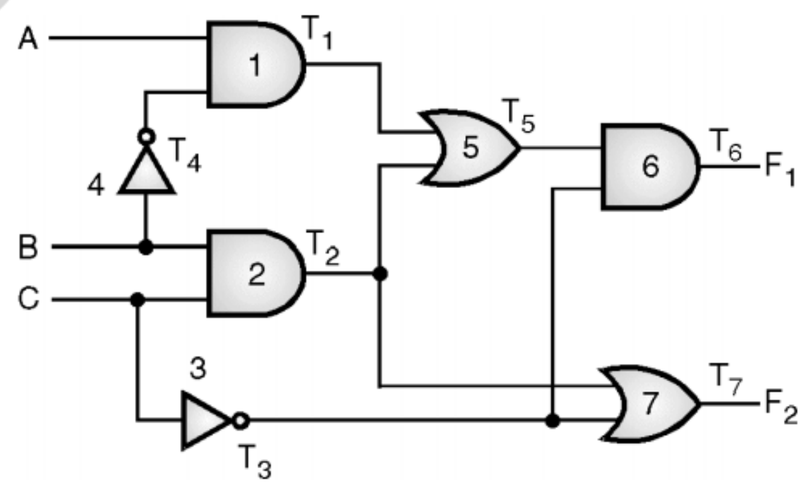
Analysis Procedure :

- The general procedure for the analysis is as follows :

1. Write down the Boolean function for the output of each input gate in the given circuit.
2. Obtain the Boolean expression at all other gates.
3. Write down the Boolean expression for the final output in terms of the input variables.

Illustration :

- Analyze the combinational circuit shown in Fig. 2.1.2.



(C-333) Fig. 2.1.2 : Given combinational circuit

Analysis :

Step 1 : Boolean expressions for the outputs of all the input gates :

- Input gates are 1, 2, 3 and 4. The Boolean expressions for their outputs are as follows :

$$T_1 = A\overline{B} \qquad T_3 = \overline{C}$$

$$T_2 = BC \qquad T_4 = \overline{B}$$

Step 2 : Boolean expressions for remaining gates :

$$T_5 = T_1 + T_2 = A\overline{B} + BC$$

$$T_6 = F_1 = T_5 \cdot T_3 = (A\overline{B} + BC)\,\overline{C}$$

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

$$T_7 = F_2 = T_2 + T_3 = BC + \overline{C}$$

Step 3 : Boolean expressions for final outputs :

$$F_1 = T_6 = A\overline{B}\,\overline{C}$$

$$F_2 = T_7 = \overline{C} + BC$$

Step 4 : Write the truth table :

(C-8244) Table 2.1.1

Inputs			Outputs						
A	B	C	F_1	F_2	T_1	T_2	T_3	T_4	T_5
0	0	0	0	1	0	0	1	1	0
0	0	1	0	0	0	0	0	1	0
0	1	0	0	1	0	0	1	0	0
0	1	1	0	1	0	1	0	0	1
1	0	0	1	1	1	0	1	1	1
1	0	1	0	0	1	0	0	1	1
1	1	0	0	1	0	0	1	0	0
1	1	1	0	1	0	1	0	0	1

– Note that the outputs of all the gates (T_1 to T_5) are first written in the truth table for various combinations of inputs and then the values of F_1 and F_2 are entered using the following equations,

$$F_1 = T_3 \cdot T_5 \text{ And } F_2 = T_2 + T_3$$

2.1.2 Design of Combinational Logic using Statements :

– The steps involved in designing a combinational logic are as follows :

Steps to be followed :

Step 1 : You will be given a problem.

Step 2 : Determine the number of inputs and outputs and assign letter symbols to input and output variables. For example F_1, F_2 … for outputs and A, B, C. …. for the inputs.

Step 3 : Prepare a truth table relating the inputs and outputs.

Step 4 : Write K-map for each output in terms of inputs and obtain the simplified Boolean expression for each output.

Step 5 : Draw the logic diagram (combinational circuit).

Ex. 2.1.1 : A circuit has four inputs and two outputs. One of the outputs is high when majority of inputs are high. The second output is high only when all inputs are of same type. Design the combinational circuit.

OR

Design a logic circuit which has three inputs A, B, C and gives a high output when majority of inputs is high.

Soln. :

Step 1 : Assign symbols to input and output variables :

– Let the four inputs be A, B, C, D and the two outputs be Y_1 and Y_2.

Step 2 : Write the truth table :

– The truth table is as given in Table P. 2.1.1.

(C-8245) Table P. 2.1.1 : Truth table relating the inputs and outputs

Decimal	Inputs				Outputs	
	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

– From the truth table we note the following things.

– $Y_1 = 1$ when number of 1 inputs is higher than the number of 0 inputs.

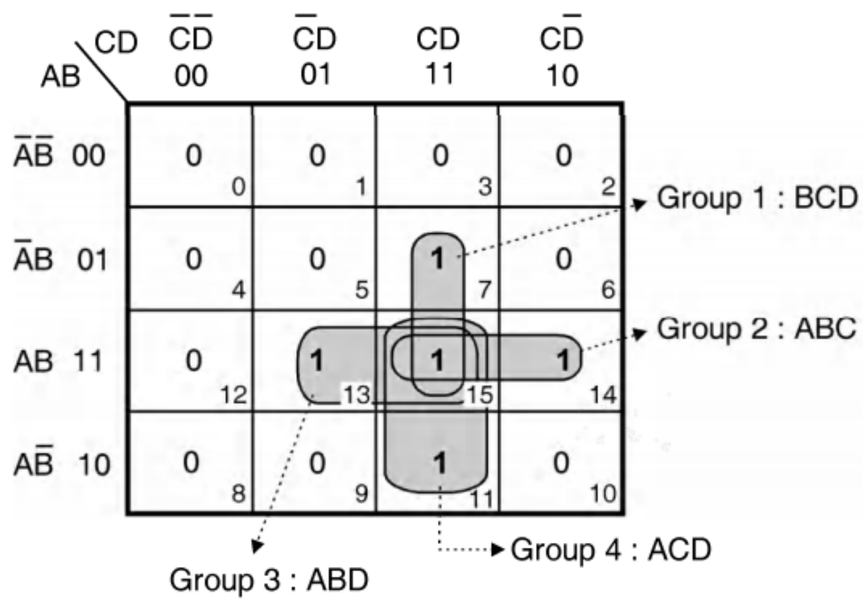
– $Y_2 = 1$ when A = B = C = D.

Step 3 : Write K-map for each output and get simplified expression :

– K-maps for the two outputs and the corresponding simplified Boolean expressions are given in Figs. P. 2.1.1(a) and (b).

Tech Knowledge Publications

For Output Y_1

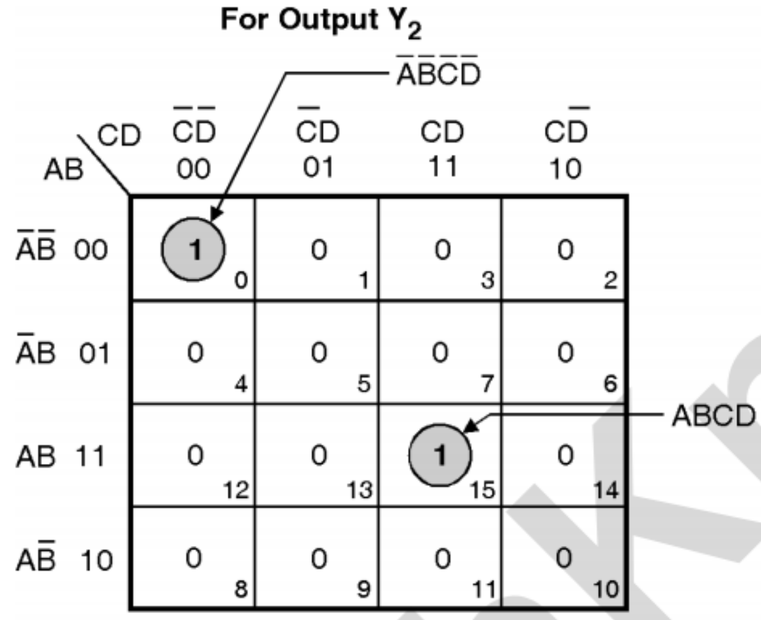


Simplified expression

$Y_1 = BCD + ABC + ABD + ACD$

(C-334) Fig. P. 2.1.1(a) : K-map and simplification for Y_1

For Output Y_2



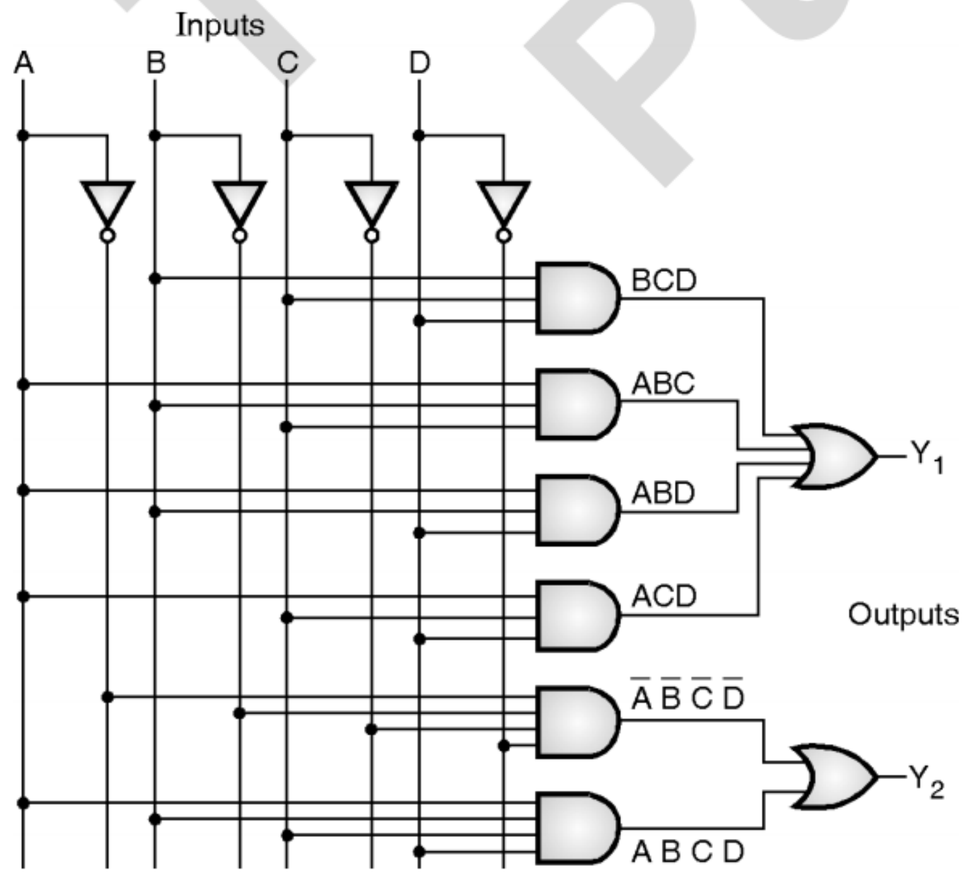
Simplified expression

$Y_2 = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\,B\,C\,D$

(C-334) Fig. P. 2.1.1(b) : K-map and simplification for Y_2

Step 4 : Draw the logic diagram :

– The logic diagram is as shown in Fig. P. 2.1.1(c).



(C-335) Fig. P. 2.1.1(c) : Logic diagram

2.2 Code Converters :

– In this section we are going to convert one type of code into another type.

2.2.1 BCD to Excess 3 Converter :

SPPU : May 06, Dec. 12, Dec. 13

University Questions

Q. 1 Design and implement BCD to Excess-3 code converter using logic gates. Starting with truth table show K-maps and circuit diagram of your design.

(May 06, 8 Marks)

Q. 2 Design 4-bit BCD to excess-3 code converter. Use logic gates as per your design and requirement.

(Dec. 12, 8 Marks)

Q. 3 Design a 4-bit BCD to Excess-3 code converter circuit using minimum number of logic gates.

(Dec. 13, 6 Marks)

– We know that Excess-3 code can be derived from the BCD code by adding 3 to each BCD number.

– For example decimal 13 is represented as 0001 0011 in BCD.

– If we add 3 i.e. (0011 0011) then the corresponding Excess 3 code is 0100 0110.

Step 1 : Write the truth table relating BCD and Excess 3 :

(C-8109) Table 2.2.1 : Truth table relating BCD and Excess-3 codes

Decimal	BCD inputs				Excess-3 output			
	D_3	D_2	D_1	D_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

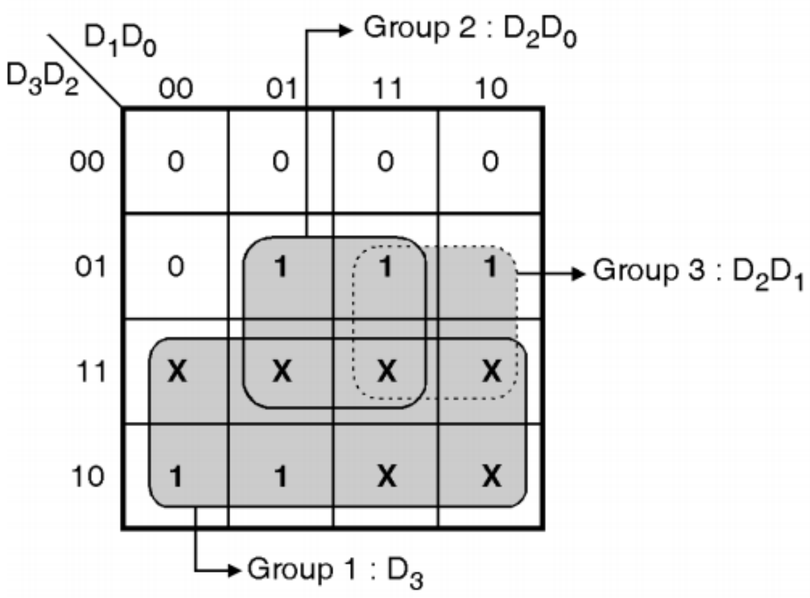
Step 2 : Write K-map for each output and obtain simplified expression :

– Refer Fig. 2.2.1(a) to (d) for the K-maps and corresponding simplified equations.

TechKnowledge Publications

– Don't care conditions (×) have been entered for $D_3 D_2 D_1 D_0$ above 1001.

For output E_3



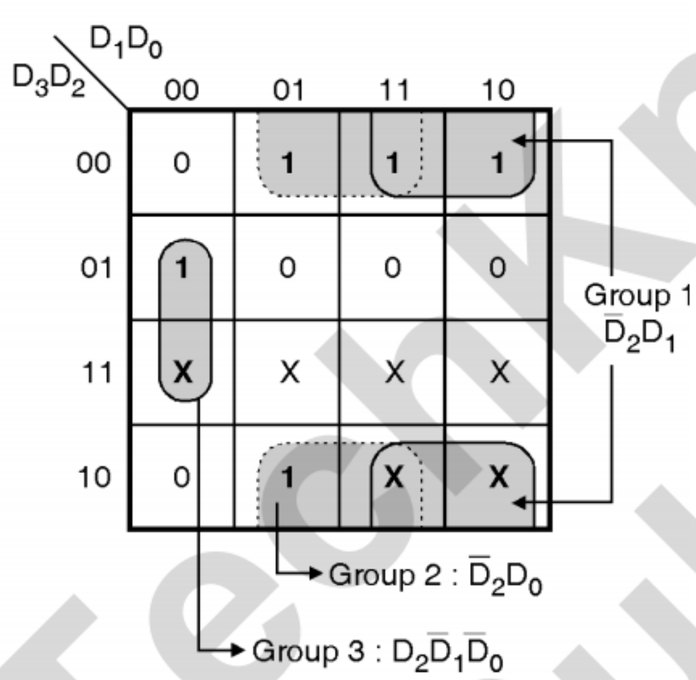
(C-365) Fig. 2.2.1(a) : K-map for E_3

– Simplified equation :

$$E_3 = D_3 + D_2D_0 + D_2D_1$$

$$\therefore E_3 = D_3 + D_2(D_0 + D_1)$$

For output E_2



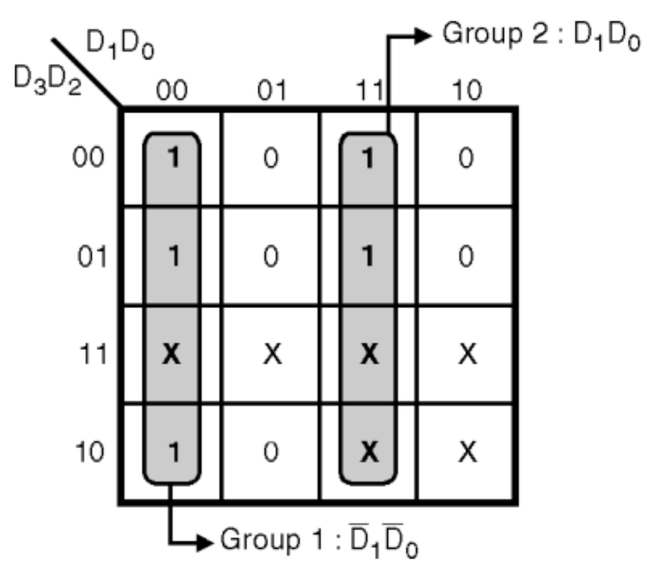
(C-365) Fig. 2.2.1(b) : K-map for E_2

– Simplified equation :

$$E_2 = \overline{D}_2 D_1 + \overline{D}_2 D_0 + D_2 \overline{D}_1 \overline{D}_0$$

$$\therefore E_2 = \overline{D}_2 (D_1 + D_0) + D_2 \overline{D}_1 \overline{D}_0$$

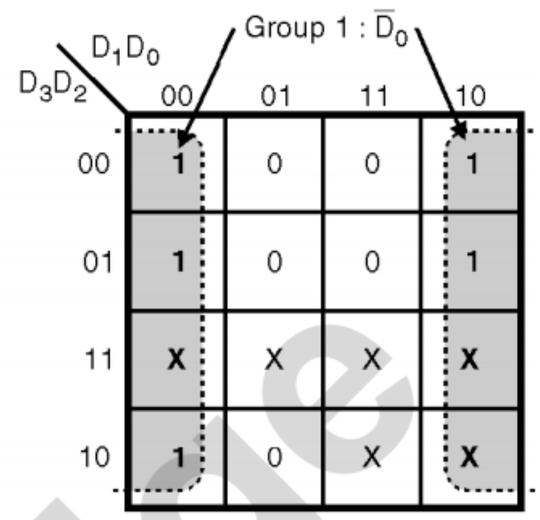
For output E_1



(C-366) Fig. 2.2.1(c) : K-map for E_1

$$\therefore E_1 = \overline{D}_1 \overline{D}_0 + D_1D_0 = (D_1 \odot D_0)$$

For output E_0

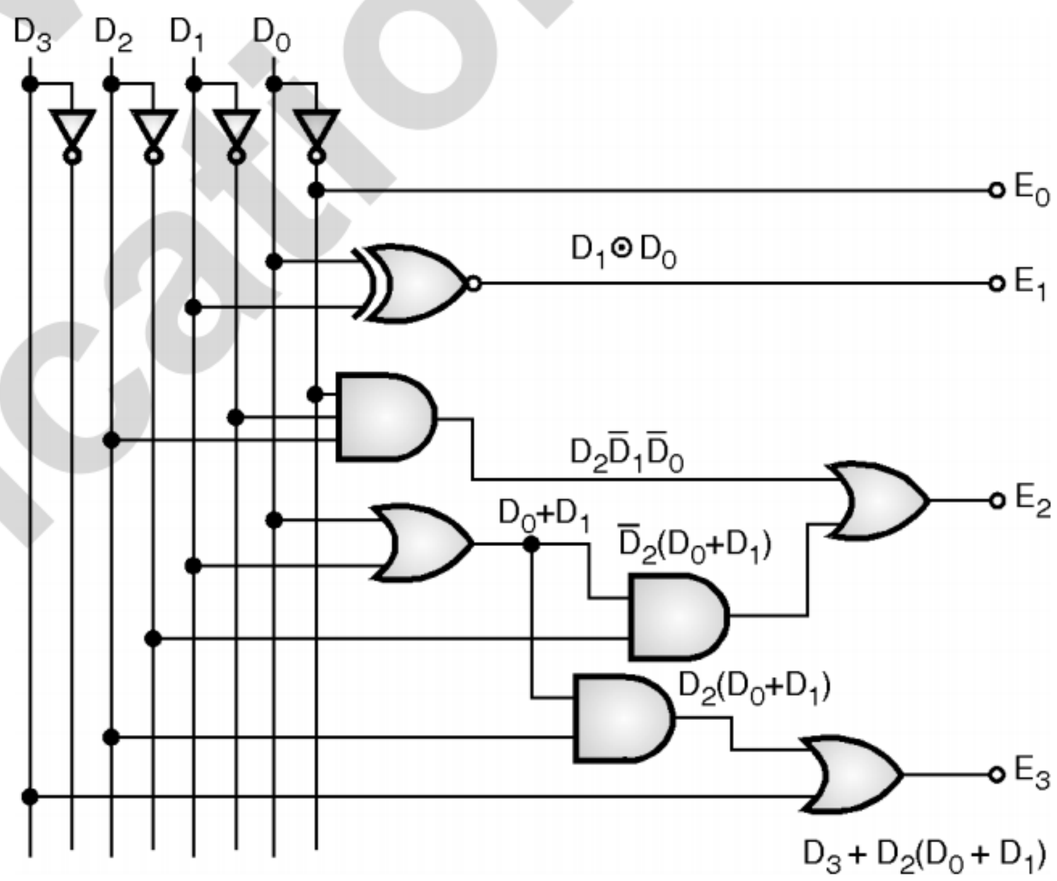


(C-366) Fig. 2.2.1(d) : K-map for E_0

$$\therefore E_0 = \overline{D}_0$$

Step 3 : Implement the BCD to Excess 3 code converter :

– The circuit diagram of BCD to Excess 3 code converter is shown in Fig. 2.2.2.



(C-367) Fig. 2.2.2 : BCD to Excess - 3 code converter

2.2.2 BCD to Gray Code Converter :

SPPU : Dec. 06

University Questions

Q. 1 Design and implement BCD to gray code converter using logic gates. Starting with truth table show K-maps and circuit diagram of your design.

(Dec. 06, 8 Marks)

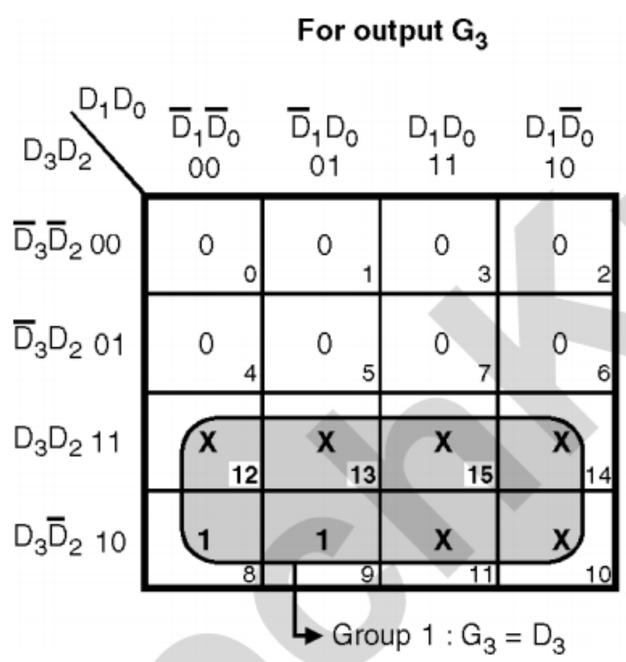
TechKnowledge Publications

Step 1 : Write the truth table relating BCD and gray codes :

(C-8145) Table 2.2.2

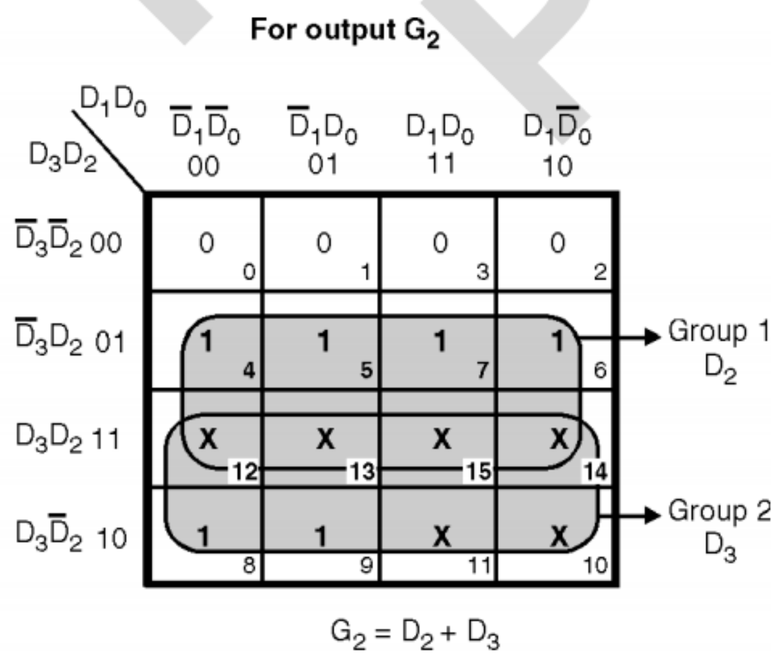
Decimal	BCD inputs				Gray code			
	D_3	D_2	D_1	D_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1

Step 2 : Write K-map for each output and get simplified equation :



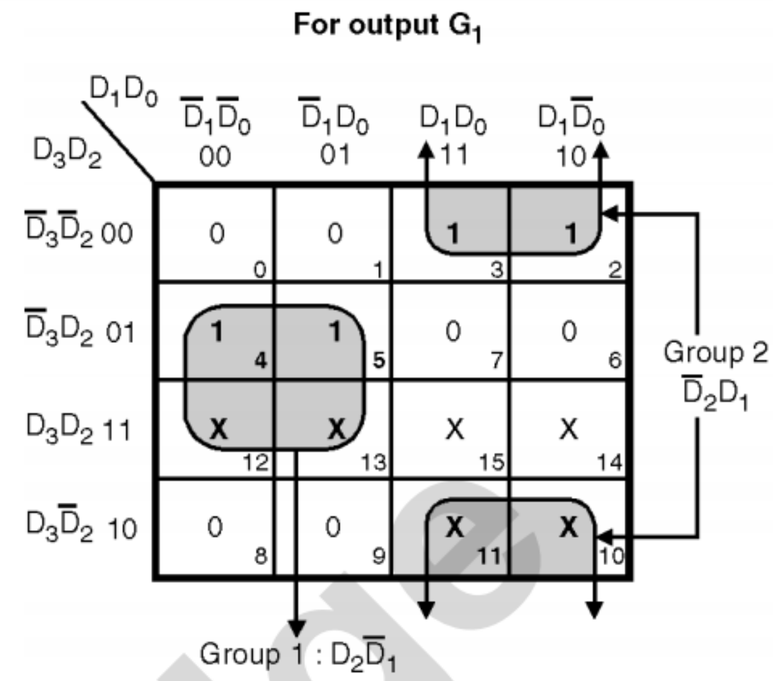
(C-371) Fig. 2.2.3(a) : K-map for G_3

$\therefore\ G_3 = D_3$



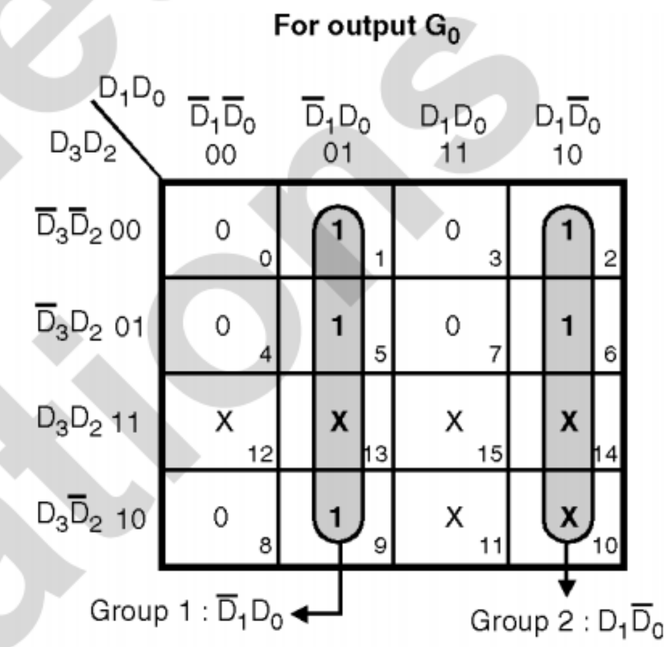
(C-371) Fig. 2.2.3(b) : K-map for G_2

$G_2 = D_2 + D_3$



(C-372) Fig. 2.2.3(c) : K-map for G_1

$\therefore\ G_1 = D_2\overline{D}_1 + \overline{D}_2D_1 = (D_2 \oplus D_1)$

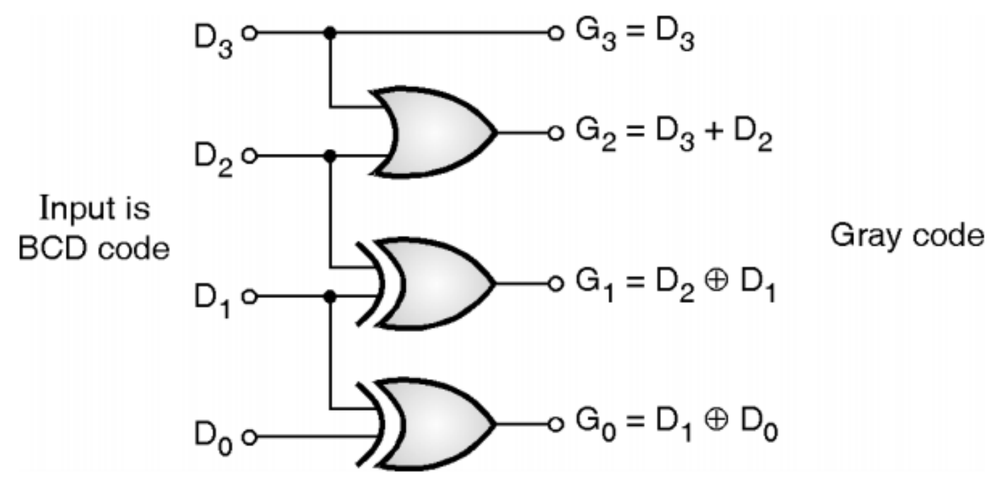


(C-372) Fig. 2.2.3(d) : K-map for G_0

$\therefore\ G_0 = (\overline{D}_1D_0 + D_1\overline{D}_0) = D_1 \oplus D_0$

Step 3 : Realization :

– The BCD to gray converter is shown in Fig. 2.2.4.



(C-373) Fig. 2.2.4 : BCD to Gray code converter

2.2.3 Binary to Gray Code Converter :

SPPU : May 12, May 15, Dec. 17

University Questions

Q. 1 Design 4-bit binary to gray code converter. State the applications of gray code. **(May 12, 8 Marks)**

Q. 2 Design 4-bit binary to gray code converter circuit using logic gates. **(May 15, 4 Marks)**

TechKnowledge Publications

Q. 3 Design and implement Binary to Gray code converter using logic gate. **(Dec. 17, 6 Marks)**

– In gray code only one bit changes at a time.

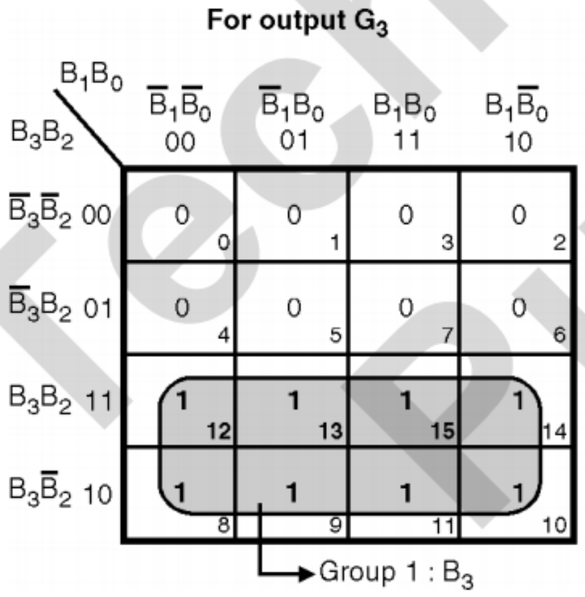
Step 1 : Write the truth table relating binary inputs and gray outputs :

(C-8252) Table 2.2.3 : Truth table relating binary and gray codes

Decimal	Binary inputs				Gray outputs			
	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

Step 2 : Write K-map for each gray output and obtain the simplified expression :

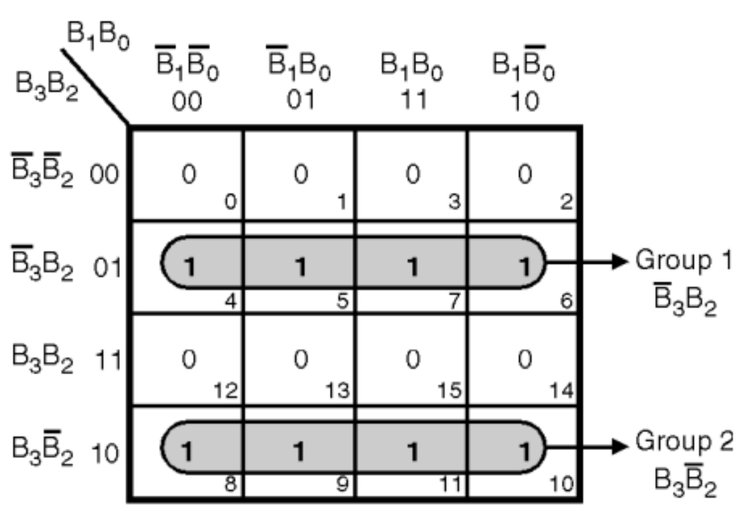
For output G_3



(C-374) Fig. 2.2.5(a) : K-map for G_3

∴ Simplified equation : $G_3 = B_3$

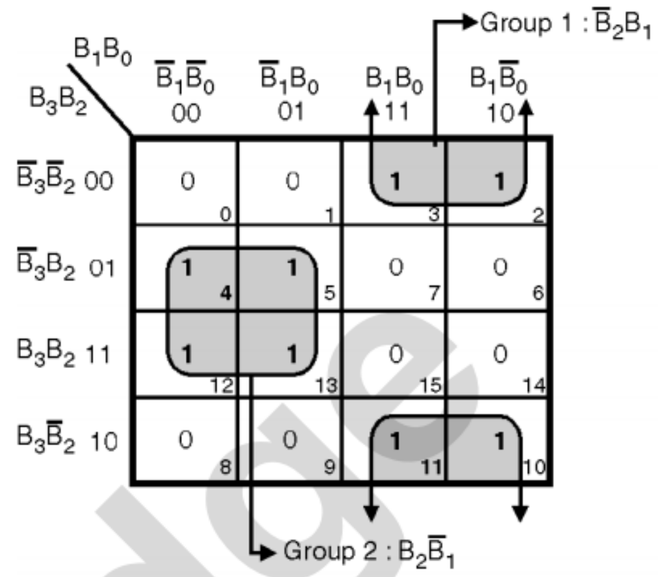
For output G_2



(C-374) Fig. 2.2.5(b) : K-map for G_2

∴ Simplified equation : $G_2 = \overline{B_3}\,B_2 + B_3\overline{B_2} = B_3 \oplus B_2$

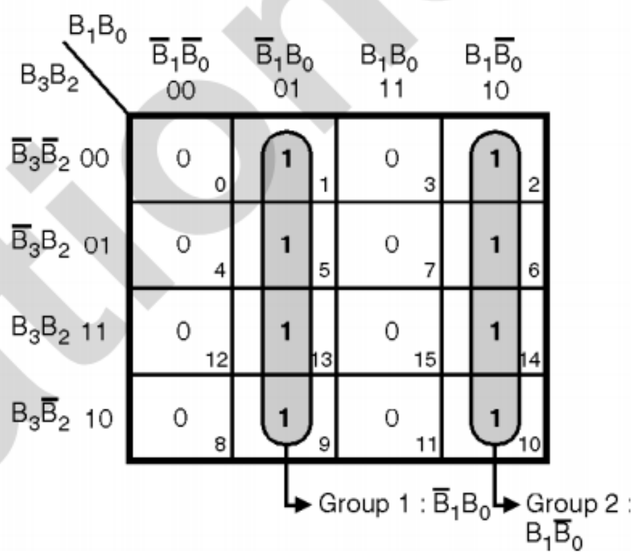
For output G_1



(C-375) Fig. 2.2.5(c) : K-map for G_1

∴ Simplified equation : $G_1 = B_2\overline{B_1} + \overline{B_2}\,B_1 = B_2 \oplus B_1$

For output G_0



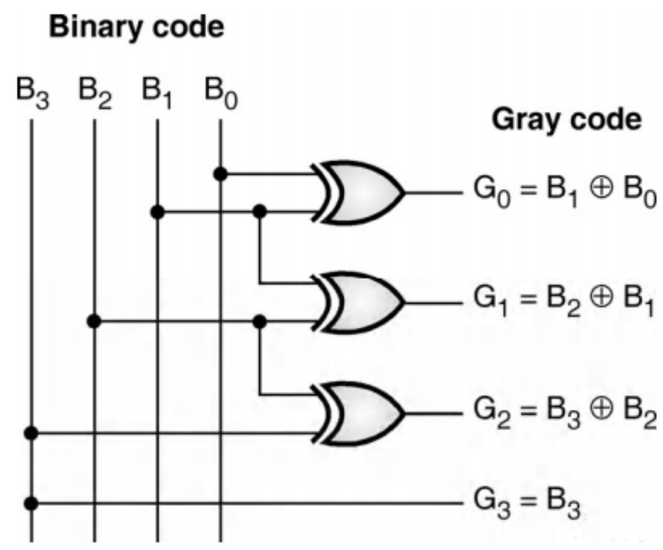
(C-375) Fig. 2.2.5(d) : K-map for G_0

∴ Simplified equation :

$$G_0 = \overline{B_1}\,B_0 + B_1\overline{B_0} = B_1 \oplus B_0$$

Step 3 : Realize Binary to gray code converter :

– Binary to gray code converter is shown in Fig. 2.2.5(e).



(C-376) Fig. 2.2.5(e) : Binary to gray code converter

2.2.4 Gray to BCD Converter :

Ex. 2.2.1 : Convert 4-bit gray code into corresponding BCD code. Show truth table and MSI circuit.

May 10, 6 Marks

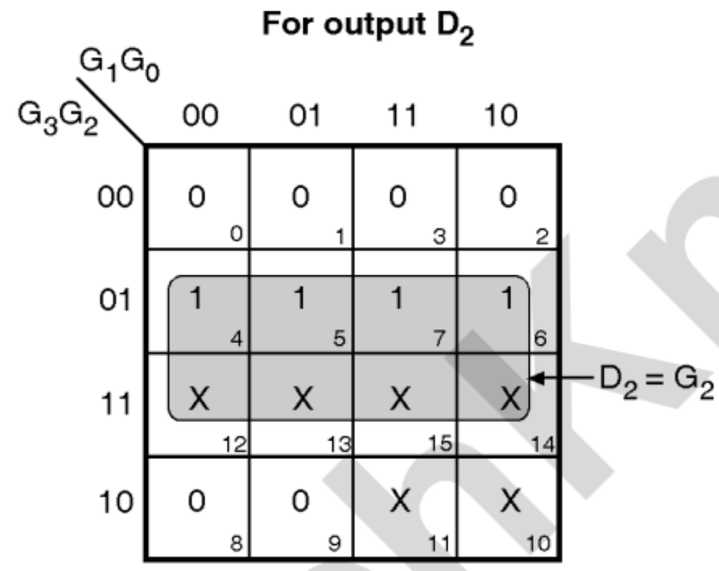
Soln. :

Step 1 : Write the truth table relating gray and BCD codes : **(C-8279)**

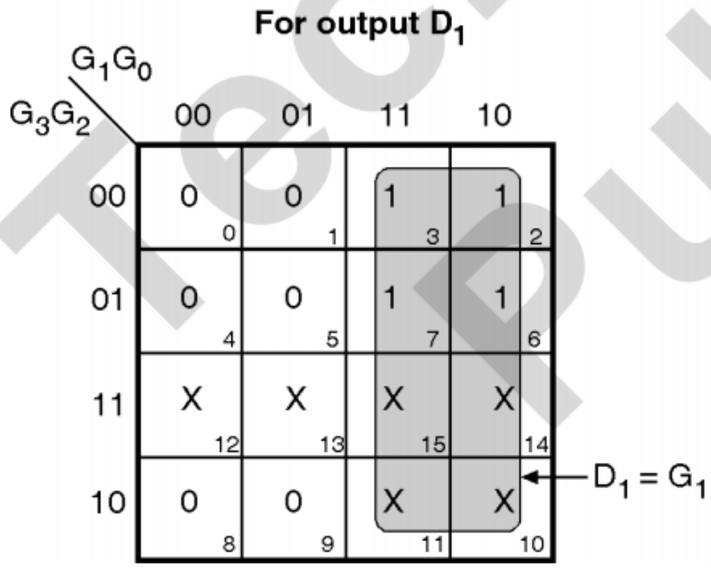
Decimal	Gray inputs				BCD outputs			
	G_3	G_2	G_1	G_0	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1

Step 2 : Write K-maps for each output and get simplified equation :

For output D_2



For output D_1



For output D_3

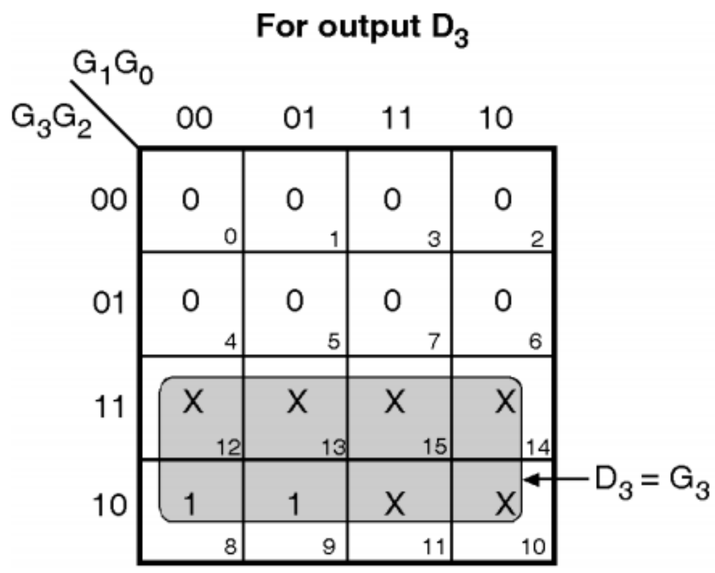
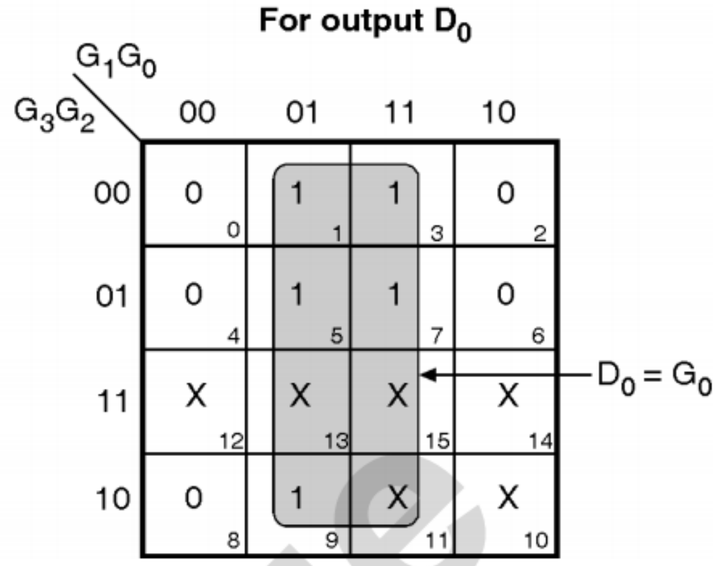


Fig. P. 2.2.1(Contd...)

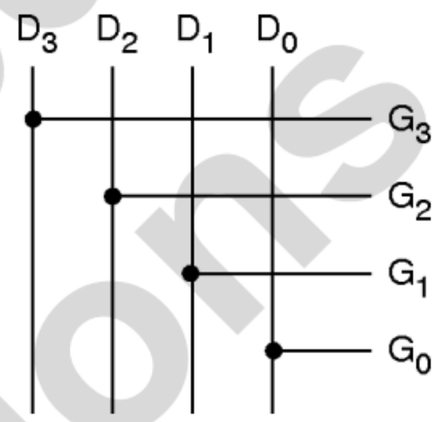
For output D_0



(C-1709) Fig. P. 2.2.1

Step 3 : Realization :

– The gray to BCD converter is shown in Fig. P. 2.2.1(a).



(C-1710) Fig. P. 2.2.1(a)

Ex. 2.2.2 : Design and explain in detail 4-bit gray code to 5-bit BCD code conversion. For this design use K-map reduction and MSI circuit using basic gates. **May 11, 16 Marks**

Soln. :

Step 1 : Write the truth table 4 bit gray to 5 bit BCD code converter :

(C-8280) Table P. 2.2.2

Decimal	Gray inputs				BCD outputs				
	G_3	G_2	G_1	G_0	B_4	B_3	B_2	B_1	B_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	1	0	0	0	1	0
3	0	0	1	0	0	0	0	1	1
4	0	1	1	0	0	0	1	0	0
5	0	1	1	1	0	0	1	0	1
6	0	1	0	1	0	0	1	1	0
7	0	1	0	0	0	0	1	1	1
8	1	1	0	0	0	1	0	0	0
9	1	1	0	1	0	1	0	0	1
10	1	1	1	1	1	0	0	0	0
11	1	1	1	0	1	0	0	0	1
12	1	0	1	0	1	0	0	1	0
13	1	0	1	1	1	0	0	1	1
14	1	0	0	1	1	0	1	0	0
15	1	0	0	0	1	0	1	0	1

TechKnowledge Publications

Step 2 : Write the K-maps and simplify :

For B_4

G_3G_2 \ G_1G_0	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (1)	0 (5)	0 (7)	0 (6)
11	0 (12)	0 (13)	1 (15)	1 (16)
10	1 (8)	1 (9)	1 (11)	1 (10)

G_3G_1

$G_3\overline{G}_2$

$\therefore \quad B_4 = G_3\overline{G}_2 + G_3G_1$

$\therefore \quad B_4 = G_3(G_1 + \overline{G}_2)$

For B_3

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	0	0	0

$G_3G_2\overline{G}_1$

$\therefore \quad B_3 = G_3G_2\overline{G}_1$

For B_2

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

$\overline{G}_3G_2$

$G_3\overline{G}_2\overline{G}_1$

$\therefore \quad B_2 = \overline{G}_3G_2 + G_3\overline{G}_2\overline{G}_1$

For B_1

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	0	0
10	0	0	1	1

$\overline{G}_2G_1$

$\overline{G}_3G_2\overline{G}_1$

$\therefore \quad B_1 = \overline{G}_3G_2\overline{G}_1 + \overline{G}_2G_1$

For output B_0

G_3G_2 \ G_1G_0	$\overline{G}_1\overline{G}_0$ 00	$\overline{G}_1G_0$ 01	G_1G_0 11	$G_1\overline{G}_0$ 10
$\overline{G}_3\overline{G}_2$ 00	0 (0)	1 (1)	0 (3)	1 (2)
$\overline{G}_3G_2$ 01	1 (4)	0 (5)	1 (7)	0 (6)
G_3G_2 11	0 (12)	1 (13)	0 (15)	1 (14)
$G_3\overline{G}_2$ 10	1 (8)	0 (9)	1 (11)	0 (10)

$\overline{G}_3\overline{G}_2\overline{G}_1G_0$

$\overline{G}_3\overline{G}_2G_1\overline{G}_0$

$\overline{G}_3G_2\overline{G}_1\overline{G}_0$

$\overline{G}_3G_2G_1G_0$

$G_3G_2\overline{G}_1G_0$

$G_3G_2G_1\overline{G}_0$

$G_3\overline{G}_2\overline{G}_1\overline{G}_0$

$G_3\overline{G}_2G_1G_0$

(C-1964) Fig. P. 2.2.2 : K-maps

Simplified expression for B_0 :

$$B_0 = \overline{G}_3\,\overline{G}_2\,\overline{G}_1\,G_0 + \overline{G}_3\,G_2\overline{G}_1\,\overline{G}_0 + \overline{G}_3\,\overline{G}_2\,G_1\overline{G}_0$$

$$+ \overline{G}_3\,G_2\,G_1\,G_0 + G_3G_2\overline{G}_1\,G_0 + G_3G_2G_1\,\overline{G}_0$$

$$+ G_3\overline{G}_2\,\overline{G}_1\,\overline{G}_0 + G_3\overline{G}_2\,G_1G_0$$

$$\therefore B_0 = \overline{G}_1\overline{G}_0\underbrace{(\overline{G}_3G_2 + G_3\overline{G}_2)}_{\text{EX-OR}} + \overline{G}_1G_0\underbrace{(\overline{G}_3\overline{G}_2 + G_3G_2)}_{\text{EX-NOR}}$$

$$+ G_1G_0\underbrace{(\overline{G}_3G_2 + G_3\overline{G}_2)}_{\text{EX-OR}} + G_1\overline{G}_0\underbrace{(\overline{G}_3\overline{G}_2 + G_3G_2)}_{\text{EX-NOR}}$$

(C-6178)

$$\therefore B_0 = \overline{G}_1\,\overline{G}_0\,(G_3 \oplus G_2) + \overline{G}_1\,G_0\,(\overline{G_3 \oplus G_2})$$

$$+ G_1G_0\,(G_3 \oplus G_2) + G_1\overline{G}_0\,(\overline{G_3 \oplus G_2})$$

$$\therefore B_0 = (G_3 \oplus G_2)\,(\overline{G}_1\,\overline{G}_0 + G_1G_0)$$

$$+ (\overline{G_3 \oplus G_2})\,(\overline{G}_1\,G_0 + G_1\overline{G}_0)$$

$$\therefore B_0 = (G_3 \oplus G_2)\,(\overline{G_1 \oplus G_0})$$

$$+ (\overline{G_3 \oplus G_2})\,(G_1 \oplus G_0)$$

Let $X = G_3 \oplus G_2$

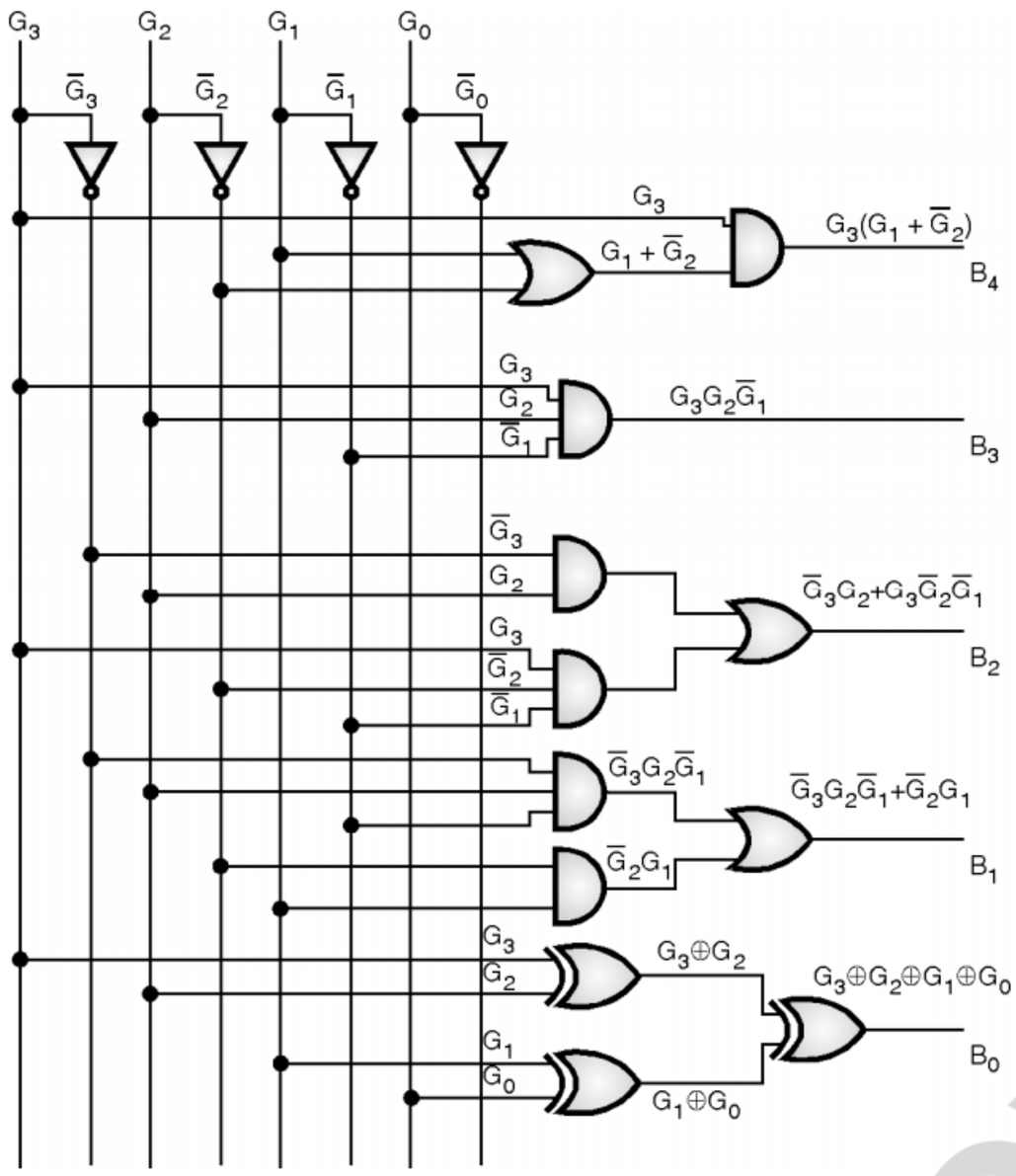
And $Y = (G_1 \oplus G_0)$

$$\therefore B_0 = X\overline{Y} + \overline{X}\,Y = X \oplus Y$$

$\therefore$ Substituting for X and Y we get,

$$B_0 = (G_3 \oplus G_2) \oplus (G_1 \oplus G_0) = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

TechKnowledge Publications



(C-1965) **Fig. P. 2.2.2(a) : Gray to 5 bit BCD code converter**

2.2.5 Excess 3 to BCD Converter :

Ex. 2.2.3 : Design a 3-bit excess 3 to 3-bit BCD code converter using logic gate. **May 15, 6 Marks**

Soln. :

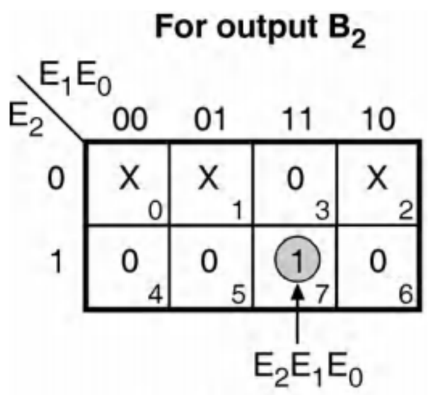
Step 1 : Write the truth table :

Input Excess 3			Output BCD			
E_2	E_1	E_0	B_2	B_1	B_0	
0	1	1	0	0	0	Valid states
1	0	0	0	0	1	
1	0	1	0	1	0	
1	1	0	0	1	1	
1	1	1	1	0	0	
0	0	0	X	X	X	Invalid states
0	0	1	X	X	X	
0	1	0	X	X	X	

(C-6179)

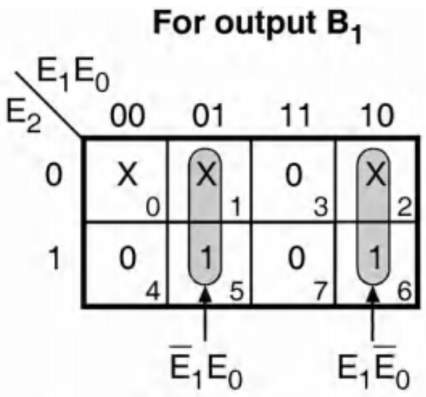
Step 2 : K-map and simplification :

For output B_2



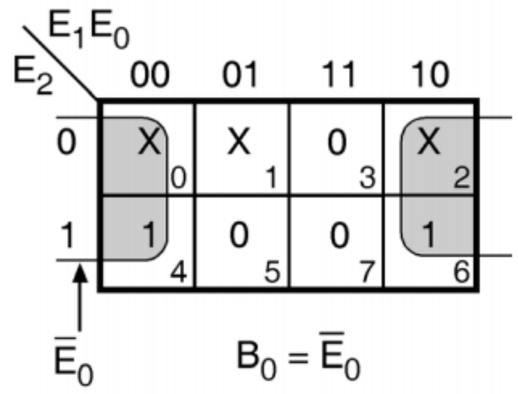
$\therefore\ B_2 = E_2E_1E_0$

For output B_1



$B_1 = \overline{E}_1E_0 + E_1\overline{E}_0 = E_1 \oplus E_0$

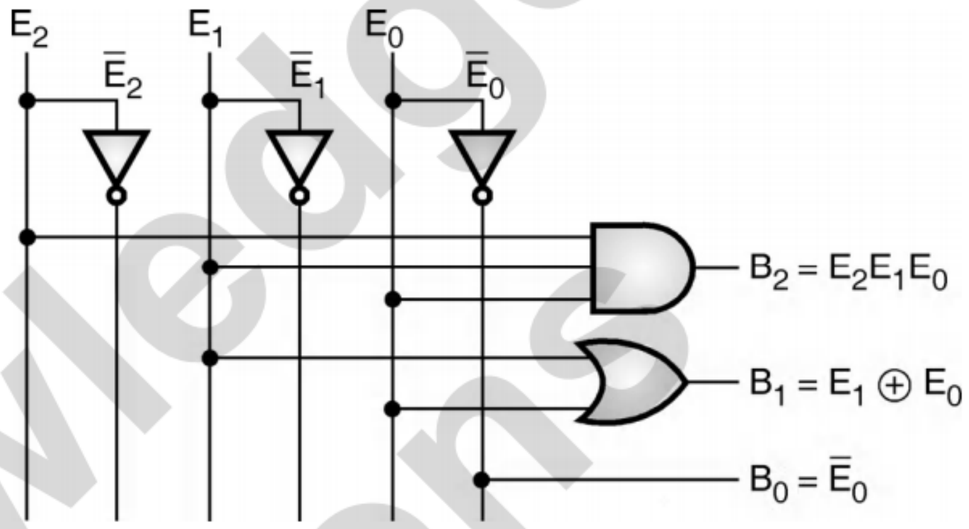
For output B_0



$B_0 = \overline{E}_0$

(C-5103) **Fig. P. 2.2.3(a)**

Step 3 : Logic gate diagram :



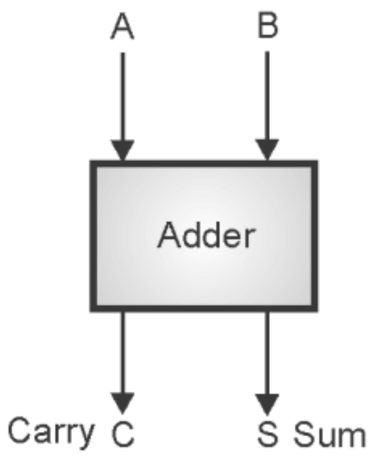
(C-5104) **Fig. P. 2.2.3(b)**

2.3 Binary Adders and Subtractors :

- Addition of two binary digits is most basic operation performed by the digital computers.
- The rules of binary addition are summarised in Table 2.3.1(A). A and B are the single bit digits to be added whereas C and S are the carry and sum outputs respectively.

Table 2.3.1(A) : Rules of binary addition

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



(C-2588) **Fig. 2.3.1(A) : Block diagram of adder**

2.3.1 Types of Binary Adders :

- In this section we are going to learn digital circuits which are used to "add" two "binary" numbers.
- The binary adders are of two types :

  1. Half adder and 2. Full adder

TechKnowledge Publications

2.3.2 Half Adder : SPPU : May 07, May 14

University Questions

Q. 1 What do you mean by half adder ?

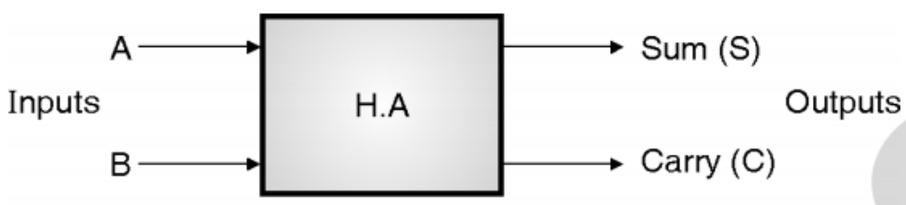
(May 07, 2 Marks)

Q. 2 What do you mean by half adder and full adder ? How will you implement full adder using half adder ? Draw the circuit diagram.

(May 14, 6 Marks)

Definition and block diagram :

- Half adder is a combinational logic circuit with two inputs and two outputs, which carries out addition of two "single" bit numbers.
- This circuit has two outputs namely "carry" and "sum". The block diagram of half adder is as shown in Fig. 2.3.1(a).



(C-344) Fig. 2.3.1 (a) : Block diagram

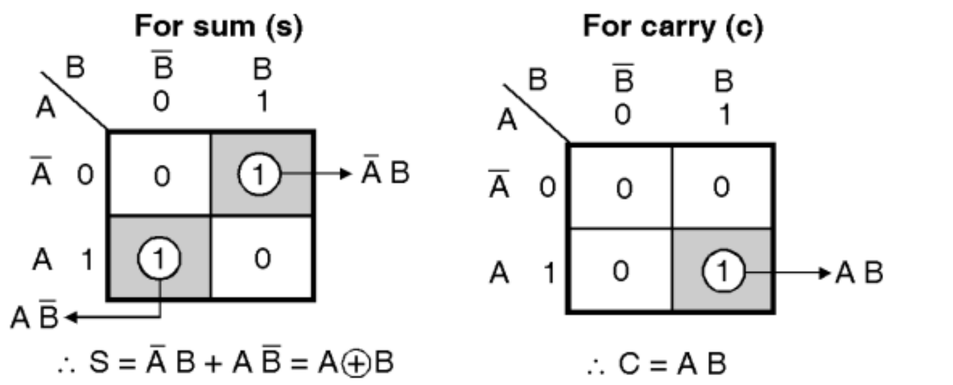
- The half adder circuit is supposed to add two single bit binary numbers A and B.
- Therefore the truth table of a half adder is as shown in Fig. 2.3.1(b).

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(C-344) Fig. 2.3.1(b) : Truth table

Design using K-maps :

- K-maps for carry and sum outputs are as shown in Figs. 2.3.2(a) and (b).



(a) K – map for sum output **(b) K – map for carry output**

(C-345) Fig. 2.3.2

- Boolean expressions for the sum (S) and carry (C) output are obtained from the K-maps as follows :

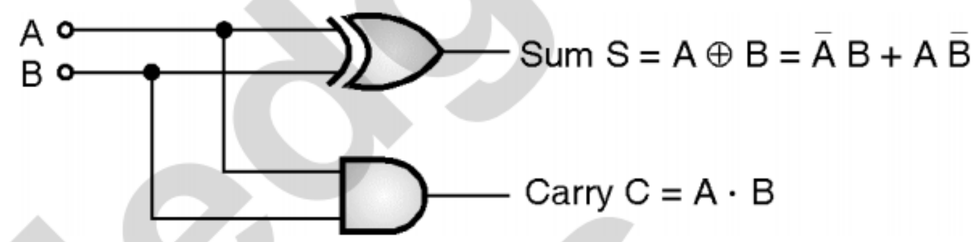
$$\left.\begin{aligned} S &= \bar{A}B + A\bar{B} = A \oplus B \\ C &= AB \end{aligned}\right\} \quad ...(2.3.1)$$

(C-346)

- The disadvantage of half adder is that addition of three bits is not possible to perform.

Logic diagram :

- The half adder circuit is as shown in Fig. 2.3.3.



(C-346) Fig. 2.3.3 : Half adder circuit

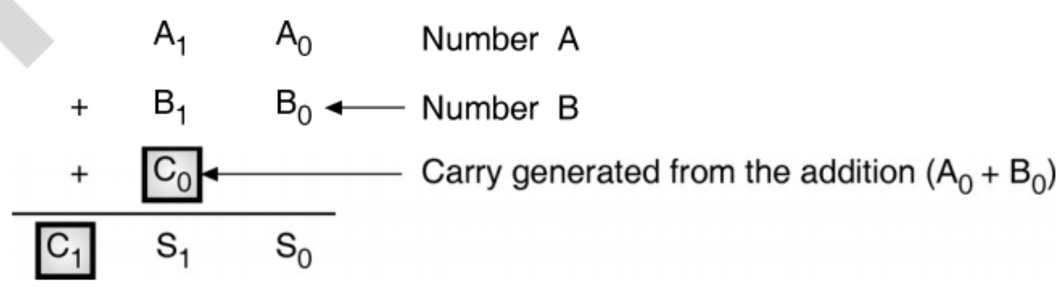
Disadvantage of half adder :

- The principle of adding two 2-bit numbers A and B is as shown in Fig. 2.3.4(a).

Let Number A = $A_1 A_0$

And Number B = $B_1 B_0$

- Then the addition should take place as shown in Fig. 2.3.4(a).



(C-347) Fig. 2.3.4(a) : Two bit binary addition

- A half adder can add A_0 and B_0 to produce S_0 and C_0. But the addition of next bits requires the addition of A_1, B_1 and C_0.
- The addition of three bits is not possible to perform by using a half adder. Hence we cannot use a half adder in practice.

Half adder using basic gates :

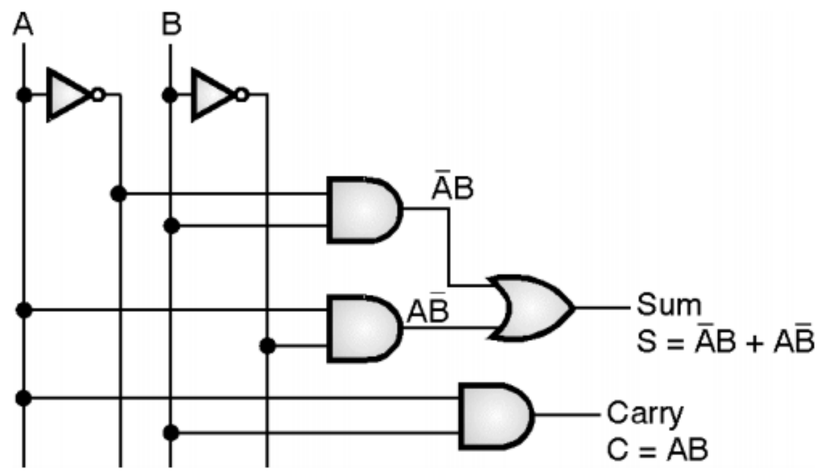
- Refer Equation (2.3.1) which states that,

$$\text{Sum } S = \bar{A}B + A\bar{B}$$

$$\text{and Carry } C = AB$$

- These equations are implemented using the basic gates as shown in Fig. 2.3.4(b).

TechKnowledge Publications



(C-348) **Fig. 2.3.4(b) : Half adder using basic gates**

Half adder using only NAND gates :

– Equation (2.3.1) states that,

Sum, $S = \bar{A}B + A\bar{B}$ and Carry, $C = AB$

– Consider the expressions separately,

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

– Take double inversion to get,

$$S = \overline{\overline{\bar{A}B + A\bar{B}}}$$

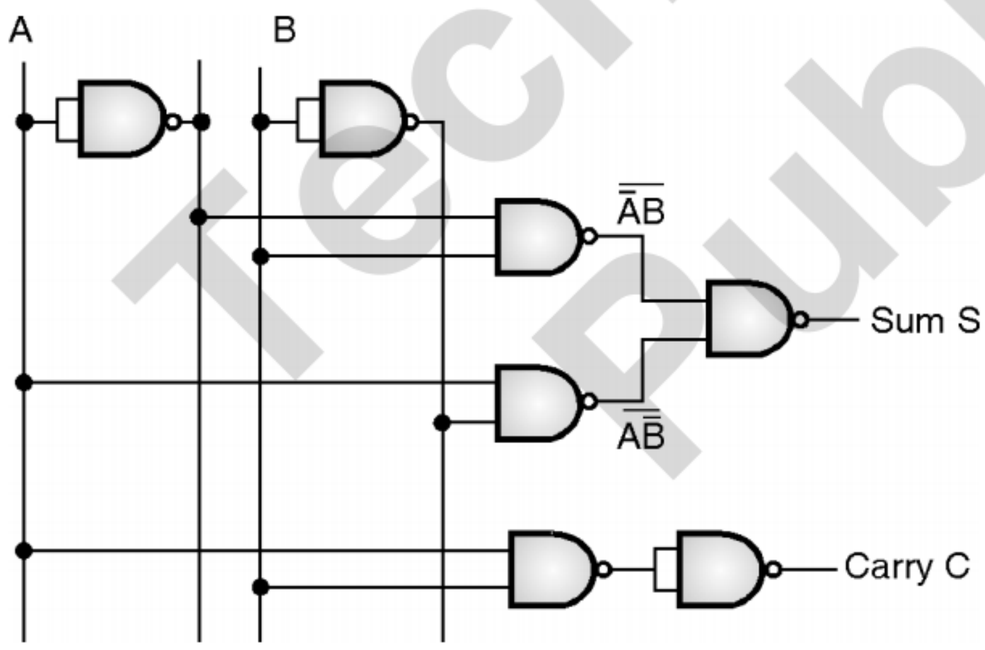
– Using the DeMorgan's theorem,

$$S = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}} \quad \text{...(Using only NAND)} \quad \text{...(2.3.2)}$$

– Similarly,

$$C = AB = \overline{\overline{AB}} \quad \text{...(Using only NAND)} \quad \text{...(2.3.3)}$$

– Equations (2.3.2) and (2.3.3) are implemented using only NAND gates as shown in Fig. 2.3.4(c).



(C-349) **Fig. 2.3.4(c) : Half adder using only NAND gates**

2.3.3 Full Adder : SPPU : May 07, May 14, Dec. 15

University Questions

Q. 1 What do you mean by full adder ?

(May 07, 6 Marks)

Q. 2 What do you mean by half adder and full adder ? How will you implement full adder using half adder ? Draw the circuit diagram.

(May 14, 6 Marks)

Q. 3 What are the full adder's inputs that will produce each of the following outputs ?

1. $\Sigma = 0$, $C_{out} = 0$
2. $\Sigma = 1$, $C_{out} = 0$
3. $\Sigma = 1$, $C_{out} = 1$
4. $\Sigma = 0$, $C_{out} = 1$. **(Dec. 15, 2 Marks)**

Definition :

– Full adder is a three input two output combinational logic circuit which can add three single bits applied at its input to produce Sum and Carry outputs.

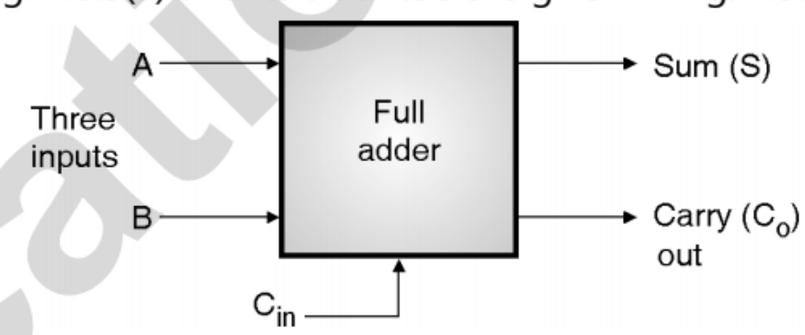
– It can add two one-bit numbers A and B, and carry C_{in}. The full adder is a three input and two output combinational circuit.

– To overcome the drawback of Half Adder circuit, a 3 single bit adder circuit called Full Adder is developed.

– It can add two one-bit numbers A and B, and carry C_{in}.

Block diagram and Truth Table :

– The block diagram of a full adder is as shown in Fig. 2.3.5(a) and its truth table is given in Fig. 2.3.5(b).



(C-350) **Fig. 2.3.5(a) : Block diagram**

Inputs			Outputs	
A	B	C_{in}	S	C_o
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

(b) Truth table

(C-350) **Fig. 2.3.5 : Full adder**

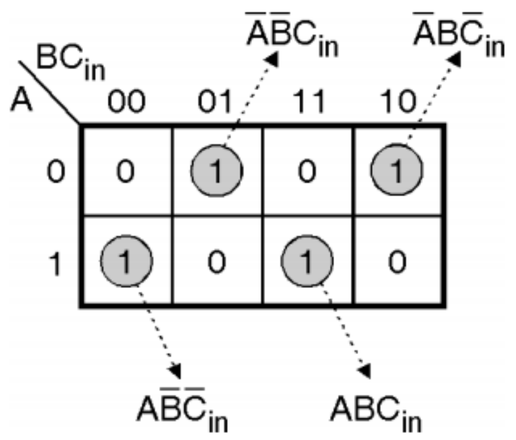
The K-maps :

– The K-maps for the sum (S) and carry out (C_o) outputs and the corresponding Boolean expressions are as shown in Fig. 2.3.6.

– Note that the K-maps have been written from the truth table of Fig. 2.3.5(b).

TechKnowledge Publications

For the sum output :



(C-351) **Fig. 2.3.6(a) : K-map for sum output**

Expression for sum output :

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + ABC_{in} + A\bar{B}\bar{C}_{in}$$

$$S = C_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{C}_{in}\underbrace{(\bar{A}B + A\bar{B})}_{\text{EX-OR}} \qquad \textbf{(C-6375)}$$

$$\therefore S = C_{in}\,(\overline{\bar{A}\bar{B} + A\bar{B}}) + \bar{C}_{in}\,(\bar{A}B + A\bar{B})$$

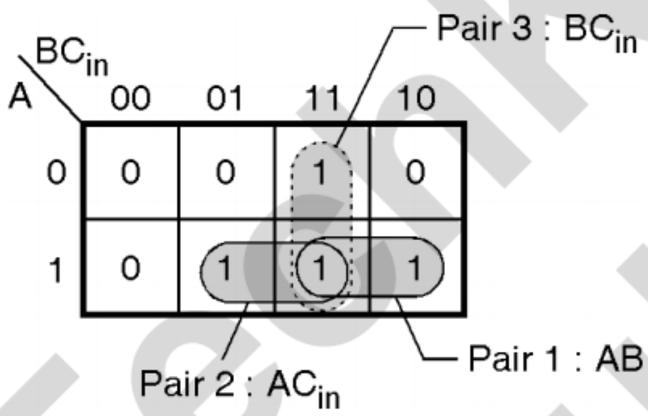
$$\text{Let } X = \bar{A}B + A\bar{B}$$

$$\therefore S = C_{in}\,\bar{X} + \bar{C}_{in}\,X = C_{in} \oplus X = C_{in} \oplus (\bar{A}B + A\bar{B})$$

But $\bar{A}B + A\bar{B} = A \oplus B$

$$\therefore S = C_{in} \oplus A \oplus B \qquad ...(2.3.4)$$

For carry output :



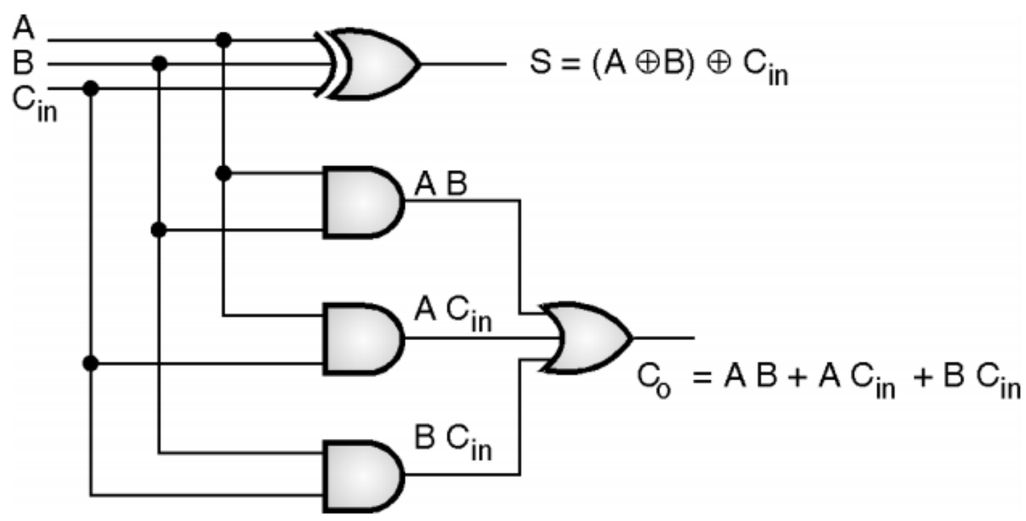
(C-352) **Fig. 2.3.6(b) : K-map for carry output**

Expression for carry output :

$$C_o = AB + AC_{in} + BC_{in} \qquad ...(2.3.5)$$

– We can use Equations (2.3.4) and (2.3.5) to draw the logic diagram of a full adder as shown in Fig. 2.3.6(c).

Logic diagram for full adder :



(C-353) **Fig. 2.3.6(c) : Full adder circuit**

2.3.4 Full Adder using Half Adder :

SPPU : May 07, May 14, Dec. 18

University Questions

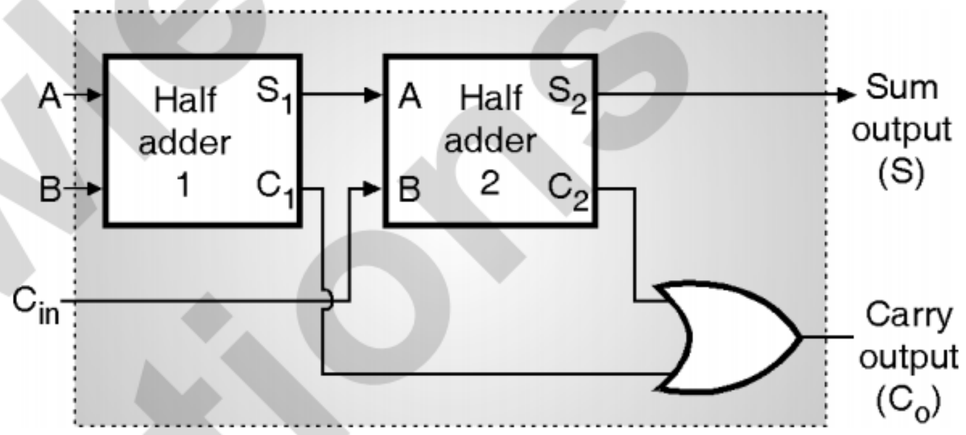
Q. 1 How will you implement full adder using half adder ? Explain with circuit diagram.

(May 07, Dec. 18, 6 Marks)

Q. 2 What do you mean by half adder and full adder ? How will you implement full adder using half adder ? Draw the circuit diagram.

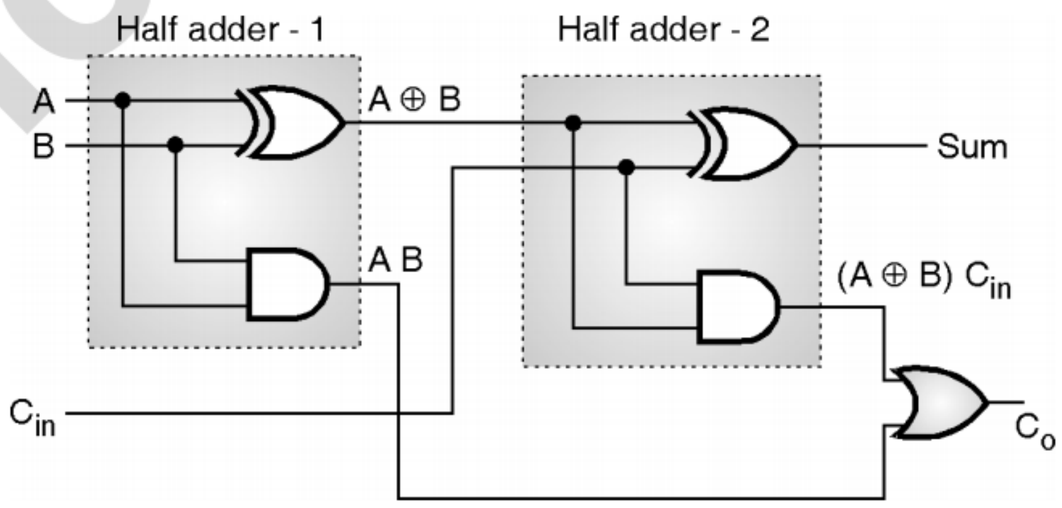
(May 14, 6 Marks)

– The full adder circuit can be constructed using two half adders as shown in Fig. 2.3.7 and the detail circuit is shown in Fig. 2.3.8.



(C-354) **Fig. 2.3.7 : Full adder using half adders**

– A full adder can be implemented using two half adders and an OR gate as shown in Fig. 2.3.8.



(C-355) **Fig. 2.3.8 : Full adder using two half adders**

– Now let us prove that this circuit acts as a full adder.

Proof :

– Refer Fig. 2.3.8 and write the expression for sum output as,

$$S = (A \oplus B) \oplus C_{in} = A \oplus B \oplus C_{in}$$

– This expression is same as that obtained for the full adder. Thus the sum output has been successfully implemented by the circuit shown in Fig. 2.3.8.

– Now write the expression for carry output C_o as,

$$C_o = (A \oplus B)\ C_{in} + AB$$

$$C_o = (\bar{A}B + A\bar{B})\ C_{in} + AB = \bar{A}BC_{in} + A\bar{B}C_{in} + AB$$

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB\,(1 + C_{in})$$

$$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

$$= BC_{in}\,(\bar{A} + A) + A\bar{B}C_{in} + AB$$

$$= BC_{in} + A\bar{B}C_{in} + AB$$

$$= BC_{in} + A\bar{B}C_{in} + AB\,(1 + C_{in})$$

$$= BC_{in} + A\bar{B}C_{in} + AB + ABC_{in}$$

$$= BC_{in} + AB + AC_{in}\,(\bar{B} + B)$$

$$\therefore C_o = BC_{in} + AB + AC_{in}$$ **...Proved.**

– This expression is same as that for a full adder. Thus we have proved that circuit shown in Fig. 2.3.8 really behaves like a full adder.

2.3.5 Applications of Full Adder :

– The full adder acts as the basic building block of the 4 bit/8 bit binary/BCD adder ICs such as 7483.

2.3.6 Binary Subtractors :

– The rules of binary subtraction are as follows :

0 – 0 = 0	0 – 1 = 1 with borrow 1
1 – 0 = 1	1 – 1 = 0

– Note that in the second case (0 – 1) it is necessary to borrow a 1.

Types of binary subtractors :

– The types of binary subtractors are :

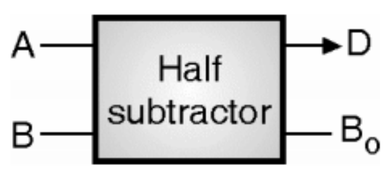
1. Half subtractor, 2. Full subtractor.

2.3.7 Half Subtractor :

Definition :

– Half subtractor is a combinational circuit with two inputs and two outputs (difference and borrow).

– It produces the difference between the two binary bits at the input and also produces an output (borrow) to indicate if a 1 has been borrowed.



(C-3514)

– In the subtraction (A – B), A is called as **minuend bit** and B is called as **subtrahend bit.**

Truth table :

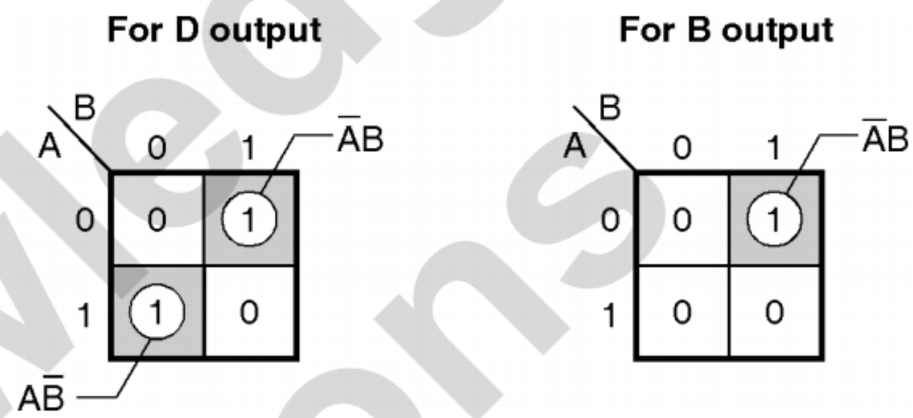
– Truth table showing the outputs of a half subtractor for all the possible combinations of input are shown in Table 2.3.1.

(C-8058) Table 2.3.1 : Truth table for half subtractor

Inputs		Outputs	
A	B	Difference D (A–B)	Borrow B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-maps for difference and borrow outputs :

– The K-maps for the two outputs of a half subtractor are as shown in Fig. 2.3.9.



(a) K-map and simplification for difference output **(b) K-map and simplification for the borrow output**

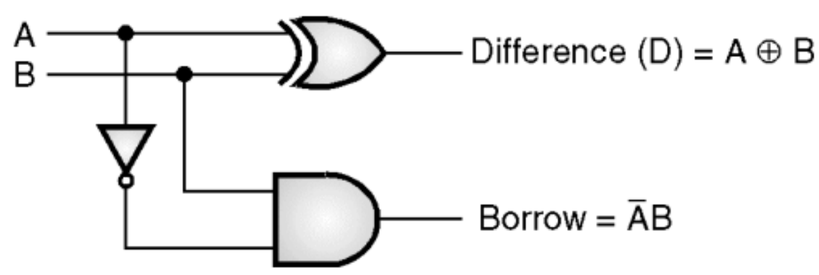
(C-356) Fig. 2.3.9

$$\therefore \text{Difference D} = \bar{A}B + A\bar{B} = A \oplus B$$

$$\therefore \text{Borrow } B_o = \bar{A}B$$

Logic diagram :

– The logic diagram using these two Boolean expressions is as shown in Fig. 2.3.9(c).



(C-357) Fig. 2.3.9(c) : Half subtractor circuit

Disadvantage of the half subtractor :

– Half subtractor can only perform the subtraction of two binary bits.

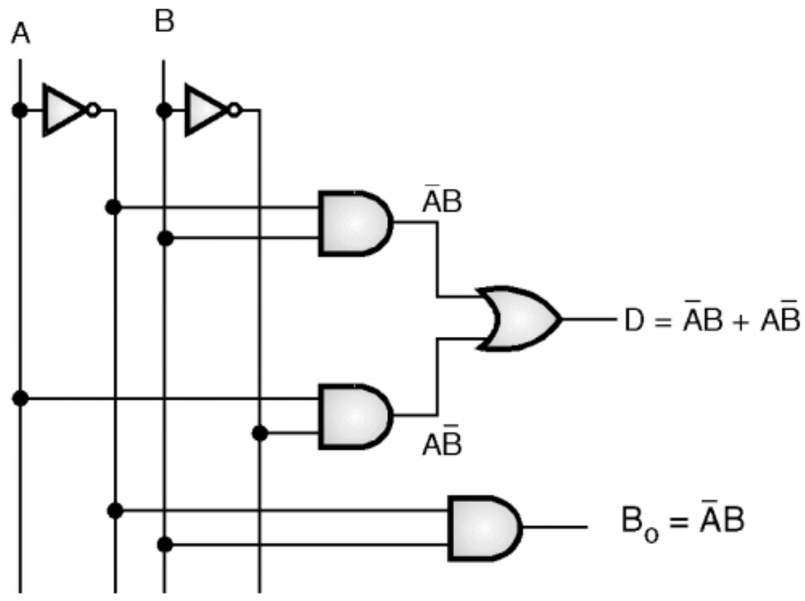
– But while performing the subtraction, it does not take into account the borrow of the lower significant stage.

Half subtractor using basic gates :

– The expressions for the difference and borrow outputs are :

$$D = \bar{A}B + A\bar{B} \quad \text{and} \quad B_o = \bar{A}B$$

– The half subtractor using basic gates is as shown in Fig. 2.3.10.



(C-358) Fig. 2.3.10 : Half subtractor using basic gates

Half subtractor using NAND gates :

– The expression for difference output is,

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

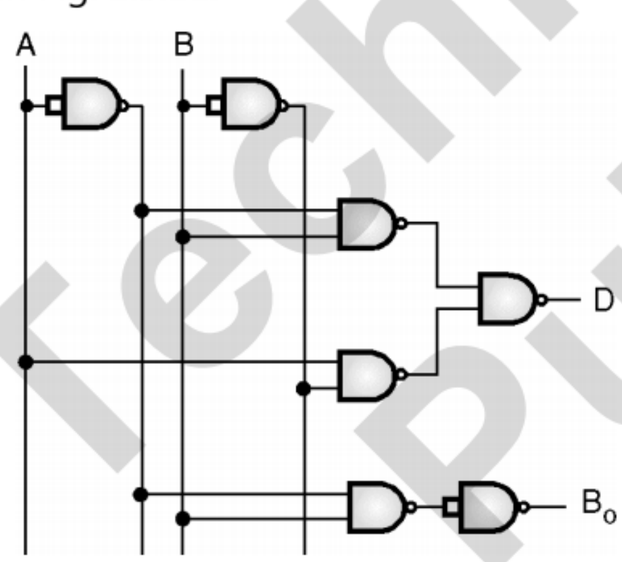
$$D = \overline{\overline{\bar{A}B + A\bar{B}}} \quad \text{...Double inversion}$$

$$D = \overline{\overline{\bar{A}B} \cdot \overline{A\bar{B}}} \quad \text{...(2.3.6)}$$

– Similarly the other output B_o is given by,

$$B_o = \bar{A}B = \overline{\overline{\bar{A}B}} \quad \text{...(2.3.7)}$$

– Equations (2.3.6) and (2.3.7) can be implemented as shown in Fig. 2.3.11.



(C-359) Fig. 2.3.11 : Half subtractor using NAND gates

2.3.8 Full Subtractor :

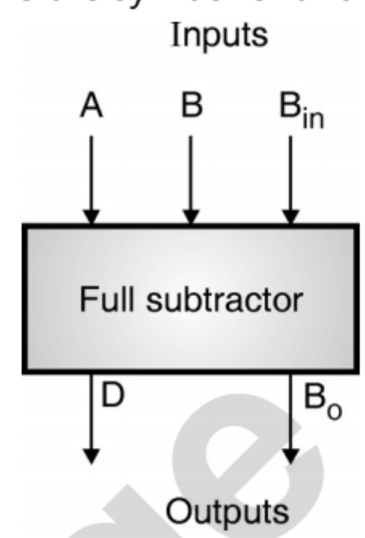
Definition and block diagram :

– The full subtractor is a combinational circuit with three inputs A, B and B_{in} and two outputs D and B_o.

– A is the minuend, B is subtrahend, B_{in} is the borrow produced by the previous stage, D is the difference output and B_o is the borrow output.

– The disadvantage of a half subtractor is overcome if we use the full subtractor.

– Fig. 2.3.12 shows the symbol of a full subtractor.



(C-2591) Fig. 2.3.12 : Symbol of a full subtractor

Truth table :

– The truth table for full subtractor is shown in Table 2.3.2.

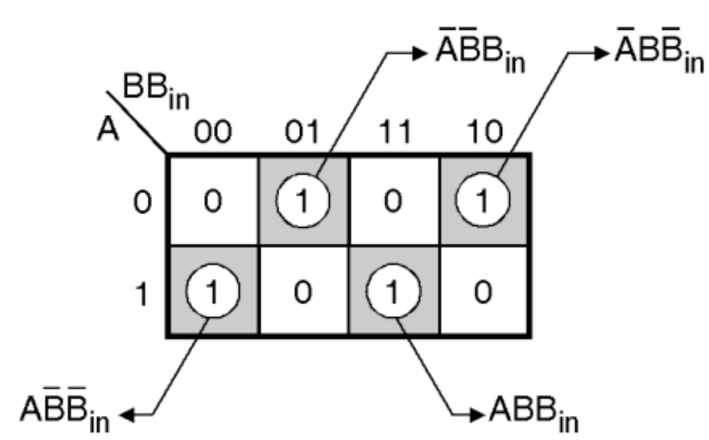
(C-8059) Table 2.3.2 : Truth table for a full subtractor

Inputs			Outputs	
A (Minuend)	B (Subtrahend)	B_{in} Previous borrow	$(A-B-B_{in})$	B_o
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

K-maps and simplifications :

– K-maps for D and B_o outputs are shown in Figs. 2.3.13(a) and (b).

For difference output :



(C-360) Fig. 2.3.13(a) : K-map for D

$$\therefore \quad D = \bar{A}\,\bar{B}\,B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

Simplification for difference output :

– From Fig. 2.3.13(a),

$$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$$

$$= B_{in}\underbrace{(\bar{A}\bar{B} + AB)}_{\text{EX-NOR}} + \bar{B}_{in}\underbrace{(\bar{A}B + A\bar{B})}_{\text{EX-OR}} \quad \textbf{(C-6376)}$$

TechKnowledge Publications

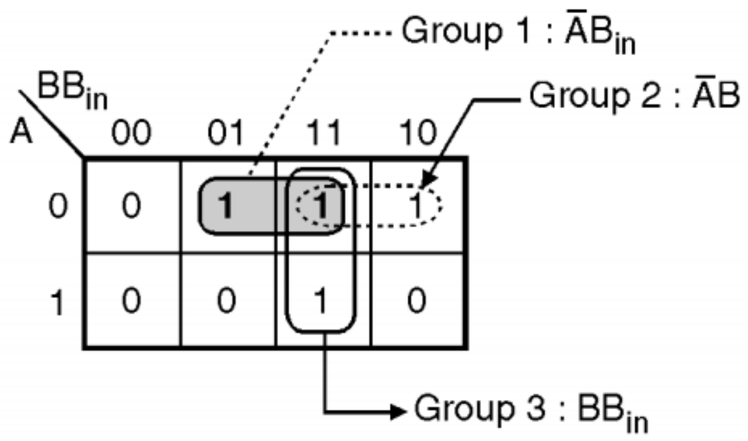
$$\therefore D = B_{in}(\overline{A \oplus B}) + \overline{B_{in}}(A \oplus B)$$

Let $A \oplus B = C$,

$$\therefore D = B_{in}\overline{C} + \overline{B_{in}}C = B_{in} \oplus C$$

$$\therefore D = B_{in} \oplus A \oplus B \quad ...(2.3.8)$$

For borrow output :



(C-360) Fig. 2.3.13(b) : K-map for B_o

$$\therefore B_o = \overline{A}B_{in} + \overline{A}B + BB_{in}$$

Simplification for borrow output :

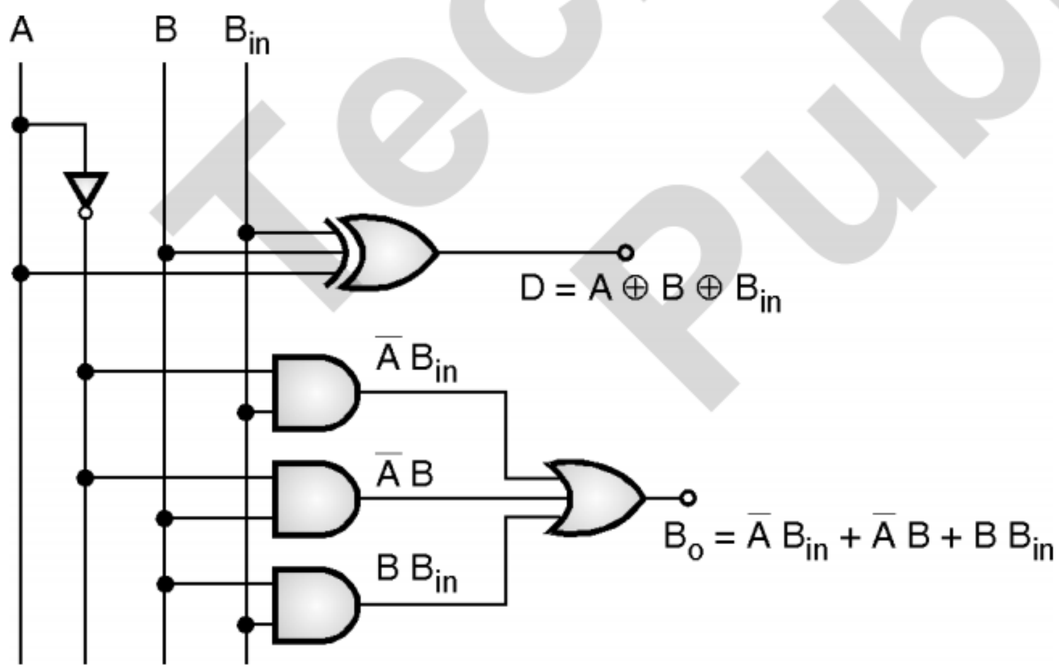
– From Fig. 2.3.13(b),

$$B_o = \overline{A}B_{in} + \overline{A}B + BB_{in} \quad ...(2.3.9)$$

– No further simplification is possible.

Logic diagram for full subtractor :

– Logic diagram for the full subtractor is shown in Fig. 2.3.14. This has been drawn by using the Boolean equations of (2.3.8) and (2.3.9).



(C-361) Fig. 2.3.14 : Logic diagram for a full subtractor

2.3.9 Full Subtractor using Half Subtractors :

SPPU : May 06, Dec. 07

University Questions

Q. 1 Draw and explain 4-bit full adder. How will you generate look ahead carry for your circuit.

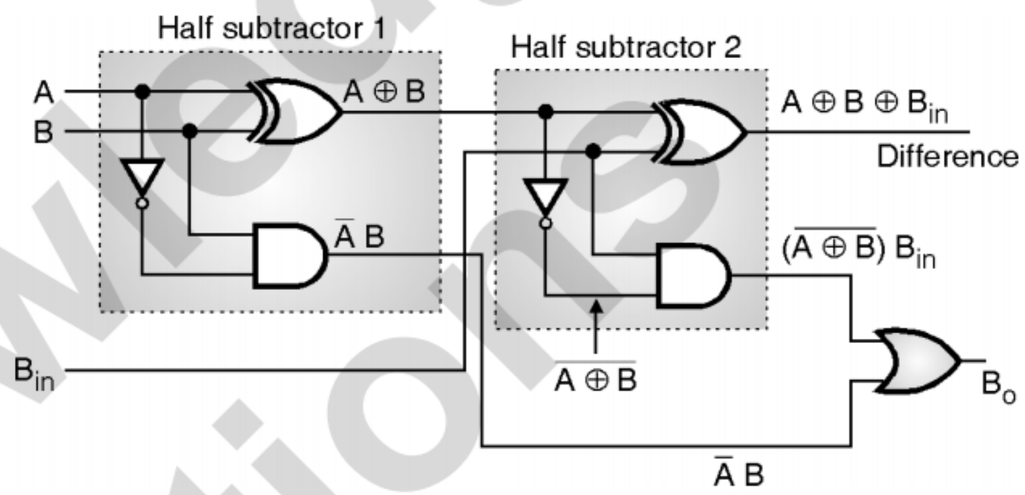
(May 06, 6 Marks)

Q. 2 What do you mean by carry generate and carry propagate ? Explain with suitable equations and block diagram. Write an equation for C_3 using carry generate and carry propagate. Simplify this equation to get minimum gate delay. How many gate delays are needed to generate C_3 with this principle ? Explain your calculations.

(Dec. 07, 10 Marks)

Logic diagram :

– Fig. 2.3.15 shows the implementation of a full subtractor using two half subtractors and an OR gate.



(C-362) Fig. 2.3.15 : Full subtractor using half subtractors

– Let us prove that the circuit shown in Fig. 2.3.15 really operates as a full subtractor.

Proof :

– Refer Fig. 2.3.14 to write the expression for difference output D as,

$$D = (A \oplus B) \oplus B_{in} = A \oplus B \oplus B_{in}$$

– This is same as the expression for D output of a full subtractor.

– Now write the expression for Borrow output B_{out}.

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

$$= (\overline{A}\,\overline{B} + AB)B_{in} + \overline{A}B = \overline{A}\,\overline{B}B_{in} + ABB_{in} + \overline{A}B$$

$$= \overline{A}\,\overline{B}B_{in} + ABB_{in} + \overline{A}B(1 + B_{in}) \quad ...\text{since } (1 + B_{in}) = 1$$

$$= \overline{A}\,\overline{B}B_{in} + ABB_{in} + \overline{A}B + \overline{A}BB_{in}$$

$$= \overline{A}\,\overline{B}B_{in} + BB_{in}(A + \overline{A}) + \overline{A}B$$

$$= \overline{A}\,\overline{B}B_{in} + BB_{in} + \overline{A}B \quad ...\text{since } (A + \overline{A}) = 1$$

$$= \overline{A}\,\overline{B}B_{in} + BB_{in} + \overline{A}B(1 + B_{in})$$

$$= \overline{A}\,\overline{B}B_{in} + BB_{in} + \overline{A}B + \overline{A}BB_{in}$$

$$= \overline{A}B_{in}(\overline{B} + B) + BB_{in} + \overline{A}B$$

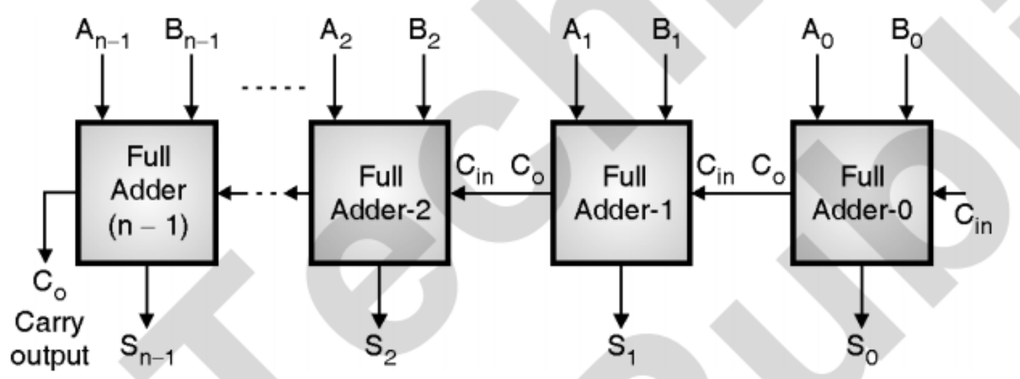
TechKnowledge Publications

$= \overline{A}\, B_{in} + BB_{in} + \overline{A}\, B$ **...Proved.**

- Note that this expression is exactly same as that for B_o of the full subtractor.
- Thus the circuit shown in Fig. 2.3.15 acts as a full subtractor.

2.4 The n-Bit Parallel Adder :

- The full adder is capable of adding only two single digit binary numbers along with a carry input.
- But in practice we need to add binary numbers which are much larger in size than just one bit.
- The two binary numbers to be added could be 4 bit, 8 bit, 16 bit long.
- In general we assume that both the numbers are n bit long.
- To add two n-bit binary numbers we need to use the n-bit parallel adder shown in Fig. 2.4.1.
- It uses a number of full adders which are connected in cascade.
- The carry output of the previous full adder is connected to the carry input of the next full adder as shown in Fig. 2.4.1.



(C-385) Fig. 2.4.1 : Block diagram of n-bit parallel adder

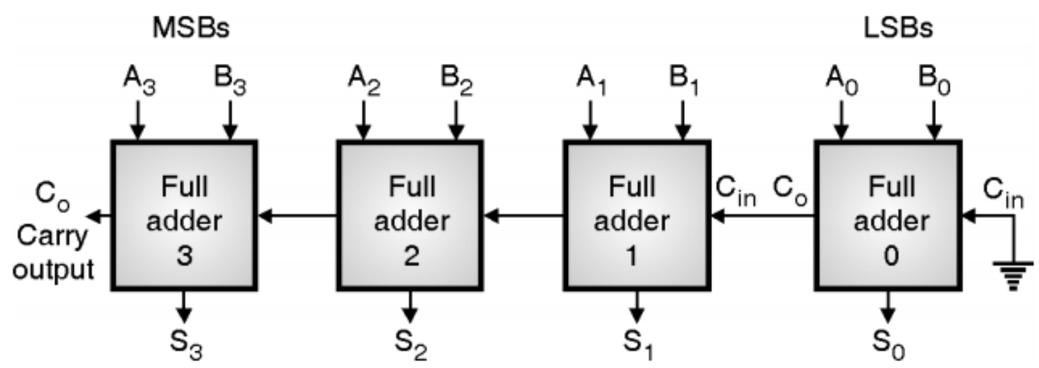
2.4.1 A Four Bit Parallel Adder Using Full Adders : SPPU : May 06, Dec. 06

University Questions

Q. 1 Draw and explain 4-bit full adder.

(May 06, Dec. 06, 4 Marks)

- The block diagram of a four bit parallel adder using full adders is shown in Fig. 2.4.2.

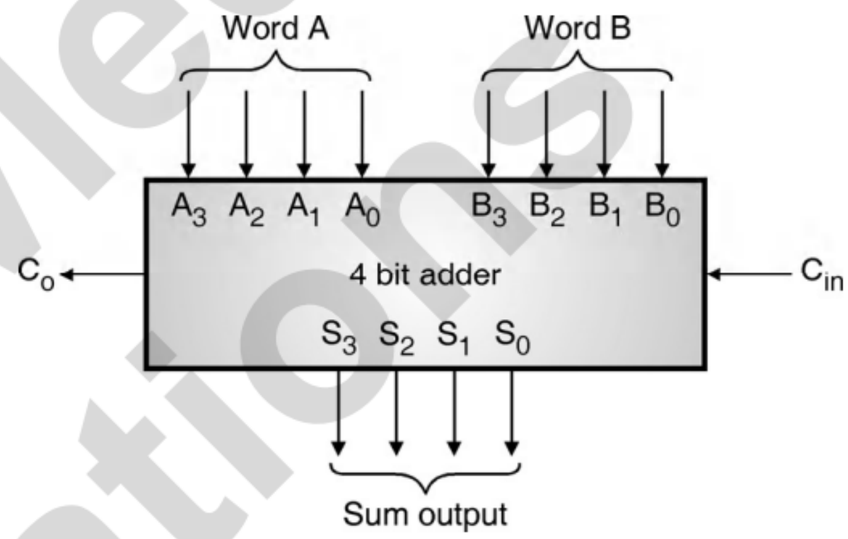


(C-386) Fig. 2.4.2 : Block diagram of a four-bit parallel adder

- Let the two four bit words that are to be added be A and B and be denoted as follows :

$$A = A_3\, A_2\, A_1\, A_0,$$
$$B = B_3\, B_2\, B_1\, B_0.$$

- A_0 and B_0 represent the LSBs of the four bit words A and B. Hence full adder-0 is the lowest stage. Hence its C_{in} has been connected to 0 permanently.
- The rest of connections are exactly same as those done for the n-bit parallel adder.
- The four-bit parallel adder is a very common logic circuit. It is normally shown by a block diagram as shown in Fig. 2.4.3.



(C-387) Fig. 2.4.3 : Block diagram of 4-bit parallel adder

- It has two 4 bit inputs A_3....A_0 and B_3 .....B_0, a carry input and carry output and 4-bit sum output $S_3\, S_2\, S_1\, S_0$.

2.4.2 Propagation Delay in Parallel Adder :

- In parallel adder carry out of the previous stage is connected to carry in of the next stage.
- Therefore the carry is said to be propagated like ripple from the LSB stage to the MSB stage. This phenomenon is called **ripple carry propagation.**
- Due to this ripple carry propagation **time delay** is introduced in the addition process. This time delay is called as the propagation delay.
- Consider the addition of the two 4-bit numbers.

$$\begin{array}{rr} & 0\,1\,1\,0 \\ + & 0\,0\,1\,1 \\ \text{Carry} & 1\,1\, \\ \hline & 1\,0\,0\,1 \end{array}$$

(C-6475)

- From this addition it is evident that the addition of 1's in the second position produces a carry which is added to the bits in the third position and the carry produced in the third position is added to the bits in the fourth position.

- Therefore the sum bit produced in the MSB position of the result depends on the carry generated due to additions in the preceding stages.
- The real problem is created due to this.
- If the propagation delay of each full adder is say 20 nS, then sum S_3 will reach its correct value after $3 \times 20 = 60$ nS from the instant when LSB carry is generated.
- Hence the total time required to perform the addition will be $4 \times 20 = 80$ nS.
- The problem of propagation delay becomes severe as the number of bits increases.
- For example if n = 16 then the total time required for performing the addition is $16 \times 20 = 320$ nS.

2.4.3 Look Ahead – Carry Adder :

SPPU : May 06, Dec. 07, May 14, Dec. 14, May 17, Dec. 17

University Questions

Q. 1 Draw and explain 4-bit full adder. How will you generate look ahead carry for your circuit.
(May 06, 6 Marks)

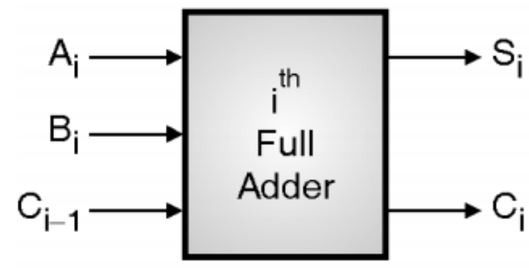
Q. 2 What do you mean by carry generate and carry propagate ? Explain with suitable equations and block diagram. Write an equation for C_3 using carry generate and carry propagate. Simplify this equation to get minimum gate delay. How many gate delays are needed to generate C_3 with this principle ? Explain your calculations.
(Dec. 07, 10 Marks)

Q. 3 Explain in detail look ahead carry generator.
(May 14, 6 Marks)

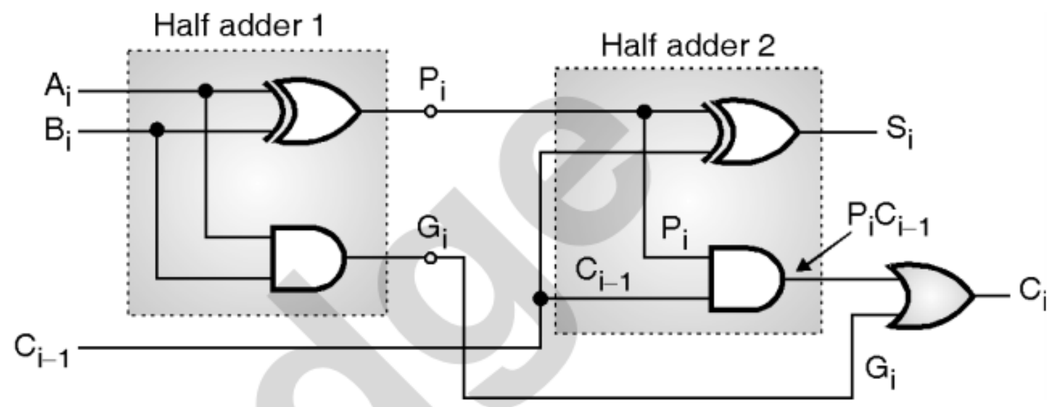
Q. 4 Explain Look Ahead Carry generator in detail.
(Dec. 14, May 17, 6 Marks, Dec. 17, 4 Marks)

Block diagram :

- The problem of propagation delay can be eliminated by using this addition technique.
- The look-ahead-carry addition will therefore speed up the addition process.
- The adder with look ahead carry needs additional hardware but the speed of that adder is independent of the number of bits.
- Consider the block diagram of full adder Fig. 2.4.4(a) and its AND-OR-EXOR realization shown in Fig. 2.4.4(b).



(a) Block diagram of ith full adder



(b) Realization using AND-OR-EXOR gates

(C-389) Fig. 2.4.4

- Refer Fig. 2.4.4(b) to write,

$$P_i = A_i \oplus B_i \quad ...(2.4.1)$$

$$\text{and } G_i = A_i B_i \quad ...(2.4.2)$$

$$\text{Also } S_i = P_i \oplus C_{i-1} = A_i \oplus B_i \oplus C_{i-1} \quad ...(2.4.3)$$

$$\text{and } C_i = G_i + P_i C_{i-1} \quad ...(2.4.4)$$

- The carry output G_i of the first half adder is equal to 1 if $A_i = B_i = 1$ and a carry is generated at the ith stage of the parallel adder. That means $C_i = 1$.
- This variable G_i is known as **Carry Generate** and its value does not depend on the input carry i.e. C_{i-1}.
- The variable P_i is called as **Carry Propagate** because this term is associated with the propagation of carry from C_{i-1} to C_i. Now consider Equation (2.4.4) i.e. $C_i = G_i + P_i C_{i-1}$.
- Using this equation, we can write the expression for the carry output of each stage in a 4 bit parallel adder as follows :

$$P_i = A_i \oplus B_i$$

Stage expression for carry output

$$0 \quad C_0 = G_0 + P_0 C_{-1} \quad ...(2.4.5)$$

$$1 \quad C_1 = G_1 + P_1 C_0$$

$$= G_1 + P_1 (G_0 + P_0 C_{-1})$$

$$\therefore C_1 = G_1 + P_1 G_0 + P_0 P_1 C_{-1} \quad ...(2.4.6)$$

Stage expression for carry output

$$2 \quad C_2 = G_2 + P_2 C_1 = G_2 + P_2 (G_1 + P_1 G_0 + P_0 P_1 C_{-1})$$

$$\therefore C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1} \quad ...(2.4.7)$$

$$3 \quad C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1} \quad ...(2.4.8)$$

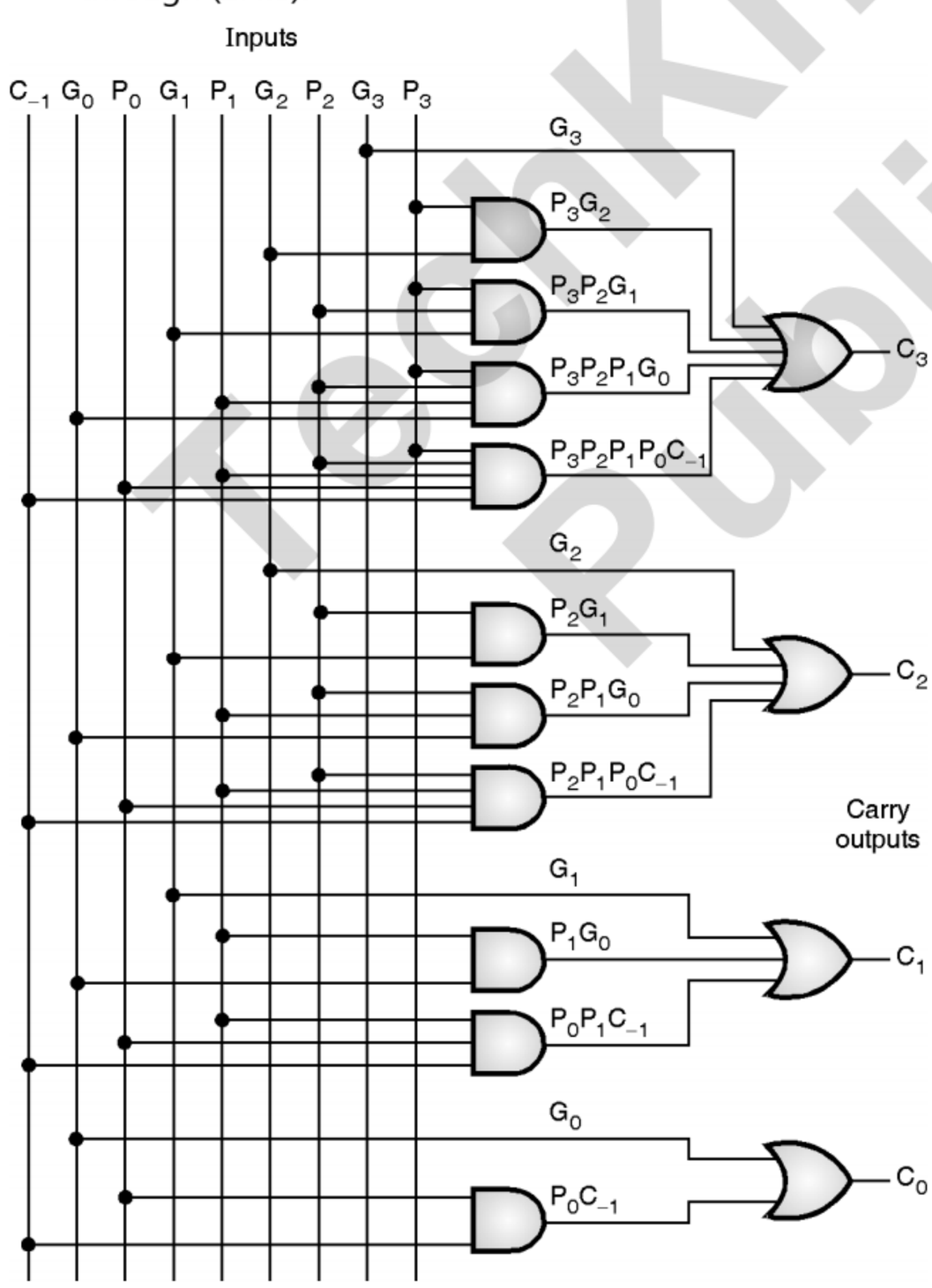
- In the expressions stated above, the variable involved are, G_0, G_1, G_2, G_3, P_0, P_1, P_2, P_3 and C_{-1}.

TechKnowledge Publications

- Out of them the G variables are generated from the A and B inputs using AND gates (as illustrated in Equation (2.4.2).
- And the P variables are obtained again directly from A and B inputs using EX-OR gates (as illustrated in Equation (2.4.1).
- If the G, P and C_{-1} are at a time available, then it is possible to produce the carry outputs C_0, C_1, C_2 and C_3 by using 2-level realization (AND-OR or NAND-NAND) etc.
- The advantage of generating the carry outputs using this method is that the propagation delay in this process corresponds to the propagation delay of only two gates.
- These carry outputs are then connected to the carry inputs of the succeeding stages.
- This eliminates the problem of carry getting propagated like ripples.

Logic diagram :

- The logic circuit of a look ahead carry generator is shown in Fig. 2.4.5 and it is based on Equations (2.4.1) through (2.4.8).

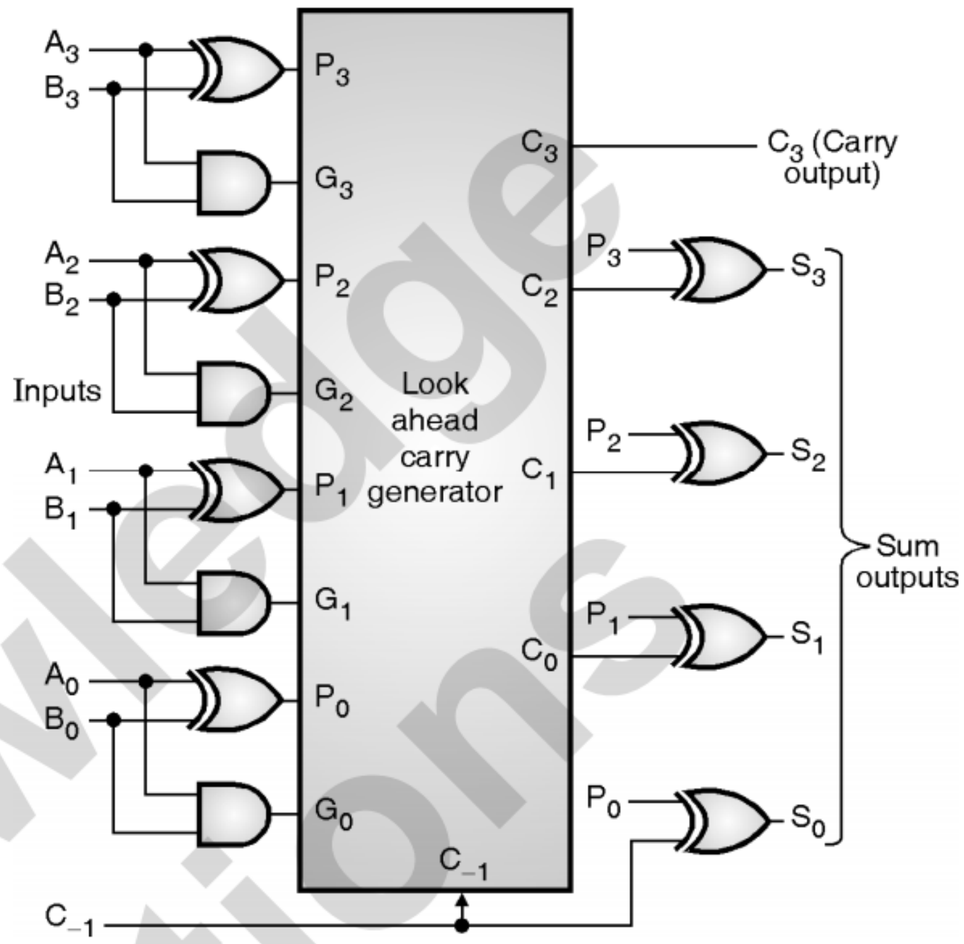


(C-390) **Fig. 2.4.5 : Logic diagram of the look ahead carry generator**

2.4.4 Four Bit Fast Adder with Look-Ahead Carry :

Block diagram :

- The block diagram of a four bit parallel adder using the look-ahead carry generator is shown in Fig. 2.4.6.



(C-391) **Fig. 2.4.6 : A 4-bit parallel adder**

- The P_i and G_i variables (i.e. P_0, P_1, ..., P_3 and G_0, G_1 .....G_3) are obtained from the inputs A_i and B_i (i.e. A_3 A_2 A_1 A_0 and B_3 B_2 B_1 B_0) by using the EX-OR and AND gates respectively.
- The carry outputs C_0 through C_3 are produced simultaneously by the look ahead carry generator, as explained earlier.
- These carry outputs and P_i variables are EX-ORed to produce the sum outputs S_0 through S_3. For example S_3 is produced by Ex-ORing P_3 and C_2, then S_2 is produced by Ex-ORing P_2 and C_1 and so on.

2.4.5 Binary Adder IC 74 LS 83 / 74 LS 283 :

SPPU : Dec. 10, Dec. 11

University Questions

Q. 1 Explain for IC 74LSXX various characteristics in brief. **(Dec. 10, 4 Marks)**

Q. 2 What is the use of 7483 chip ? **(Dec. 11, 4 Marks)**

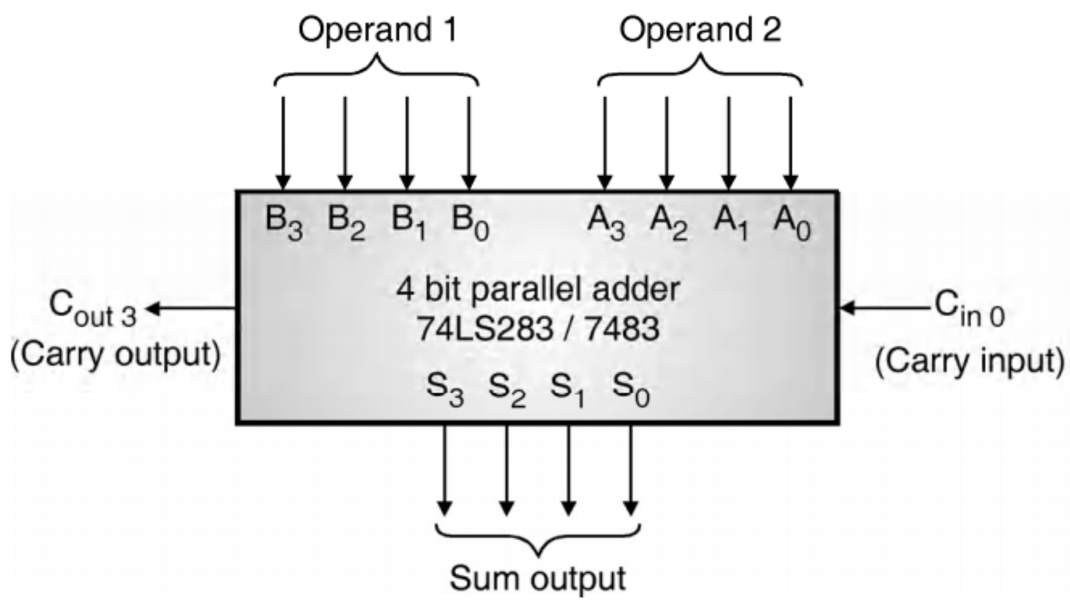
Block diagram :

- The most common binary parallel adder in the integrated circuit form is IC 74 LS 83 / 74 LS 283.
- It is a 4-Bit parallel adder, which consists of four interconnected full adders alongwith the look-ahead carry circuit.

TechKnowledge Publications

– The IC 7483 and 74283 are TTL MSI (Medium Scale Integration) for 4-bit parallel adders and both of them have the same pin configuration.

– Fig. 2.4.7 shows the functional symbol of IC 74 LS 283. A_3 A_2 A_1 A_0 is a four bit word A and B_3 B_2 B_1 B_0 is another word B.



(C-392) **Fig. 2.4.7 : Functional symbol for 74 LS 283**

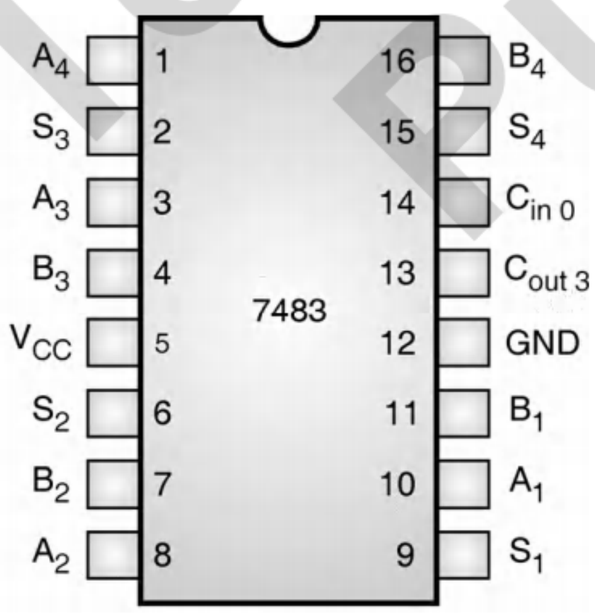
– Both these words are applied at the inputs of the I.C. 7483/74 LS 283.

– $C_{in\ 0}$ is the input carry and $C_{out\ 3}$ represents the output carry. S_3, S_2, S_1, S_0 represent the sum outputs with S_3 MSB.

Pin diagram :

– The pin diagram of IC 7483 is shown in Fig. 2.4.8.

– This IC adds the two four bit words A and B and the bit at $C_{in\ 0}$ and produces a four bit sum output along with carry output at $C_{out\ 3}$.



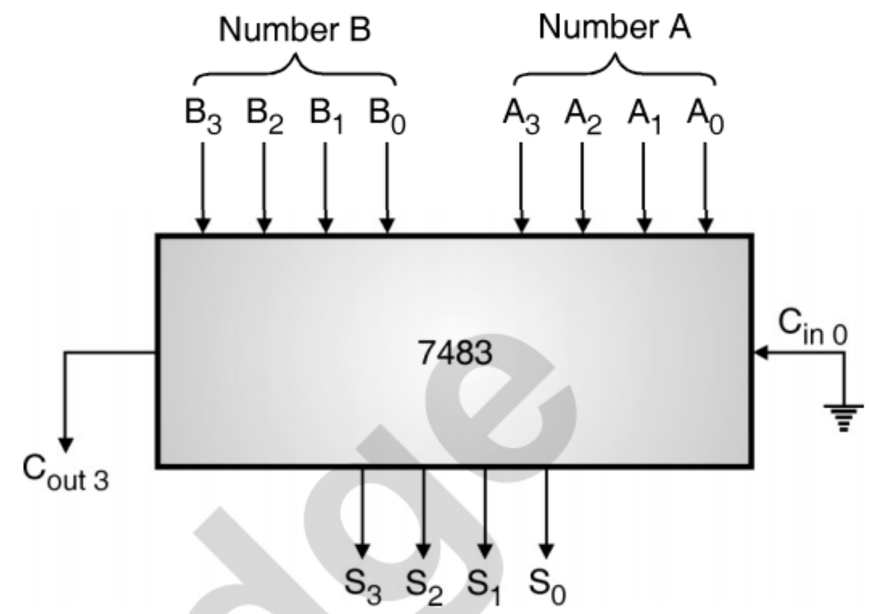
(C-392) **Fig. 2.4.8 : Pin diagram of IC – 7483**

2.4.6 Four Bit Binary Adder using IC 7483 :

Block diagram :

– A four bit binary adder is shown in Fig. 2.4.9. Note that the carry input Cin0 has been connected to ground.

– So at the outputs we get the addition of two four bit numbers A and B.



(C-393) **Fig. 2.4.9 : 4 bit binary adder using IC 7483**

2.4.7 Cascading of Adders :

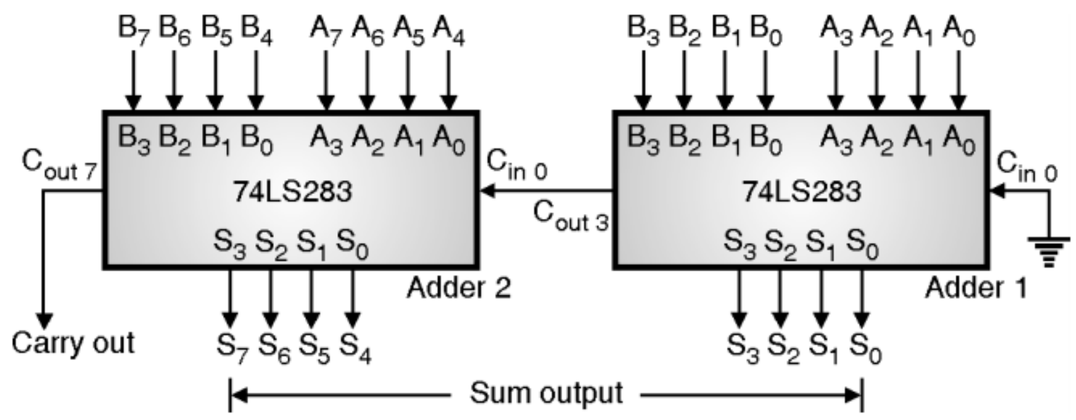
– If we want to add two 8-bit numbers using the 4-bit parallel adder 74283, then we have to cascade two such four bit adders.

– Fig. 2.4.10 shows an 8-bit adder using two 4-bit adders. Similarly it is possible to connect a number of adders to make an n-bit adder.

– $A_7 – A_0$ and $B_7 – B_0$ are the two eight bit numbers to be added.

– Adder-1 in Fig. 2.4.10 adds the four LSB bits of the two numbers i.e. $A_3 – A_0$ and $B_3 – B_0$.

– The carry input of first adder has been connected to ground (logic 0).



(C-394) **Fig. 2.4.10 : 8-bit addition using 4-bit adders**

– The $C_{out\ 3}$ i.e. the carry output of adder-1 is connected to $C_{in\ 0}$ input of adder-2. The second adder adds this carry and the four MSB bits of the two numbers.

– $C_{out\ 7}$ of adder-2 acts as the final output carry and the sum output is from S_7 through S_0.

2.5 n-bit Parallel Subtractor (Use of Adder as Subtractor) :

– The subtraction can be carried out by taking the 1's or 2's complement of the number to be subtracted.

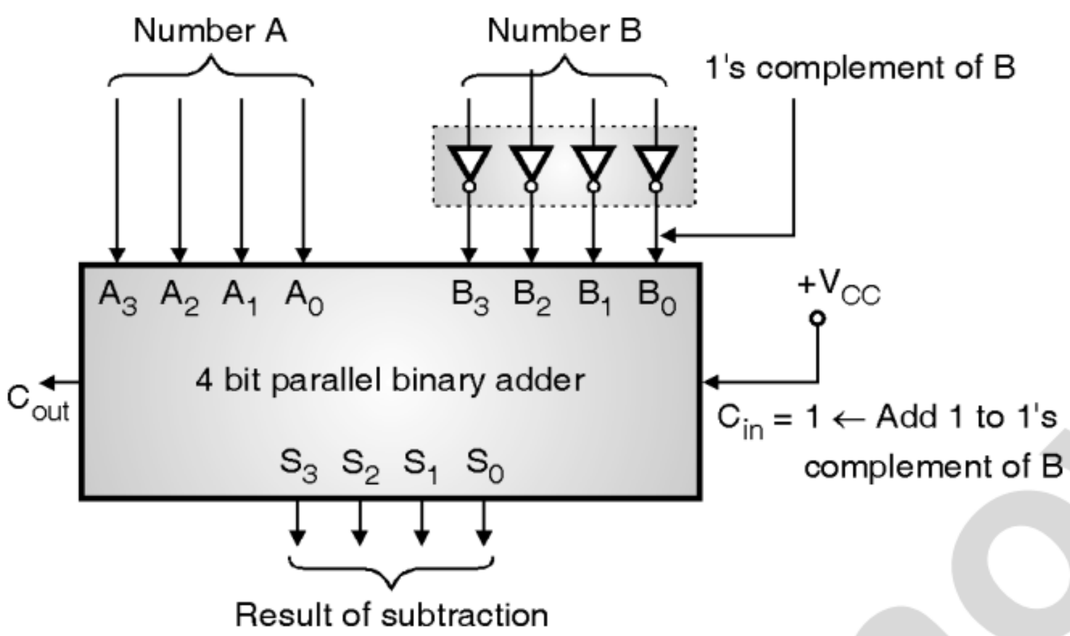
TechKnowledge Publications

– For example we can perform the subtraction (A – B) by adding either 1's complement or 2's complement of B to A. That means we can use a binary adder to perform the binary subtraction.

2.5.1 4 Bit Parallel Subtractor using 2's Complement :

Block diagram :

– A 4-bit parallel subtractor using a 4-bit parallel adder is shown in Fig. 2.5.1.



(C-395) Fig. 2.5.1 : 4-bit parallel binary subtractor using 2's complement

Description :

– The number to be subtracted (B) is first passed through inverters to obtain its 1's complement.

– One inverter per bit of word B is used so that all the bits of B get inverted.

– Then 1 is added to 1's complement of B, by making C_{in} = 1. Thus we obtain the 2's complement of B.

– The 4-bit adder then adds A and 2's complement of B to produce the subtraction at its sum outputs S_3 S_2 S_1 S_0.

– The word S_3 S_2 S_1 S_0 represent the result of binary subtraction (A – B) and carry output C_{out} represents the polarity of the result.

– If A > B then C_{out} = 0 and the result is in true binary form but if A < B then C_{out} = 1 and the result is negative and in the 2's complement form.

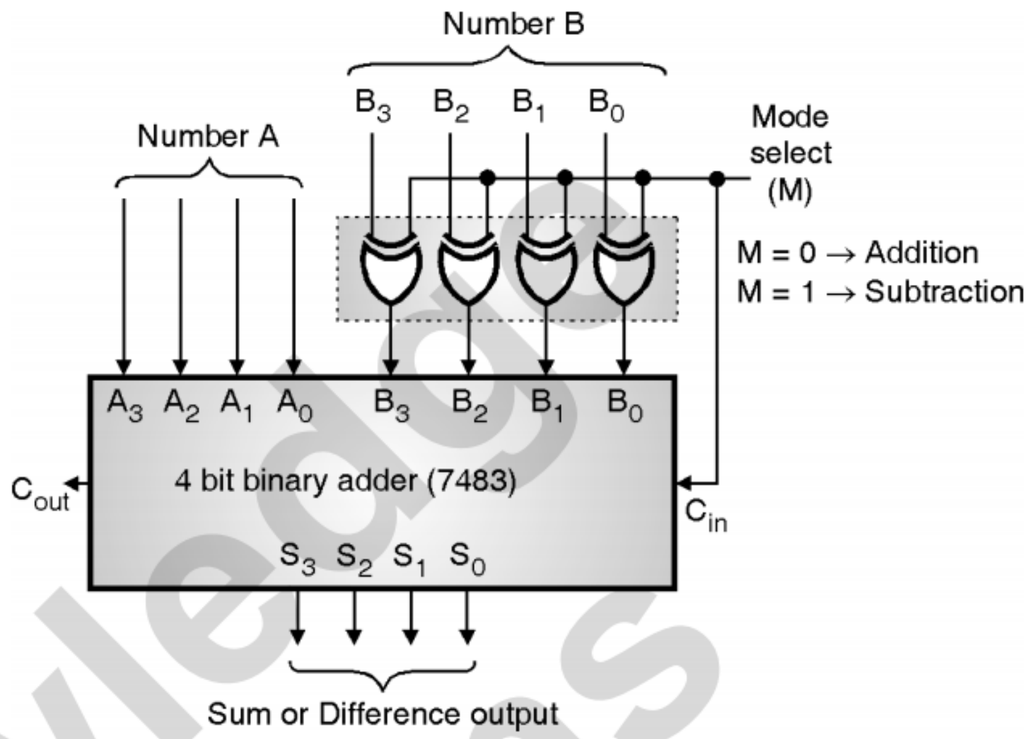
2.5.2 4-Bit Binary Parallel Adder/Subtractor :

Block diagram :

– The addition or subtraction of two 4-bit binary numbers can be obtained using the same circuit shown in Fig. 2.5.2.

– Therefore it is called as an adder/subtractor circuit.

– The operation performed by this circuit (addition or subtraction) depends on the state of the mode of select input i.e. M in Fig. 2.5.2.



(C-396) Fig. 2.5.2 : 4-bit binary parallel adder/subtractor

– Number B is applied to the adder through four EX-OR gates. One input of each EX-OR gate is connected to the mode-select input (M).

Operation as Adder (M = 0) :

– For operation as an adder, the mode select (M) input is connected to ground,

$\therefore$ M = 0.

– Since M = 0, the output of the EX-OR gates will be the same number B which was applied at their inputs. This is because $0 \oplus 0 = 0$ and $0 \oplus 1 = 1$.

– Hence B_3, B_2, B_1 and B_0 will pass unchanged through the EX-OR gates. The carry input C_{in} is connected to M,

$\therefore C_{in} = 0$.

– Therefore the adder adds A + B + C_{in} = A + B since C_{in} = 0. Thus with M = 0 addition of A and B will take place.

Operation as Subtractor (M = 1) :

– For operation as a subtractor, the mode select (M) input is connected to V_{CC},

$\therefore$ M = 1.

– Since M = 1, one input of each EX-OR gate is now 1. Hence each EX-OR gate acts as an inverter. This is because $1 \oplus 0 = 1$ and $1 \oplus 1 = 0$.

– Thus each bit of word B is inverted by the EX-OR inverters. Thus we get the 1's complement of number B at the output of EX-OR gates.

TechKnowledge Publications

– The carry input terminal C_{in} is connected to M,

$\therefore\ C_{in} = 1$

– The inverted number B adds with C_{in} = 1 to give the 2's complement of B. Hence the adder will add A with the 2's complement of B and the result is actually the subtraction A – B.

– If C_{out} = 0 then the subtraction (A – B) is positive and in true form. But if C_{out} = 1 then the subtraction (A – B) is negative and in the 2's complement form.

– Thus with M = 1, this circuit works as a 2's complement subtractor.

2.6 BCD Addition : SPPU : Dec. 14, Dec. 16

University Questions

Q. 1 Explain rules for BCD addition with suitable example and design one digit BCD adder using IC 7483. **(Dec. 14, 6 Marks)**

Q. 2 Write the rules for BCD addition and give example. **(Dec. 16, 2 Marks)**

– In BCD addition we have to deal with three different situations. Assume that two 4 bit BCD numbers A and B are being added. Then the three cases to be considered are,

Case 1 : Sum is equal to or less than 9 and carry is 0.

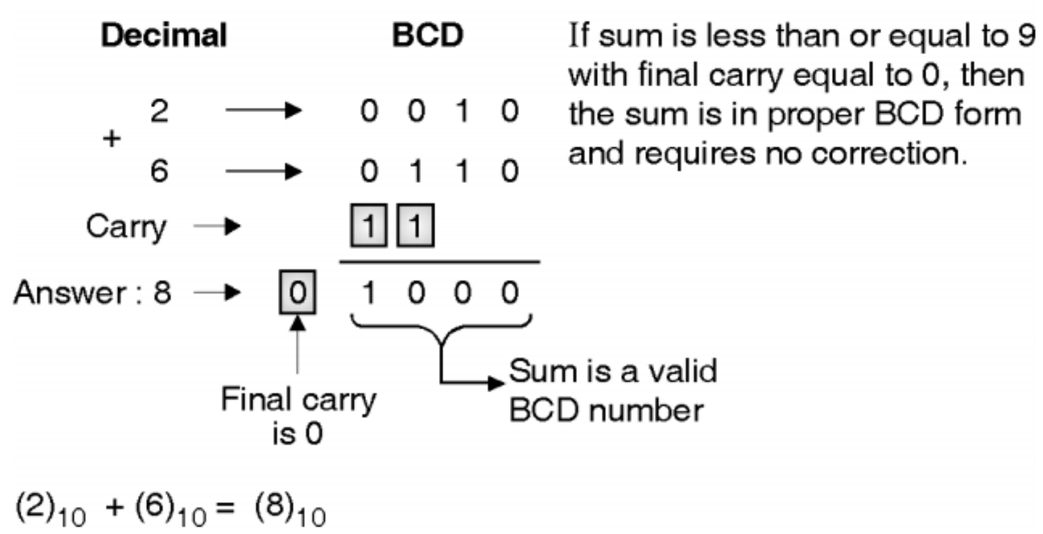
Case 2 : Sum is greater than 9 and carry is 0.

Case 3 : Sum is less than or equal to 9 but carry is 1.

– The BCD addition is to be carried out in an identical manner as normal binary addition.

Case 1 : Sum equal to or less than 9 with carry 0

– The addition of $(2)_{10}$ and $(6)_{10}$ in BCD is shown in Fig. 2.6.1.



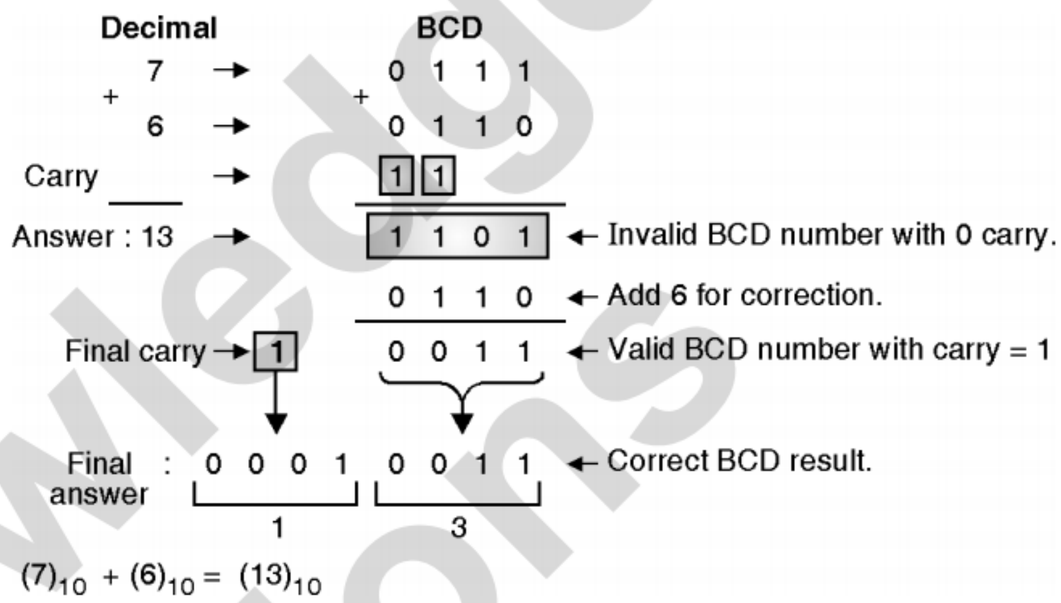
(C-76) **Fig. 2.6.1 : Illustration of case 1 in BCD addition**

Case 2 : Sum greater than 9 but carry = 0

– In this case the sum of the two BCD numbers is greater than 9 that means it is an invalid BCD number. But the final carry is 0.

– So we have to correct the sum by adding decimal 6 or BCD 0110 to it. After doing this we get the correct BCD sum.

– Case 2 of BCD addition is illustrated in Fig. 2.6.2.



(C-77) **Fig. 2.6.2 : Illustration of case 2 in BCD addition**

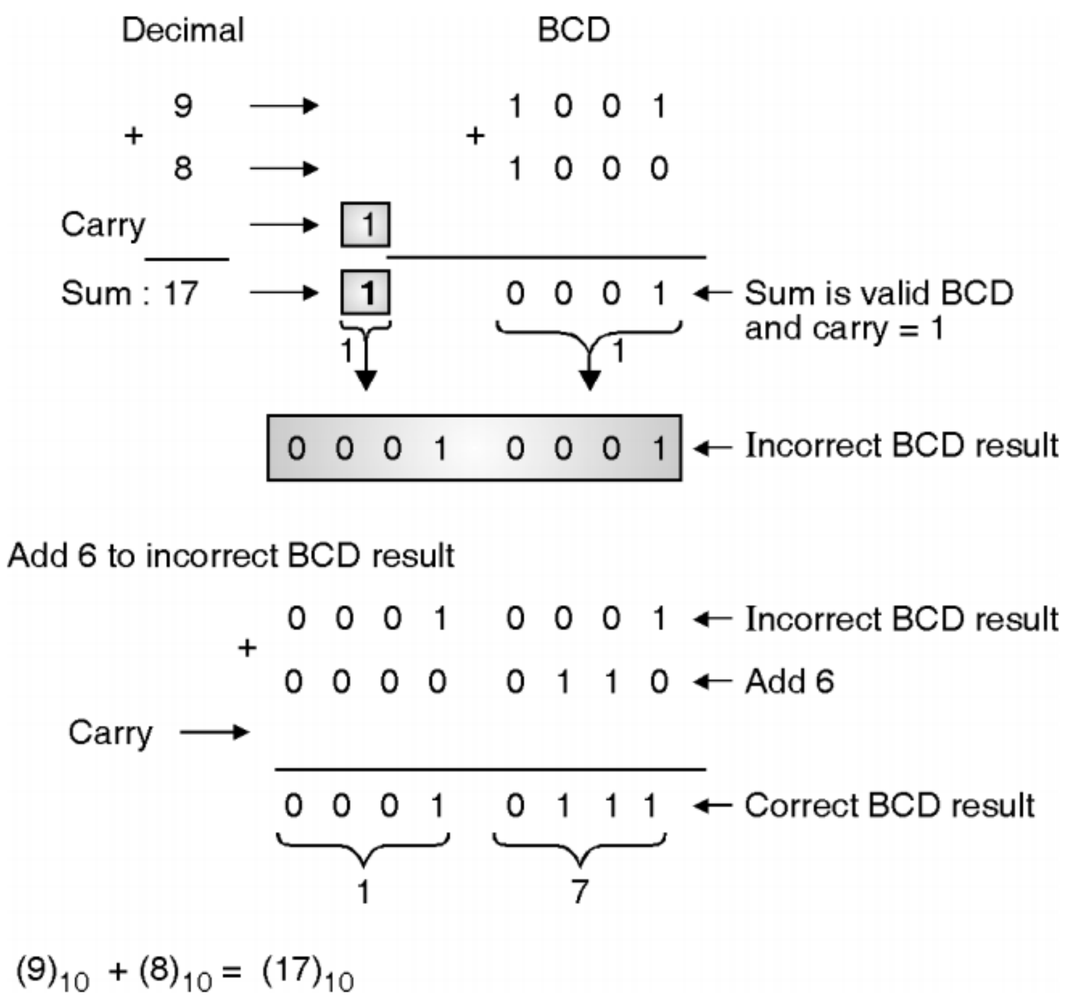
Case 3 : Sum less than or equal to 9 but carry = 1

– Here the sum of two BCD numbers is less than or equal to 9 i.e. it is a valid BCD number, and final carry = 1.

– A nonzero carry indicates that the answer is wrong and needs correction.

– Add $(6)_{10}$ or (0110) BCD to the sum to correct the answer.

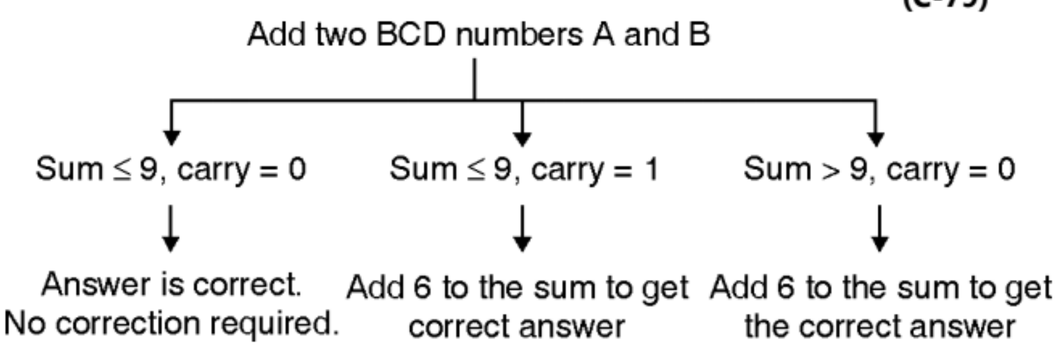
– Case 3 of BCD addition is illustrated in Fig. 2.6.3.



(C-78) **Fig. 2.6.3 : Illustration of case 3 in BCD addition**

Summary of BCD addition :

– The BCD addition discussed so far can be summarised as follows : **(C-79)**



Ex. 2.6.1 : Add $(83)_{10}$ and $(34)_{10}$ in BCD.

Soln. :

Decimal		BCD		
$(83)_{10}$		1 0 0 0	0 0 1 1	
+ $(34)_{10}$	+	0 0 1 1	0 1 0 0	
	Carry			
$(117)_{10}$	Sum :	1 0 1 1 (Invalid BCD)	1 0 1 1 (Valid BCD)	Sum > 9, Carry = 0 ∴ Correction required

Add 6 to the invalid BCD, for correction.

		1 0 1 1	0 1 1 1	
	+	0 1 1 0	0 0 0 0	6 is added only to the invalid BCD.
	Carry	1 1		
	1	0 0 0 1	0 1 1 1	
	0 0 0 1	0 0 0 1	0 1 1 1	Correct BCD result.
	1	1	7	

(C-6578)

∴ $(83)_{10} + (34)_{10} = (117)_{10}$ **...Ans.**

Ex. 2.6.2 : Perform addition in BCD format $(79)_{BCD} + (16)_{BCD}$.

Soln. :

$(79)_{BCD} + (16)_{BCD}$:

Decimal		BCD	
		0 1 1 1	1 0 0 1
+ 79			
16		0 0 0 1	0 1 1 0
	Carry	1 1 1	
95		1 0 0 0 (Valid BCD)	1 1 1 1 (Invalid BCD)

(C-5229)

Add $(6)_{10}$ to invalid BCD.

∴		1 0 0 0	1 1 1 1
	+	0 0 0 0	0 1 1 0
	Carry	1	1 1
		1 0 0 1	0 1 0 1
		9	5

(C-5230)

∴ $(79)_{BCD} + (16)_{BCD} = (95)_{BCD}$ **...Ans.**

2.6.1 BCD Adder :

SPPU : May 11, May 12, Dec. 12, Dec. 14

University Questions

Q. 1 Draw and explain the basic circuit of single digit BCD adder using IC 7483. How will you make two digit BCD adder ? Explain the logic of the circuit. **(May 11, 4 Marks)**

Q. 2 Describe the working of BCD adder using 7483 with the help of diagram. **(May 12, 8 Marks)**

Q. 3 Draw and explain 4-bit BCD adder using IC7483. Also explain with example addition of numbers with carry. **(Dec. 12, 8 Marks)**

Q. 4 Explain rules for BCD addition with suitable example and design one digit BCD adder using IC 7483. **(Dec. 14, 6 Marks)**

– BCD adder adds two BCD digits and produces a BCD digit. But remember that a BCD digit cannot be greater than 9.

– The two given 4-bit BCD numbers are to be added using the rules of binary addition.

– If sum is less than or equal to 9 and carry = 0, then no correction is necessary. The sum obtained is correct and in the true BCD form.

– But if sum is invalid BCD or carry = 1, then the result is wrong and needs correction.

– The wrong result can be corrected by adding six i.e. (0110) to it.

Block Diagram of BCD Adder :

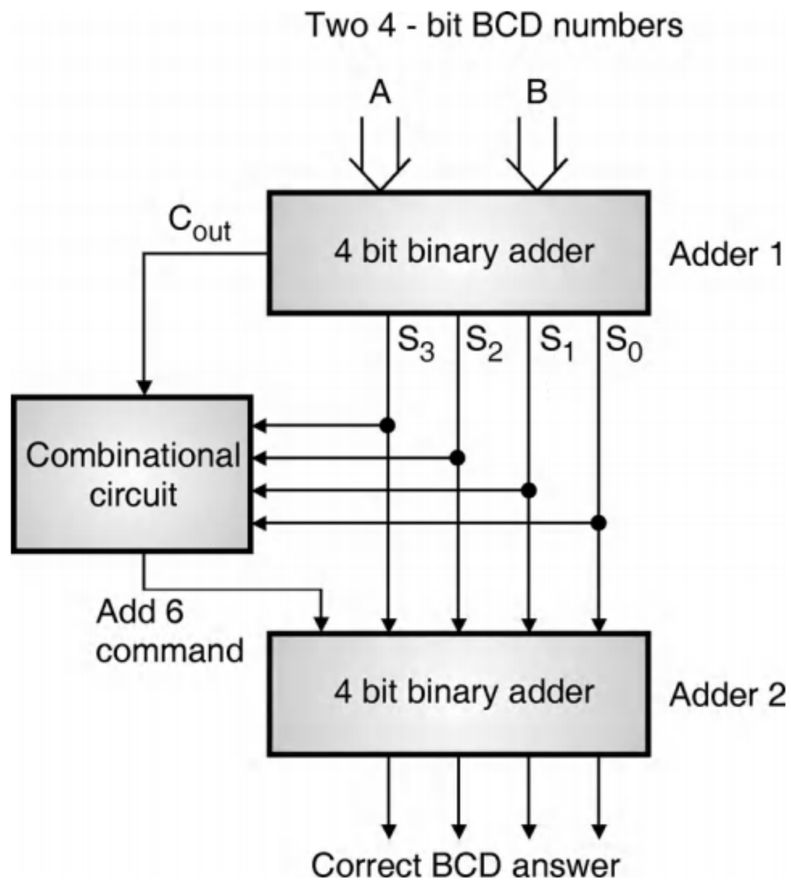
– From the points stated above, we understand that the 4-bit, BCD adder should consist of the following blocks :

1. A 4-bit binary adder to add the given two 4-bit BCD numbers A and B.
2. A combinational circuit to check if sum is greater than 9 or carry = 1.
3. Another 4-bit binary adder to add six (0110) to the incorrect sum if sum > 9 or carry = 1.

– The block diagram of such a BCD adder is shown in Fig. 2.6.4.

– So we have to design the combinational circuit that finds out whether the sum is greater than 9 or carry = 1.

TechKnowledge Publications



(C-397) Fig. 2.6.4 : Block diagram of BCD adder

Design of Combinational Circuit :

– The output of combinational circuit should be 1 if the sum produced by adder 1 is greater than 9 i.e. 1001.

– The truth table is as follows :

(C-6180) Table 2.6.1 : Truth table for combinational circuit design

Sum bits of adder - 1 →

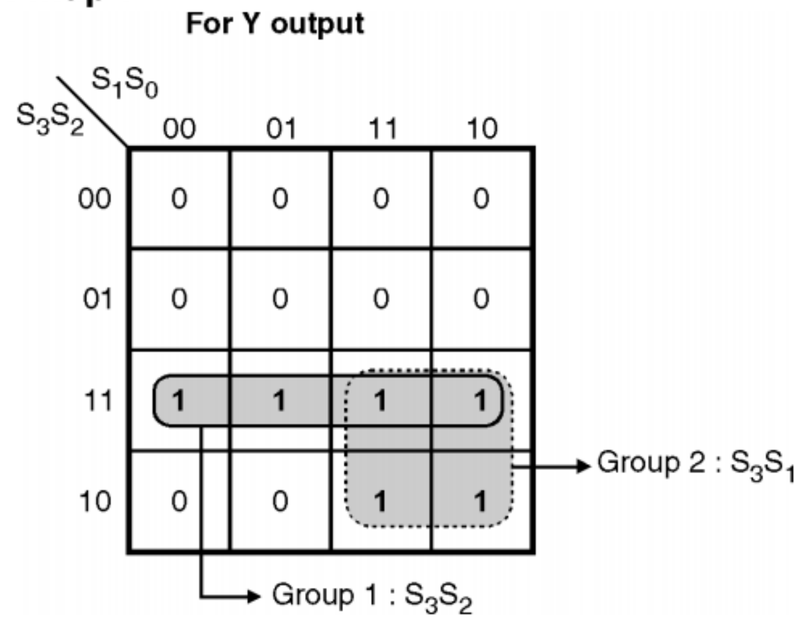
Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Sum is valid BCD number ∴ Y = 0

Sum is invalid BCD number ∴ Y = 1

Write K-map :

For Y output

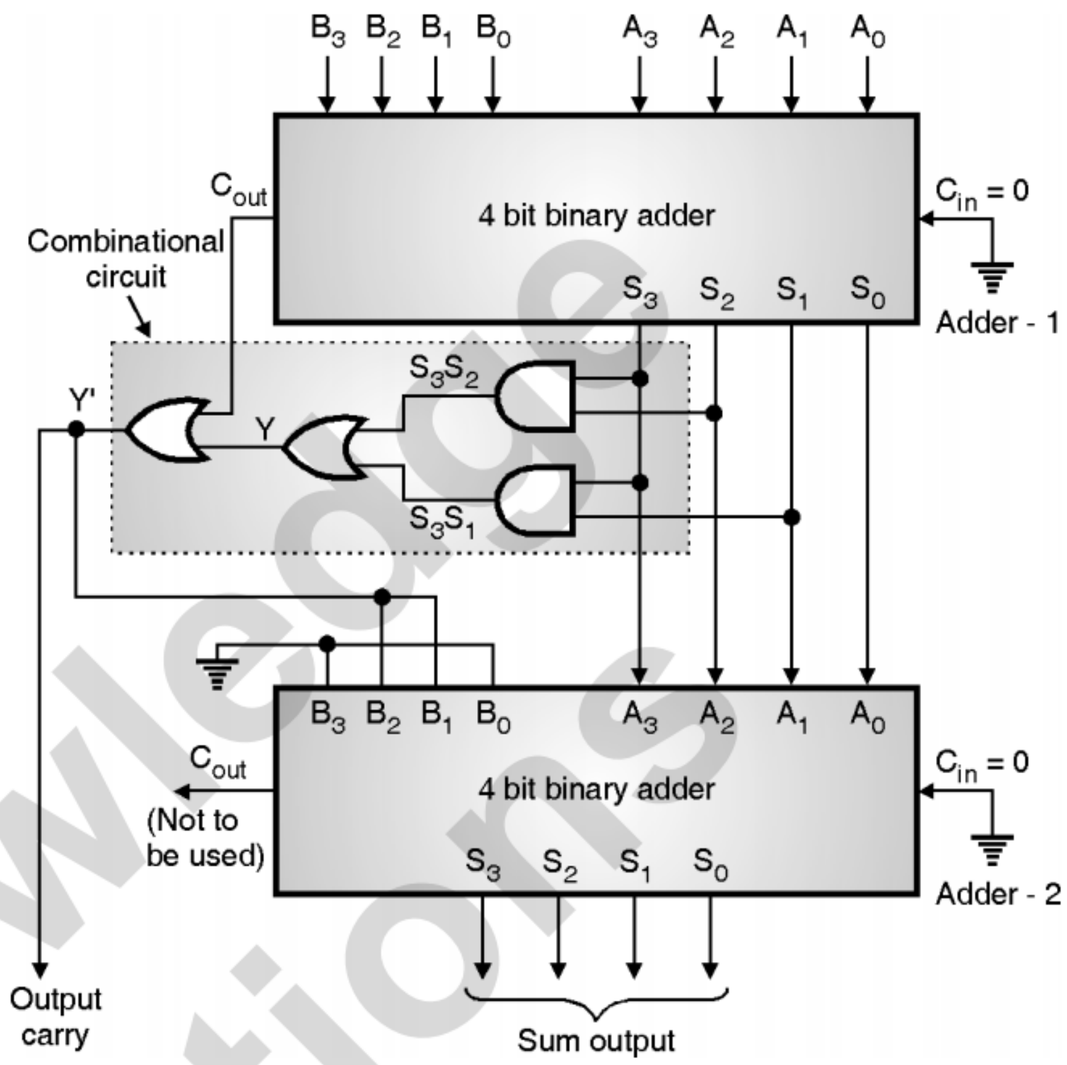


(C-398) Fig. 2.6.5 : K-map for Y output

– The boolean expression is,

$$Y = S_3S_2 + S_3S_1$$

– The complete BCD adder is shown in Fig. 2.6.6.



(C-399) Fig. 2.6.6 : 4-bit BCD adder

– The output of the combinational circuit should be 1 if C_{out} of adder-1 is high or if the output of adder-1 is greater than 9.

– Therefore Y is ORed with C_{out} of adder 1 as shown in Fig. 2.6.6.

– The output of combinational circuit is connected to B_1B_2 inputs of adder-2 and $B_3 = B_1 = 0$ as they are connected to ground permanently.

– This makes $B_3\ B_2\ B_1\ B_0 = 0\ 1\ 1\ 0$ if $Y' = 1$.

– The sum outputs of adder-1 are applied to $A_3A_2A_1A_0$ of adder-2.

– The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.

Operation :

Case I : Sum ≤ 9 and carry = 0

– The output of combinational circuit $Y' = 0$. Hence $B_3\ B_2\ B_1\ B_0 = 0000$ for adder-2.

– Hence output of adder-2 is same as that of adder-1.

TechKnowledge Publications

Case II : Sum > 9 and carry = 0

- If $S_3\ S_2\ S_1\ S_0$ of adder-1 is greater than 9, then output Y′ of combinational circuit becomes 1.

  $\therefore\ B_3\ B_2\ B_1\ B_0\ =\ 0110$ (of adder-2)

- Hence six (0110) will be added to the sum output of adder-1.
- We get the corrected BCD result at the sum output of adder-2.

Case III : Sum ≤ 9 but carry = 1

- As carry output of adder-1 is high, Y′ = 1.

  $\therefore\ B_3\ B_2\ B_1\ B_0\ =\ 0\ 1\ 1\ 0$ (of adder-2)

  $\therefore$ 0 1 1 0 will be added to the sum output of adder-1.

- We get the corrected BCD result at the sum output of adder-2.
- Thus the four bit BCD addition can be carried out using the binary adder.

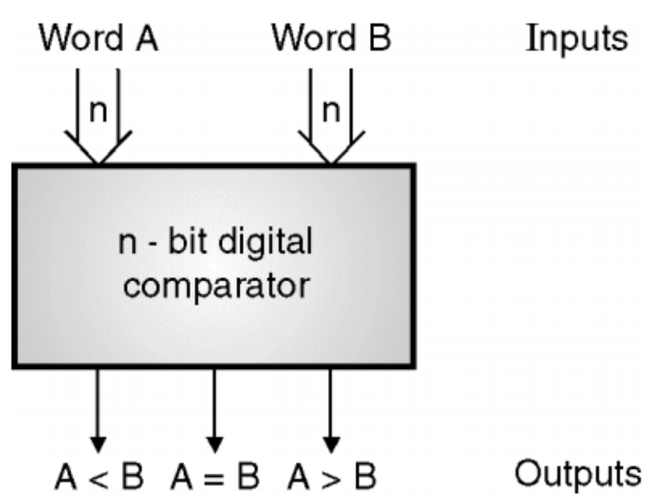
2.7 Magnitude Comparators :

Definition :

- Digital(or magnitude) comparator is a combinational circuit, which compares the two n-bit binary words A and B applied at its input and produces three different outputs : A < B, A = B and A > B.

Block diagram :

- The block diagram of an n bit digital comparator is shown in Fig. 2.7.1.
- The comparator has three outputs namely A > B, A = B and A < B.
- Depending on the result of comparison, one of these outputs will go high.



(C-403) **Fig. 2.7.1 : Block diagram of an n-bit comparator**

2.7.1 1-Bit Binary Comparator :

Definition :

- The one-bit comparator is a combinational logic circuit with two inputs A and B and three outputs namely A < B, A = B and A > B.
- A one bit comparator compares the two single bit numbers A and B and produces an output that indicates the result of the comparison.
- This is clear from the truth table as given in Table 2.7.1.

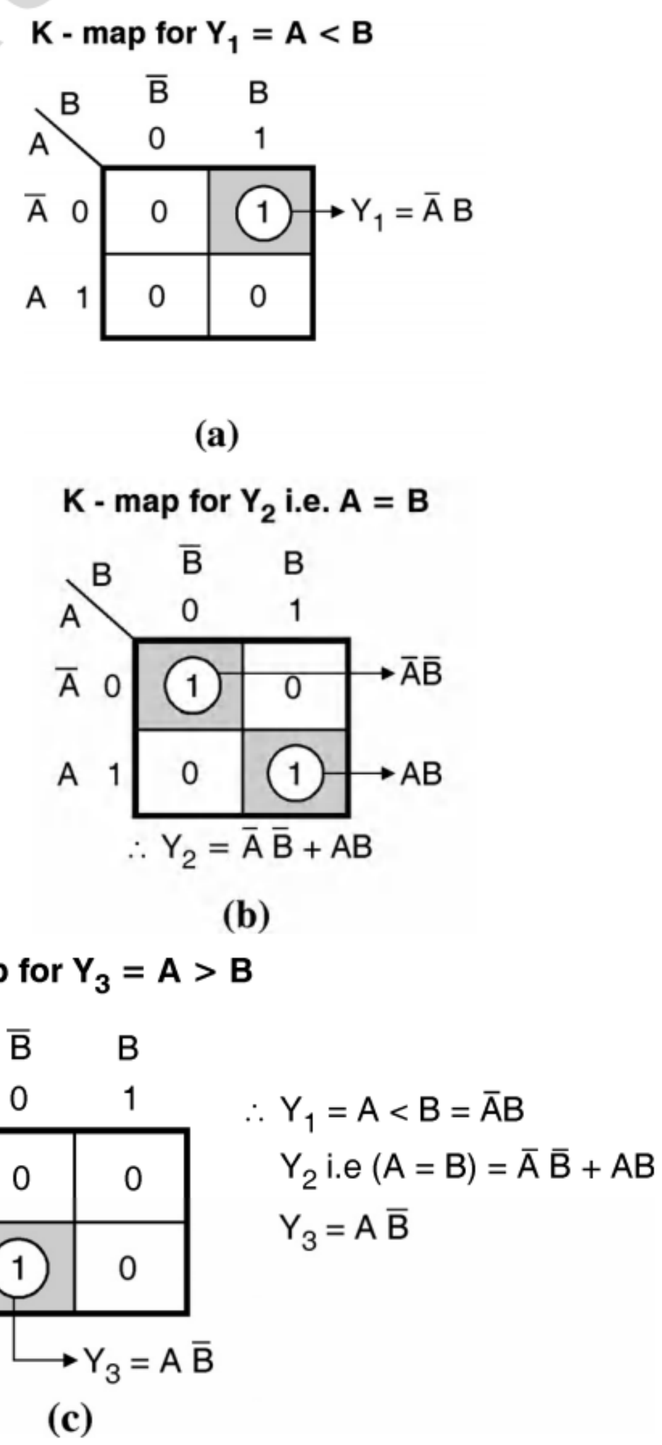
Truth table of one-bit comparator :

(C-8102) **Table 2.7.1 : Truth table of a one-bit comparator**

Inputs		Outputs		
A	B	$Y_1 = A < B$	$Y_2 = A = B$	$Y_3 = A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-maps for each output :

- The K-maps for the three outputs Y_1, Y_2 and Y_3 are as shown in Fig. 2.7.2.



(C-404) **Fig. 2.7.2 : K-maps for the three outputs of a one-bit comparator**

TechKnowledge Publications

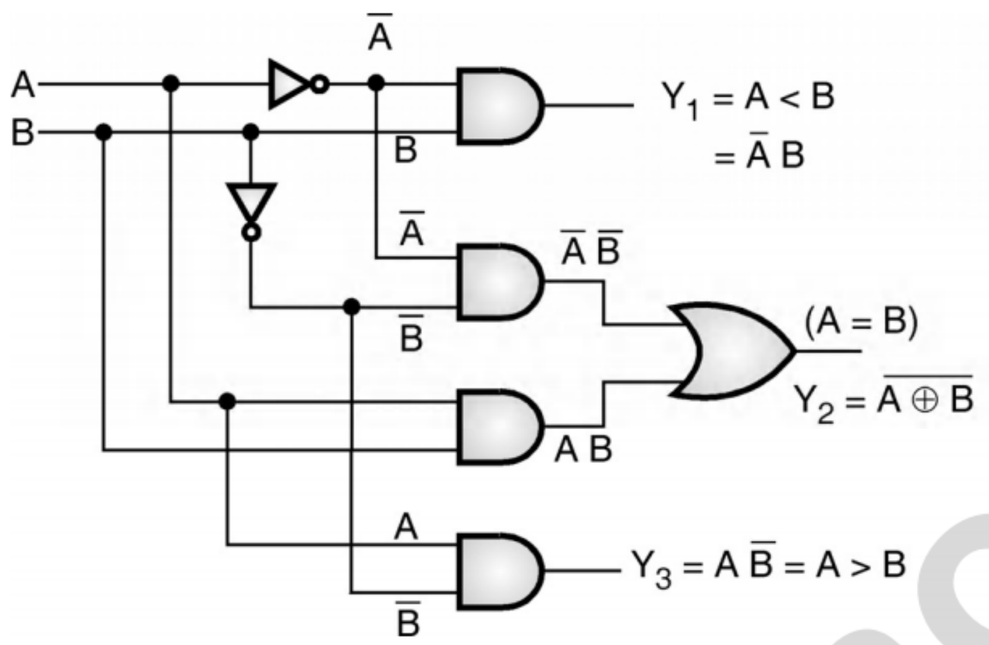
– From Fig. 2.7.2 we can write the expressions for the three outputs as,

$$Y_1 = (A < B) = \bar{A}B$$

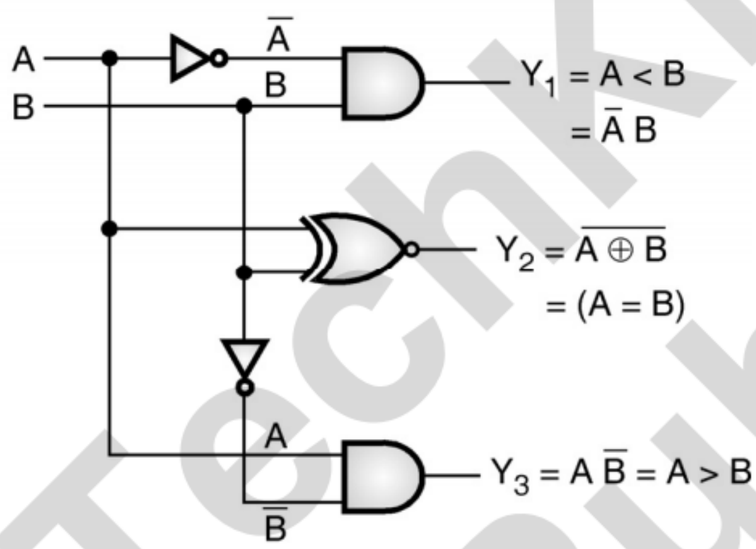
$$Y_2 = (A = B) = \bar{A}\,\bar{B} + AB = \overline{A \oplus B}$$

$$Y_3 = A\bar{B}$$

– The expression for Y_2 is nothing but the expression for an EX-NOR gate. Hence the single bit comparator can be realized using basic gates as shown in Fig. 2.7.3.



(a) Using basic gates



(b) Using AND and EX-NOR gates

(C-405) Fig. 2.7.3 : Realization of one bit comparator

2.7.2 A 2-Bit Comparator :

SPPU : Dec. 12, May 18

University Questions

Q. 1 Design 2-bit magnitude comparator using logic gates. Assume that A and B are 2-bit inputs. The outputs of comparator should be A > B, A = B, A < B. **(Dec. 12, 8 Marks)**

Q. 2 Design two bit comparator using gates (consider A1 MSB and A0 LSB). **(May 18, 4 Marks)**

Truth table :

– For a 2-bit comparator, each input word A and B is 2 bit long.

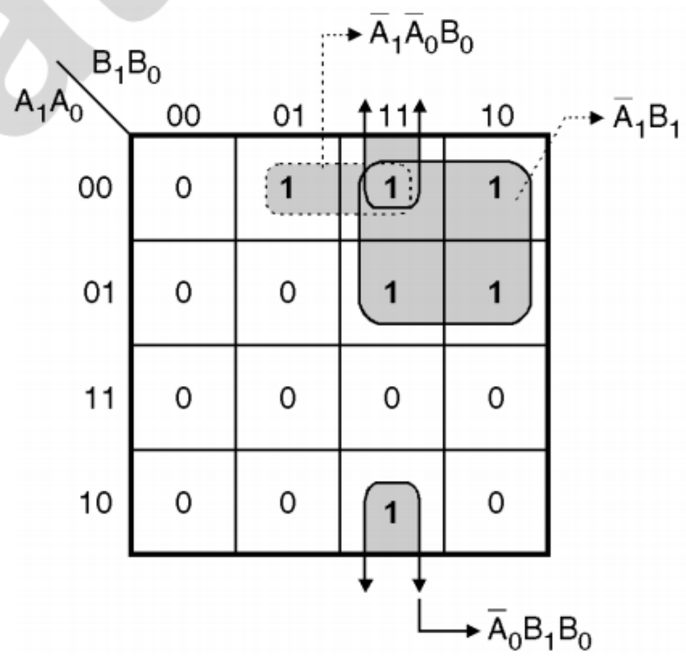
– The truth table of a 2-bit comparator is shown in Table 2.7.2.

(C-406(a)) Table 2.7.2 : Truth table for a 2-bit comparator

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	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

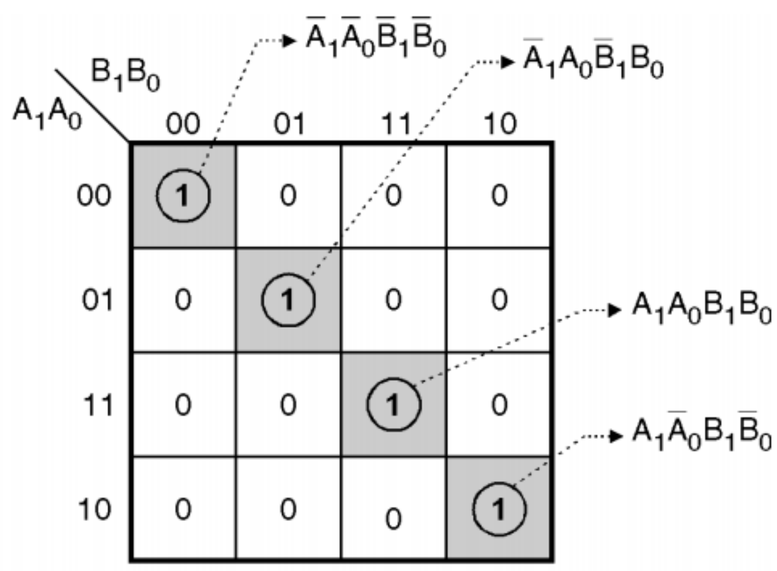
K-maps :

For A < B :



(C-406) Fig. 2.7.4(a) : K-map for output A < B

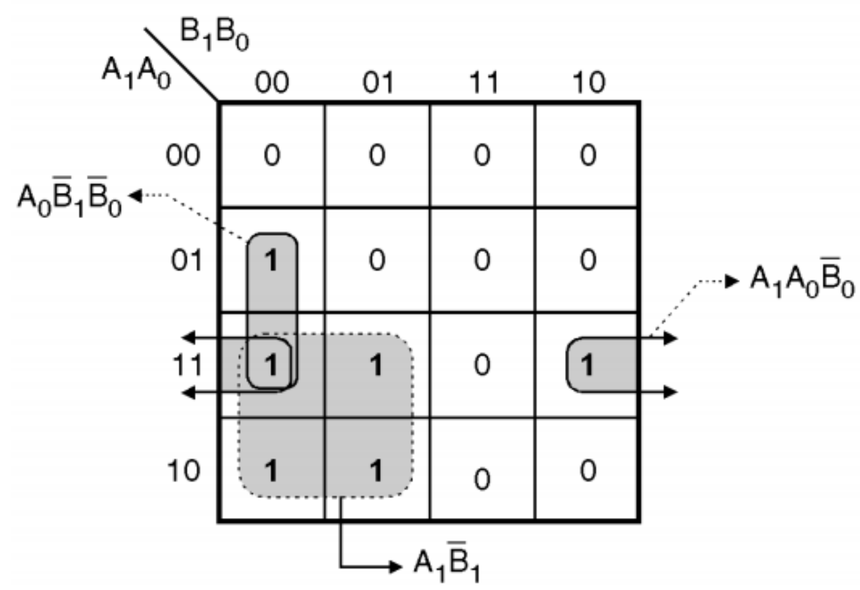
For A = B :



(C-406) Fig. 2.7.4(b) : K-map for output A = B

TechKnowledge Publications

For A > B :



(C-407) Fig. 2.7.4(c) : K-map for output A > B

– The K-maps for the three outputs and corresponding simplified expressions are shown in Figs. 2.7.4(a), (b) and (c).

– From these K-maps we get the simplified expressions for the three outputs of comparator are as follows :

For A < B

– Simplified expression :

$$A < B = \overline{A}_1\,\overline{A}_0\,B_0 + \overline{A}_1\,B_1 + \overline{A}_0\,B_1\,B_0$$

For A > B

– Simplified expression :

$$A > B = A_0\,\overline{B}_1\,\overline{B}_0 + A_1\,A_0\,\overline{B}_0 + A_1\,\overline{B}_1$$

Simplification for output A = B :

– Refer the K-map for output A = B shown in Fig. 2.7.4(b).

– The expression for A = B is given by,

$$(A = B) = \overline{A}_1\,\overline{A}_0\,\overline{B}_1\,\overline{B}_0 + \overline{A}_1\,A_0\,\overline{B}_1 B_0 + A_1\,A_0\,B_1\,B_0 + A_1\overline{A}_0\,B_1\,\overline{B}_0$$

$$= \overline{A}_0\,\overline{B}_0\,(\overline{A}_1\,\overline{B}_1 + A_1\,B_1) + A_0\,B_0\,(\overline{A}_1\,\overline{B}_1 + A_1 B_1)$$

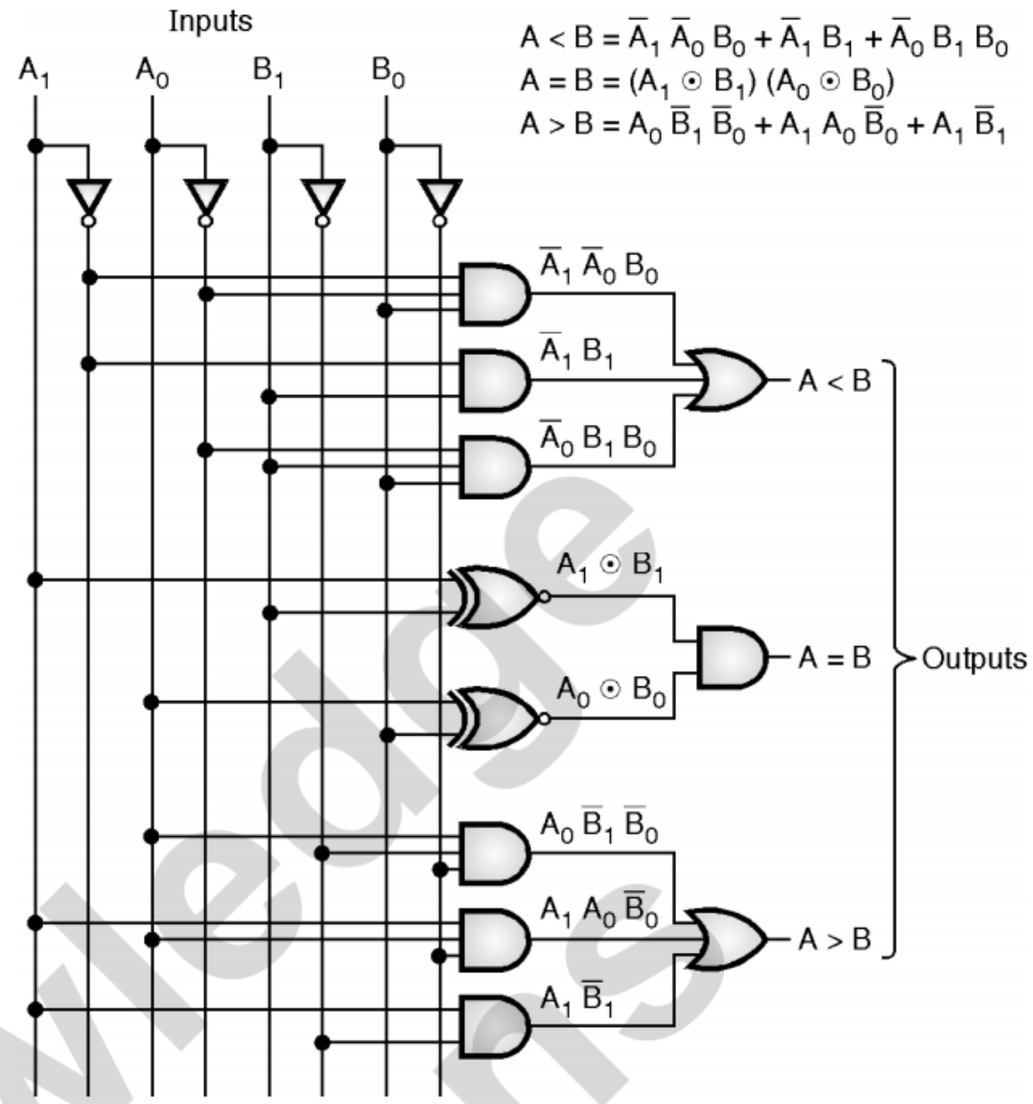
$$= (\overline{A}_1\overline{B}_1 + A_1\,B_1)\,(\overline{A}_0\,\overline{B}_0 + A_0\,B_0)$$

$\therefore$ (A = B) = $(A_1 \odot B_1)\,(A_0 \odot B_0)$ where $\odot$ = EX-NOR

Logic diagram for a two bit comparator :

– The logic diagram for the 2-bit digital comparator is shown in Fig. 2.7.5.

– This diagram is drawn by referring to the simplified expressions of the outputs.



(C-408) Fig. 2.7.5 : Logic diagram for 2-bit comparator

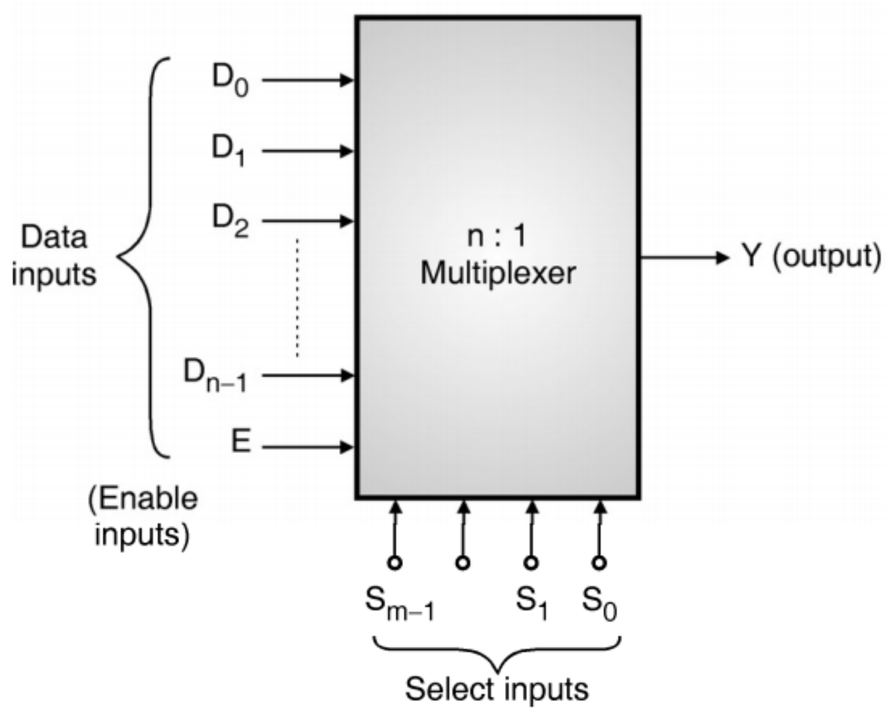
2.8 Multiplexer (Data Selector) :

Definition and block diagram :

– A multiplexer is a combinational circuit with n data inputs one output and m select inputs, which selects one of the n data inputs and routes (connects) it to the output.

– The selection of one of the n inputs is done with the help of the select inputs.

– Multiplexer is a special type of combinational circuit. The block diagram of an n-to-1 multiplexer is shown in Fig. 2.8.1(a).



(C-413) Fig. 2.8.1(a) : Block diagram of an n : 1 multiplexer

– As shown, there are n-data inputs (D_0, D_1, ...D_{n-1}), one output Y and m select inputs, S_0, S_1 .....S_{m-1}.

TechKnowledge Publications

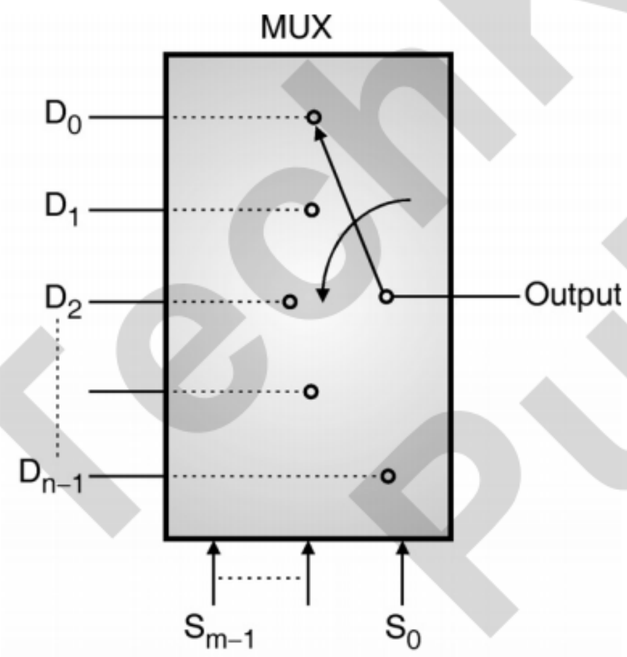
- The relation between the number of data inputs (n) and number of select inputs (m) is as follows :

$$n = 2^m$$

- A multiplexer is a digital circuit which selects one of the n data inputs and routes (connects) it to the output.
- The selection of one of the n inputs is done with the help of the select inputs.
- To select n inputs we need m select lines such that $2^m = n$.
- Depending on the digital code applied at the select inputs, one out of n data sources is selected and transmitted to the single output Y.
- E is called as a strobe or enable input which is useful for cascading.
- It is generally an active low terminal, that means it will perform the required operation when it is low.

Equivalent Circuit :

- The equivalent circuit of an n:1 multiplexer is as shown in Fig. 2.8.1(b).
- As shown in Fig. 2.8.1(b) the multiplexer acts like a digitally controlled single pole, multiple way switch.



(C-413) Fig. 2.8.1(b) : Equivalent circuit of a multiplexer

- The output gets connected to only one of the n data inputs at given instant of time via the single pole multiple throw rotary switch.
- The position of the rotary arm will be dependent on the status of the m select inputs. It will connect the output to the selected input.

2.8.1 Necessity of Multiplexers :

- In most of the electronic systems, the digital data is available from more than one sources. It is necessary to route this data over a single line.
- Under such circumstances we require a circuit which selects one of the many sources at a time.
- This circuit is nothing else but a multiplexer, which has many inputs, one output and some select inputs.
- Multiplexer improves the reliability of the digital system because it reduces the number of external wired connections.

2.8.2 Advantages of Multiplexers :

1. It reduces the number of wires, required to be used.
2. A multiplexer reduces the circuit complexity and cost.
3. We can implement many combinational circuits using MUX.
4. It simplifies the logic design.
5. It does not need the K maps for simplification.

2.9 Types of Multiplexers :

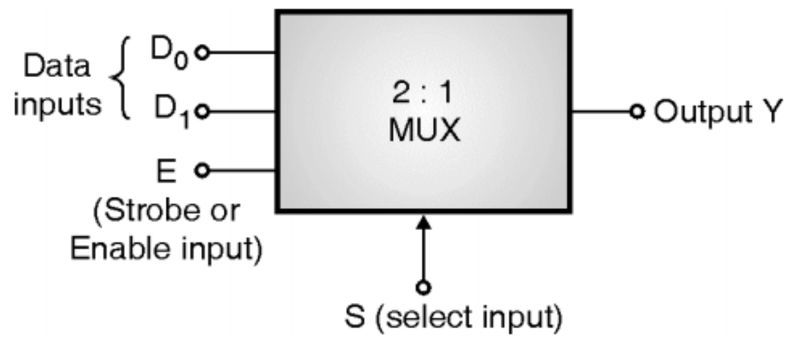
- The types of multiplexers are :

1. 2 : 1 multiplexer.
2. 4 : 1 multiplexer.
3. 8 : 1 multiplexer.
4. 16 : 1 multiplexer.
5. 32 : 1 multiplexer.

2.9.1 2 : 1 Multiplexer :

Block diagram :

- The block schematic of a 2 : 1 multiplexer is shown in Fig. 2.9.1(a). It has two data inputs D_0 and D_1, one select input S, an enable input and one output.
- The truth table of this MUX is shown in Fig. 2.9.1(b).



(a) Block diagram

Enable	Select Input S	Output Y
0	X	0
1	0	D_0
1	1	D_1

X = Don't care

(b) Truth table

(C-414) (a) Fig. 2.9.1 : 2 : 1 multiplexer

TechKnowledge Publications

Truth table :

– Write a more elaborate truth table for 2 : 1 MUX as shown in Table 2.9.1.

(C-414(a)) Table 2.9.1 : Truth table of a 2 : 1 MUX

Enable E	Select S	D_1	D_0	Output Y
0	×	×	×	0
1	0	×	0	0
1	0	×	1	1
1	1	0	×	0
1	1	1	×	1

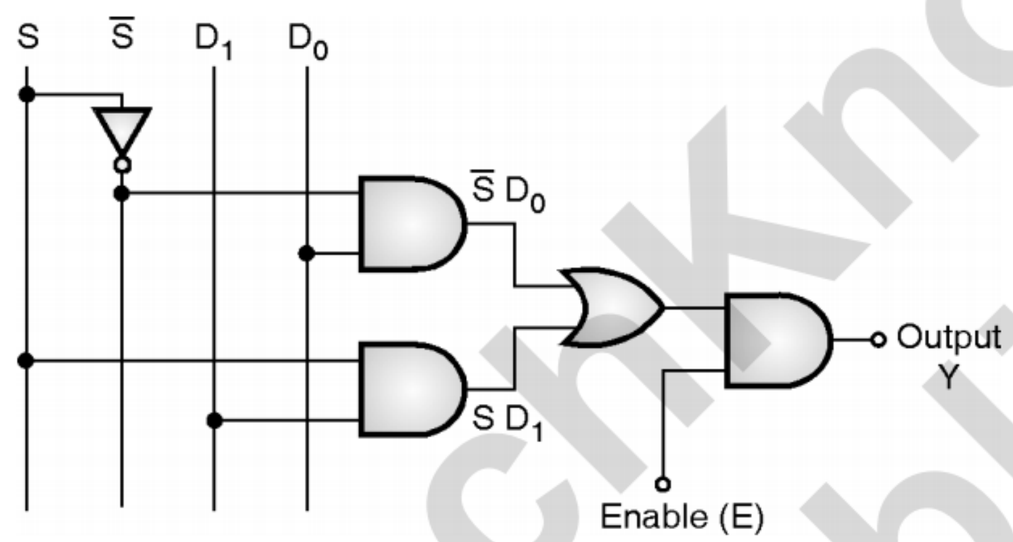
– From the truth table it is clear that Y = 1 for only two conditions shown by the shaded boxes.

– We can write down the Boolean expression by taking into consideration these two conditions as follows :

$$\therefore Y = E\overline{S}D_0 + ESD_1 = E(\overline{S}D_0 + SD_1)$$

Realization of a 2 : 1 MUX using gates :

– The realization using gates is shown in Fig. 2.9.2.



(C-415) Fig. 2.9.2 : Realization of 2 : 1 MUX using gates

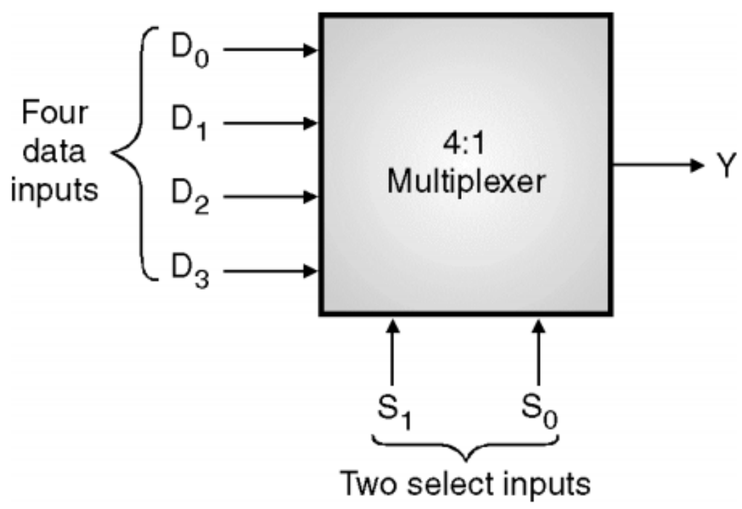
2.9.2 A 4 : 1 Multiplexer : SPPU : May 07

University Questions

Q. 1 Draw basic circuit diagram of 4 : 1 MUX and its truth table. **(May 07, 6 Marks)**

Block diagram and truth table :

– Fig. 2.9.3 shows the block diagram of a 4 : 1 multiplexer and Table 2.9.2 gives its truth table.



(C-416) Fig. 2.9.3 : 4 : 1 multiplexer

(C-416)(a) Table 2.9.2 : Truth table

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

– Note that n = 4 hence number of select lines i.e. m = 2 so that 2^m = n. The strobe terminal (G) has not been included to reduce the complexity.

– The truth table tells us that if $S_1 S_0$ = 00, the data bit D_0 is selected and routed to output.

$\therefore$ Y = D_0 .... when $S_1 S_0$ = 00

– Similarly if $S_1 S_0$ = 01, then D_1 is selected and routed to the output.

$\therefore$ Y = D_1 .... when $S_1 S_0$ = 01

Y = D_2 for $S_1 S_0$ = 10 and

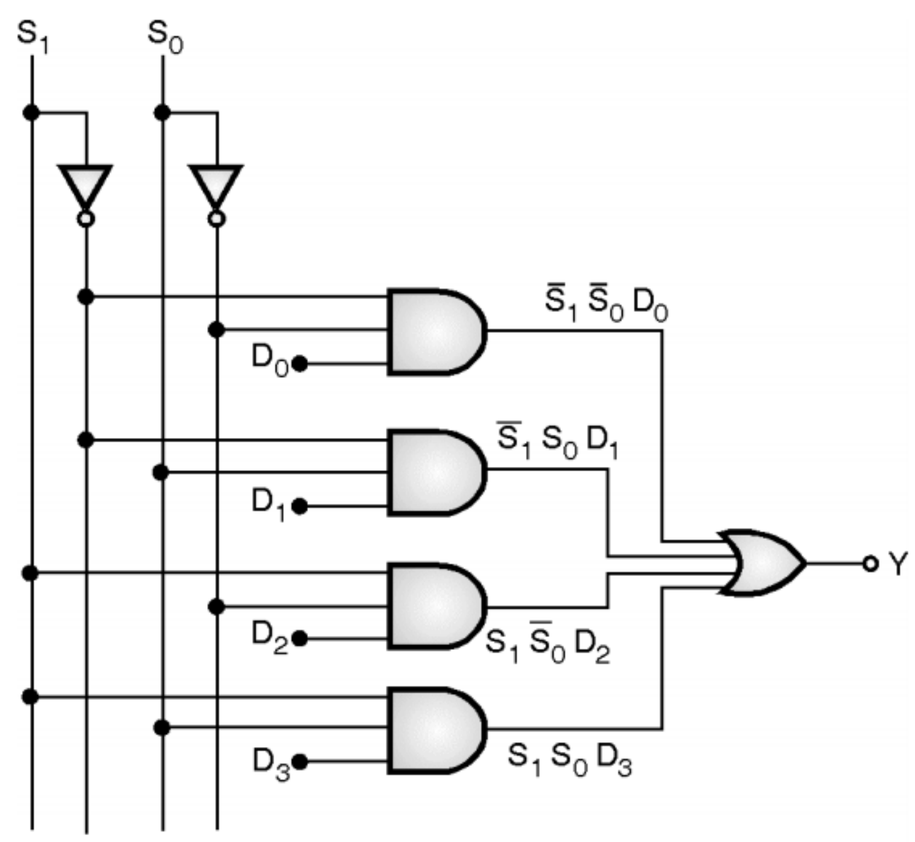
Y = D_3 for $S_1 S_0$ = 11

– The output will be high when the selected input (D_0 , D_1 etc.) is 1. Hence the logical expression for output in the SOP form is as follows :

$$Y = \overline{S}_1\overline{S}_0 D_0 + \overline{S}_1 S_0 D_1 + S_1\overline{S}_0 D_2 + S_1 S_0 D_3$$

Realization of a 4 : 1 MUX using gates :

– This expression can be realized using basic gates as shown in Fig. 2.9.4.



$$Y = \overline{S}_1\overline{S}_0D_0 + \overline{S}_1S_0D_1 + S_1\overline{S}_0D_2 + S_1S_0D_3$$

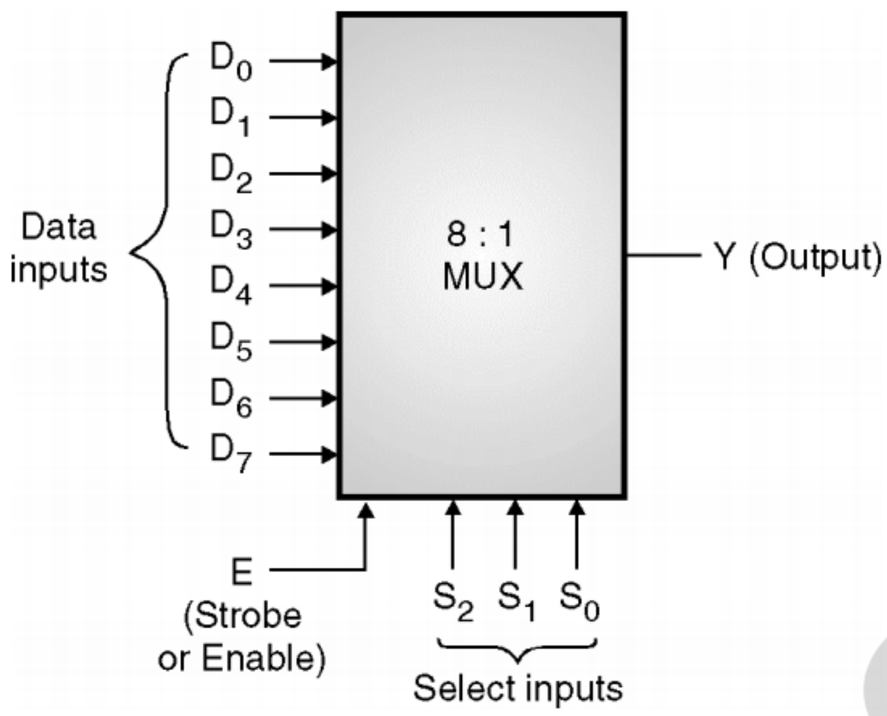
(C-417) Fig. 2.9.4 : Realization of 4 : 1 multiplexer using basic gates

TechKnowledge Publications

2.9.3 8 : 1 Multiplexer :

Block diagram and truth table :

- The block diagram of an 8 : 1 MUX is shown in Fig. 2.9.5(a) and its truth table is shown in Fig. 2.9.5(b).
- It has eight data inputs, one enable input, three select inputs and one output.



(a) Block diagram

Enable E	Select input S_2	S_1	S_0	Output Y
0	×	×	×	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

(b) Truth table

(C-419(a)) **Fig. 2.9.5 : 8 : 1 multiplexer**

Operating principle :

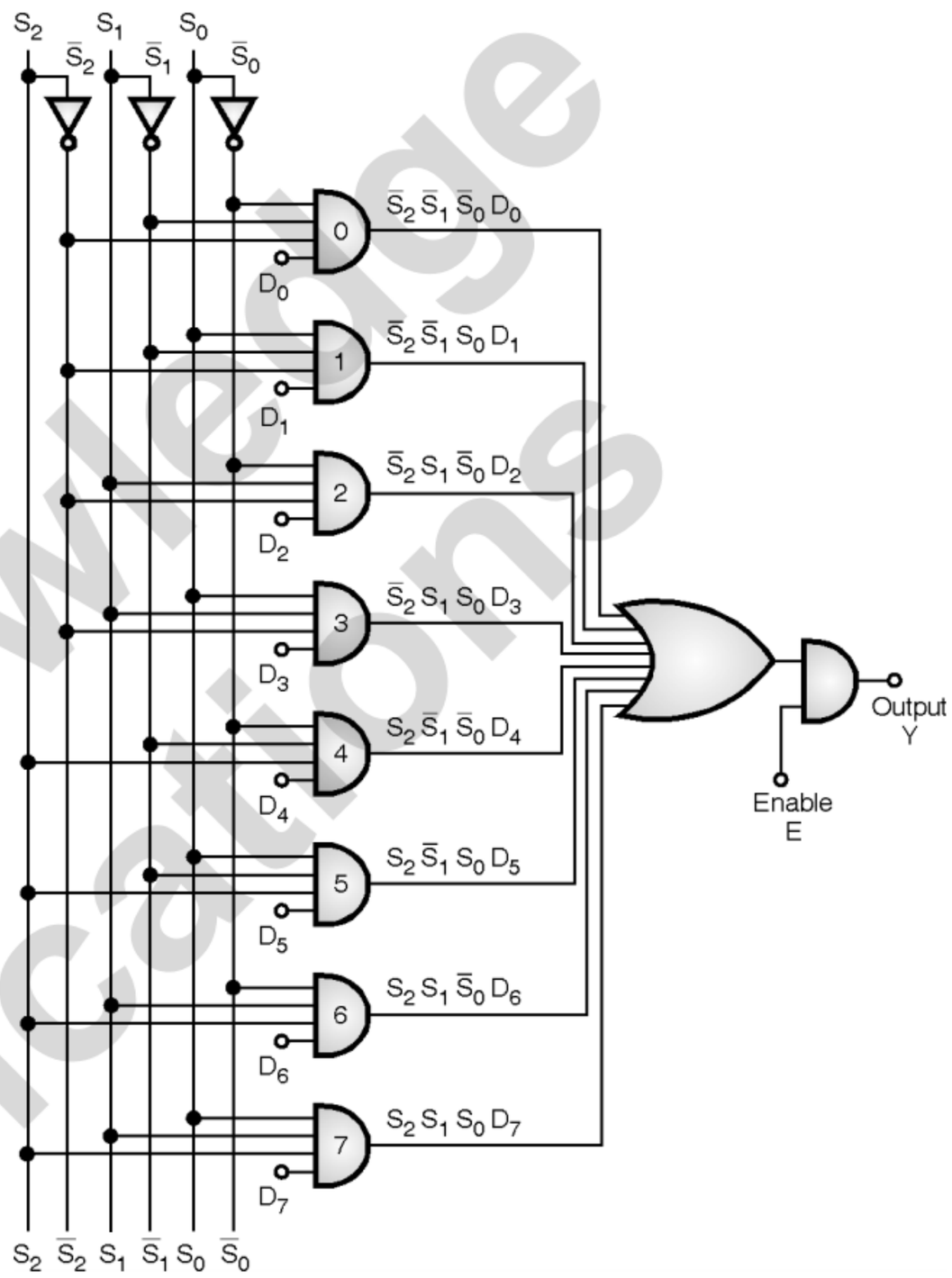
- When the strobe or enable input is 0, the output of the multiplexer will be 0 irrespective of any input.
- With E = 1, we can select any one of the eight data inputs and connect it to the output.
- For example if $S_2\ S_1\ S_0$ = 0 1 1 then the data input D_3 is selected and output Y will follow the selected input D_3.

Realization using gates :

- The expression for output Y can be obtained from the truth table of Fig. 2.9.5(b) as,

$$Y = E \cdot [\overline{S}_2\ \overline{S}_1\ \overline{S}_0\ D_0 + \overline{S}_2\ \overline{S}_1\ S_0 D_1 + \overline{S}_2\ S_1\ \overline{S}_0\ D_2 + \overline{S}_2\ S_1\ S_0\ D_3 + S_2\ \overline{S}_1\ \overline{S}_0\ D_4 + S_2\ \overline{S}_1\ S_0\ D_5 + S_2\ S_1\ \overline{S}_0\ D_6 + S_2\ S_1\ S_0 D_7]$$

- The realization of 8 : 1 MUX using gates is shown in Fig. 2.9.6.



(C-420) **Fig. 2.9.6 : 8 : 1 MUX using gates**

2.9.4 Applications of a Multiplexer :

SPPU : Dec. 06

University Questions

Q. 1 Explain any one application of multiplexer. **(Dec. 06, 8 Marks)**

- Some of the important applications of a multiplexer are as follows :

1. It is used as a data selector to select one out of many data inputs.
2. It is used for simplification of logic design.
3. In the data acquisition system.
4. In designing the combinational circuits.
5. In the D/A converters.
6. To minimize the number of connections in a logic circuit.

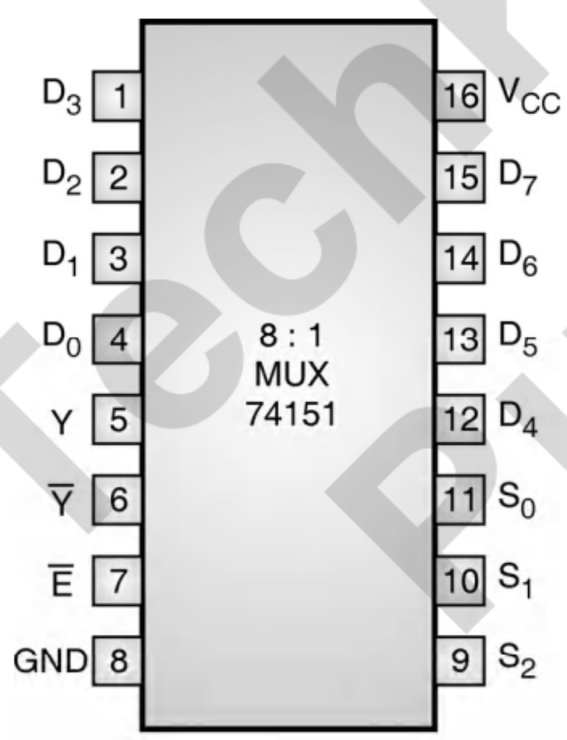
TechKnowledge Publications

2.10 Study of Different Multiplexer ICs :

- The multiplexer ICs available in market are given in Table 2.10.1.
- Out of these the two most commonly used multiplexer ICs are : 74150 (16 : 1 multiplexer) and 74151 (8 : 1 multiplexer), and 74153 (Dual 4:1 multiplexer).
- The pin configuration for these ICs is shown in Fig. 2.10.1.

Table 2.10.1

IC Number	Description	Output
74157	Quad 2 : 1 Mux	Same as input (No inversion)
74158	Quad 2 : 1 Mux	Inverted output
74153	Dual 4 : 1 Mux	Same as input (No inversion)
74352	Dual 4 : 1 Mux	Inverted output
74151A	8 : 1 Mux	Complementary outputs
74152	8 : 1 Mux	Inverted output
74150	16 : 1 Mux	Inverted output



$D_0 - D_{15}$: Data inputs
$S_0 - S_3$: Select inputs
$\overline{E}$: Active low enable input
$\overline{Y}$ = Complemented output

(C-422(a)) Fig. 2.10.1 : Pin configuration of 74151

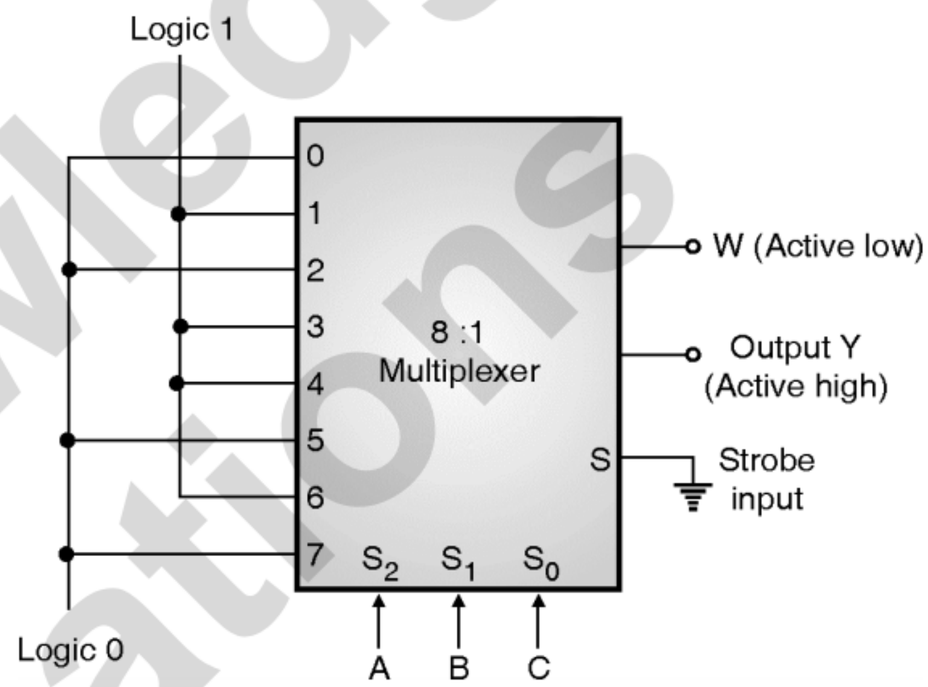
2.10.1 8 : 1 Multiplexer (74151) :

- There are 8-data inputs (D_0 to D_7), three select inputs (S_0 to S_2), a pair of outputs (Y and $\overline{Y}$) i.e. complementary are outputs and an active low enable input ($\overline{E}$).

54LS151 / DM54LS151 / DM74LS151 (8 : 1 MUX) :

- This is an 8 : 1 multiplexer or data selector. It has a strobe (enable) input which must be at logic 0 to enable the multiplexer because it is an active low input.
- This IC has two outputs, Y and W. Out of which W is the inverted output.
- If the strobe input goes high (it is inactive) then it will force Y output to logic 0 and W output to logic 1.
- The Y and W are complementary outputs. Fig. 2.10.2 shows the function diagram and Table 2.10.2 shows its truth table.

Connection diagram :



(C-423) Fig. 2.10.2 : Function diagram of 54151

(C-8132) Table 2.10.2 : Truth table of 54151

Inputs			Strobe	Outputs	
Select				Y	W
C	B	A	S		
X	X	X	1	0	1
0	0	0	0	D_0	$\overline{D}_0$
0	0	1	0	D_1	$\overline{D}_1$
0	1	0	0	D_2	$\overline{D}_2$
0	1	1	0	D_3	$\overline{D}_3$
1	0	0	0	D_4	$\overline{D}_4$
1	0	1	0	D_5	$\overline{D}_5$
1	1	0	0	D_6	$\overline{D}_6$
1	1	1	0	D_7	$\overline{D}_7$

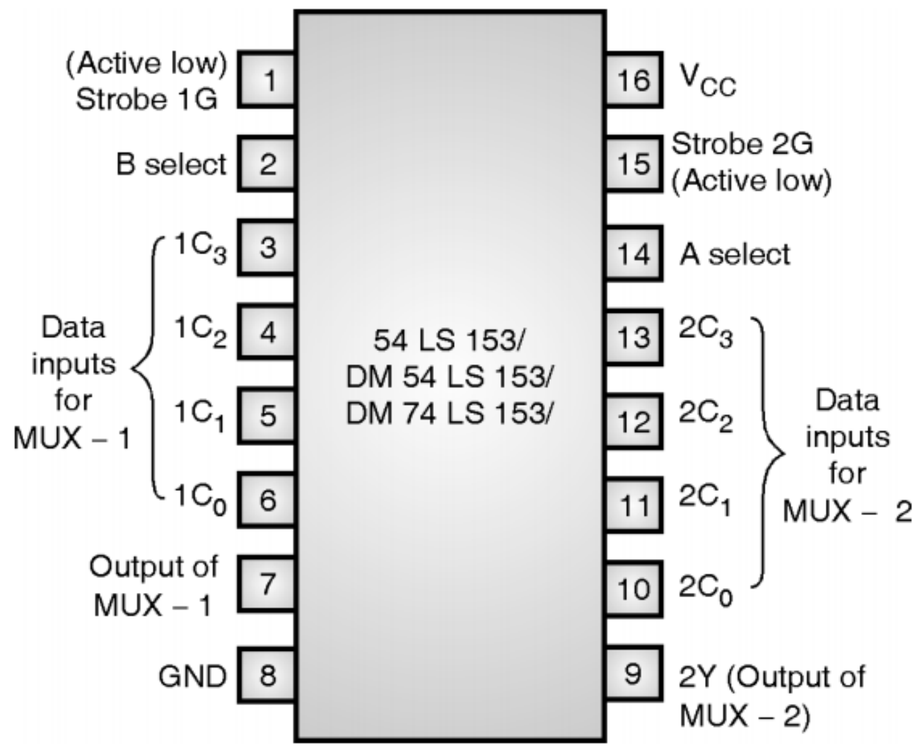
Features :

- Select one-of-eight data lines.
- Performs parallel-to-series conversion.
- Allows multiplexing from N lines to one line.
- It can be used as Boolean function generator.
- Typical average propagation delay time data input to W output 12.5 nS.
- Typical power dissipation 30 mW.

Tech Knowledge Publications

2.10.2 54LS 153/DM 54LS 153/DM 74LS 153 (Dual 4 : 1 Multiplexer) :

– The block diagram of these multiplexers is shown in Fig. 2.10.3(a).



(C-3578) Fig. 2.10.3(a) : Pin configuration of Dual 4 : 1 multiplexer IC 54LS 153 / DM 54LS 153 / DM 74LS 153

– These chips consist of two 4 : 1 multiplexers. Each multiplexer has a separate strobe (enable) input.

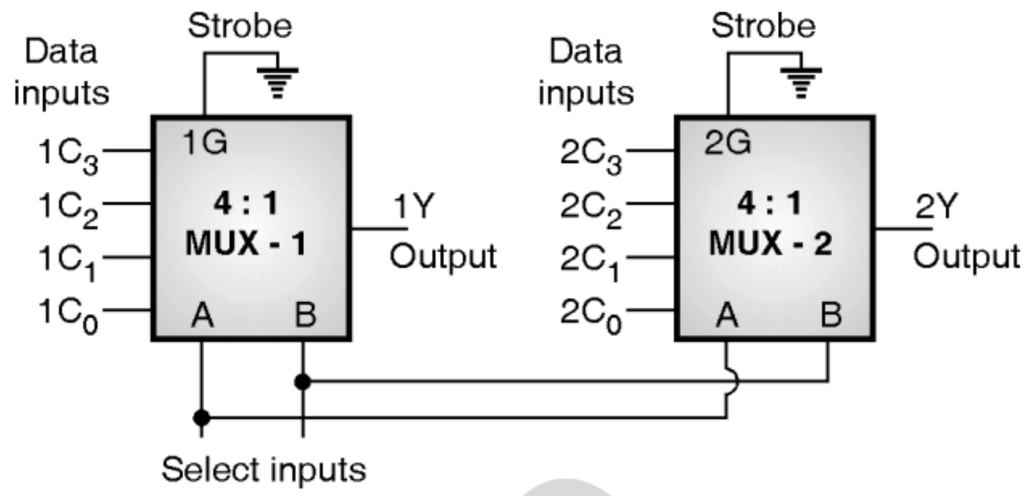
– There are two select inputs A and B which are common to both the 4 : 1 multiplexers inside the ICs.

– The strobe lines 1G and 2G are the active low enable lines for the two 4 : 1 multiplexer inside the IC. The truth table for each 4 : 1 MUX is shown in Table 2.10.3.

(C-6185) Table 2.10.3 : Truth table of 4:1 MUX

	Select inputs		Data inputs				Strobe	Output
	B	A	C_0	C_1	C_2	C_3	G	Y
	X	X	X	X	X	X	1	0
C_0 selected	0	0	0	X	X	X	0	0
	0	0	1	X	X	X	0	1
C_1 selected	0	1	X	0	X	X	0	0
	0	1	X	1	X	X	0	1
C_2 selected	1	0	X	X	0	X	0	0
	1	0	X	X	1	X	0	1
C_3 selected	1	1	X	X	X	0	0	0
	1	1	X	X	X	1	0	1

– The functional diagram of IC 74153 is as shown in Fig. 2.10.3(b).



(C-3579) Fig. 2.10.3(b) : Functional diagram of IC-74153

– The function table is shown in Table 2.10.4.

(C-8133) Table 2.10.4 : Function table of a IC-74153

Inputs			Output
Select inputs		Strobe	Y
B	A	$\overline{G}$	
X	X	1	0
0	0	0	C_0
0	1	0	C_1
1	0	0	C_2
1	1	0	C_3

2.11 Multiplexer Tree / Cascading of Multiplexer :

– The multiplexers having more number of inputs can be obtained by cascading two or more multiplexers with less number of inputs.

– This is called as a multiplexer tree. This concept will be clear after solving the following examples.

Ex. 2.11.1 : Obtain an 8 : 1 multiplexer using two 4 : 1 multiplexers.

Soln. :

– The cascading of two 4 : 1 multiplexer results in 8 : 1 multiplexer as shown in Fig. P. 2.11.1.

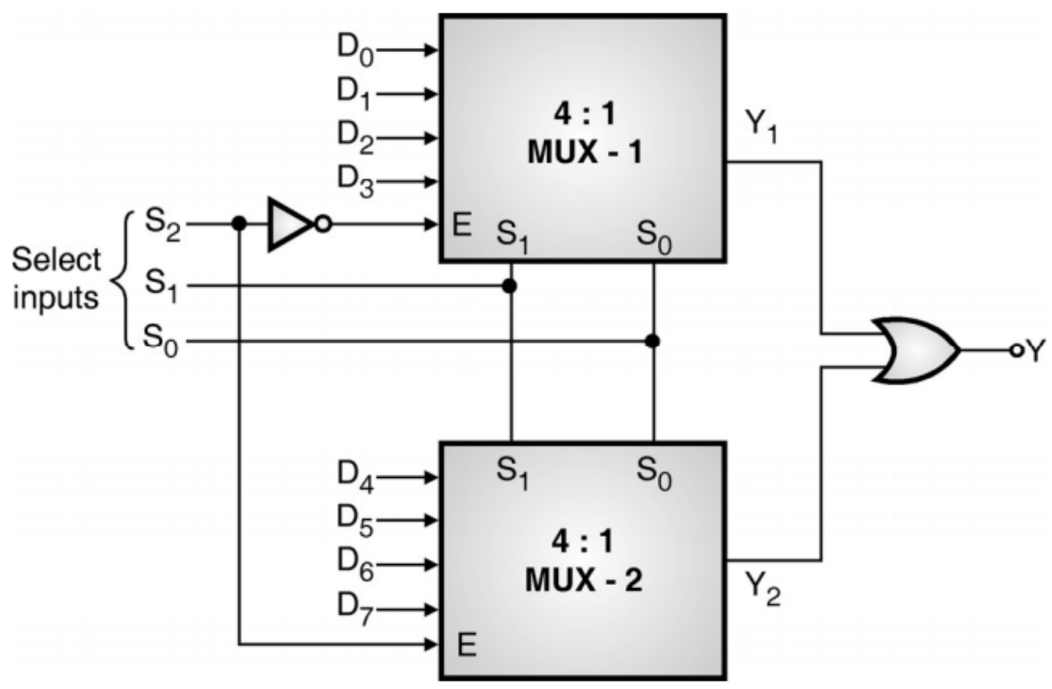
– There are in all eight data inputs (D_0 through D_7).

– The select lines S_1 and S_0 of both 4 : 1 multiplexers are connected in parallel whereas a third select input S_2 is used for enabling one multiplexer at a time.

– S_2 is connected directly to the enable (E) terminal of MUX-1 whereas $\overline{S}_2$ is connected to the enable terminal of MUX-2.

TechKnowledge Publications

– The outputs of the two multiplexers are ORed to obtain the final output Y.



(C-424) Fig. P. 2.11.1 : 8 : 1 multiplexer by cascading two 4 : 1 multiplexers

(C-6186) Table P. 2.11.1 : Truth table

Select inputs			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

MUX 1 is enable

MUX 2 is enable

Ex. 2.11.2 : Implement a 16 : 1 multiplexer using 4 : 1 multiplexers.

Dec. 05, 6 Marks, Dec. 07, 7 Marks, Dec. 08, May 11, 8 Marks, Dec. 16, 6 Marks

Soln. :

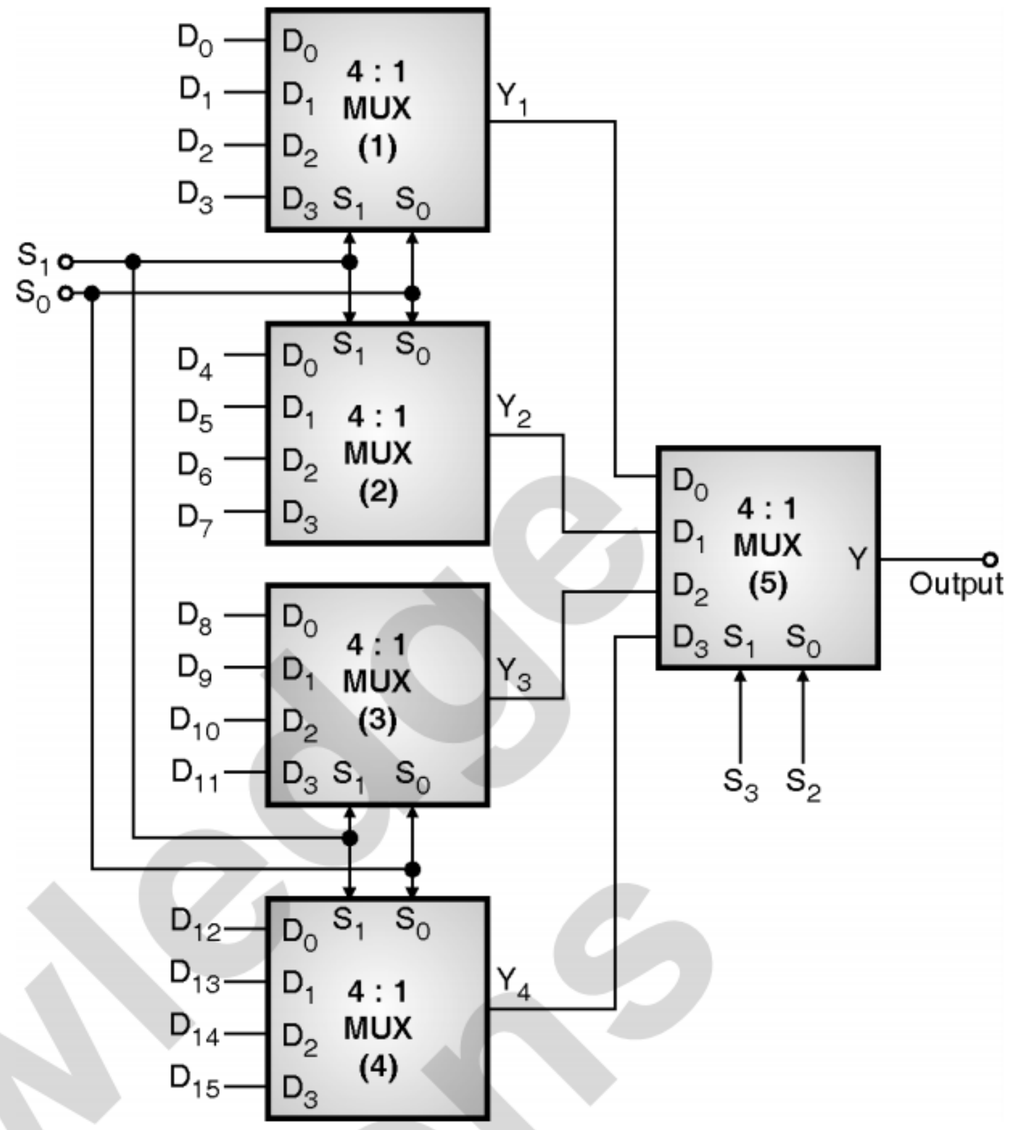
– Refer Fig. P. 2.11.2 for implementation.

Description :

– The select inputs S_1 and S_0 of the multiplexers 1, 2, 3 and 4 are connected together.

– The select inputs S_3 and S_2 are applied to the select inputs S_1 and S_0 of MUX-5.

– The outputs Y_1 ,Y_2 ,Y_3 ,Y_4 are applied to the data inputs D_0, D_1 , D_2 and D_3 of MUX-5 as shown in Fig. P. 2.11.2.



(C-425) Fig. P. 2.11.2 : 16 : 1 multiplexer using 4 : 1 multiplexers

– The operation can be summarized using Table P. 2.11.2.

(C-6187) Table P. 2.11.2 : Summary of operation

Select inputs				Mux. Outputs				Final Output
S_3	S_2	S_1	S_0	Y_1	Y_2	Y_3	Y_4	Y
0	0	0	0	**D_0**	D_4	D_8	D_{12}	**D_0**
0	0	0	1	**D_1**	D_5	D_9	D_{13}	**D_1**
0	0	1	0	**D_2**	D_6	D_{10}	D_{14}	**D_2**
0	0	1	1	**D_3**	D_7	D_{11}	D_{15}	**D_3**
0	1	0	0	D_0	**D_4**	D_8	D_{12}	**D_4**
0	1	0	1	D_1	**D_5**	D_9	D_{13}	**D_5**
0	1	1	0	D_2	**D_6**	D_{10}	D_{14}	**D_6**
0	1	1	1	D_3	**D_7**	D_{11}	D_{15}	**D_7**
1	0	0	0	D_0	D_4	**D_8**	D_{12}	**D_8**
1	0	0	1	D_1	D_5	**D_9**	D_{13}	**D_9**
1	0	1	0	D_2	D_6	**D_{10}**	D_{14}	**D_{10}**
1	0	1	1	D_3	D_7	**D_{11}**	D_{15}	**D_{11}**
1	1	0	0	D_0	D_4	D_8	**D_{12}**	**D_{12}**
1	1	0	1	D_1	D_5	D_9	**D_{13}**	**D_{13}**
1	1	1	0	D_2	D_6	D_{10}	**D_{14}**	**D_{14}**
1	1	1	1	D_3	D_7	D_{11}	**D_{15}**	**D_{15}**

$S_3S_2 = 00$ ∴ MUX-5 selects Y_1

$S_3S_2 = 01$ ∴ MUX-5 selects Y_2

$S_3S_2 = 10$ ∴ MUX-5 selects Y_3

$S_3S_2 = 11$ ∴ MUX-5 selects Y_4

Ex. 2.11.3 : Design 12 : 1 mux using 4 : 1 multiplexers (with enable inputs). Explain the truth table of your circuit in short. **Dec. 09, 8 Marks**

Soln. :

– The 12:1 MUX using 4 : 1 MUXes is as shown in Fig. P. 2.11.3 and its truth table is as shown in Table P. 2.11.3.

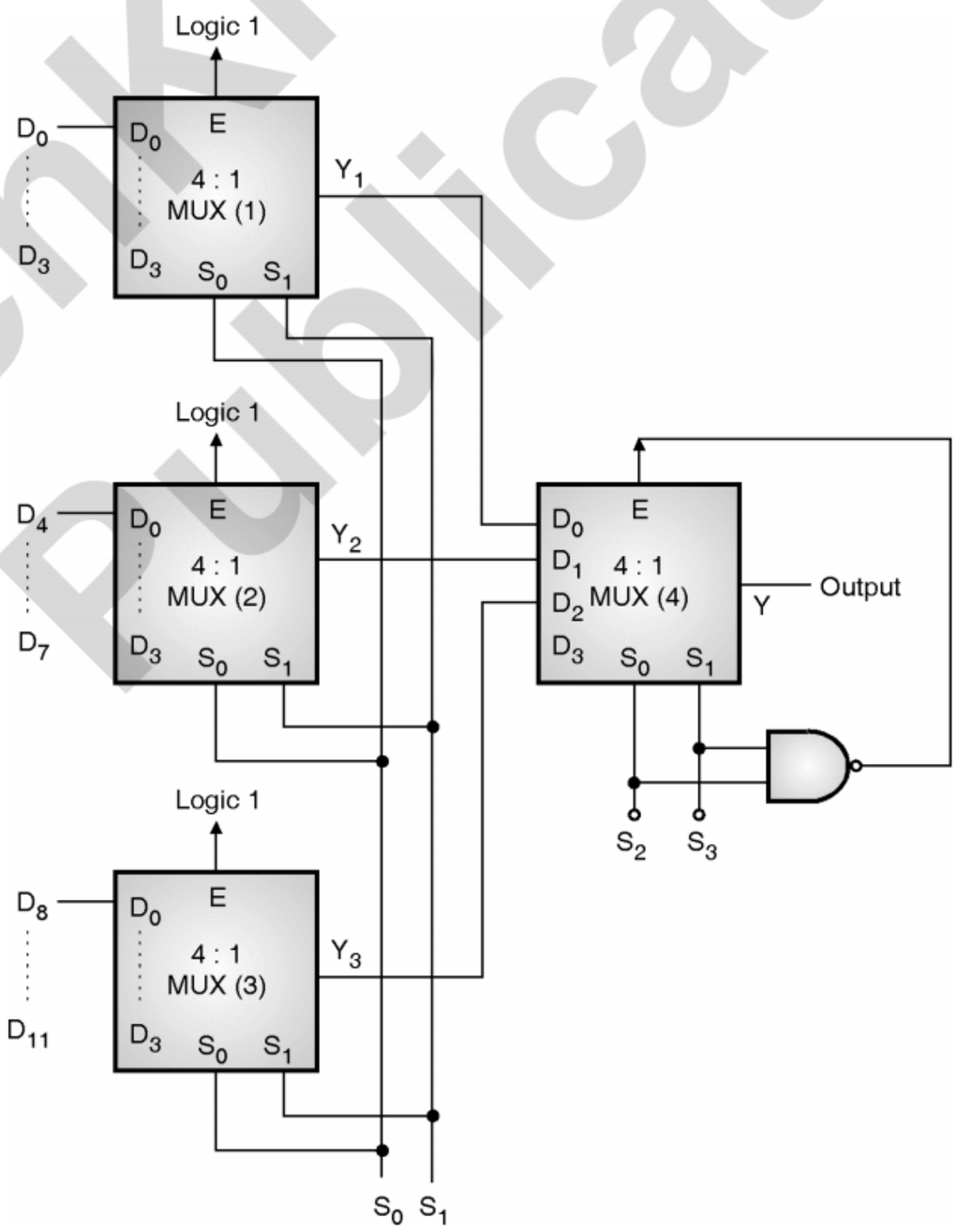
TechKnowledge Publications

(C-8134) Table P. 2.11.3 : Truth table

Select inputs				MUX outputs			Final output Y	Comment
S_3	S_2	S_1	S_0	Y_1	Y_2	Y_3		
0	0	0	0	D_0	D_4	D_8	D_0	$S_3S_2 = 00$ ∴ MUX–4 select Y_1
0	0	0	1	D_1	D_5	D_9	D_1	
0	0	1	0	D_2	D_6	D_{10}	D_2	
0	0	1	1	D_3	D_7	D_{11}	D_3	
0	1	0	0	D_0	D_4	D_8	D_4	$S_3S_2 = 01$ ∴ MUX–4 select Y_2
0	1	0	1	D_1	D_5	D_9	D_5	
0	1	1	0	D_2	D_6	D_{10}	D_6	
0	1	1	1	D_3	D_7	D_{11}	D_7	
1	0	0	0	D_0	D_4	D_8	D_8	$S_3S_2 = 10$ ∴ MUX–4 select Y_3
1	0	0	1	D_1	D_5	D_9	D_9	
1	0	1	0	D_2	D_6	D_{10}	D_{10}	
1	0	1	1	D_3	D_7	D_{11}	D_{11}	
1	1	0	0	X	X	X	0	$S_3S_2 = 11$ ∴ MUX–4 disabled.
1	1	0	1	X	X	X	0	
1	1	1	0	X	X	X	0	
1	1	1	1	X	X	X	0	

– The enable input E of the first three 4:1 multiplexers are connected to logic 1 permanently in order to enable them.

– The select lines S_0 and S_1 respectively of MUX 1, 2 and 3 are connected together as shown in Fig. P. 2.11.3.



(C-1708) Fig. P. 2.11.3 : 12:1 multiplexer using 4:1 multiplexers

TechKnowledge Publications

Ex. 2.11.4 : Design 14 : 1 mux using 4 : 1 mux (with enable inputs). Explain the truth table of your circuit in short. **May 10, 8 Marks**

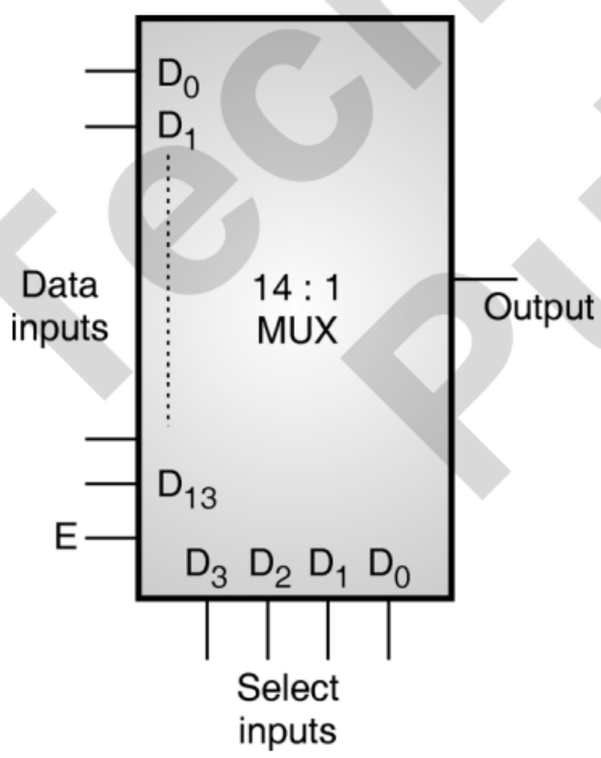
Soln. :

– The truth table of 14 : 1 MUX is as shown below :

(C-8135) Table P. 2.11.4 : Truth table of a 14 : 1 MUX

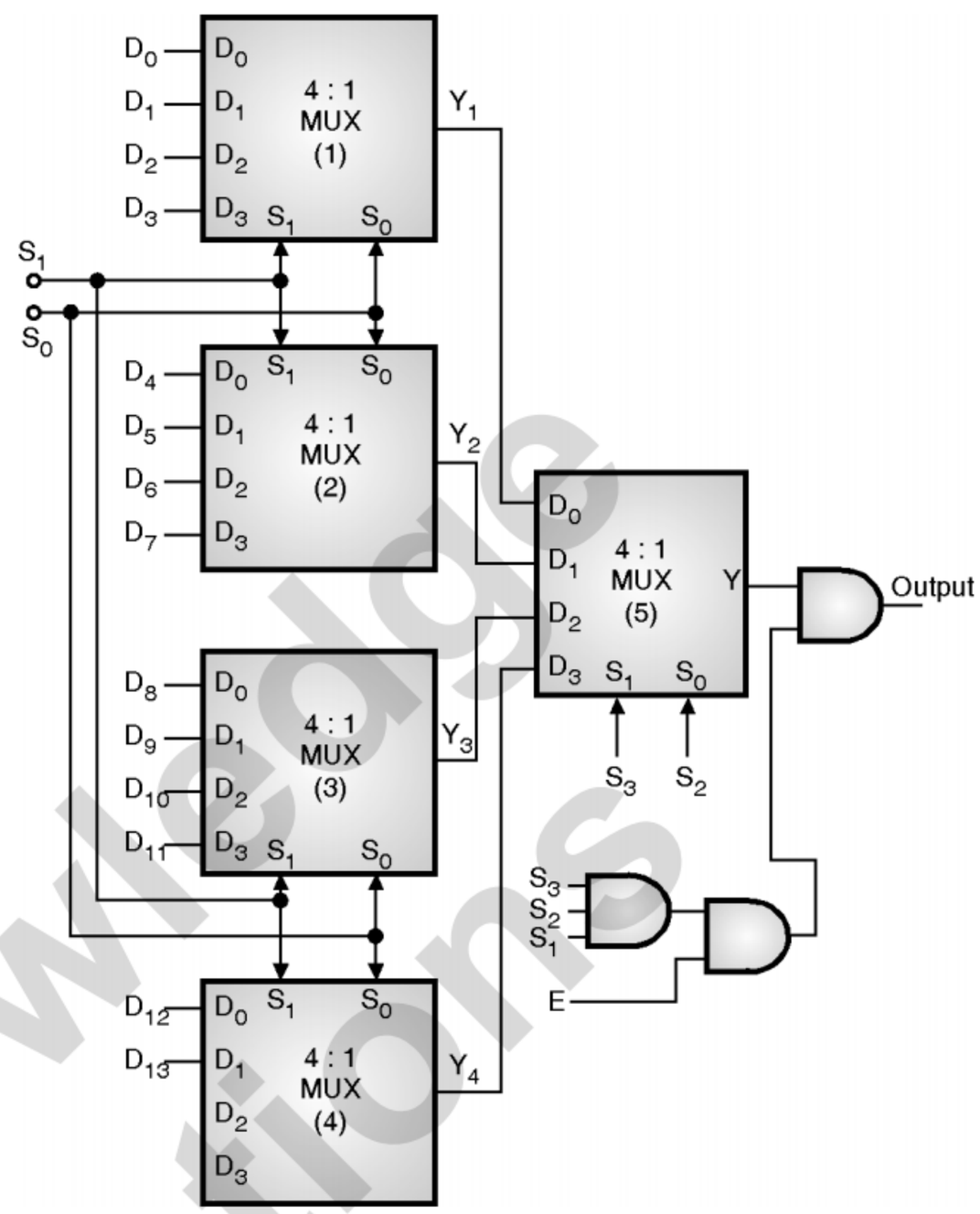
Enable input E	Select inputs S_3	S_2	S_1	S_0	Output Y
0	X	X	X	X	0
1	0	0	0	0	D_0
1	0	0	0	1	D_1
1	0	0	1	0	D_2
1	0	0	1	1	D_3
1	0	1	0	0	D_4
1	0	1	0	1	D_5
1	0	1	1	0	D_6
1	0	1	1	1	D_7
1	1	0	0	0	D_8
1	1	0	0	1	D_9
1	1	0	1	0	D_{10}
1	1	0	1	1	D_{11}
1	1	1	0	0	D_{12}
1	1	1	0	1	D_{13}
X	1	1	1	0	0
X	1	1	1	1	0

Output is 0 for the invalid combinations of select inputs



(C-1711) Fig. P. 2.11.4(a)

– The realization of 14 : 1 MUX using 4 : 1 MUX is shown in Fig. P. 2.11.4(b).



(C-1712) Fig. P. 2.11.4(b) : 14 : 1 MUX using 4 : 1 multiplexers

Ex. 2.11.5 : Design 28 : 1 mux using 8 : 1 mux (with enable inputs). Explain truth table of your design in short. [Hint : you can use separate mux for enable of respective IC's].

Dec. 10, 8 Marks

Soln. :

– In order to design a 28 : 1 MUX we will use four (4) 8 : 1 multiplexers and one 4 : 1 multiplexer.

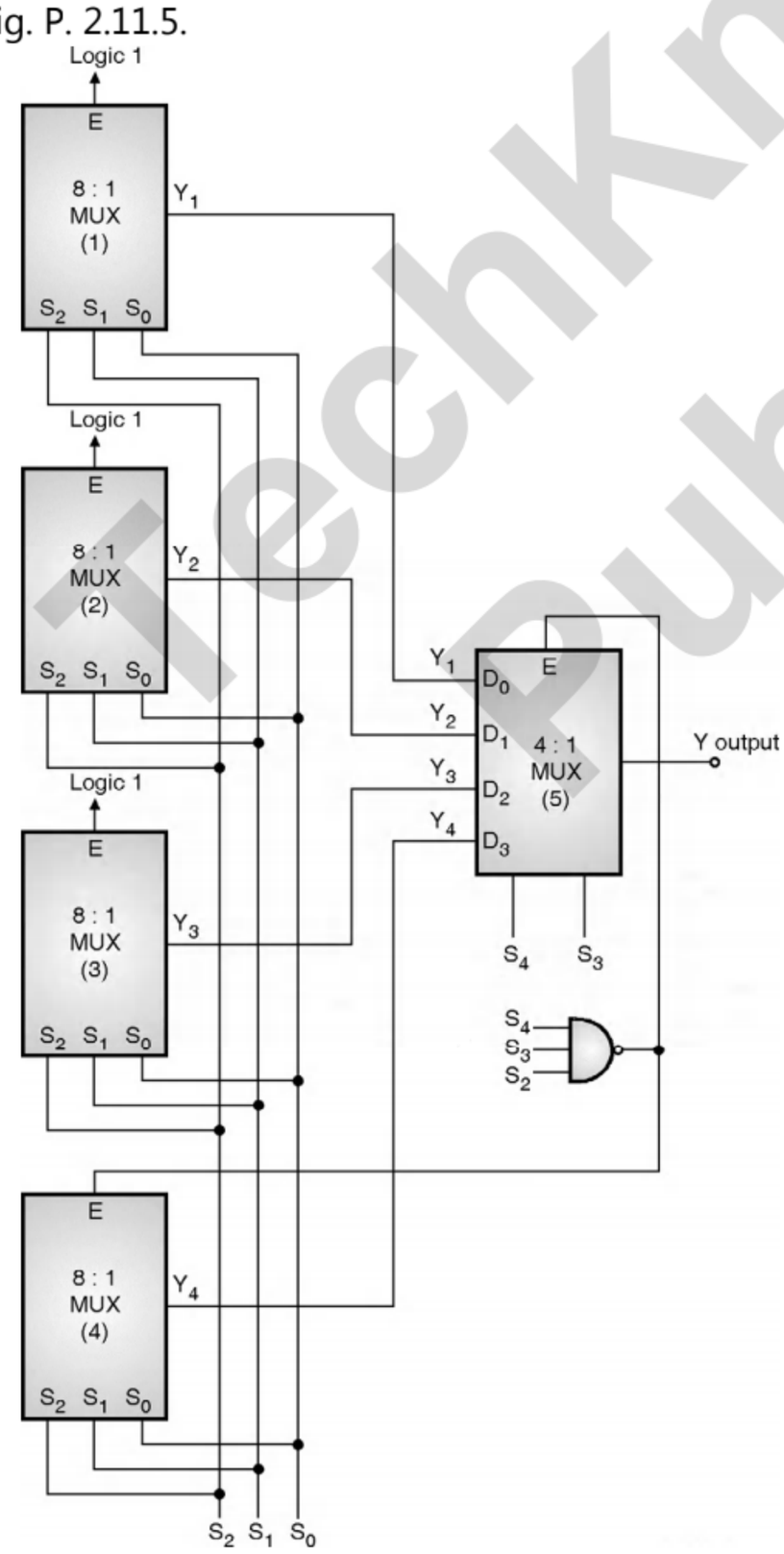
– The truth table is as follows :

TechKnowledge Publications

(C-8136) Table P. 2.11.5 : Truth table

Select inputs					MUX outputs				Final output Y	Comment
S_4	S_3	S_2	S_1	S_0	Y_1	Y_2	Y_3	Y_4		
0	0	0	0	0	D_0	D_8	D_{16}	D_{24}	D_0	$S_4 S_3 = 00$
0	0	0	0	1	D_1	D_9	D_{17}	D_{25}	D_1	∴ MUX 5 select Y_1
		⋮			⋮	⋮	⋮	⋮	⋮	
0	0	1	1	1	D_7	D_{15}	D_{23}	D_{31}	D_7	∴ $Y = Y_1$
0	1	0	0	0	D_0	D_8	D_{16}	D_{24}	D_8	$S_4 S_3 = 01$
0	1	0	1	1	⋮	D_9	D_{17}	D_{25}	D_9	∴ MUX 5 select Y_2
		⋮			⋮	⋮	⋮	⋮	⋮	
0	1	1	1	1	D_7	D_{15}	D_{23}	D_{31}	D_{15}	∴ $Y = Y_2$
1	0	0	0	0	D_0	D_8	D_{16}	D_{24}	D_{16}	$S_4 S_3 = 10$
1	0	0	0	1	D_1	D_9	D_{17}	D_{25}	D_{17}	∴ MUX 5 select Y_3
		⋮			⋮	⋮	⋮	⋮	⋮	
1	0	1	1	1	D_7	D_{15}	D_{23}	D_{31}	D_{23}	∴ $Y = Y_3$
1	1	0	0	0	D_0	D_8	D_{17}	D_{24}	D_{24}	$S_4 S_3 = 11$
1	1	0	0	1	D_1	D_9	D_{18}	D_{25}	D_{25}	∴ MUX 5 select Y_4
1	1	0	1	0	D_2	D_{10}	D_{19}	D_{26}	D_{26}	
1	1	0	1	1	D_3	D_{11}	D_{20}	D_{27}	D_{27}	∴ $Y = Y_4$
1	1	1	0	0	D_4	D_{12}	D_{21}	MUX	–	MUX 4 and 5 are disabled
1	1	1	0	1	D_5	D_{13}	D_{22}	4 and	–	
1	1	1	1	0	D_6	D_{14}	D_{23}	5 are	–	
1	1	1	1	1	D_7	D_{15}	D_{24}	Disabled	–	

– The realization using 8 : 1 multiplexers is as shown in Fig. P. 2.11.5.



(C-1716) Fig. P. 2.11.5 : Realization of 28 : 1 MUX

– Note that when $S_4 = S_3 = S_2 = 1$, the enable (E) of MUX 4 and 5 goes to 0 to disable those multiplexers.

2.12 Use of Multiplexers in Combinational Logic Design :

– Multiplexers ICs for 2 : 1 , 4 : 1, 8 : 1 and 16 : 1 multiplexers are available.

– We can use them to implement the given Boolean expressions representing the combinational circuits.

– Thus it is possible to design and implement many combinational circuits by using multiplexer and a few logic gates.

Advantages :

– Advantages of using multiplexers for logic design are as follows :

1. Logic design is simplified.
2. It is not necessary to simplify the logic expression.
3. It minimizes the number of ICs required to be used.

2.12.1 Implementation of a Logical Expression in the Standard SOP Form :

Use of MUX for combinational circuit design :

1. A truth table or logic expression in standard SOP or POS form is given to us.
2. We have to follow the design procedure given below to use MUX for implementing the given logical expression.

TechKnowledge Publications

Design procedure :

Step 1 : Identify the decimal number corresponding to each minterm in the given expression as illustrated below :

$$\text{If } Y = \underbrace{\bar{A}\bar{B}\bar{C}}_{0} + \underbrace{A\bar{B}C}_{5} + \underbrace{ABC}_{7}$$

(C-6264)

Step 2 : The input lines of the multiplexer, corresponding to these numbers (0, 5 and 7) are connected to logic 1 level.

Step 3 : All the other input lines of multiplexer are connected to logic 0 level.

Step 4 : The inputs (A, B, C) are to be connected to the select inputs.

– The following example will make this concept clear.

Ex. 2.12.1 : Realize the logic function of the truth table given in Table P. 2.12.1 using a multiplexer.

(C-7617) Table P. 2.12.1

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

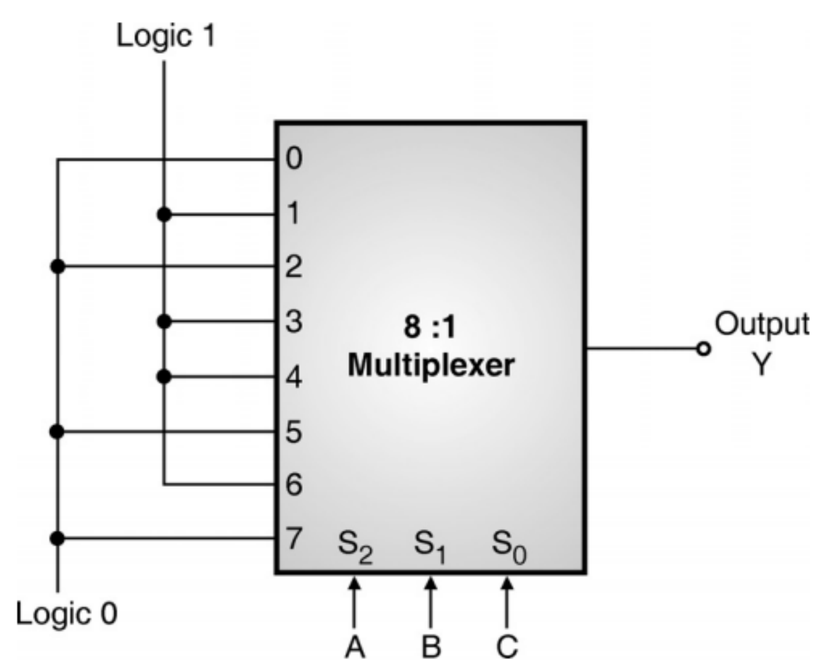
Soln. :

Step 1 : Express the output Y in the standard SOP form $Y = \sum m(1, 3, 4, 6)$

Step 2 : Connect the data inputs 1, 3, 4 and 6 to logic 1 and the remaining to logic 0.

Step 3 : Connect the inputs A, B, C to the select lines S_2 S_1 and S_0 respectively.

Step 4 : Draw the logic diagram as shown in Fig. P. 2.12.1.



(C-429) Fig. P. 2.12.1 : Logic diagram for Ex. 2.12.1

Ex. 2.12.2 : Implement the following expression using a multiplexer.

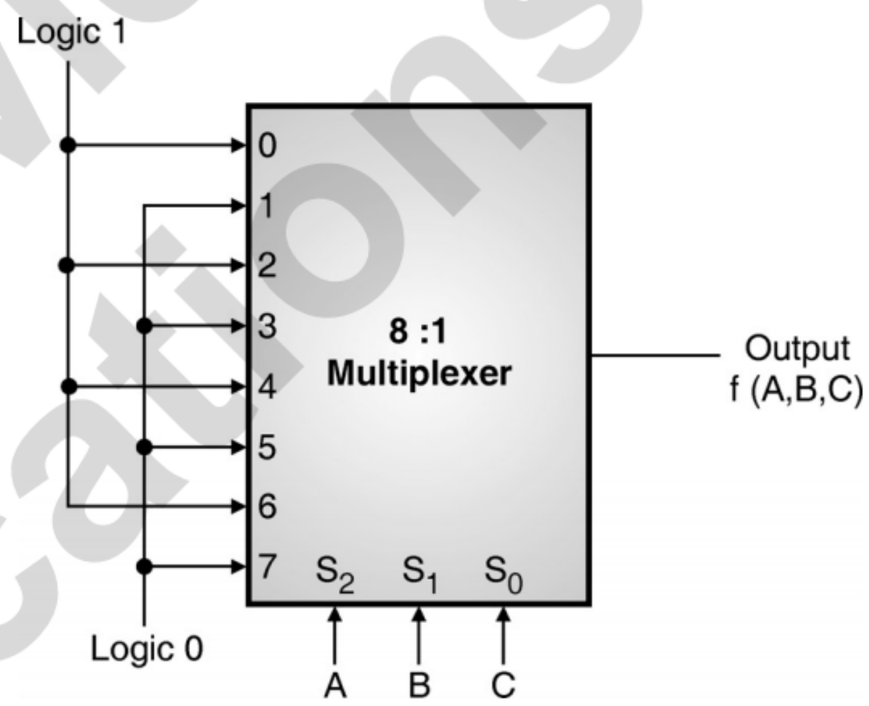
$f(A, B, C) = \Sigma m(0, 2, 4, 6)$

Soln. :

– Since there are three variables, the multiplexer having three select inputs should be used. Hence an 8 : 1 multiplexer as shown in Fig. P. 2.12.2 should be used.

Step 1 : Identify the decimal number corresponding to each minterm. The decimal numbers corresponding the minterms are 0, 2, 4 and 6.

Step 2 : Connect the data input lines 0, 2, 4 and 6 to logic 1 and the remaining lines (1, 3, 5, 7) to logic 0 as shown in Fig. P. 2.12.2.



(C-430) Fig. P. 2.12.2 : Implementation of a logic expression using a multiplexer

Step 3 : Connect the variables A, B and C to the select inputs.

Note : We can use IC 74151 which is an 8 : 1 multiplexer to implement Boolean equations using 8 : 1 MUX.

Ex. 2.12.3 : Implement the logic function f (A, B, C) $= \sum m(1, 3, 4, 6)$ using a 4 : 1 multiplexer.

Soln. :

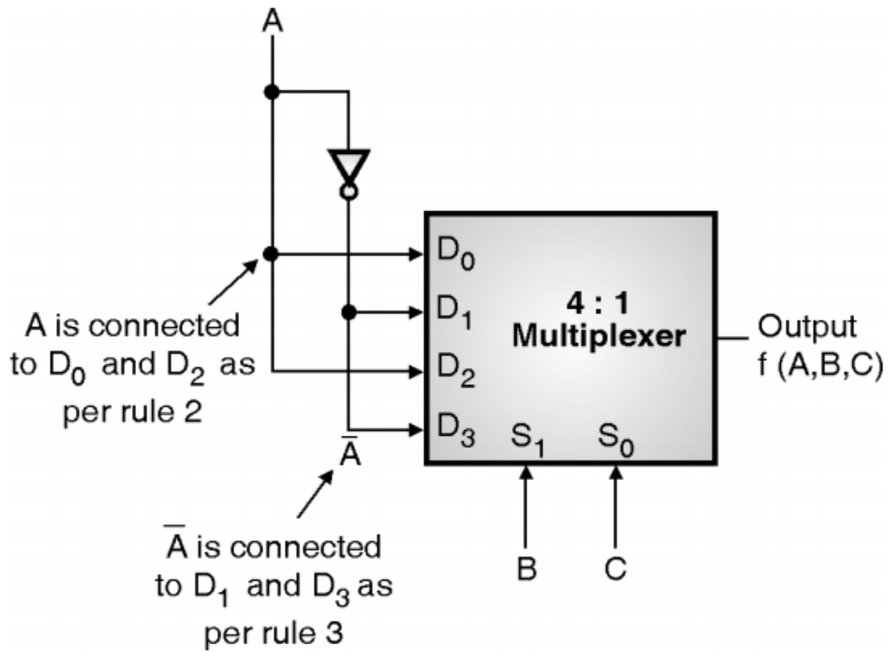
The logic function to be implemented is

$$f(A, B, C) = \Sigma m(1, 3, 4, 6)$$

Step 1 : Apply two variables B and C to the select inputs :

– As shown in Fig. P. 2.12.3(a), the two input variables B and C are applied to the select lines S_1 and S_0 respectively.

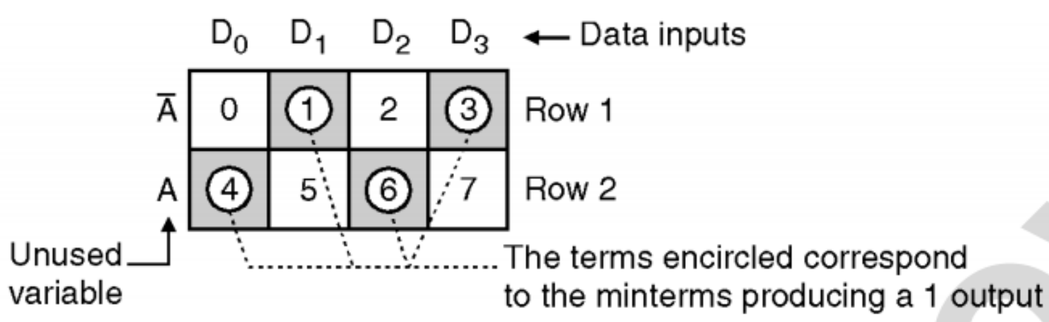
TechKnowledge Publications



(C-431) Fig. P. 2.12.3(a) : Logic diagram

Step 2 : Write the design table :

– The design table is shown in Fig. P. 2.12.3(b).



(C-432) Fig. P. 2.12.3(b) : Design table

– The data inputs D_0 to D_3 have been written at the top of the table and the two possible values A and $\bar{A}$ of the unused variable A have been written.

– In the eight boxes we enter the decimal numbers corresponding to the minterms (0 to 7) serially.

– Encircle those minterms corresponding to which the output is 1 (minterms 1, 3, 4, 6).

Step 3 : Check each column in the design table :

– The columns of design table are inspected using the following rules :

Rule 1 : If both the minterms in a column are not circled, then apply logic 0 to the corresponding data input. Note that there is no such column in our implementation table.

Rule 2 : If only the minterm in the second row is encircled (see columns 1 and 3) then "A" should be applied to that data input. Hence we should apply A to the D_0 and D_2 inputs.

Rule 3 : If only the minterm in the first row is encircled, (see columns 2 and 4), then $\bar{A}$ should be connected to that data input. Hence we should apply $\bar{A}$ to the D_1 and D_3 inputs.

Rule 4 : If both the minterms in a column are encircled, then apply a logic 1 to the corresponding data input.

– Note that there is no such column in our implementation table.

Step 4 : Draw the logic diagram :

– The logic diagram is as shown in Fig. P. 2.12.3(a).

Note : We can use IC 74153 which is a dual 4 : 1 multiplexer in order to implement the Boolean expressions using 4 : 1 MUX.

Ex. 2.12.4 : Implement a full adder using 8 : 1 multiplexer.

Dec. 17, 6 Marks

Soln. :

Step 1 : Write the truth table of a full adder :

– The truth table of a full adder is shown in Table P. 2.12.4.

(C-7621) Table P. 2.12.4 : Truth table for a full adder

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Step 2 : Identify the decimal number corresponding to each minterm :

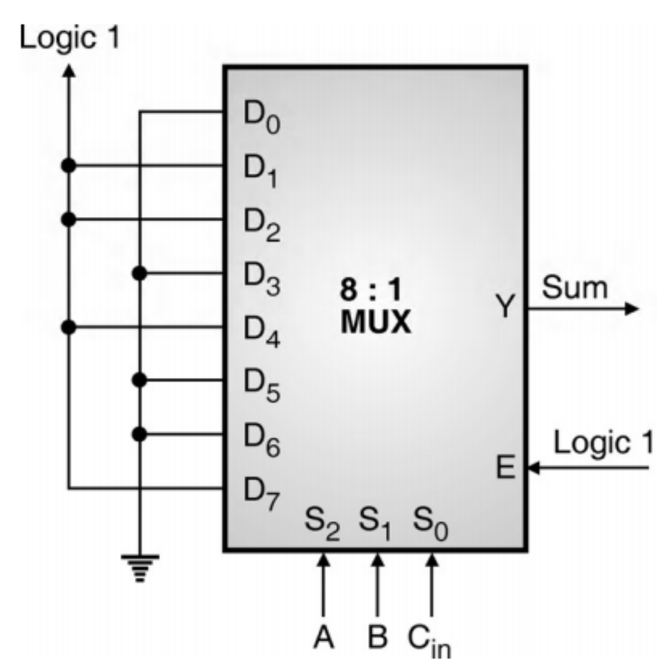
– The sum and carry outputs can be expressed in the standard SOP form as follows :

$$S = \Sigma\ m\ (1, 2, 4, 7) \text{ and } C = \Sigma\ m\ (3, 5, 6, 7)$$

Step 3 : Realize the circuit using two 8 : 1 multiplexers :

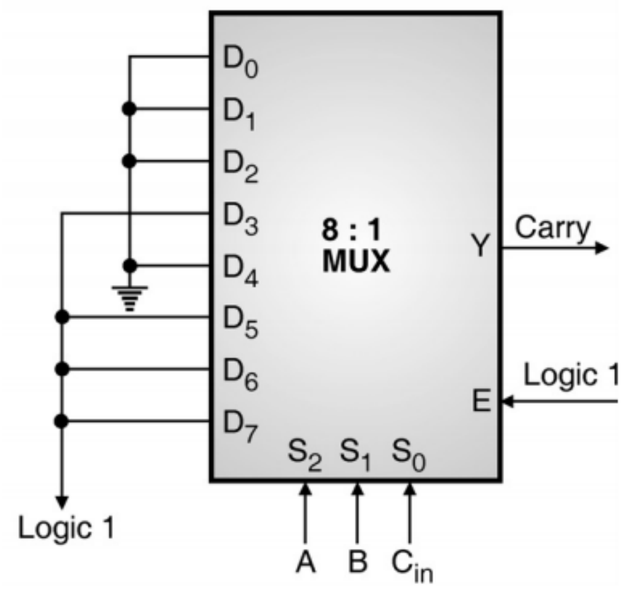
– We have to use two 8 : 1 multiplexers, one each for the sum and carry outputs.

– The realization is shown in Fig. P. 2.12.4.



(C-433) Fig. P. 2.12.4(a) (Contd...)

TechKnowledge Publications



(b)

(C-433) **Fig. P. 2.12.4 : Full adder using 8 : 1 multiplexer**

2.12.2 Use of 8 : 1 MUX to Realize a 4 Variable Function :

– It is very easy to use the 8 : 1 MUX to a 4 variable function or the 4 : 1 MUX to a 3 variable function or 16 : 1 MUX to a 5 variable function.

– The procedure to be followed for this has been illustrated through the following examples.

Ex. 2.12.5 : Implement the following Boolean function using 8 : 1 multiplexer.

$f(A, B, C, D) = \Sigma\, m\,(2, 4, 5, 7, 10, 14)$

Soln. :

Step 1 : Apply the variables B, C and D to the select inputs :

As shown in Fig. P. 2.12.5(b), the three variables B, C and D are connected to the select lines S_2 , S_1 and S_0 respectively.

Step 2 : Write the design table :

– The design table is as shown in Fig. P. 2.12.5(a).

	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	Row 1
A	8	9	10	11	12	13	14	15	Row 2
Input to MUX :	0	0	1	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$	

The terms encircled correspond to the minterms producing a 1 output

(C-434) **Fig. P. 2.12.5(a) : Design table**

– In the sixteen boxes we have entered the decimal numbers corresponding to the minterms 0 to 15 serially.

– Encircle those minterms which correspond to an output = 1 (minterms 2, 4, 5, 7, 10, 14).

Step 3 : Inspect each column in the design table :

Rule 1 : If both the minterms in a column are not circled, then apply logic 0 to the corresponding data input. Note that there is no such column in our implementation table.

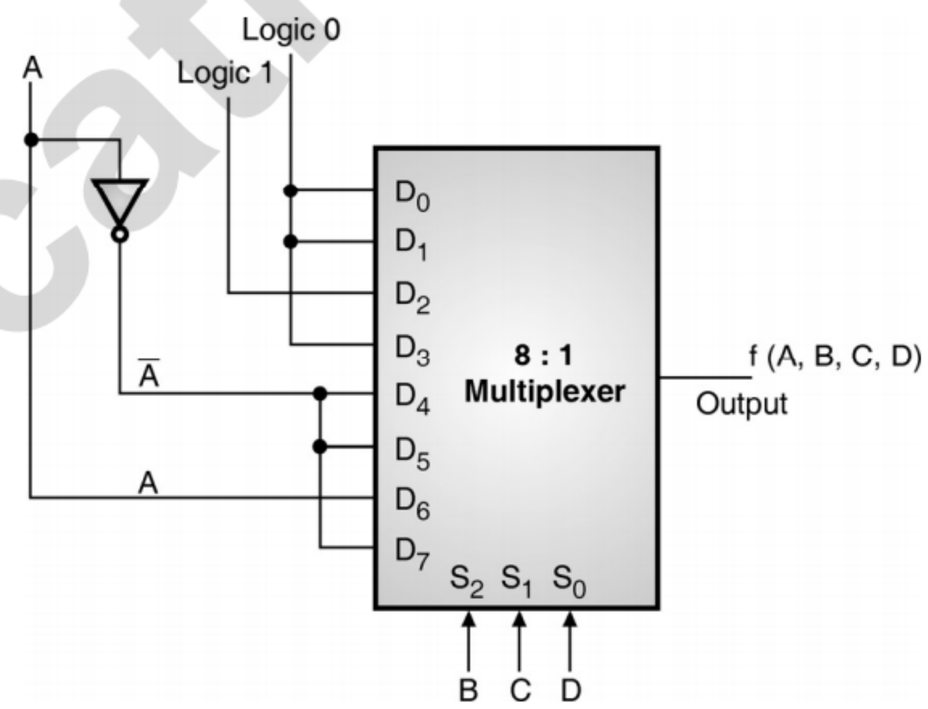
Rule 2 : If only the minterm in the second row is encircled then "A" should be applied to that data input. **Hence we should apply A to the D_6 input.**

Rule 3 : If only the minterm in the first row is encircled, then $\bar{A}$ should be connected to that data input. Hence we should apply $\bar{A}$ to the D_4, D_5 and D_7 inputs.

Rule 4 : If both the minterms in a column are encircled, then apply a logic 1 to the corresponding data 1 input. Note that there is no such column in our implementation table.

Step 4 : Draw the logic diagram :

– The logic diagram is as shown in Fig. P. 2.12.5(b).



(C-435) **Fig. P. 2.12.5(b) : Logic diagram**

Ex. 2.12.6 : Implement the following function using 16 : 1 multiplexer

$f(A, B, C, D, E) = \Sigma\, m\,(2, 4, 5, 7, 10, 14, 15, 16, 17, 25, 26, 30, 31)$

Soln. :

Step 1 : Apply the variables B, C, D, E to the select inputs :

As shown in Fig. P. 2.12.6 the variables B, C, D, E are applied to the select lines S_3 , S_2, S_1 , S_0 respectively.

Step 2 : Write the design table :

– The design table is as shown in Fig. P. 2.12.6.

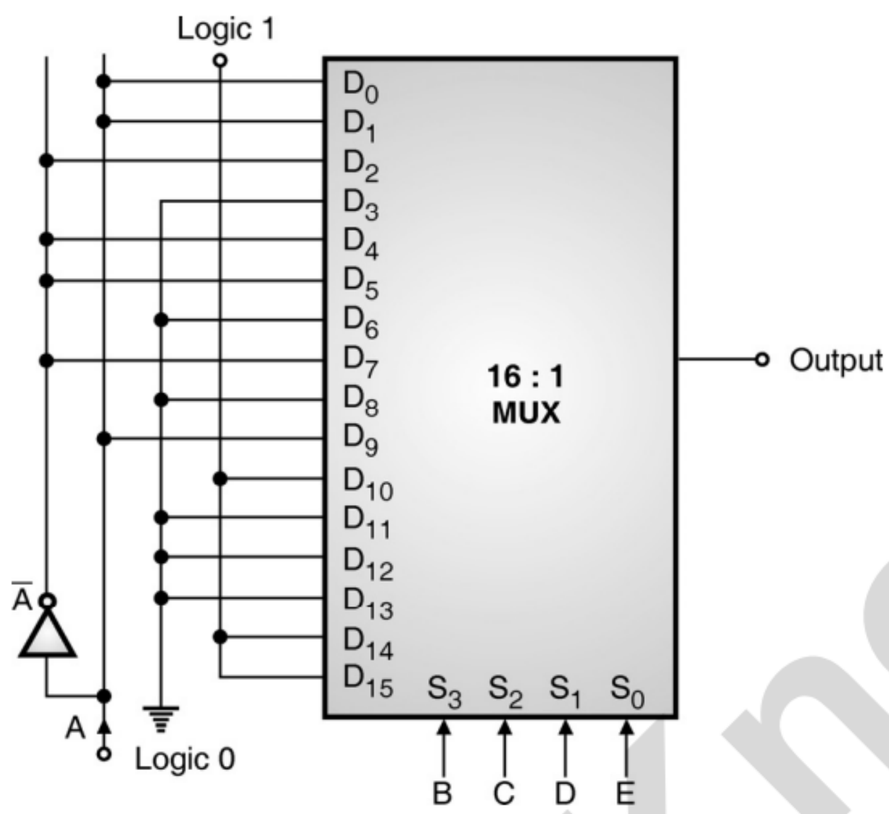
Techknowledge Publications

(C-435(a)) **Table P. 2.12.6 : Design table**

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}
$\overline{A}$	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
Input to MUX	A	A	$\overline{A}$	0	$\overline{A}$	$\overline{A}$	0	$\overline{A}$	0	A	1	0	0	0	1	1

Step 3 : Realize the logic circuit :

– The logic circuit is as shown in Fig. P. 2.12.6.



(C-436) **Fig. P. 2.12.6 : Logic diagram**

Ex. 2.12.7 : Implement the following function using 4 : 1 multiplexers with active low strobe input :

f (A, B, C, D) = ∑m (2, 3, 5, 7, 8, 9, 12, 13, 14, 15).

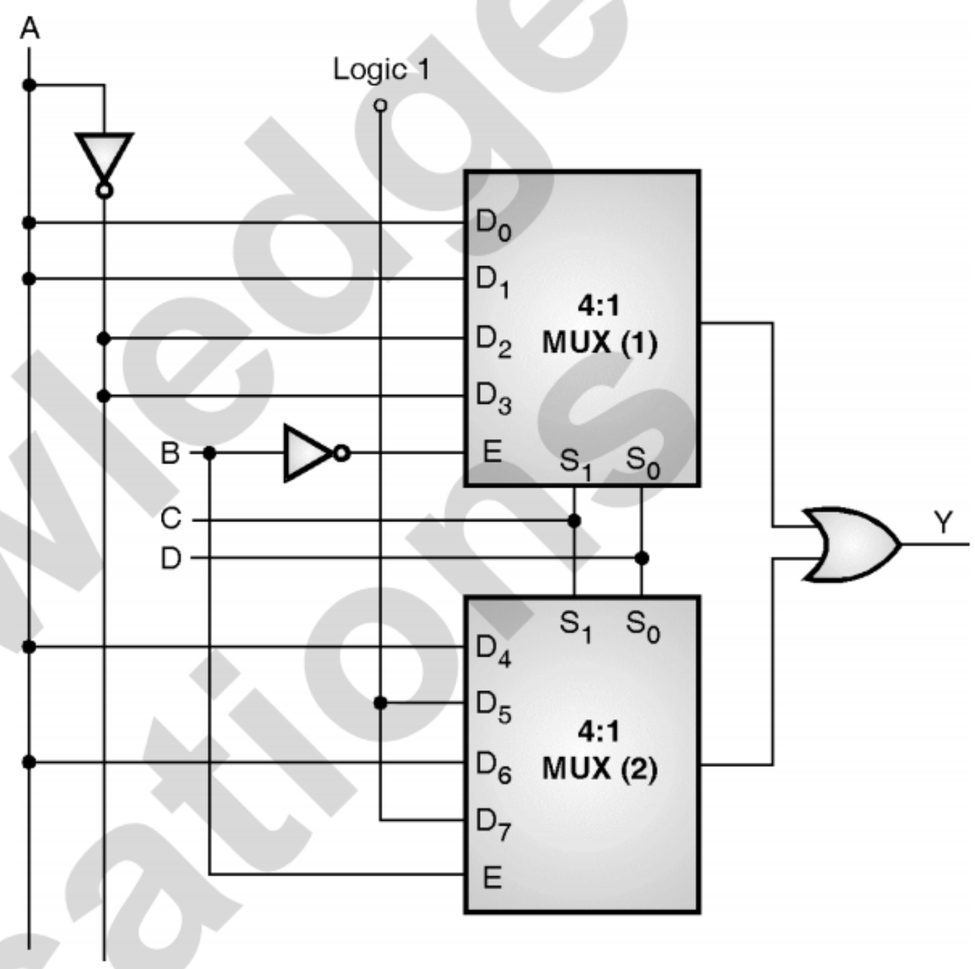
Soln. :

Step 1 : Write the design table :

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
Input to MUX :	A	A	$\overline{A}$	$\overline{A}$	A	1	A	1
	← MUX - 1 →				← MUX - 2 →			

(C-1329(a))

Step 2 : Implementation of logic circuit :



(C-1330) **Fig. P. 2.12.7**

Ex. 2.12.8 : Implement the following function using single 4 : 1 MUX and logic gates :

F (A, B, C, D) = Σm (0, 1, 5, 9, 10, 15)

May 06, 6 Marks

Soln. :

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

Step 1 : Write the design table :

	D_0	D_1	D_2	D_3
$\overline{A}\overline{B}$	0	1	2	3
$\overline{A}B$	4	5	6	7
$A\overline{B}$	8	9	10	11
AB	12	13	14	15
	$\overline{A}\overline{B}$		$A\overline{B}$	AB

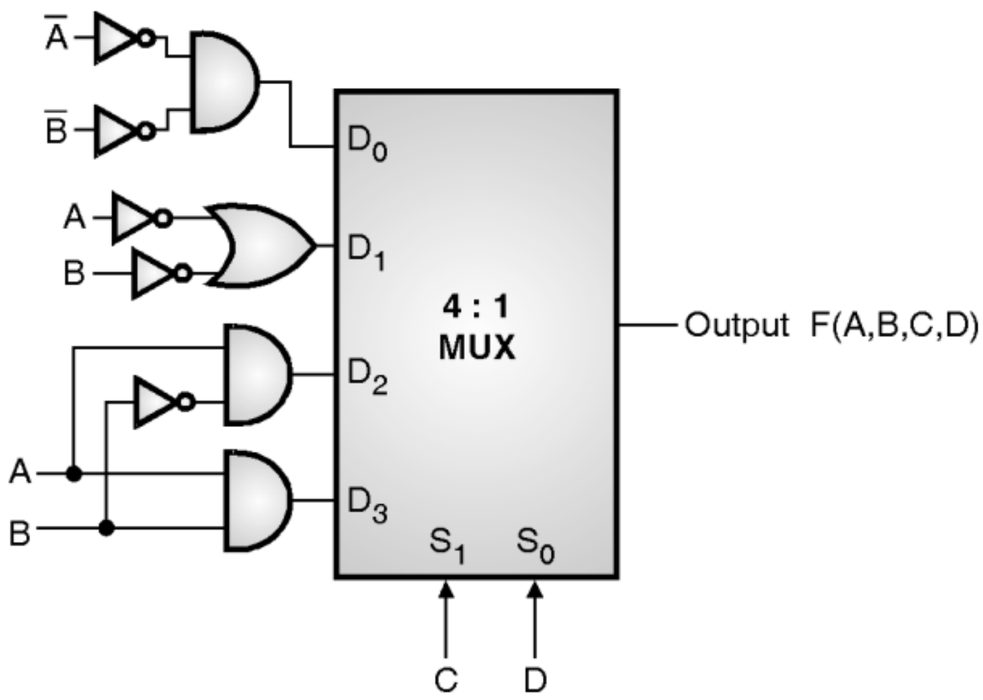
(C-1335)

$$D_1 = \overline{A}\overline{B} + \overline{A}B + A\overline{B} = \overline{A}(\overline{B} + B) + A\overline{B}$$

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

TechKnowledge Publications

Step 2 : Implementation using 4 : 1 MUX :



(C-1336) **Fig. P. 2.12.8**

Ex. 2.12.9 : Implement the following Boolean function with 4 : 1 MUX and external gates. Connect input A (MSB) and B to the selection lines S_1 (MSB) and S_0 respectively :

$F(A, B, C, D) = \sum (1, 3, 4, 11, 12, 13, 14, 15)$

Dec. 07, 6 Marks

Ans. : Similar to Ex. 2.12.8.

Ex. 2.12.10 : Implement the expression using a 8 : 1 multiplexer

$f(A, B, C, D) = \sum m (0, 2, 3, 6, 8, 9, 12, 14)$.

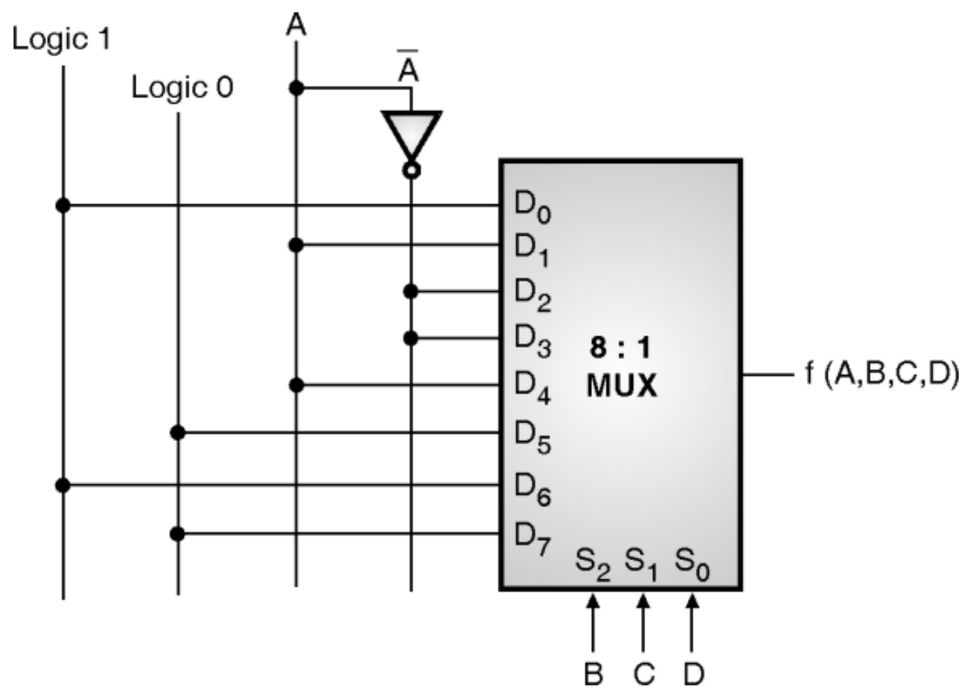
May 17, 4 Marks

Soln. :

Step 1 : Write the design table : (C-5910)

Inputs	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
Input to MUX	1	A	$\overline{A}$	$\overline{A}$	A	0	1	0

Step 2 : Implementation using 8 : 1 MUX :



(C-5911) **Fig. P. 2.12.10 : Implementation using 8 : 1 MUX**

Ex. 2.12.11 : Design full subtractor using multiplexer IC 74151. **Dec. 18, 4 Marks**

Soln. :

Step 1 : Write the truth table of full subtractor :

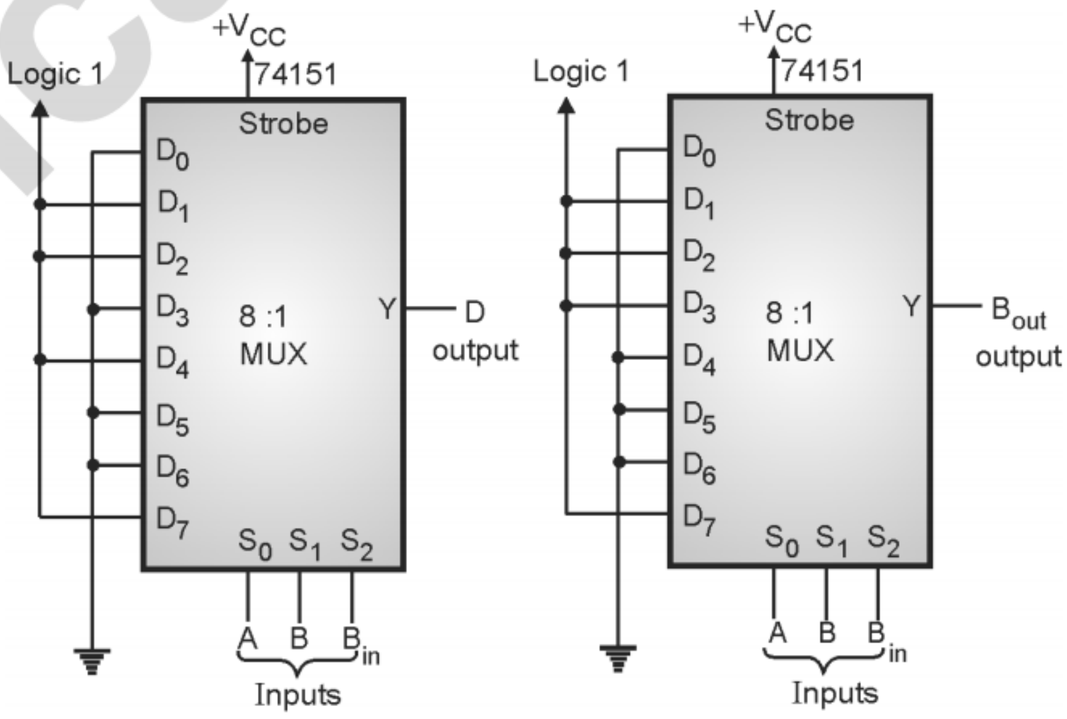
(C-7461) **Table P. 2.12.11**

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

$\therefore$ $D = \Sigma m (1, 2, 4, 7)$ and $B_{out} = \Sigma m (1, 2, 3, 7)$

Step 2 : Implement using MUX IC 74151 :

– IC 74151 is an 8 : 1 MUX with complementary outputs. We will use two IC 74151 one each for D and B_{out} outputs as shown in Fig. P. 2.12.11.



(C-7462) **Fig. P. 2.12.11 : Full subtractor using IC 74151**

2.12.3 Implementation of a Logical Expression in the Non-standard SOP Form :

– Till now we have seen the implementation of expressions in the standard SOP form.

– But if the given expression is not in the standard form, then we have to first bring it to the standard form and then proceed.

– Example 2.12.12 illustrates this concept.

Ex. 2.12.12 : Implement the following Boolean expression using 8 : 1 multiplexer.

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

OR

Implement the following Boolean expression using 8 : 1 MUX.

$Y = \sum m\,(0, 2, 3, 7, 11, 14, 15)$

Soln. :

Step 1 : Convert the given expression to standard SOP form :

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

This is the expression in the standard SOP form.

$$\therefore\ f(A, B, C, D) = \sum m\,(2, 0, 15, 14, 11, 3, 7, 3)$$

$$\text{As } X + X = X$$

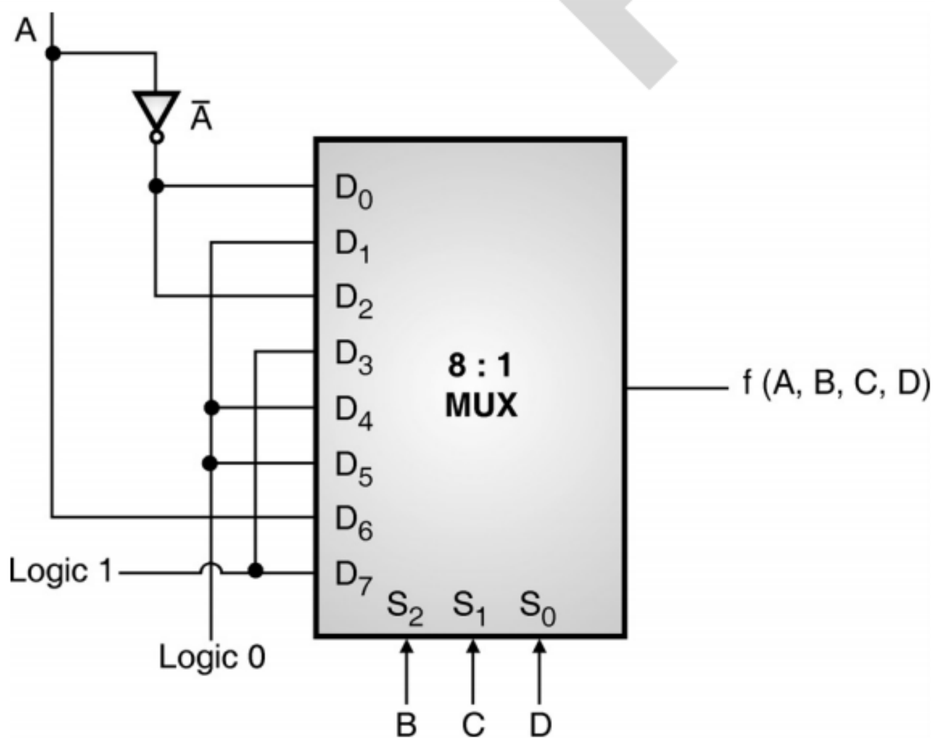
$$\therefore\ \bar{A}\,\bar{B}\,C\,D + \bar{A}\,\bar{B}\,C\,D = \bar{A}\,\bar{B}\,C\,D$$

$$\therefore\ f(A, B, C, D) = \sum m\,(0, 2, 3, 7, 11, 14, 15)$$

Step 2 : Write the design table :

(C-437(a)) Table P. 2.12.12 : 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
Inputs to MUX	$\bar{A}$	0	$\bar{A}$	1	0	0	A	1



(C-438) Fig. P. 2.12.12 : Implementation

Step 3 : Implementation using 8 : 1 multiplexer :

– The implementation is as shown in Fig. P. 2.12.12.

2.12.4 Implementing a Standard POS Expression using Multiplexer :

– Let us consider the Boolean expression in the standard POS form and its implementation using a multiplexer.

Ex. 2.12.13 : Implement the following logic function using the 8 : 1 multiplexer.

$f(A, B, C) = \Pi M\,(0, 1, 3, 5, 7)$

Soln. :

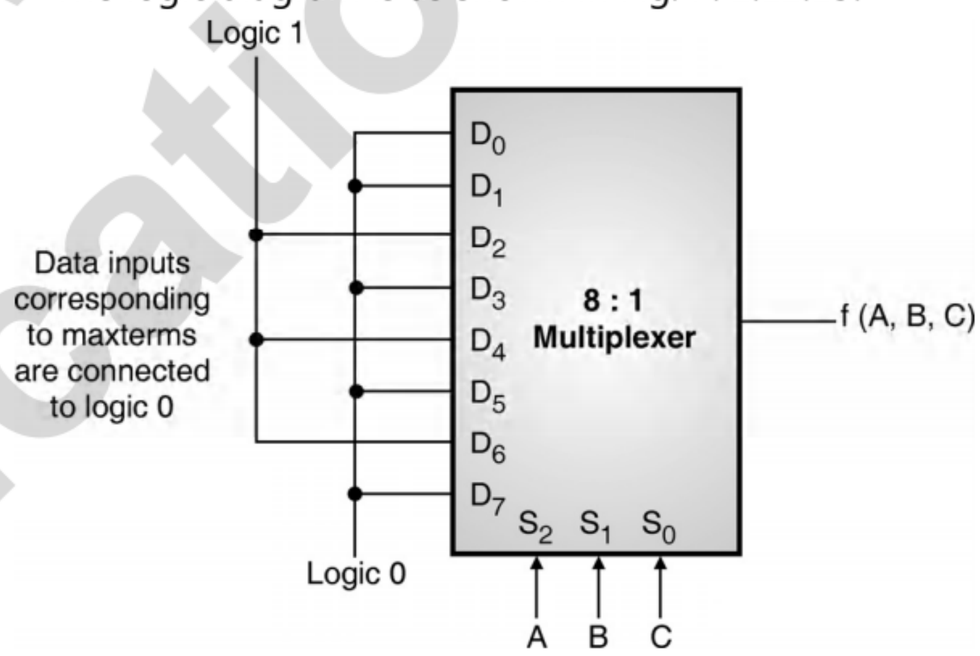
– In the given expression, the maxterms have been specified.

– Hence we should connect the corresponding data inputs (D_0, D_1, D_3, D_5 and D_7) to logic 0.

– And connect the remaining data inputs to logic 1.

– Connect the input variables A, B, C to the select inputs S_2, S_1 and S_0 respectively.

– The logic diagram is as shown in Fig. P. 2.12.13.



(C-439) Fig. P. 2.12.13 : Implementation of standard POS expression

Ex. 2.12.14 : Implement the following logic function using 4 : 1 multiplexer.

$f(A, B, C) = \Pi\,M\,(0, 1, 3, 5, 7)$

Soln. :

– Connect B and C to the select inputs S_1 and S_0 respectively.

– For the connections of data inputs (D_0 to D_3) and A write down the design table as shown in Fig. P. 2.12.14(a).

	D_0	D_1	D_2	D_3	
$\bar{A}$	0	1	2	3	Row 1
A	4	5	6	7	Row 2
Input to MUX:	A	0	1	0	

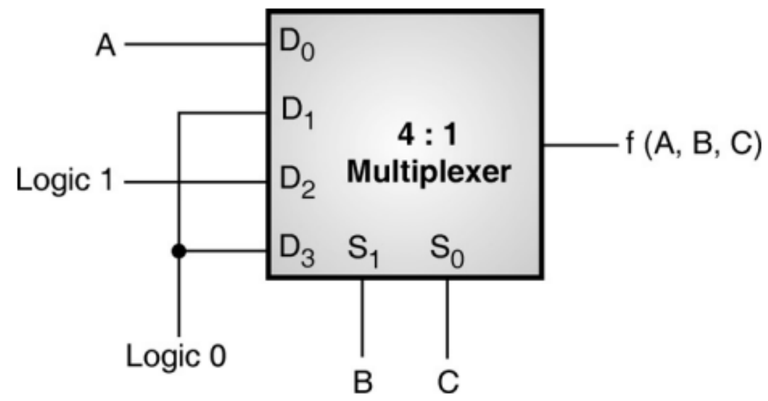
Encircle the maxterms which are not included in the given Boolean expression.

(C-440) Fig. P. 2.12.14(a) : Design table

– Note that as the given expression is in the POS form, we should encircle those maxterms which are not included in the given Boolean expression.

– The rules for deciding the input to the data lines (D_0 to D_3) are however the same as those stated earlier for the SOP form.

– The logic diagram is as shown in Fig. P. 2.12.14(b).



(C-441) Fig. P. 2.12.14(b) : Implementation using 4 : 1 multiplexer

Ex. 2.12.15 : Implement the following function using an 8 : 1 multiplexer.

f (A, B, C, D) = ΠM (0, 2, 4, 6, 8, 10, 12, 14)

Soln. :

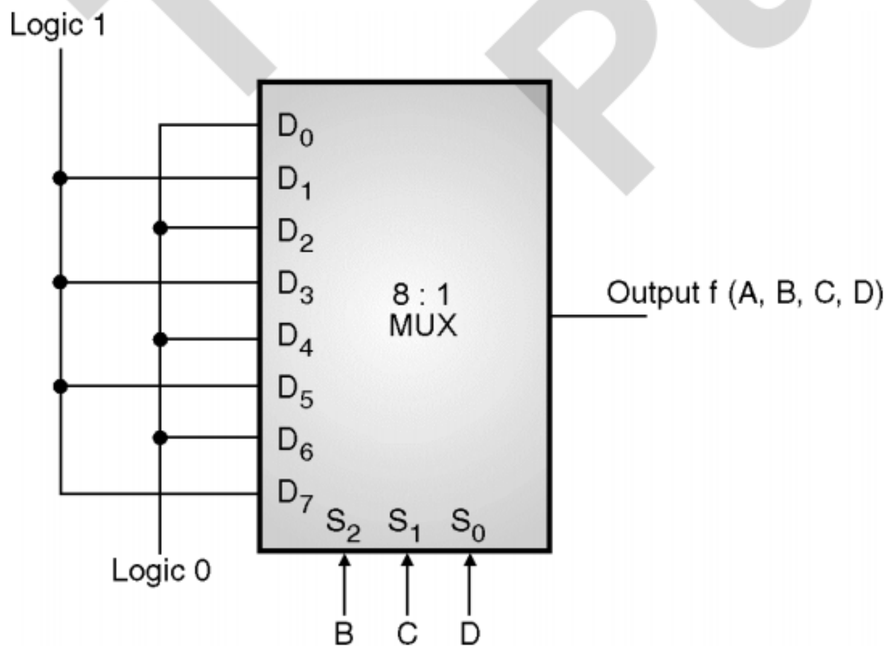
Step 1 : Write the design table :

Table P. 2.12.15 : Design table

Inputs	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
Inputs to MUX	0	1	0	1	0	1	0	1

Step 2 : Implement using an 8 : 1 multiplexer :

– The implementation is as shown in Fig. P. 2.12.15.



(C-442) Fig. P. 2.12.15 : Implementation using 8 : 1 multiplexer

Ex. 2.12.16 : Implement a full adder circuit using two 4 : 1 multiplexers. **Dec. 04, 3 Marks**

Soln. :

Step 1 : Write the truth table of full adder :

(C-7621) Table P. 2.12.16 : Truth table of a full adder

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Step 2 : Write the design tables for sum and carry outputs :

Sum

	D_0	D_1	D_2	D_3
$\overline{A}$	0	1	2	3
A	4	5	6	7
Input to Mux	A	$\overline{A}$	$\overline{A}$	A

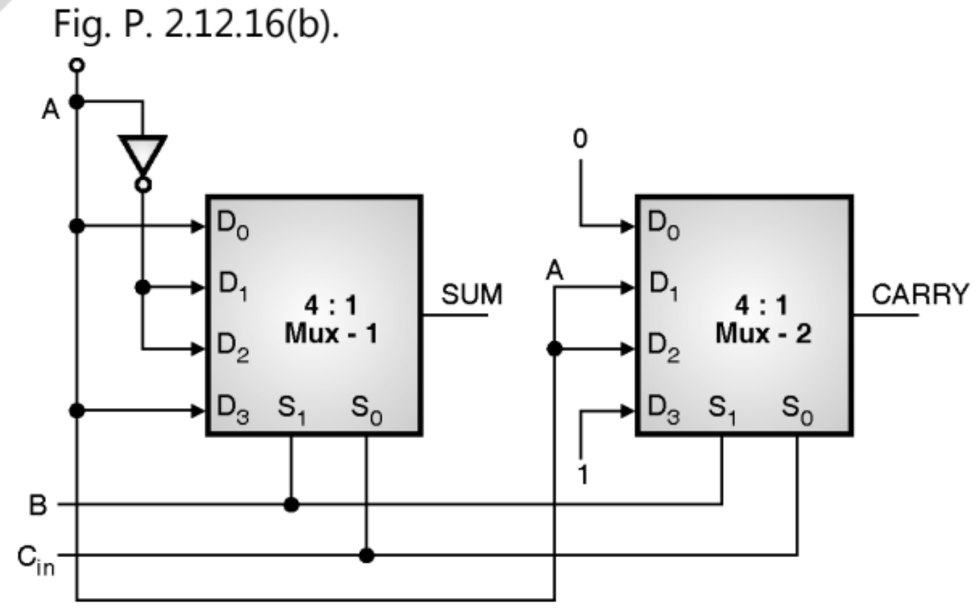
Carry

	D_0	D_1	D_2	D_3
$\overline{A}$	0	1	2	3
A	4	5	6	7
Input to Mux	0	A	A	1

(C-443) Fig. P. 2.12.16(a) : Design tables for sum and carry outputs

Step 3 : Draw the logic diagram :

– The full adder using 4 : 1 multiplexer is shown in Fig. P. 2.12.16(b).



(C-444) Fig. P. 2.12.16(b) : Implementation of full adder using two 4 : 1 multiplexers

2.12.5 Implementation of Boolean SOP Expression with Don't Care Conditions :

– Sometimes the Boolean expressions are given with don't care conditions.

– We know that the don't care conditions can be treated as logic 1's or 0's.

– Ex. 2.12.17 illustrates the use of multiplexer to implement such equations.

TechKnowledge Publications

Ex. 2.12.17 : Implement the following Boolean expression using an 8 : 1 multiplexer.

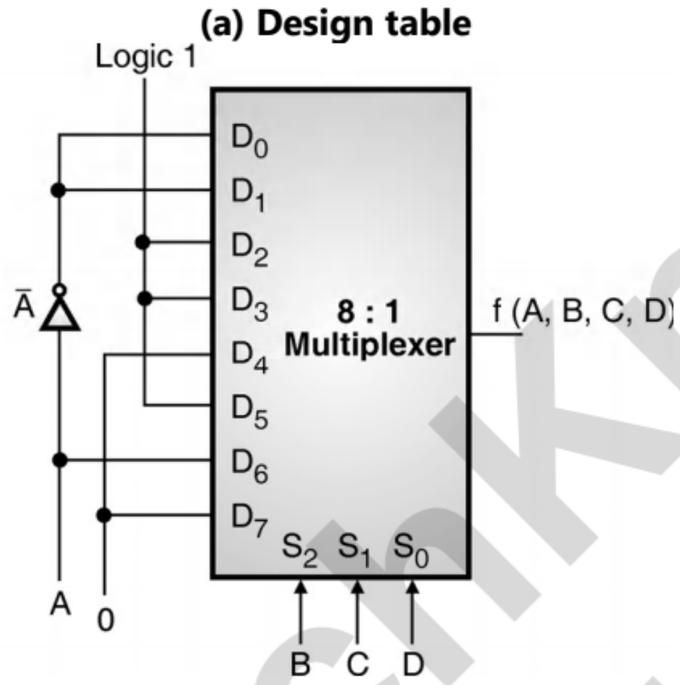
$f(A, B, C, D) = \Sigma m(1, 3, 5, 10, 11, 13, 14) + d(0, 2)$.

Soln. :

– The don't care conditions are assumed to be logic 1's. The design table is shown in Fig. P. 2.12.17(a) and the corresponding logic diagram is shown in Fig. P. 2.12.17(b).

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
Input to Mux	$\overline{A}$	$\overline{A}$	1	1	0	1	A	0

(a) Design table



(b) Implementation

(C-445) Fig. P. 2.12.17

Ex. 2.12.18 : Implement the following function using 8 : 1 MUX and logic gates.

$F(A, B, C, D) = \Sigma(0, 2, 5, 8, 10, 15)$.

Dec. 19, 6 Marks

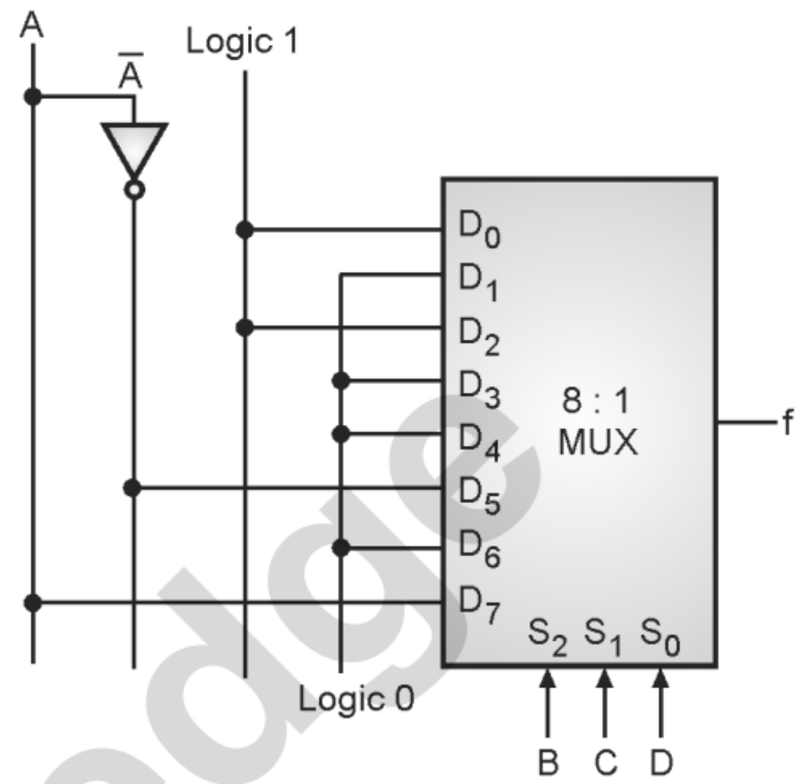
Soln. :

Step 1 : Write the design table :

(CT-3) Table P. 2.12.18 : Design table

Inputs	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
Input to MUX	1	0	1	0	0	$\overline{A}$	0	A

Step 2 : Implementation using 8 : 1 multiplexer :



(CT-5) Fig. P. 2.12.18 : Implementation using 8 : 1 MUX

Ex. 2.12.19 : Design an 8 : 1 multiplexer using two 4 : 1 multiplexers. Explain with the help of the truth table. Implement the function $f(A, B, C) = \sum m(1, 3, 7)$ using the same. **May 12, 8 Marks**

Soln. : Solve it yourself.

Ex. 2.12.20 : Design 16 : 1 multiplexer using only one 8 : 1 multiplexer and required discrete logic gate for the following function :

$F(A, B, C, D) = \Sigma m(0, 4, 6, 9, 12, 13)$.

Dec. 11, 8 Marks

Soln. : Solve it yourself.

Ex. 2.12.21 : Implement the following expression using 8 : 1 multiplexer : $f(A, B, C, D) = \sum m(2, 4, 6, 7, 9, 10, 11, 12, 15)$ **Dec. 12, 8 Marks**

Soln. : Solve it yourself.

Ex. 2.12.22 : Determine the output for the following input states : $D_0 = 0$, $D_1 = 1$, $D_2 = 1$, $D_3 = 0$, $s_0 = 1$, $s_1 = 0$. Use 4 : 1 MUX. **Dec. 15, 2 Marks**

Soln. :

$D_3 D_2 D_1 D_0 = 0110$

$S_1 S_0 = 01$

Output = $D_1 = 1$, because D_1 will get selected and connected to output.

2.13 Demultiplexers :

2.13.1 Demultiplexer Principle :

Definition :

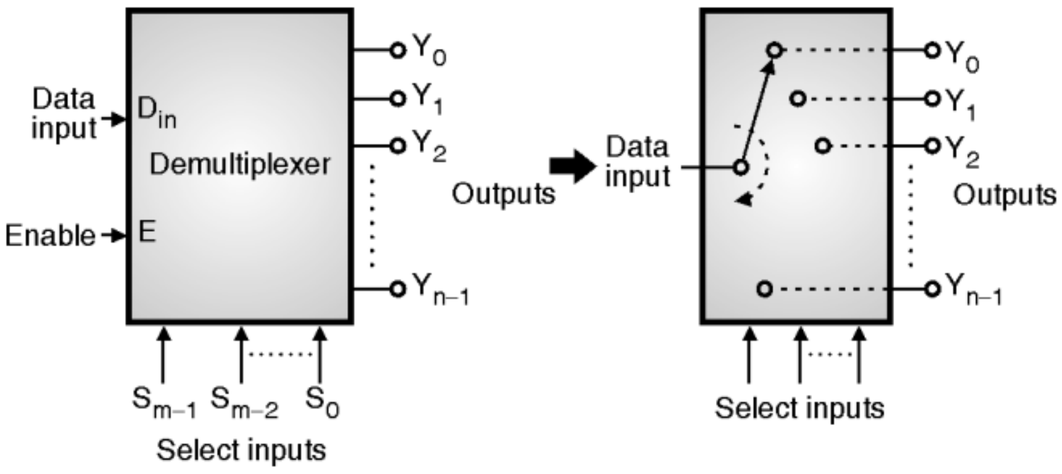
– A de-multiplexer is a combinational circuit with one data input, n outputs and m select inputs, which selects one of the n data outputs and routes (connects) it to the input.

TechKnowledge Publications

– The selection of one of the n outputs is done with the help of the select inputs.

Block diagram :

– The block diagram of a 1 to n demultiplexer is shown in Fig. 2.13.1(a) and its equivalent circuit is as shown in Fig. 2.13.1(b).



(a) 1 : n demultiplexer (b) Equivalent circuit

(C-447) **Fig. 2.13.1**

– It has one data input, n outputs, m select inputs and one enable input.

– A demultiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs.

– At a time only one output line is selected by the select lines and the input is connected to the selected output line.

– The enable input will enable the demultiplexer. If the enable (E) input is not active, then the demultiplexer does not work.

– The relation between the n output lines and m select lines is as follows :

$$n = 2^m$$

Equivalent Circuit :

– A demultiplexer is equivalent to a single pole multiple way switch as shown in Fig. 2.13.1(b).

– As shown in Fig. 2.13.1(b) a demultiplexer is equivalent to a digitally controlled single pole, multiple way switch.

– The data input gets connected to only one of the n outputs at given instant of time via the single pole multiple throw rotary switch as shown.

– The position of the rotary arm will be dependent on the status of the m select inputs. It will connect the input to the selected output.

Need of Demultiplexers :

– In most of the electronic systems, the digital data from more than one sources is added together and this common signal is transmitted over a common communication channel.

– It is necessary to separate out this data at the receiving end.

– Under such circumstances we require a circuit which separates the multiple signals from the common one.

– This circuit is nothing else but a demultiplexer, which has one input, many outputs, and some select inputs.

– Demultiplexer improves the reliability of the digital system because it reduces the number of external wired connections.

2.14 Types of Demultiplexers :

– Similar to the multiplexers, the demultiplexers are classified as follows :

1. 1 : 2 demultiplexer. 2. 1 : 4 demultiplexer.
3. 1 : 8 demultiplexer. 4. 1 : 16 demultiplexer.

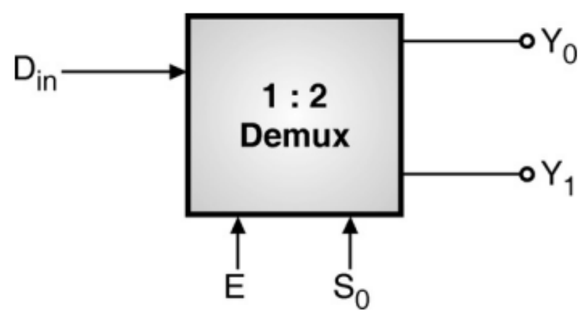
2.14.1 1 : 2 Demultiplexer :

Block diagram :

– The block diagram of 1 : 2 demultiplexer is shown in Fig. 2.14.1. It has one data input D_{in}, one select input S_0, one enable (E) input and two outputs Y_0 and Y_1.

– D_{in} is connected to Y_0 if $S_0 = 0$ and E = 1. Similarly D_{in} is connected to Y_1 if $S_0 = 1$ and E = 1.

– If E = 0, then both the outputs will be 0 irrespective of the inputs, because the DEMUX is disabled.



(C-448) **Fig. 2.14.1 : A 1:2 demultiplexer**

Truth Table :

– The truth table of the 1 : 2 demultiplexer is as follows :

(C-7620) **Table 2.14.1 : Truth table of demux 1 : 2**

Enable	Select	Outputs	
E	S_0	Y_1	Y_0
0	X	0	0
1	0	0	D_{in}
1	1	D_{in}	0

Realization of 1 : 2 demux :

Step 1 : To realize the 1 : 2 demultiplexer using gates we have to write the truth table as given in Table 2.14.2 :

(C-6188) Table 2.14.2 : Detail truth table of demux 1 : 2

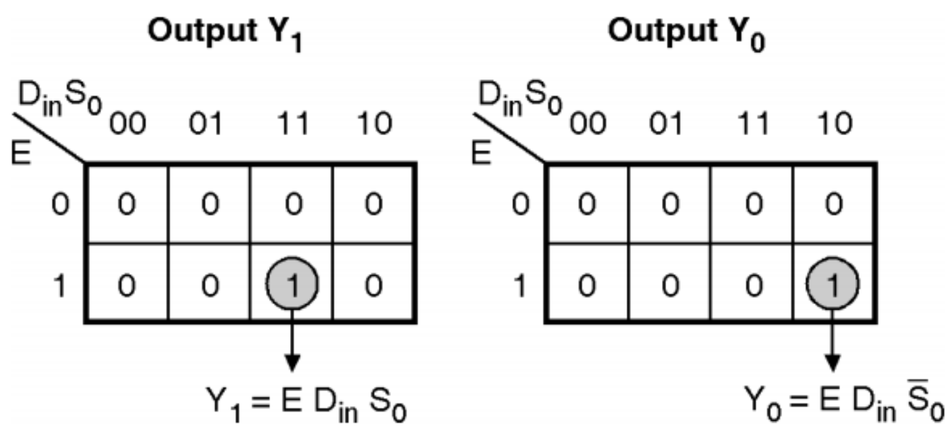
Enable	Data	Select	Outputs	
E	D_{in}	S_0	Y_1	Y_0
0	X	X	0	0
1	0	0	0	0
1	1	0	0	1
1	0	1	0	0
1	1	1	1	0

Y_0 is connected to D_{in}

Y_1 is connected to D_{in}

Step 2 : Write the K maps :

– The K maps for the two outputs are as follows :

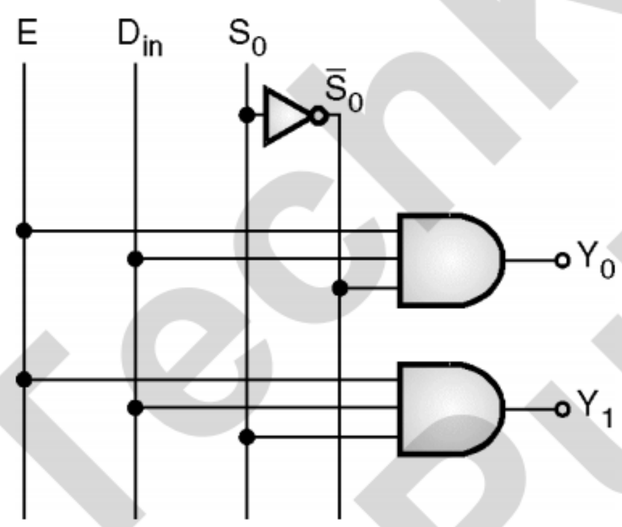


(C-449) Fig. 2.14.1(a) : K-maps

– Thus the expressions for Y_0 and Y_1 are as follows :

$$Y_0 = ED_{in}\,\overline{S_0} \text{ and } Y_1 = E\,D_{in}\,S_0$$

– The 1 : 2 demux using gates is shown in Fig. 2.14.2.

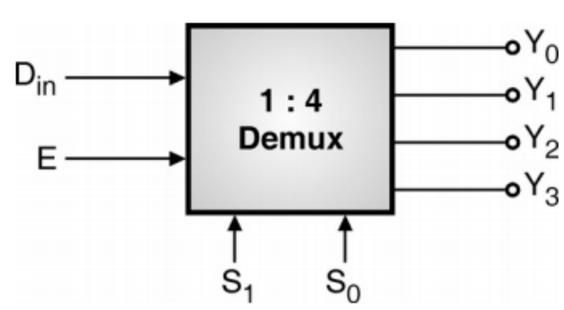


(C-450) Fig. 2.14.2 : 1 : 2 demux using gates

2.14.2 1 : 4 Demultiplexer :

Block diagram and truth table :

– The 1 : 4 demultiplexer is shown in Fig. 2.14.3 and its truth table is given in Table 2.14.3.



(C-451) Fig. 2.14.3 : Block diagram of 1 : 4 demultiplexer

(C-6189) Table 2.14.3 : Truth table for 1 : 4 demultiplexer

		Inputs		Outputs			
E	D_{in}	S_1	S_0	Y_0	Y_1	Y_2	Y_3
1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	0
1	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0
1	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1

Y_0 is connected to D_{in}

Y_1 is connected to D_{in}

Y_2 is connected to D_{in}

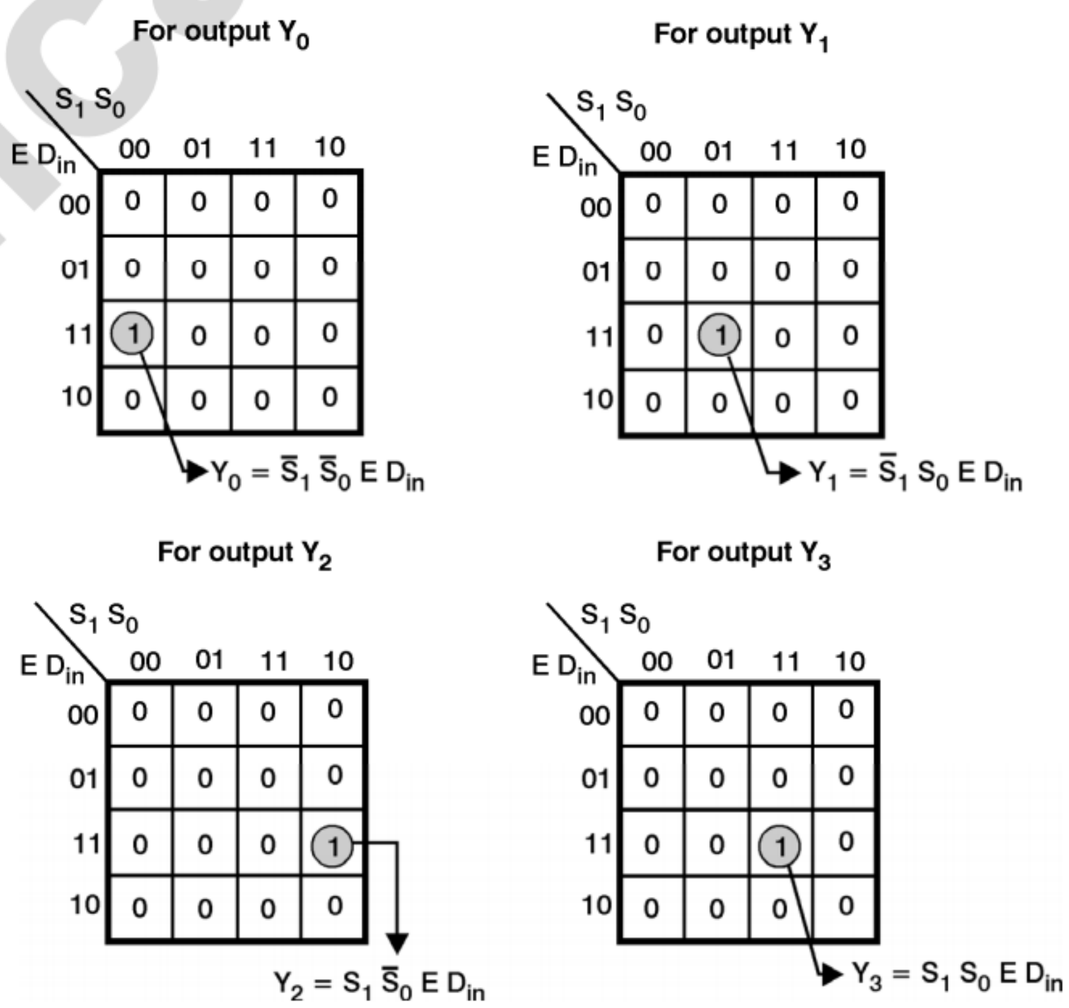
Y_3 is connected to D_{in}

– D_{in} is connected to Y_0 when $S_1S_0 = 00$, it is connected to Y_1 when $S_1S_0 = 01$ and so on. The other outputs will remain zero.

– The enable input needs to be high in order to enable the demux. If E = 0 then all the outputs will be low irrespective of everything.

K-maps :

– The K-maps for the four outputs are shown in Fig. 2.14.4.

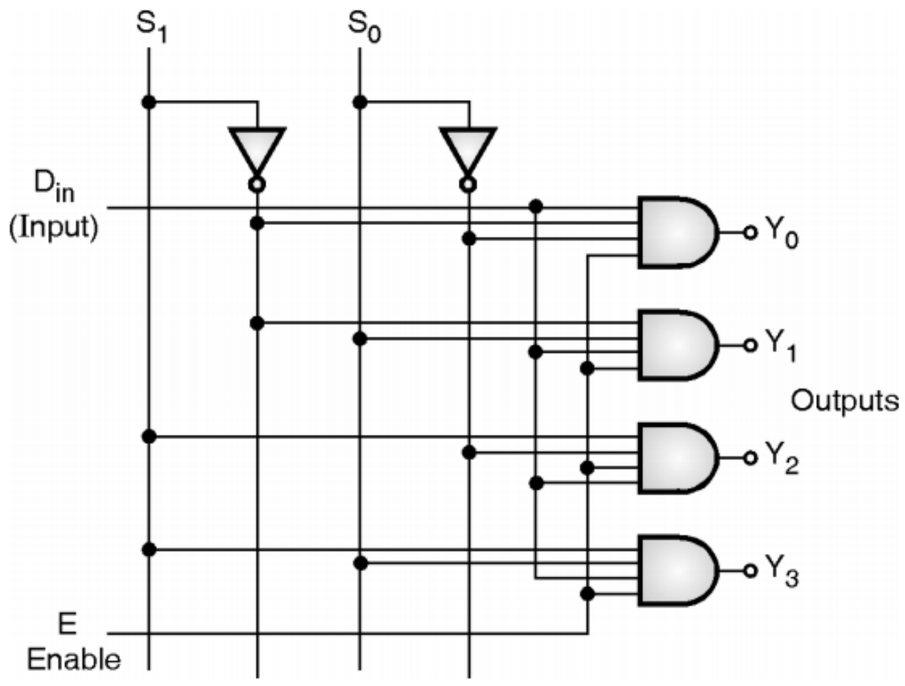


(C-452) Fig. 2.14.4 : K-maps for various outputs

Implementation :

– The 1 : 4 demux is implemented as shown in Fig. 2.14.5.

TechKnowledge Publications

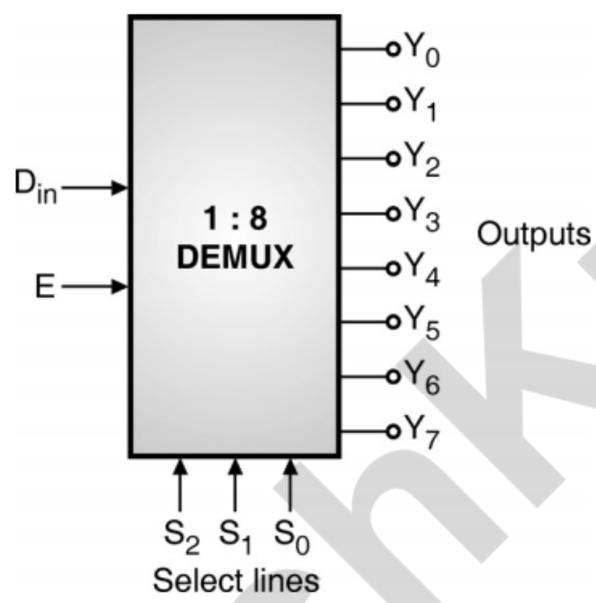


(C-453) Fig. 2.14.5 : 1:4 demultiplexer

2.14.3 1 : 8 Demultiplexer :

Block diagram and truth table :

– The block diagram of 1 : 8 demux is shown in Fig. 2.14.6(a).



(C-454) Fig. 2.14.6(a) : 1 : 8 demux

– It has one data input, eight outputs, three select inputs and an enable input E.

– The truth table is shown in Fig. 2.14.6(b).

Enable	Select			Outputs							
E	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	D_{in}
1	0	0	1	0	0	0	0	0	0	D_{in}	0
1	0	1	0	0	0	0	0	0	D_{in}	0	0
1	0	1	1	0	0	0	0	D_{in}	0	0	0
1	1	0	0	0	0	0	D_{in}	0	0	0	0
1	1	0	1	0	0	D_{in}	0	0	0	0	0
1	1	1	0	0	D_{in}	0	0	0	0	0	0
1	1	1	1	D_{in}	0	0	0	0	0	0	0

(C-8103) Fig. 2.14.6(b) : Truth table for 1 : 8 demux

– Depending on the combination of the select inputs S_2 S_1 S_0, the data input D_{in} is connected to one of the eight outputs.

– For example if S_2 S_1 S_0 = 1 1 0 then D_{in} is connected to output Y_6.

2.14.4 IC 74138 as 1 : 8 DE-MUX :

– The pin configuration of the 3 : 8 decoder IC 74138 is shown in Fig. 2.14.7.

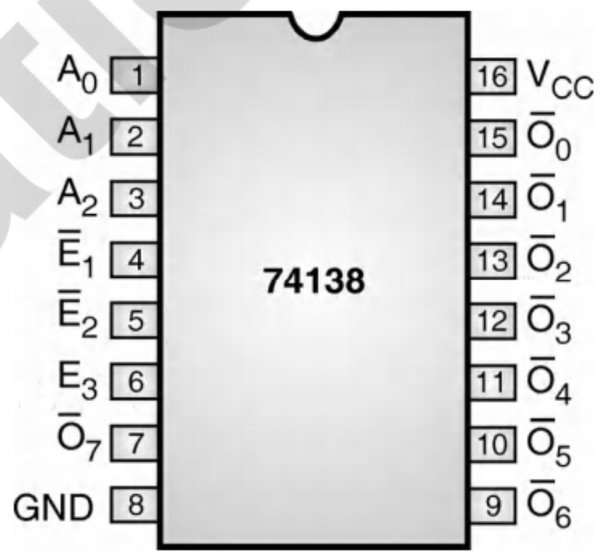
– A_0 , A_1 , A_2 are the three address lines or the three input lines. We have to apply the 3-bit binary data to these inputs.

– But when this IC is to be used as a 1 : 8 DEMUX we have to use these lines as the three select lines.

– $\overline{O}_0$ to $\overline{O}_7$ are the 8-output active low lines.

– There are three enable inputs out of which $\overline{E}_1$ and $\overline{E}_2$ are the active low enable inputs whereas E_3 is an active high enable input.

– We have to make $\overline{E}_1$ = $\overline{E}_2$ = 0 and E_3 = 1 in order to enable the IC.



(C-489) Fig. 2.14.7(a) : Pin configuration of IC 74138

Pin names	Description
$A_0 – A_2$	Address inputs (Select lines)
$\overline{E}_1 – \overline{E}_2$	Enable inputs (Active Low)
E_3	Enable Input (Active HIGH)
$\overline{O}_0 – \overline{O}_7$	Outputs (Active Low)

Fig. 2.14.7(b) : Pin names and description

Features of IC 74138 :

– Schottky process for achieving a high speed.

– It can work as a decoder or as a demultiplexer.

– Multiple enable inputs ensure easy expansion.

– Active low, mutually exclusive outputs.

TechKnowledge Publications

– The truth table for IC 74138 is shown in Table 2.14.4.

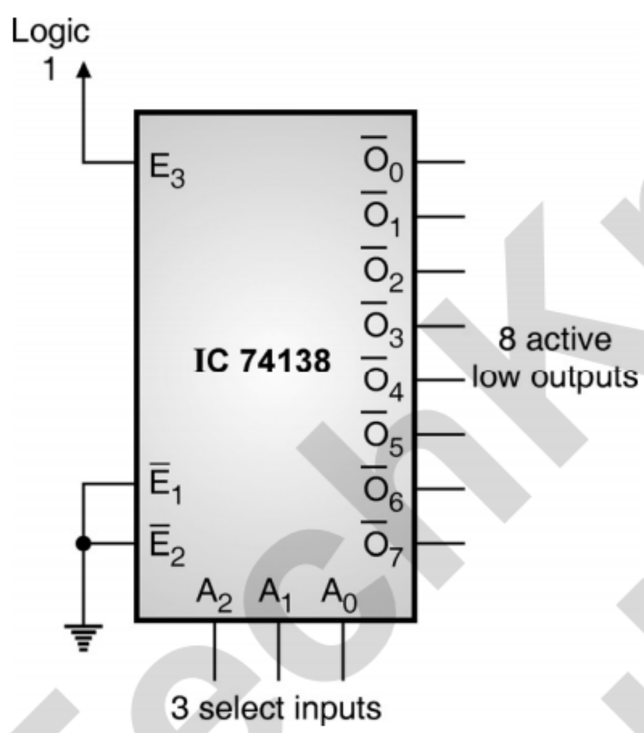
(C-6190) Table 2.14.4 : Truth table of 74138

Inputs						Outputs								
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$	
H	×	×	×	×	×	H	H	H	H	H	H	H	H	Decoder is disabled
×	H	×	×	×	×	H	H	H	H	H	H	H	H	
×	×	L	×	×	×	H	H	H	H	H	H	H	H	
L	L	H	L	L	L	L	H	H	H	H	H	H	H	Decoder is enabled
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	

A_0 = LSB, A_2 = MSB, H = HIGH Voltage Level,

L = LOW Voltage Level × = Don't care condition.

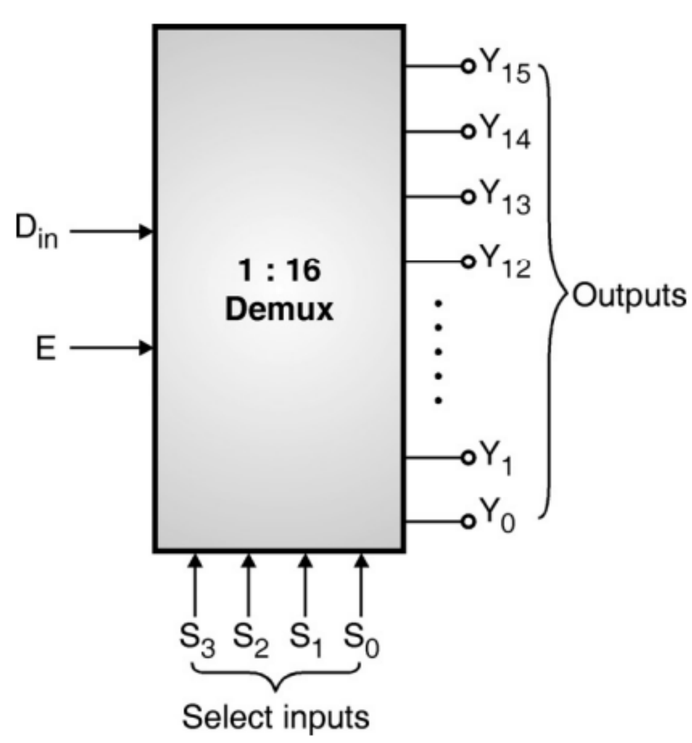
– The connection diagram of IC 74138 as 1:8 DEMUX is as shown in Fig. 2.14.8.



(C-3580) Fig. 2.14.8 : IC74138 as 1:8 DEMUX

2.14.5 1 : 16 Demultiplexer :

– The 1 : 16 demultiplexer block diagram is shown in Fig. 2.14.9.



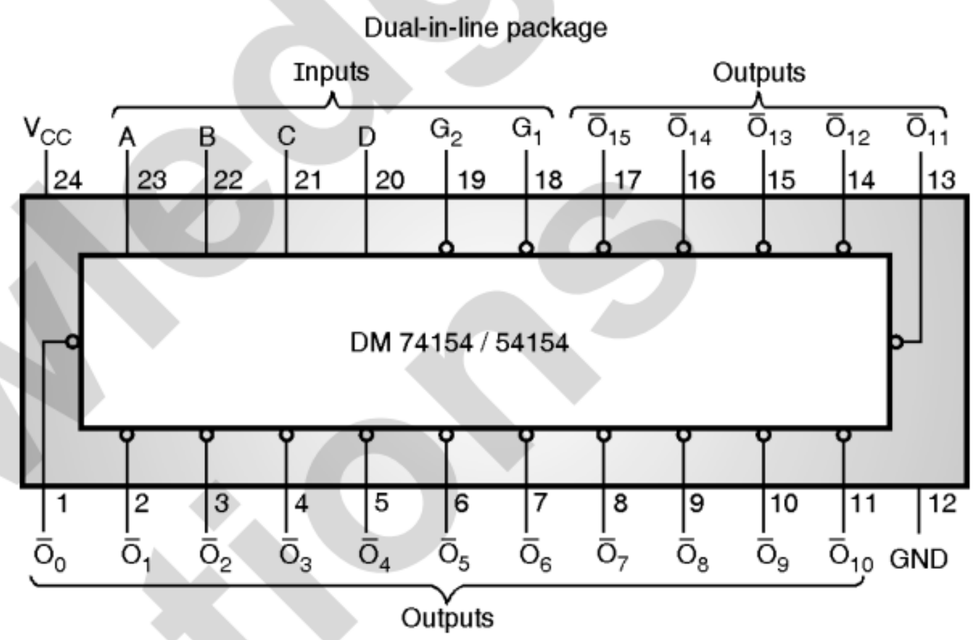
(C-456) Fig. 2.14.9 : 1 : 16 demultiplexer

– It has one data input D_{in}, four select inputs S_3, S_2, S_1 and S_0 sixteen outputs and one enable input.

– As discussed earlier, the combination of select inputs will select one of the sixteen outputs and connect it to D_{in}.

2.14.6 DM 54 LS 154 / DM 74 LS 154 : 4 Line to 16 Line Decoder / Demux :

– The pin configuration of this IC is as shown in Fig. 2.14.10.



(C-457) Fig. 2.14.10 : Pin configuration of IC 74154 / 54154

– A, B, C, D are the four input lines, G_1 and G_2 are the active low strobe lines.

– O_0 to O_{15} are the 16-active low outputs.

– This is a TTL IC hence needs a unipolar power supply of + 5 Volts.

– These ICs can be used as 1 : 16 demultiplexer. To do so, one of the strobe inputs (G_1 or G_2) is used as data input (D_{in}) line and the input lines A, B, C, D act as select lines.

– The other strobe input should be connected to logic low level.

– If G_1 or G_2 or both are high then all the outputs will be high (inactive).

– 74LS154 / 54 LS 154 are ideally suitable for implementing high performance memory decoders.

– All the inputs are buffered and input clamping diodes are provided.

Features :

1. Decodes 4 binary inputs into 16 mutually exclusive active low outputs.
2. Can perform the demultiplexer operation.
3. High fan out, low impedance, totem pole outputs.

Technknowledge Publications

4. Propagation delay : 19 to 23 nS.
5. Power dissipation : 45 mW (typical).

Truth table :

– The truth table for IC-74154 is shown in Table 2.14.5.

(C-8137) Table 2.14.5 : Function table of a IC-74154

Inputs						Outputs															
$\overline{G}_2$	$\overline{G}_1$	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

2.15 Demultiplexer Tree :

– Similar to multiplexer we can construct the demultiplexer having more number of lines using demultiplexers having less number lines.

– This is called as demultiplexer tree. It is also called as cascading of demultiplexers.

– This concept will be clear by solving the following examples.

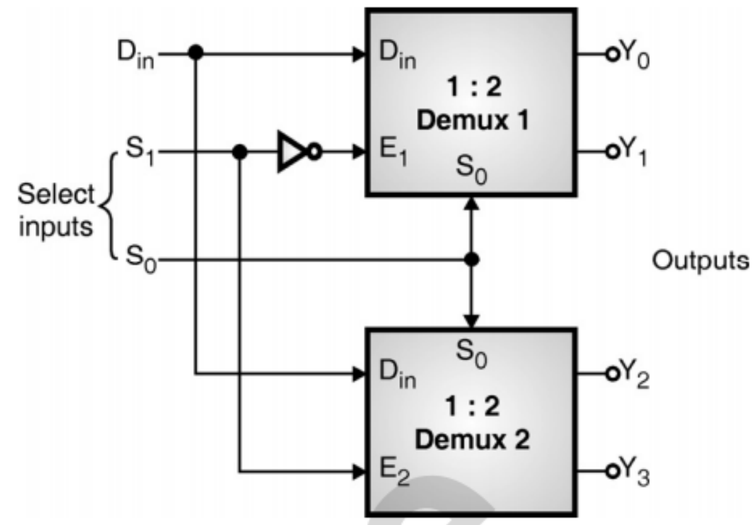
Ex. 2.15.1 : Obtain a 1 : 4 line demux using 1 : 2 demultiplexers.

Soln. :

– Refer Fig. P. 2.15.1 for the solution.

– The select lines S_0 of both the 1 : 2 demultiplexers are connected together and brought out as S_0 line of the 1 : 4 demux.

– The second select line S_1 is connected directly to the enable (E_2) input of demux 2 whereas inverted S_1 is connected to the enable input of demux – 1.



(C-459) Fig. P. 2.15.1 : 1 : 4 demultiplexer using 1 : 2 demultiplexers

– The truth table of the circuit is shown in the Table P. 2.15.1.

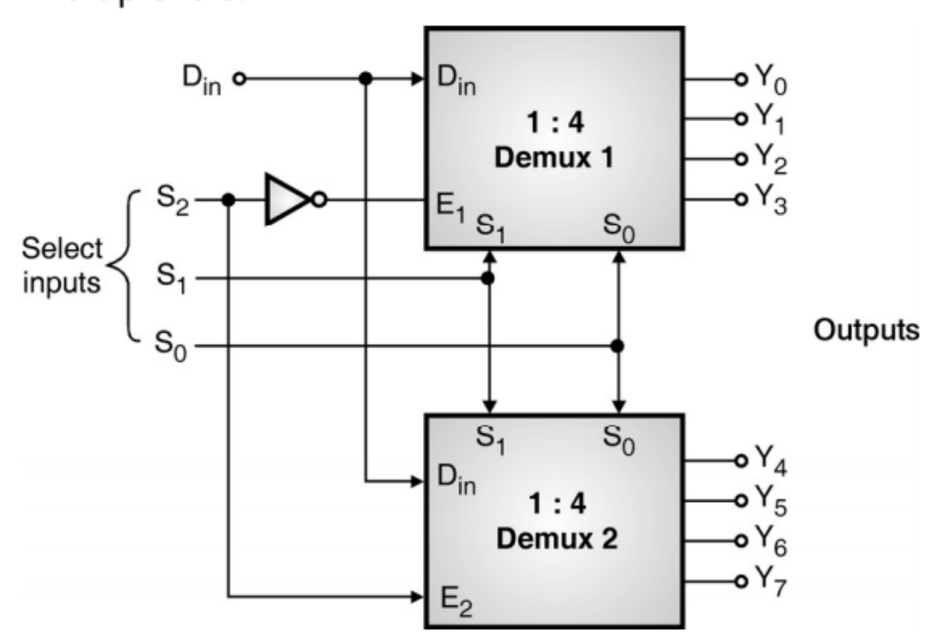
(C-6191) Table P. 2.15.1 : Truth table of 1 : 4 demux using two 1 : 2 demux

Select inputs		Outputs				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	
0	0	0	0	0	D_{in}	Demux - 1 enabled
0	1	0	0	D_{in}	0	
1	0	0	D_{in}	0	0	Demux - 2 enabled
1	1	D_{in}	0	0	0	

Ex. 2.15.2 : Obtain a 1 : 8 line demultiplexer using two 1 : 4 line demultiplexers.

Soln. :

– The 1 : 8 demultiplexer using two 1 : 4 demultiplexers is shown in Fig. P. 2.15.2. This is similar to cascading of multiplexers.



(C-460) Fig. P. 2.15.2 : 1 : 8 demultiplexer using two 1 : 4 demultiplexers

– The select lines S_1 and S_0 of the two 1 : 4 demultiplexers are connected in parallel with each other and S_2 is used for selecting one of the two 1 : 4 demultiplexers.

– S_2 is connected directly to enable (E) input of Demux – 2 whereas inverted S_2 is connected to the enable input of demux – 1.

Tech Knowledge Publications

– The truth table of this circuit is shown in Table P. 2.15.2.

(C-6192) Table P. 2.15.2 : Truth table of 1 : 8 demultiplexer using two 1 : 4 demultiplexers

Select inputs			Outputs								
S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	
0	0	0	D_{in}	0	0	0	0	0	0	0	Demux-1 enabled
0	0	1	0	D_{in}	0	0	0	0	0	0	
0	1	0	0	0	D_{in}	0	0	0	0	0	
0	1	1	0	0	0	D_{in}	0	0	0	0	
1	0	0	0	0	0	0	D_{in}	0	0	0	Demux-2 enabled
1	0	1	0	0	0	0	0	D_{in}	0	0	
1	1	0	0	0	0	0	0	0	D_{in}	0	
1	1	1	0	0	0	0	0	0	0	D_{in}	

Ex. 2.15.3 : Obtain 1 : 32 demultiplexer using 1 : 8 demultiplexers.

Soln. : Solve it yourself.

2.15.1 Comparison of Multiplexer and Demultiplexer :

Table 2.15.1 : Comparison of MUX and DEMUX

Sr. No.	Parameter	Multiplexer	Demultiplexer
1.	Type of logic circuit	Combinational	Combinational
2.	Number of data inputs	n	1
3.	Number of select inputs	m	m
4.	Number of data output	1	n
5.	Relation between input / output lines and select lines	$n = 2^m$	$n = 2^m$
6.	Operation principle	Many to 1 or as data selector	1 to many or data distributor
7.	Application	– As a universal logic circuit we can implement any combinational circuit using a MUX. – In time division multiplexing at the sending end.	– We can implement some combinational circuit. – In TDM system at receiving end.

2.15.2 Use of DEMUX in Combinational Logic Design :

– Like multiplexers, we can use the demultiplexers for designing the combinational circuits.

– Demultiplexers ICs for 1:2 , 1:4, 1:8 and 1:16 demultiplexers are available.

– We can use them to implement the given Boolean expressions representing a combinational circuits.

– Thus it is possible to design and implement many combinational circuits by using demultiplexer and a few logic gates.

Advantages :

– Advantages of using demultiplexers for logic design are as follows :

1. Logic design is simplified.
2. It is not necessary to simplify the logic expression.
3. It minimizes the number of ICs required to be used.

Use of DEMUX for combinational circuit design :

1. A truth table or logic expression is given to us.
2. We have to follow the design procedure given below to use DEMUX for implementing the given logical expression.

Design procedure :

Step 1 : Identify the decimal number corresponding to each minterm in the given expression as illustrated below.

$$\text{If } Y = \underbrace{\bar{A}\bar{B}\bar{C}}_{0} + \underbrace{A\bar{B}C}_{5} + \underbrace{ABC}_{7}$$ **(C-6264)**

Step 2 : The input data line of the demultiplexer is connected to a logic 1. The outputs corresponding to these numbers (0, 5 and 7) are connected as inputs to a suitable multi input OR gate.

Step 3 : The inputs (A, B, C) of the combinational circuit being designed are connected to the select inputs.

– The following example will demonstrate this concept.

Ex. 2.15.4 : Implement a full adder using demultiplexer.

Dec. 09, Dec. 10, 4 Marks, May 10, 3 Marks

TechKnowledge Publications

Soln. :

Step 1 : Write the truth table of the full adder :

(C-7621) **Table P. 2.15.4 : Truth table of a full adder**

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Step 2 : Write expressions for sum and carry :

$$\text{Sum} = \Sigma\, m\,(1, 2, 4, 7)$$

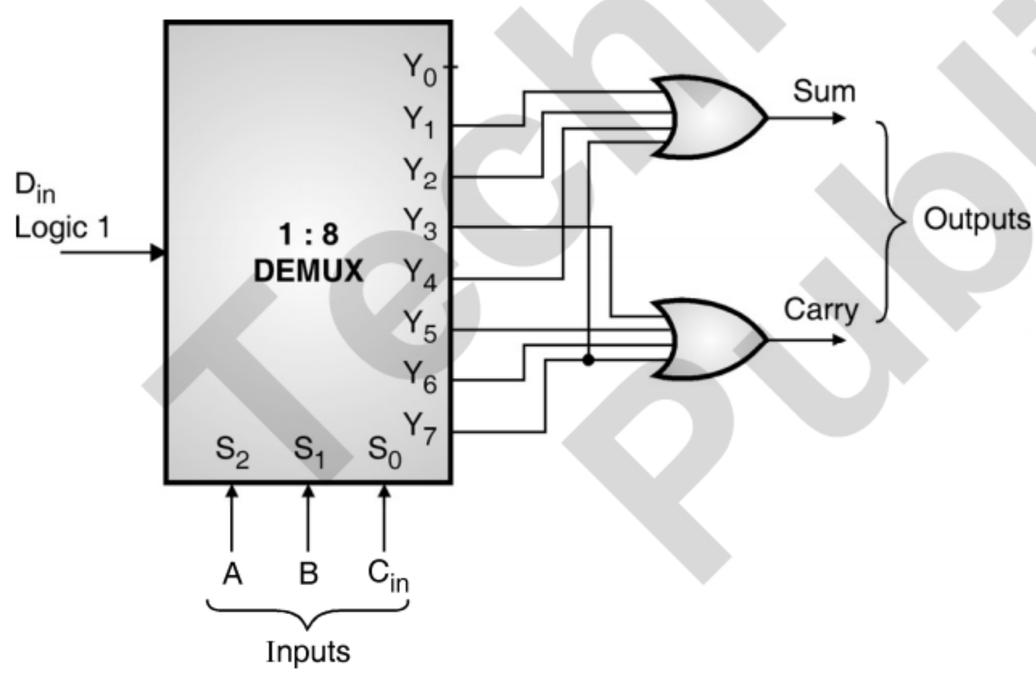
$\therefore$ Outputs Y_1, Y_2, Y_4 and Y_7 should be Ored to get sum output.

$$\text{Carry} = \Sigma\, m\,(3, 5, 6, 7)$$

$\therefore$ Outputs Y_3, Y_5, Y_6 and Y_7 should be Ored to get the carry output.

Step 3 : Implementation using a 1 : 8 demultiplexer :

– Fig. P. 2.15.4 shows the implementation.



(C-465) **Fig. P. 2.15.4 : Full adder using a demultiplexer**

Ex. 2.15.5 : Implement the full subtractor using a 1 : 8 demultiplexer.

Dec. 09, Dec. 10, 4 Marks, May 10, 3 Marks

Soln. :

Step 1 : Write the truth table of the full subtractor :

– The truth table of a full subtractor is as follows :

(C-7461) **Table P. 2.15.5 : Truth table of a full subtractor**

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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

Step 2 : Write expressions for sum and carry :

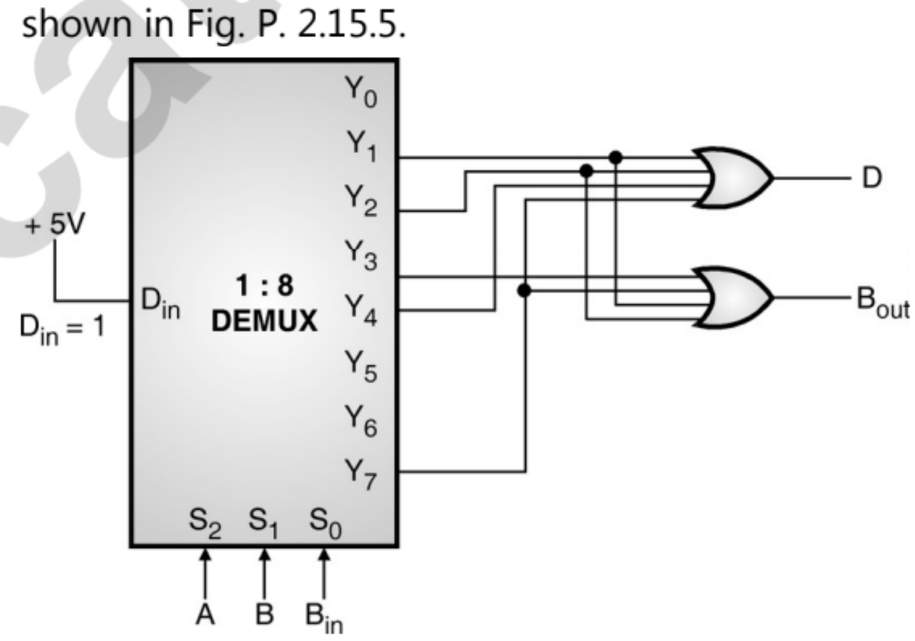
– From the truth table we can express the difference and borrow out outputs in the standard SOP forms as,

$$D = f\,(A, B, B_{in}) = \Sigma\, m\,(1, 2, 4, 7)$$

$$\text{and } B_{out} = f\,(A, B, B_{in}) = \Sigma\, m\,(1, 2, 3, 7)$$

Step 3 : Implementation using a 1 : 8 demultiplexer :

– Connect D_{in} to logic 1 permanently and connect A, B and B_{in} to the select inputs S_2, S_1, S_0 respectively as shown in Fig. P. 2.15.5.



(C-464) **Fig. P. 2.15.5 : Full subtractor using 1 : 8 demux**

– As D_{in} is connected to 1, we get the required minterms at the Demux outputs.

– We have to OR the required minterms to obtain the D and B_{out} outputs as shown in Fig. P. 2.15.5.

Note : We can use IC 74138 as 1 : 8 DEMUX to implement the Boolean expression with 1 : 8 DEMUX.

Ex. 2.15.6 : Implement the following functions using demultiplexer :

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

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

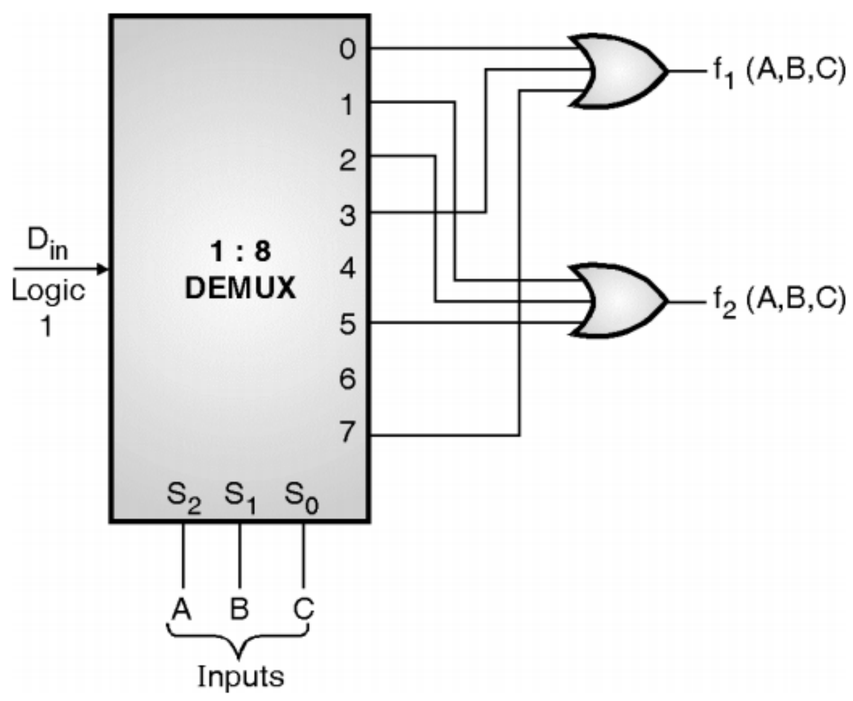
Soln. :

Step 1 : Select the DEMUX :

– We have to use a 1 : 8 demux because there are 8 inputs.

Step 2 : Implement the circuit :

– Fig. P. 2.15.6 shows the implementation.



(C-1323) **Fig. P. 2.15.6**

Ex. 2.15.7 : Implement two bit comparator using 1 : 16 demultiplexer (active low output). Draw the truth table of two bit comparator and explain the design in steps. **Dec. 11, 8 Marks**

Soln. :

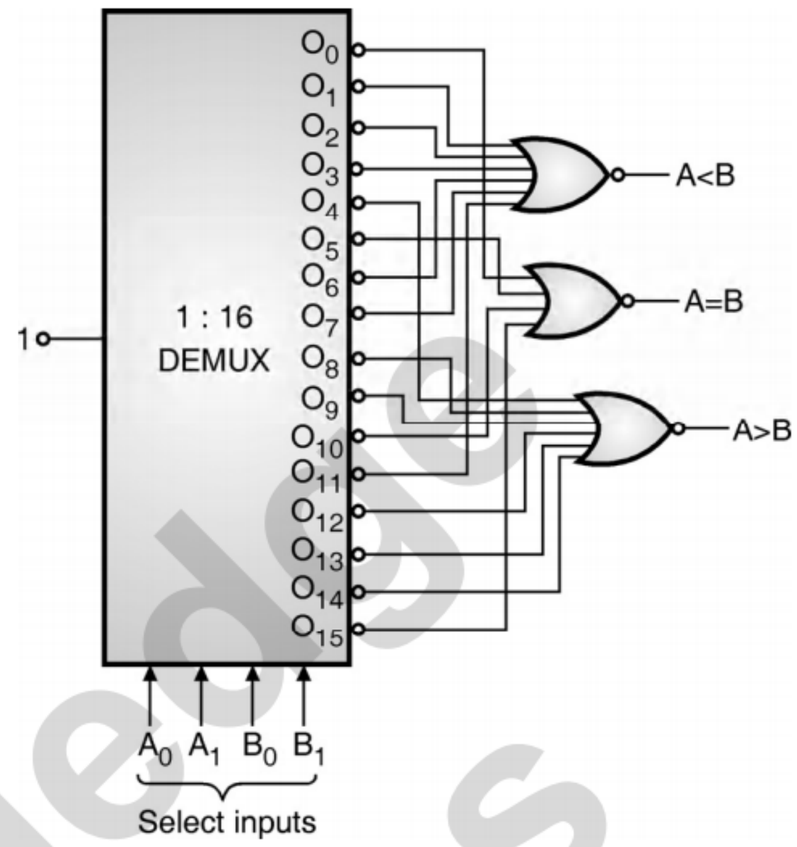
– For a 2-bit comparator, each input word is 2 bit long.

– The truth table of a 2-bit comparator is shown in Table P. 2.15.7.

(C-406(a)) **Table P. 2.15.7 : Truth table for a 2-bit comparator**

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	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

– Implementation using 1 : 16 demultiplexer is as shown in Fig. P. 2.15.7.



(C-1969) **Fig. P. 2.15.7 : 2 bit comparator using 1 : 16 DEMUX**

2.16 Encoders :

Definition :

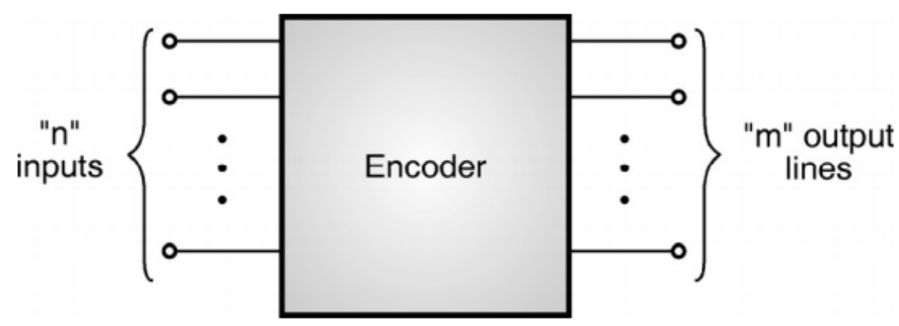
– Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder.

– An encoder has "n" number of input lines and "m" number of output lines.

– An encoder produces an m bit binary code corresponding to the n bit digital number, applied at its input.

Block diagram :

– Block diagram of an encoder is shown in Fig. 2.16.1.



(C-466) **Fig. 2.16.1 : Block diagram of an encoder**

– The encoder accepts an n input digital word and converts it into an m bit another digital word.

– For example a BCD number applied at the input can be converted into a binary number at the output.

– The internal combinational circuit of the encoder is designed accordingly.

2.16.1 Types of Encoders :

The types of encoders which we are going to discuss are as follows :

1. Priority encoders.

2. Decimal to BCD encoder.
3. Octal to binary encoder.
4. Hexadecimal to binary encoder.

2.17 Priority Encoder : SPPU : Dec. 08

University Questions

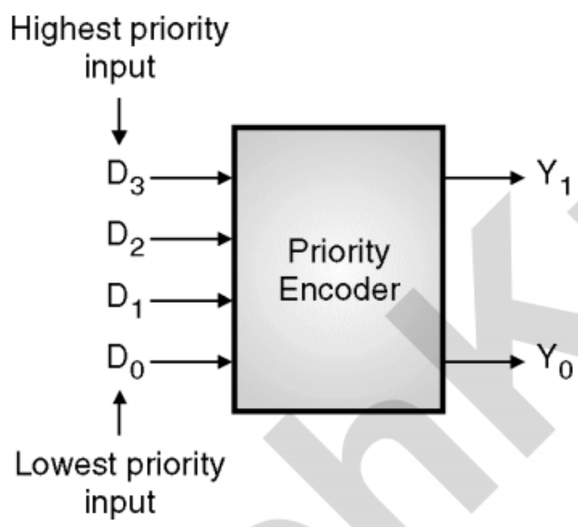
Q. 1 What is priority encoder ? **(Dec. 08, 2 Marks)**

Definition :

– This is a special type of encoder with priorities assigned to all its input lines. If two or more input lines are "1" at the same time, then the input line with highest priority will be considered.

Block diagram and truth table :

– The block diagram of a priority encoder is shown in Fig. 2.17.1(a) and its truth table is shown in Fig. 2.17.1(b).



(a) Block diagram of a priority encoder

Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_1	Y_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

(b) Truth table of a priority encoder

(C-467) Fig. 2.17.1

– Priorities are given to the input lines. If two or more input lines are "1" at the same time, then the input line with highest priority will be considered.

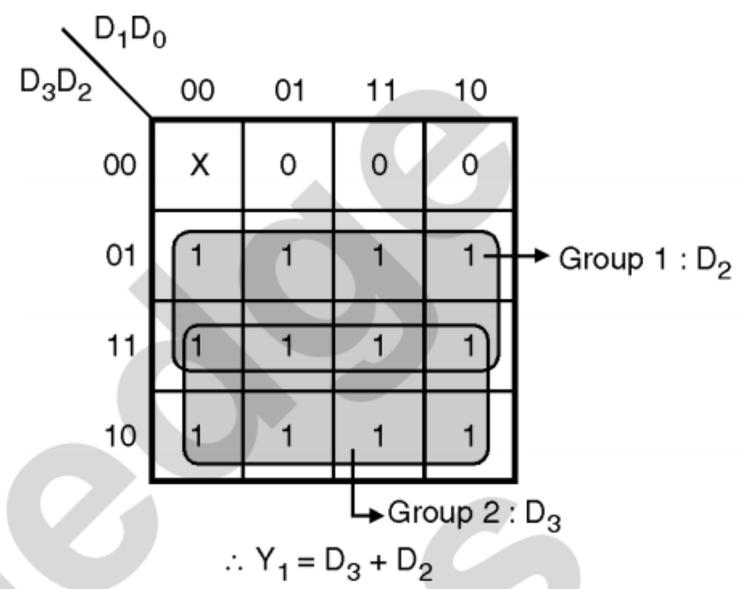
– There are four inputs, D_0 through D_3 and two outputs Y_1 and Y_0 . Out of the four inputs D_3 has the highest priority and D_0 has the lowest priority.

– That means if D_3 = 1 then $Y_1 Y_0$ = 11 irrespective of the other inputs. Similarly if D_3 = 0 and D_2 = 1 then $Y_1 Y_0$ = 10 irrespective of the other inputs.

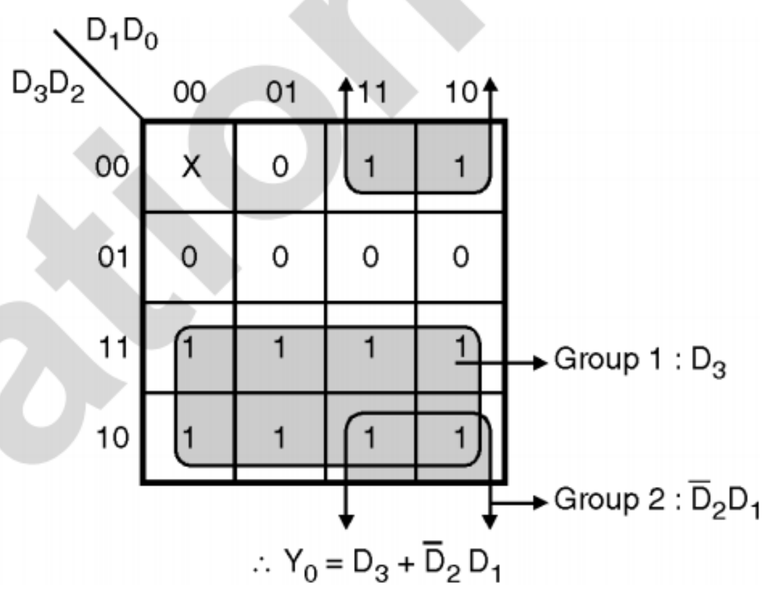
– Carefully go through the truth table shown in Fig. 2.17.1(b) to get the feel of priority encoder operation.

K maps for the outputs :

– The K maps for the two outputs are as shown in Fig. 2.17.2.



(a) K map for Y_1

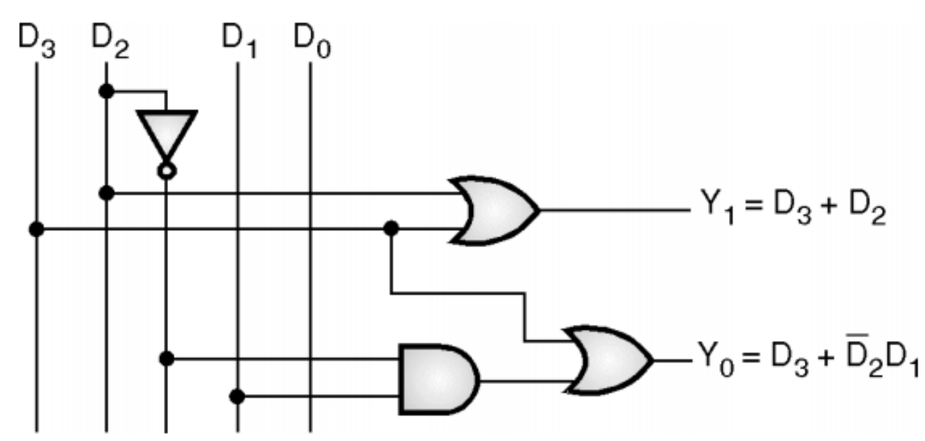


(b) K map for Y_0

(C-468) Fig. 2.17.2

Implementation :

– The logic diagram for the priority encoder is shown in Fig. 2.17.3.



(C-469) Fig. 2.17.3 : Logic diagram of a priority encoder

2.17.1 Priority Encoders in the IC Form :

– The priority encoders are available in the integrated circuit form. The available encoders are :

74147	10 : 4 Priority encoder (Decimal to BCD)
74148	Octal to binary priority encoder

– 74147 is basically a decimal to BCD priority encoder.

TechKnowledge Publications

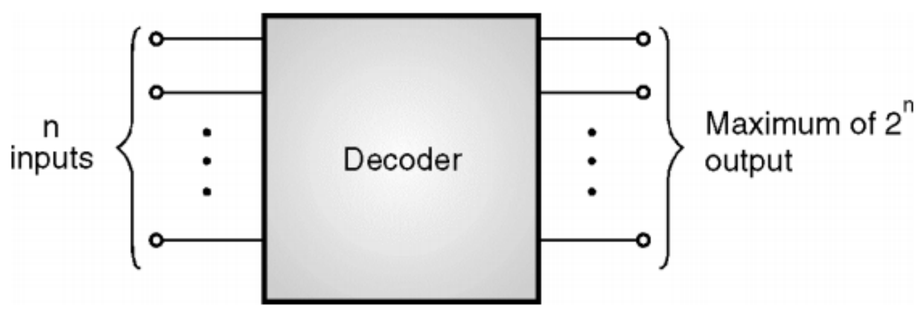
2.18 Decoder :

Definition :

– A decoder is a combinational circuit with "n" inputs and to a maximum 2^n outputs that are related to each other with a certain rule.

Block diagram :

– Fig. 2.18.1 shows the block diagram of a decoder. It has "n" inputs and to a maximum 2^n outputs.



(C-479) **Fig. 2.18.1 : Block diagram of a decoder**

– Decoder is identical to a demultiplexer without any data input. It performs operations which are exactly opposite to those of an encoder.

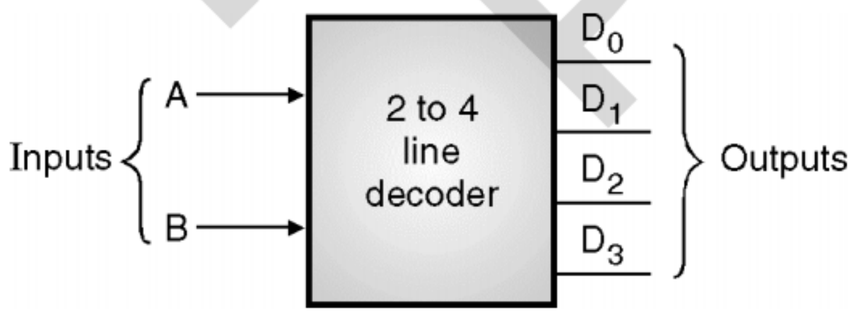
Typical applications :

1. Code converters.
2. BCD to seven segment decoders.
3. Nixie tube decoders. 4. Relay actuators.

2.18.1 2 to 4 Line Decoder :

Block diagram :

– The block diagram of a 2 to 4 line decoder is shown in Fig. 2.18.2(a). A and B are the two inputs whereas D_0 through D_3 are the four outputs.



(C-480) **Fig. 2.18.2(a) : Block diagram of a 2 line to 4 line decoder**

Truth table :

– Table 2.18.1 shows the truth table which explains the operation of the decoder. It shows that each output is "1" for only a specific combination of inputs.

(C-480(a)) **Table 2.18.1 : Truth table of a 2 line to 4 line decoder**

Inputs		Outputs			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

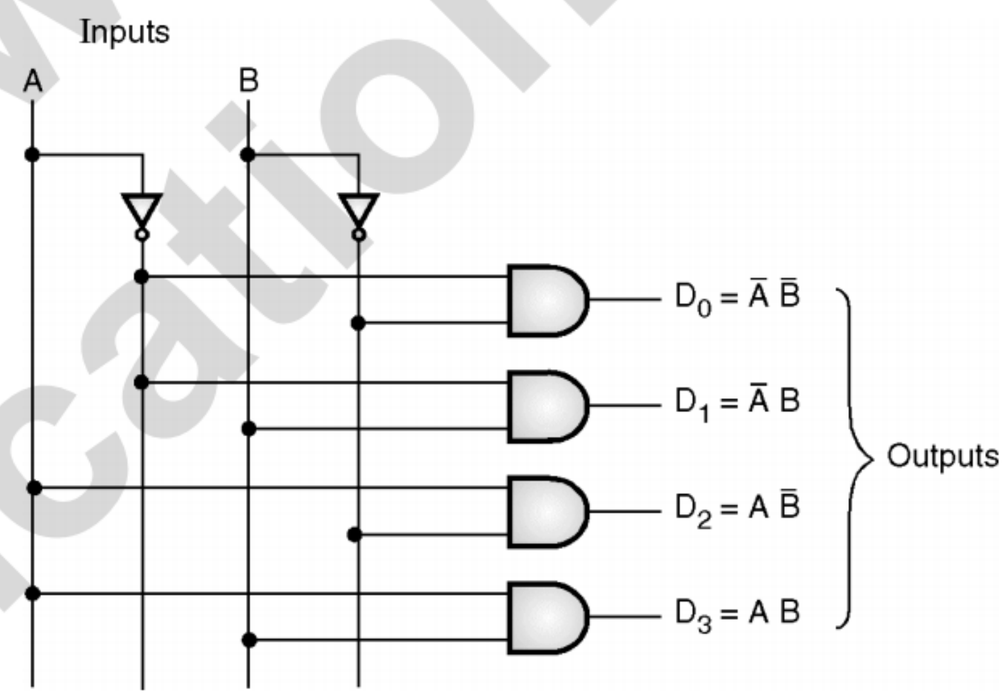
– The Boolean expressions for the four outputs are,

$$D_0 = \bar{A}\,\bar{B} \qquad D_1 = \bar{A}\,B$$

$$D_2 = A\,\bar{B} \quad \text{and} \quad D_3 = A\,B$$

Logic diagram :

– The logic diagram for a 2 to 4 line decoder is shown in Fig. 2.18.2(b).

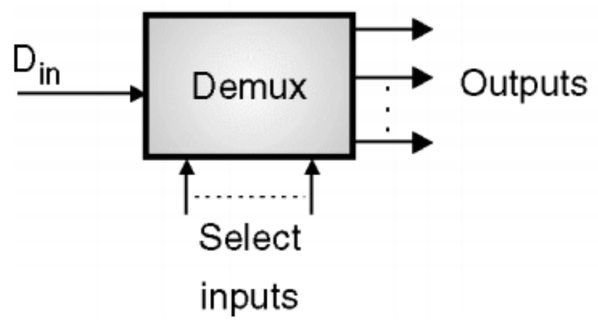


(C-481) **Fig. 2.18.2(b) : 2 to 4 line decoder**

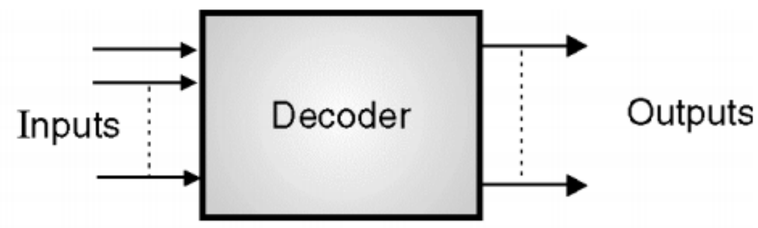
2.18.2 Difference between Decoder and Demultiplexer :

Sr. No.	Parameter	Demux	Decoder
1.	Block schematic	See Fig. A	See Fig. B.
2.	Number of data inputs	One	More than 1.
3.	Select inputs	Present	Absent
4.	Applications	As a distributor switch, to implement Boolean expressions.	BCD to seven segment decoder, Decimal to BCD, Binary to Hex etc.

TechKnowledge Publications



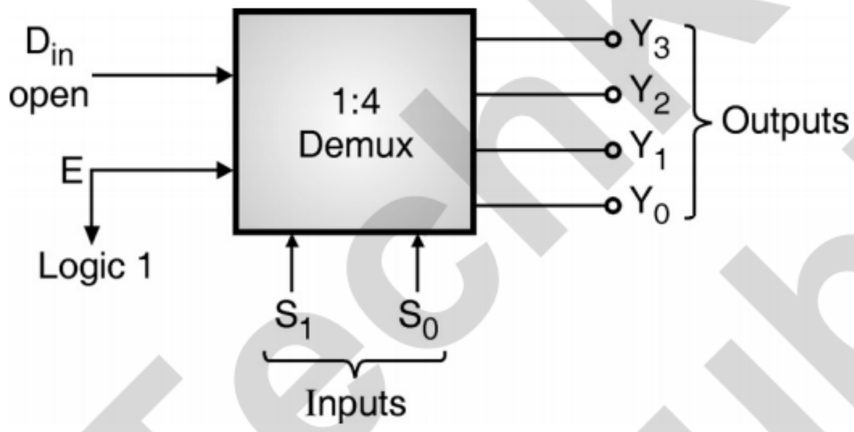
(C-482) Fig. A



(C-482) Fig. B

2.18.3 Demultiplexer as Decoder :

- We can use a demultiplexer as a decoder.
- Let us see how to operate a 1 : 4 demux as 2 : 4 decoder.
- Consider Fig. 2.18.3 which shows a 1 : 4 demultiplexer.
- D_{in} is the data input, S_1 S_0 are the select lines and Y_3 through Y_0 are the outputs.
- In order to operate it as 2 : 4 decoder, we have to use S_1 S_0 as inputs, keep D_{in} open and use Y_3 to Y_0 as outputs as shown in Fig. 2.18.3.

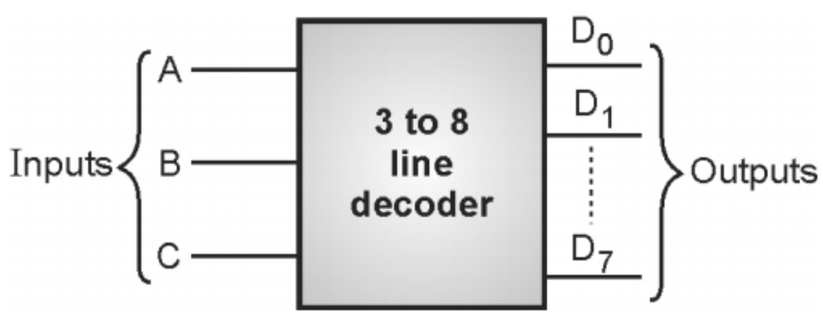


(C-483) Fig. 2.18.3 : 1 : 4 demux as 2 : 4 decoder

2.18.4 3 to 8 Line Decoder :

Block diagram :

- The block diagram of a 3 to 8 line decoder is shown in Fig. 2.18.4(a).
- A, B and C are the three inputs whereas D_0 through D_7 are the eight outputs.



(C-7773) Fig. 2.18.4(a) : Block diagram of a 3 line to 8 line decoder

Truth table :

- The truth table of 3 to 8 line decoder is shown in Table 2.18.2.

(C-6477) Table 2.18.2 : Truth table for 3 to 8 line decoder

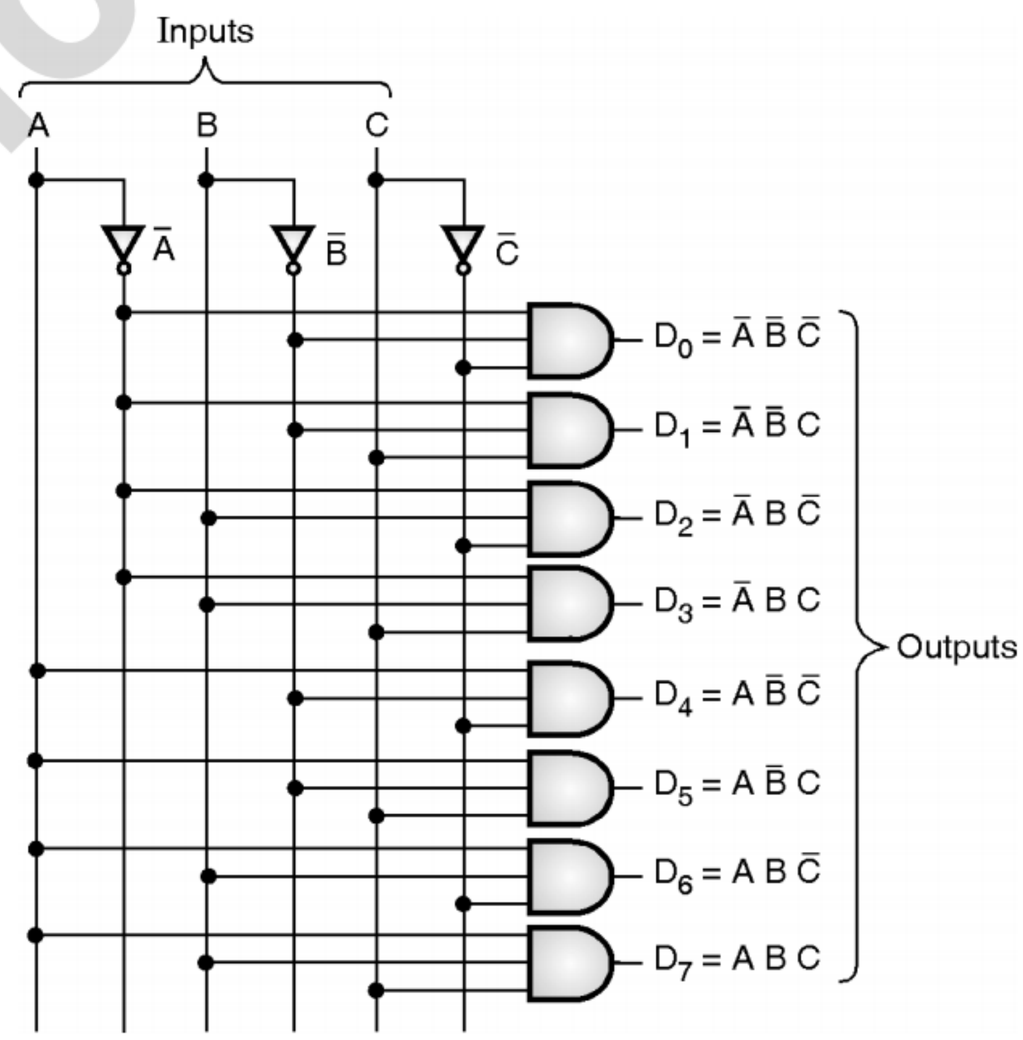
Inputs			Outputs								
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \bar{A}\bar{B}C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \bar{A}B\bar{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \bar{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\bar{B}C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = ABC$

Boolean expressions :

- The Boolean expressions for the eight outputs in terms of the three inputs are as given in Table 2.18.2.

Logic diagram :

- The logic diagram for 3 to 8 line decoder is shown in Fig. 2.18.4.



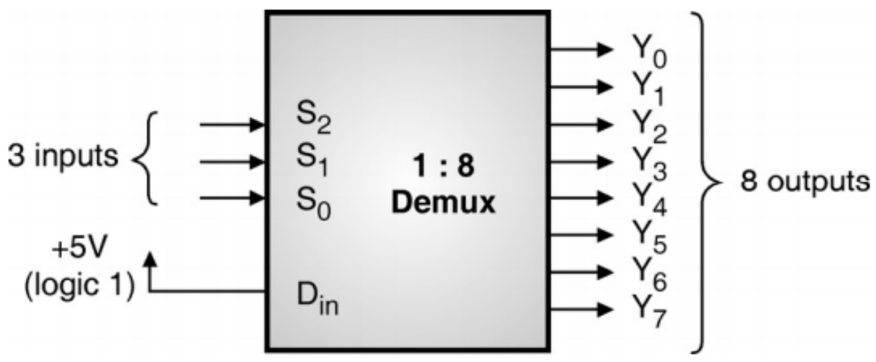
(C-484) Fig. 2.18.4 : 3 to 8 line decoder

2.18.5 1 : 8 DEMUX as 3:8 Decoder :

- In order to operate 1 : 8 Demux as a 3:8 line decoder, the connections are to be made as shown in Fig. 2.18.5.

Technknowledge Publications

– The data input D_{in} is connected to logic 1 permanently. The three select inputs S_0, S_1 and S_2 will act as three input lines of the decoder and Y_0 to Y_7 are the 8-output lines.



(C-485) Fig. 2.18.5 : Use of 1:8 demux as 3:8 decoder

– The truth table of this circuit is as given in Table 2.18.3 which shows that the circuit works as a 3 : 8 decoder.

(C-485(a)) Table 2.18.3 : Truth table

D_{in}	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
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	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

2.18.6 IC 74138 as 3 : 8 Decoder :

– The pin configuration of the 3 : 8 decoder IC 74138 is shown in Fig. 2.18.6.

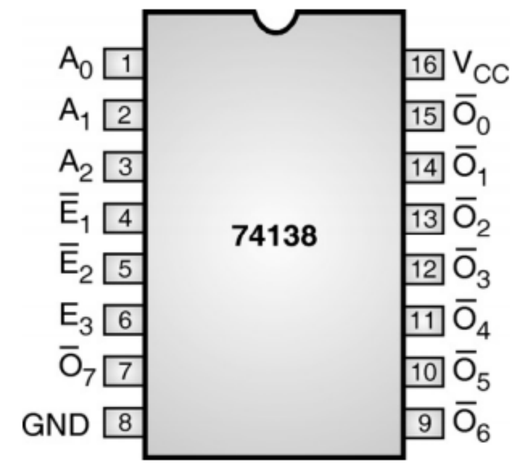
– We have already discussed this IC as a demultiplexer. Let us now see how to use it as a decoder.

– A_0 , A_1 , A_2 are the three address lines or the three input lines. We have to apply the 3-bit binary data to these inputs. So these lines act as the 3-data input lines of the 3:8 line decoder.

– $\overline{O}_0$ to $\overline{O}_7$ are the 8-output active low lines. These lines act as 8-data output lines of the 3 : 8 line decoder.

– There are three enable inputs out of which $\overline{E}_1$ and $\overline{E}_2$ are the active low enable inputs whereas E_3 is an active high enable input.

– We have to make $\overline{E}_1 = \overline{E}_2 = 0$ and $E_3 = 1$ in order to enable the IC.



(a) Pin configuration of IC 74138

Pin names	Description
$A_0 - A_2$	Address inputs (Select lines)
$\overline{E}_1 - \overline{E}_2$	Enable inputs (Active Low)
E_3	Enable Input (Active HIGH)
$\overline{O}_0 - \overline{O}_7$	Outputs (Active Low)

(b) Pin names and description

(C-489) Fig. 2.18.6

– The truth table for IC 74138 is shown in Table 2.18.4.

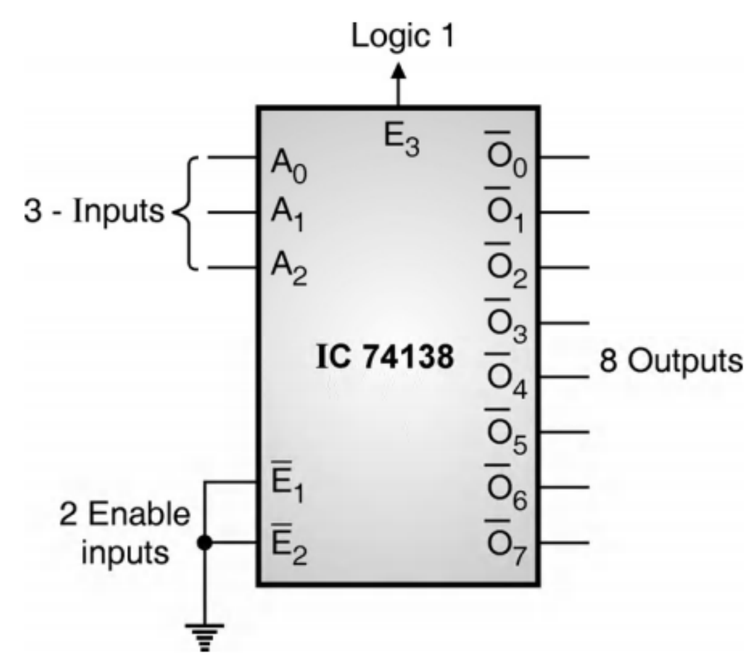
(C-6190) Table 2.18.4 : Truth table of 74138

Inputs						Outputs								
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$	
H	×	×	×	×	×	H	H	H	H	H	H	H	H	Decoder is disabled
×	H	×	×	×	×	H	H	H	H	H	H	H	H	
×	×	L	×	×	×	H	H	H	H	H	H	H	H	
L	L	H	L	L	L	L	H	H	H	H	H	H	H	Decoder is enabled
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	

A_0 = LSB, A_2 = MSB, H = HIGH Voltage Level,

L = LOW Voltage Level, × = Don't care condition

– Fig. 2.18.7 shows the connection diagram for IC 74138 used as a 3 : 8 decoder.



(C-3581) Fig. 2.18.7 : IC 74138 connected as 3:8 decoder

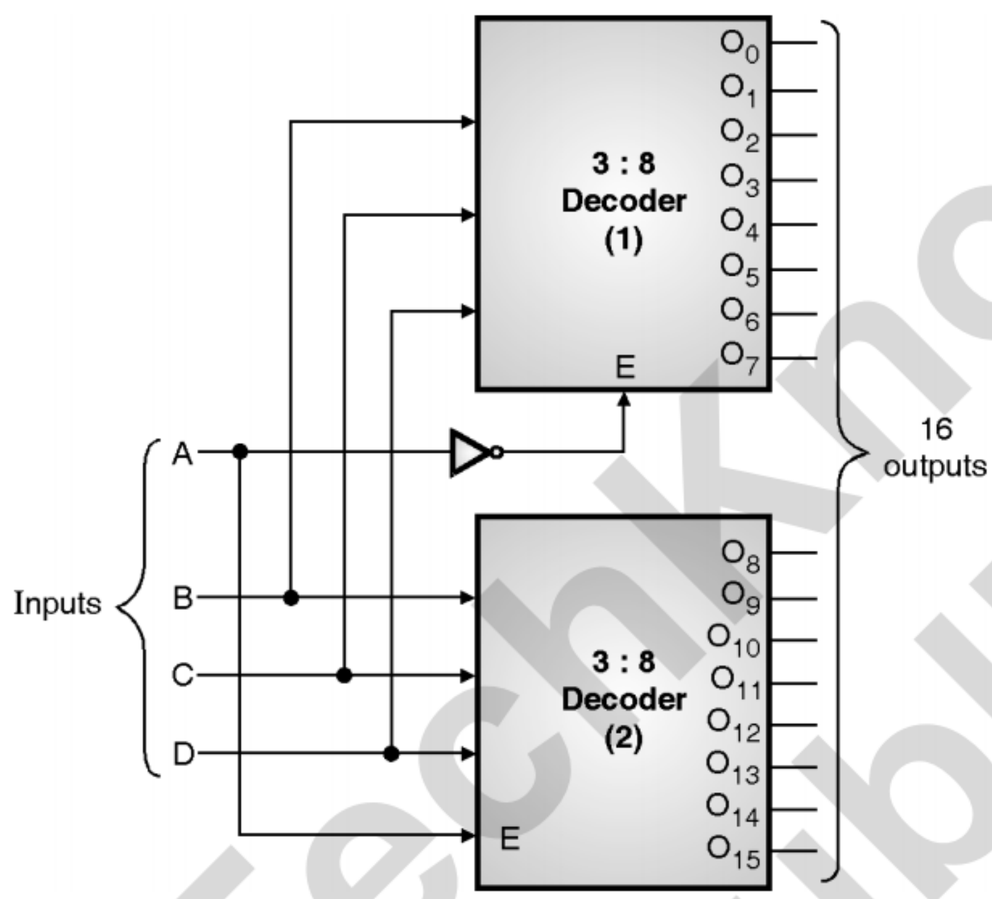
2.18.7 4 Line to 16 Line Decoder using 3 : 8 Decoder :

SPPU : May 06

University Questions

Q. 1 Design 4 : 16 decoder using 3 : 8 decoders (with enable inputs). Explain the truth table of your circuit in short. Explain any one application of decoder. **(May 06, 8 Marks)**

- The connection diagram for a 4 : 16 decoder using a 3 : 8 decoder is shown in Fig. 2.18.8.
- Note that the three inputs are connected in parallel with each other and then connected to the BCD inputs.
- The fourth input A is used to enable one of the two 3 : 8 decoders. $\overline{A}$ is applied to the enable input of decoder 1 whereas A is applied directly to decoder 2.



(C-488) Fig. 2.18.8 : 4 to 16 decoder using two 3 : 8 decoders

2.18.8 Combinational Logic Design Using Decoders :

- In order to implement a logic function using a decoder, the outputs corresponding to all the minterms will be **NANDed or bubbled ORed** if the decoder outputs are active low type (for example decoder IC 74138 has active low outputs).
- Following examples demonstrate the use of decoder for logic circuit design.

Ex. 2.18.1 : Implement the following logic function using a 4 : 16 decoder :

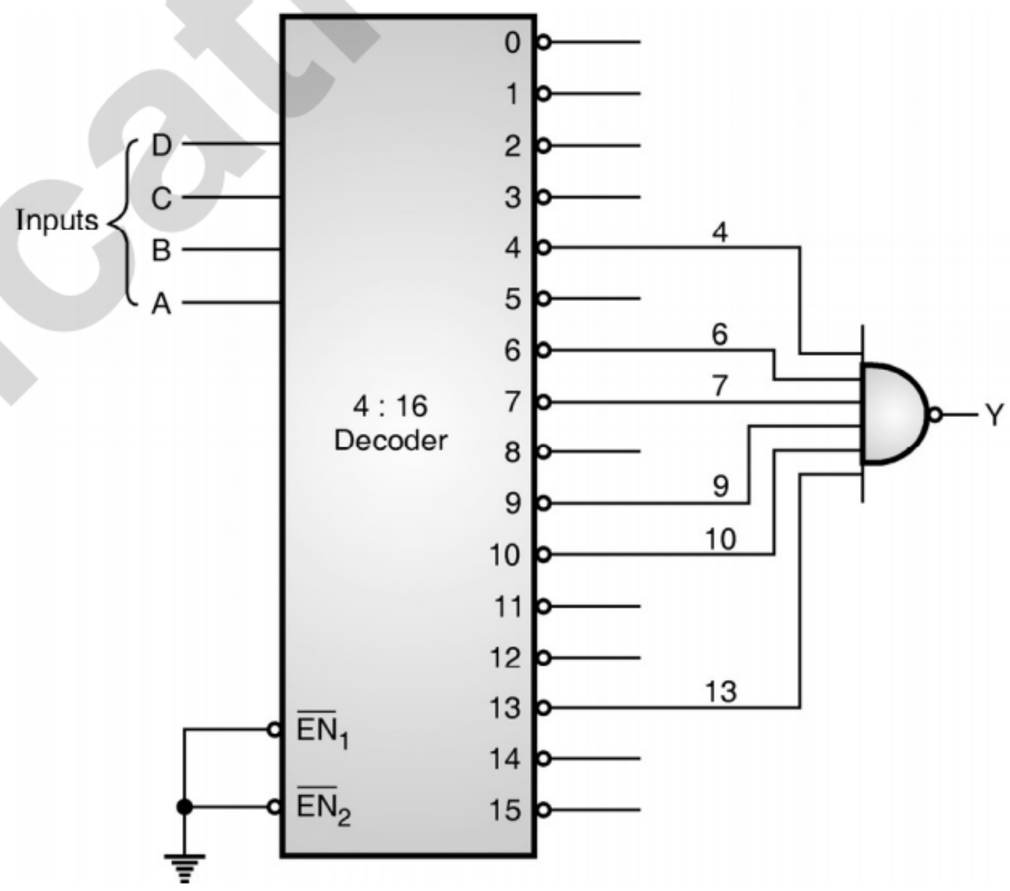
$Y = \sum m\,(4, 6, 7, 9, 10, 13)$

Soln. :

- Fig. P. 2.18.1(a) demonstrates the implementation. The given logic function indicates that it is a 4 variable function. Let the inputs be A, B, C and D.
- We can express the given function as follows :

$$Y = 4 + 6 + 7 + 9 + 10 + 13$$

- These output lines are ORed to form the desired function.
- As the outputs of decoder are asserted low (active low) a low asserted OR gate or a high asserted NAND gate is used on the output side.
- The decoder outputs 4, 6, 7, 9, 10 and 13 i.e. corresponding to all the minterms are connected to a bubbled OR or a NAND gate.
- Both enable inputs are grounded to enable to decoder.
- Fig. P. 2.18.1(a) shows the implementation.

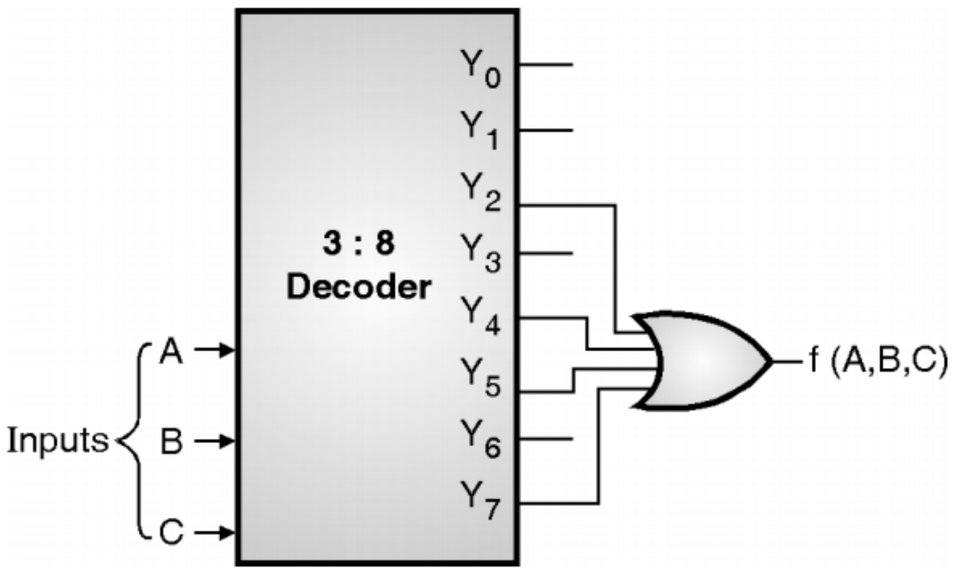


(C-4547) Fig. P. 2.18.1(a) : Realization

Ex. 2.18.2 : Implement the following Boolean function using a 3 : 8 decoder and external gates. $f(A, B, C) = \Sigma\,(2, 4, 5, 7)$

Soln. :

- The decoder produces minterms. The outputs Y_2, Y_4, Y_5 and Y_7 are ORed to produce the required output.
- The logic implementation of the given logic function is shown in Fig. P. 2.18.2.



(C-486) Fig. P. 2.18.2 : Implementation of Boolean equation using decoder and gate

Ex. 2.18.3 : Implement the following multiple output function using a suitable decoder.

f_1 (A, B, C) = Σ m (0, 4, 7) + d (2, 3),

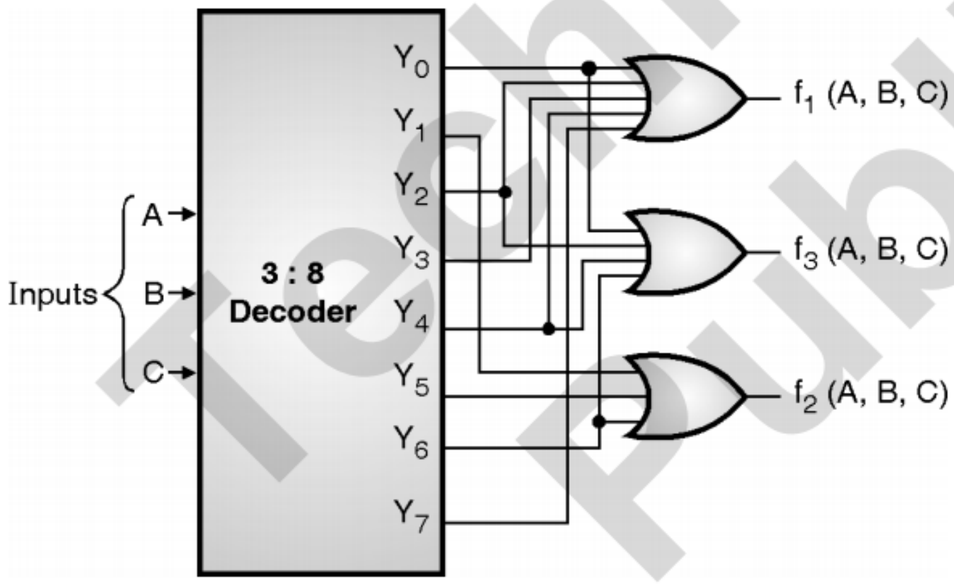
f_2 (A, B, C) = Σ m (1, 5, 6)

f_3 (A, B, C) = Σ m (0, 2, 4, 6).

Soln. :

– Let us use 3 line to 8 line decoder. f_1 consists of don't care conditions. So we will consider them to be logic 1.

– The implementation of the given three functions using a 3 : 8 decoder and a few OR gates is shown in Fig. P. 2.18.3.



(C-487) Fig. P. 2.18.3 : Implementation of the given Boolean functions using a 3 : 8 decoder and OR gates

Ex. 2.18.4 : Implement a 3-bit binary to gray code converter using decoder IC 74138.

May 11, 8 Marks, Dec. 13, 4 Marks

Soln. :

Step 1 : Write the truth table relating the binary and gray codes :

– The truth table is as follows :

(C-8138) Table P. 2.18.4 : Truth table relating the binary and gray codes

Decimal	Binary inputs			Gray outputs		
	B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Step 2 : Obtain the expressions for G_2 , G_1 and G_0 :

– Refer to the shaded portions of Table P. 2.18.4. Normally the expressions for G_2 , G_1 and G_0 would have been written in the SOP form as,

$$G_2 = O_4 + O_5 + O_6 + O_7$$

$$G_1 = O_2 + O_3 + O_4 + O_5$$

$$G_0 = O_1 + O_2 + O_5 + O_6$$

– But the outputs of IC 74138 are active low. Hence we will have to convert these equations in terms of inverted "O" outputs as follows :

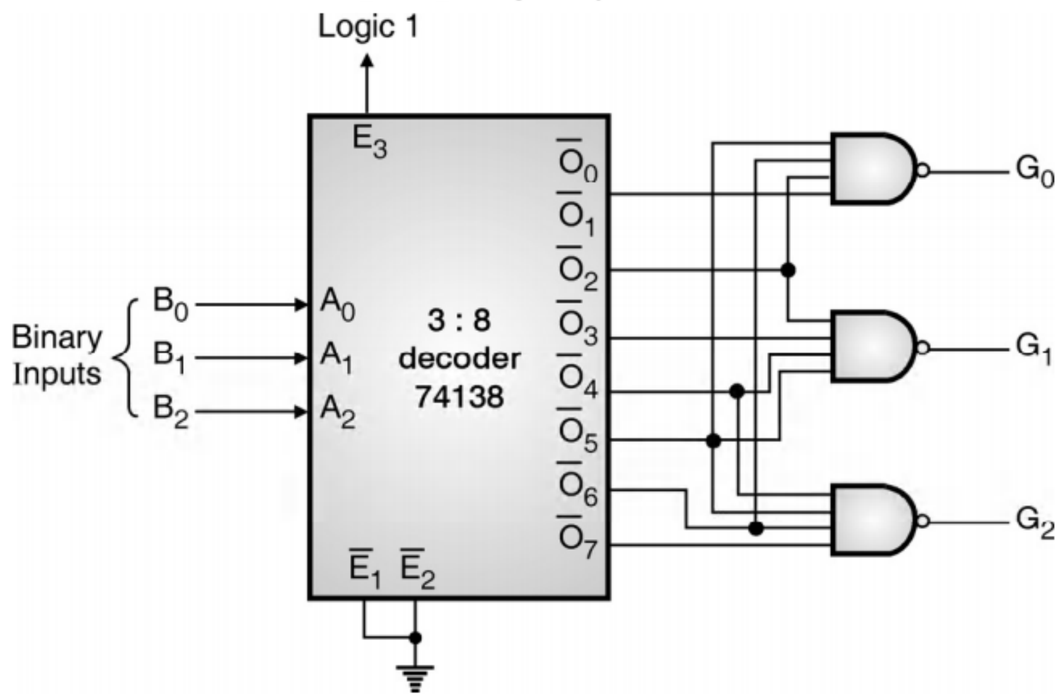
$$G_2 = O_4 + O_5 + O_6 + O_7 = \overline{\overline{O_4 + O_5 + O_6 + O_7}}$$

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

$$\therefore G_2 = \overline{\bar{O}_4 \cdot \bar{O}_5 \cdot \bar{O}_6 \cdot \bar{O}_7} \quad \text{... (1)}$$

$$\text{Similarly } G_1 = \overline{\bar{O}_2 \cdot \bar{O}_3 \cdot \bar{O}_4 \cdot \bar{O}_5} \quad \text{... (2)}$$

$$\text{And } G_0 = \overline{\bar{O}_1 \cdot \bar{O}_2 \cdot \bar{O}_5 \cdot \bar{O}_6} \quad \text{... (3)}$$



(C-492) Fig. P. 2.18.4 : Binary to gray code converter using decoder 74138

TechKnowledge Publications

– Equations (1), (2) and (3) are implemented as shown in Fig. P. 2.18.4. Note that each equation represents a 4-input NAND gate.

Note : We can use IC 74138 as 3 : 8 decoder in order to implement the Boolean equations using 3 : 8 decoder.

Ex. 2.18.5 : Implement Gray to Binary code converter using suitable decoder.

Soln. :

Step 1 : Truth table for 3-bit gray to binary code converter :

– Refer Table P. 2.18.5.

(C-8108) Table P. 2.18.5

Decimal	Gray input			Binary output		
	G_2	G_1	G_0	B_2	B_1	B_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
3	0	1	1	0	1	0
2	0	1	0	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1
5	1	0	1	1	1	0
4	1	0	0	1	1	1

Step 2 : To obtain the output expressions for B_2, B_1 and B_0 :

$$B_2 = O_4 + O_5 + O_6 + O_7$$

$$B_1 = O_2 + O_3 + O_6 + O_7$$

$$B_0 = O_1 + O_3 + O_5 + O_7$$

Step 3 : Implementation of gray to binary code converter using 3 : 8 decoder IC 74138 :

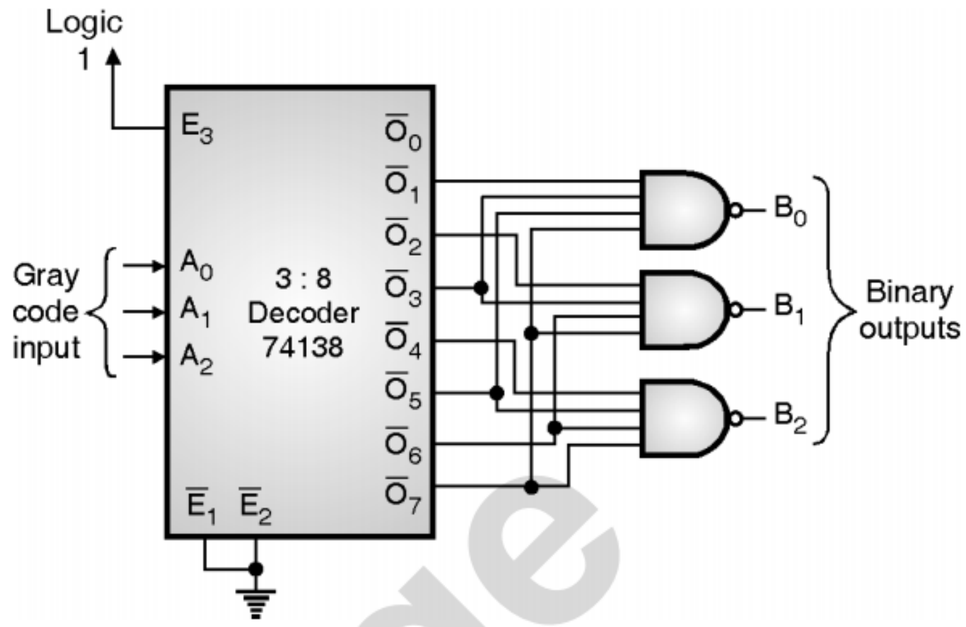
– Since the outputs of IC74138 are active low, the output expressions obtained are modified as follows :

$$B_2 = \overline{\overline{O_4 + O_5 + O_6 + O_7}} = \overline{\overline{O}_4 \cdot \overline{O}_5 \cdot \overline{O}_6 \cdot \overline{O}_7}$$

$$B_1 = \overline{\overline{O_2 + O_3 + O_6 + O_7}} = \overline{\overline{O}_2 \cdot \overline{O}_3 \cdot \overline{O}_6 \cdot \overline{O}_7}$$

$$B_0 = \overline{\overline{O_1 + O_3 + O_5 + O_7}} = \overline{\overline{O}_1 \cdot \overline{O}_3 \cdot \overline{O}_5 \cdot \overline{O}_7}$$

– Hence the logic diagram for Gray to Binary code converter using a 3 : 8 decoder and 4 input NAND gates is as shown in Fig. P. 2.18.5.



(C-1318) Fig. P. 2.18.5 : Logic diagram for Gray to Binary code converter

Ex. 2.18.6 : Implement full subtractor using decoder IC 74138 and 2-input NAND gate ICs 7400.

Soln. :

Truth table :

– The truth table for full subtractor is as shown in Table P. 2.18.6.

(C-7461) Table P. 2.18.6 : Truth table for full subtractor

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

– The A, B, B_{in} inputs are applied to the A_0, A_1 and A_2 inputs of the decoder.

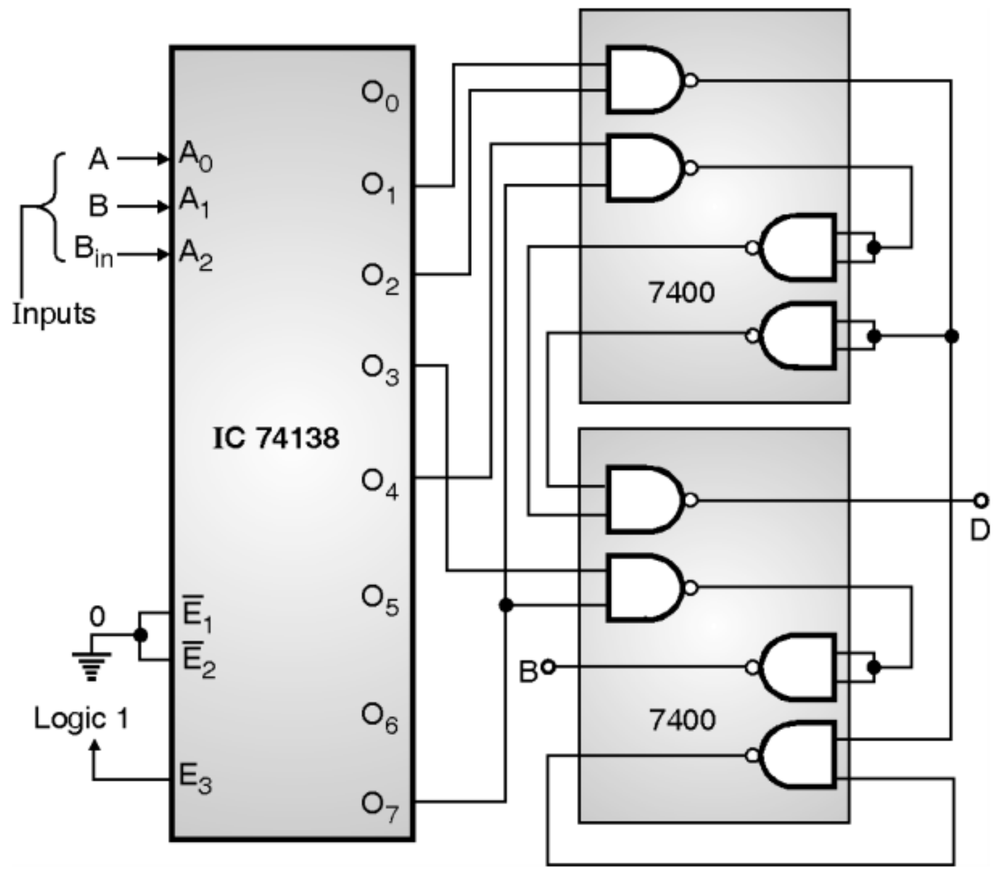
– The outputs of IC 74138 are active low. Hence they should be applied to NAND gates as shown in Fig. P. 2.18.6 to obtain the difference and borrow outputs.

Logic diagram :

– Implementation of full subtractor is as shown in Fig. P. 2.18.6.

– All the enable terminals are connected to their respective active levels to enable the decoder IC.

Tech Knowledge Publications



(C-1317) **Fig. P. 2.18.6**

– Implementation is to be done using 2 input NAND gates. Hence the output equations for "Borrow" and "Difference" are modified as,

$$\text{Difference } D = \overline{\overline{O_1}\,\overline{O_2}\,\overline{O_4}\,\overline{O_7}}$$

$$= \overline{\overline{\overline{(\overline{O_1}\,\overline{O_2})}}\;\overline{\overline{(\overline{O_4}\,\overline{O_7})}}}$$

$$= \overline{\overline{\overline{\overline{O_1}\,\overline{O_2}}}\;\overline{\overline{\overline{O_4}\,\overline{O_7}}}}$$

$$\text{Borrow } B = \overline{\overline{O_1}\,\overline{O_2}}\;\;\overline{\overline{O_3}\,\overline{O_7}}$$

$$= \overline{\overline{\overline{\overline{O_1}\,\overline{O_2}}}\;\overline{\overline{\overline{O_3}\,\overline{O_7}}}}$$

$$= \overline{\overline{\overline{\overline{O_1}\,\overline{O_2}}}\;\overline{\overline{\overline{O_3}\,\overline{O_7}}}}$$

Ex. 2.18.7 : Implement the following Boolean functions with a multiplexer and decoder.

$F(w, x, y, z) = \sum (2, 3, 5, 6, 11, 14, 15)$

Soln. :

Given : $F(w, x, y, z) = \Sigma (2, 3, 5, 6, 11, 14, 15)$

To do : Implement using : 1. Multiplexer 2. Decoder

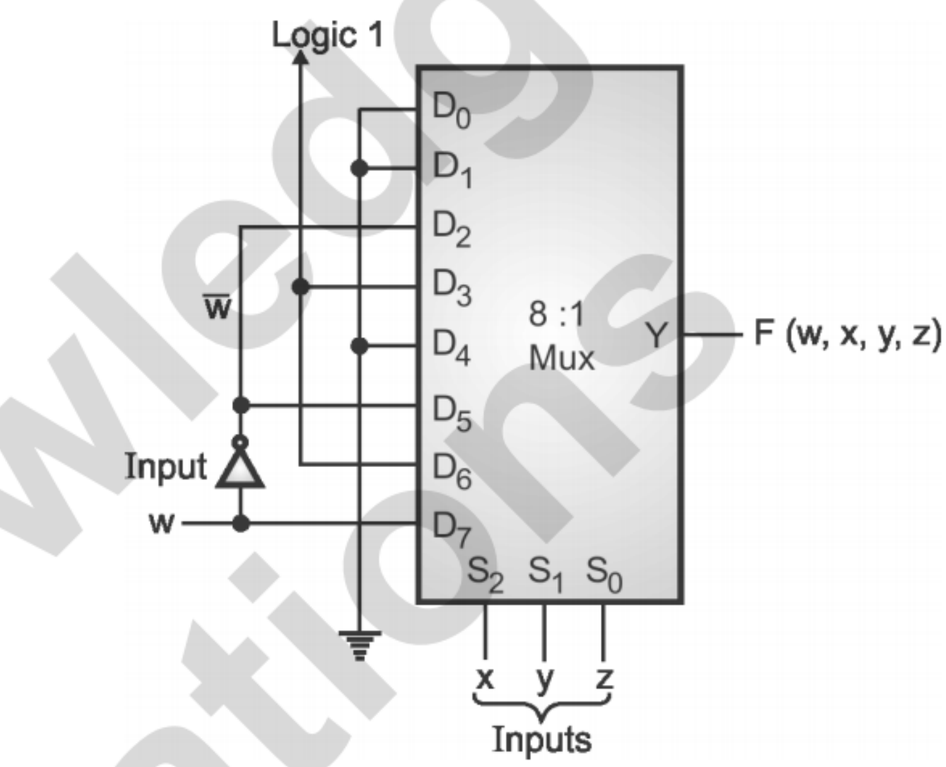
1. Implementation using MUX :

Step 1 : Write the design table :

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\overline{w}$	0	1	2	3	4	5	6	7
w	8	9	10	11	12	13	14	15
Input to MUX	0	0	$\overline{w}$	1	0	$\overline{w}$	1	w

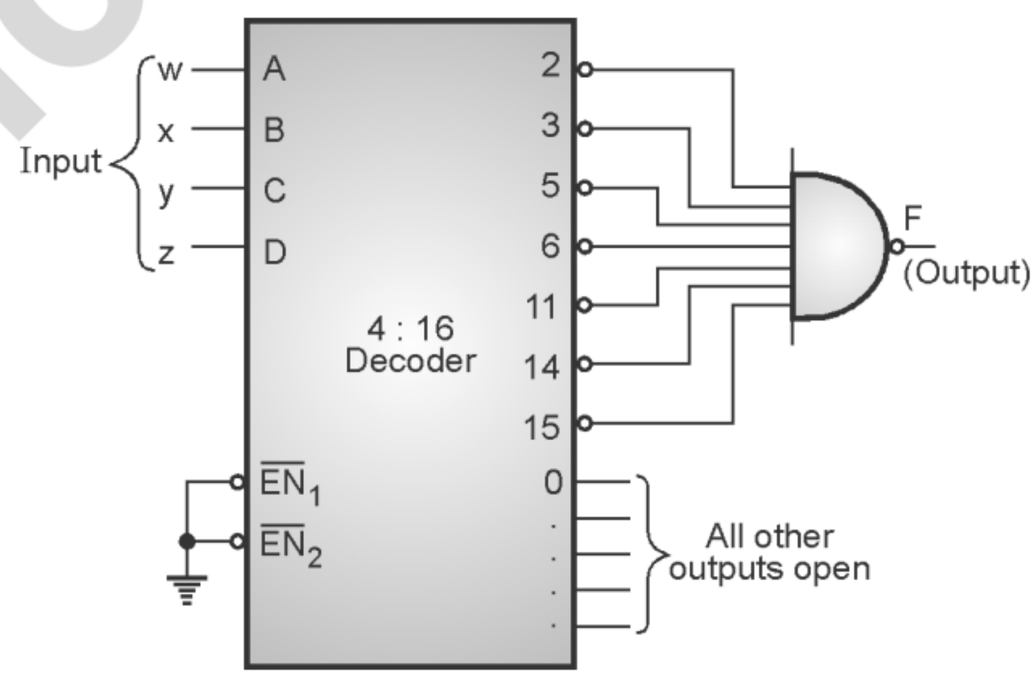
(C-7452)

Step 2 : Implementation :



(C-7453) **Fig. P. 2.18.7(a) : Implementation using a multiplexer**

2. Implementation using a decoder :



(C-7454) **Fig. P. 2.18.7(b) : Implementation using a decoder**

2.18.9 Advantage of Decoder Realization :

– The decoder may not be as efficient as the MUX in realizing a single Boolean function because a NAND gate is required to be used on the output side.

– But the decoder can be preferred over a MUX in realizing multiple Boolean functions simultaneously.

Tech Knowledge Publications

2.19 Parity Generators / Checkers :

SPPU : May 05, Dec. 11

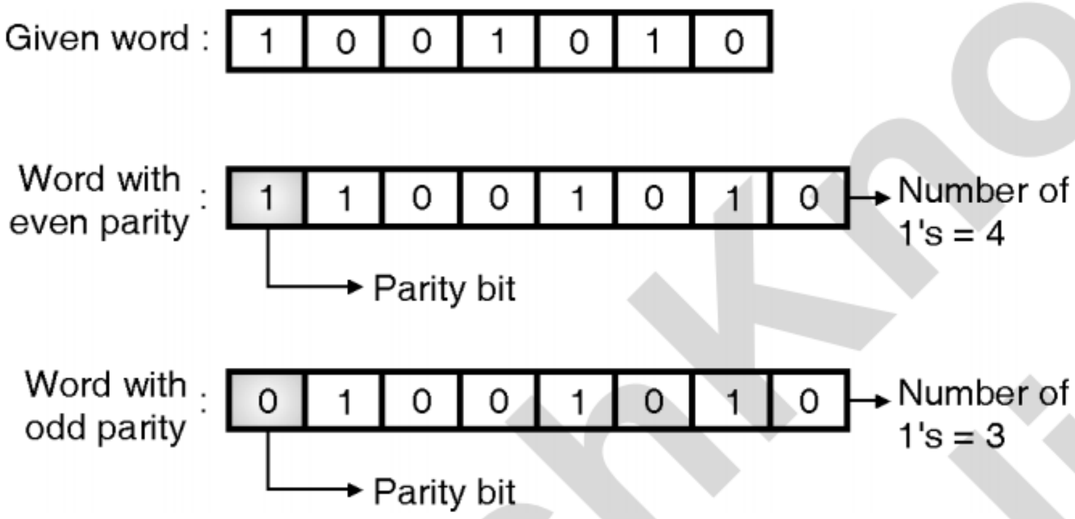
University Questions

Q. 1 Design an even parity generator for 4-bit input. Use 4 : 1 multiplexers and some logic gates. Explain your logic and optimization using truth table. **(May 05, 8 Marks)**

Q. 2 What do you mean by parity ? How does IC 74180 work ? **(Dec. 11, 4 Marks)**

Concept of Parity bit :

- A parity bit is an additional bit which is added to a binary word in order to make the number of 1's in the new word formed, even (even parity) or odd (odd parity).
- The even and odd parity has been illustrated in Fig. 2.19.1.



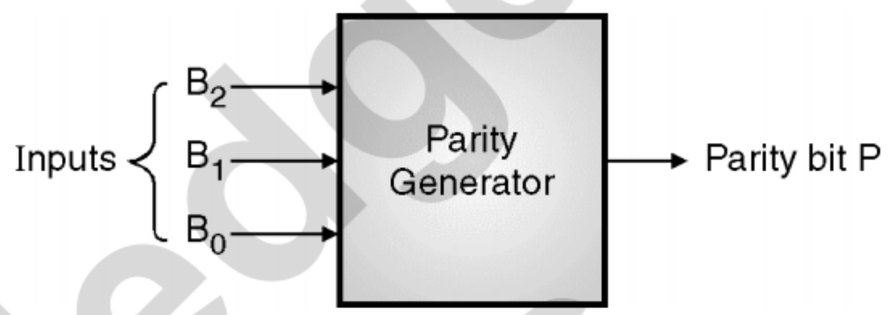
(C-503) **Fig. 2.19.1 : Illustration of even and odd parity**

- Parity bits are added in order to detect any error occurred in the process of transmission.
- The words with parity bits are transmitted by the transmitter. At the receiver, these words are received and checked for errors.
- The receiver checks parity of the received word and compares it with the parity of the transmitted word. If they are same then there is no error.
- But if the parity of received word is different from that of transmitted word, then the receiver understands that an error is present.
- In the even parity system, the added parity bit will make the total number of 1's an even number.
- In the odd parity system, the added parity bit will make the total number of 1's an odd number.

2.19.1 Parity Generator :

Definition :

- Parity generator is a logic circuit which generates the parity bits for even parity or odd parity.
- Block diagram of the parity generator is shown in Fig. 2.19.2.
- The digital word is applied at the input and a parity bit for either the even parity or odd parity.



(C-504) **Fig. 2.19.2 : Block diagram of parity generator**

Ex. 2.19.1 : Design a parity generator using the basic gates to produce digital words with odd parity. Assume the input to be three bit binary words.

Soln. :

Step 1 : Write truth table :

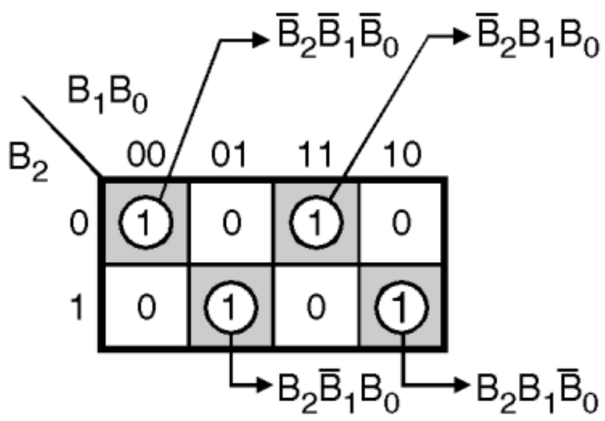
- Table P. 2.19.1 shows the required truth table.

(C-6265) **Table P. 2.19.1**

Inputs			Output
B_2	B_1	B_0	Odd parity bit P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

These parity bits make the total number of 1's odd

Step 2 : K-map and simplification :



(C-1265) **Fig. P. 2.19.1(a) : K-map and simplification**

$$P = \bar{B}_2\,\bar{B}_1\,\bar{B}_0 + \bar{B}_2\,B_1B_0 + B_2\bar{B}_1\,B_0 + B_2B_1\,\bar{B}_0$$

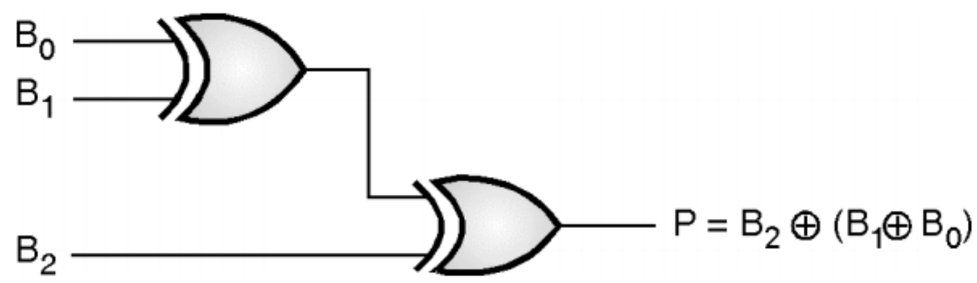
$$P = \bar{B}_2\,(\bar{B}_1\,\bar{B}_0 + B_1B_0) + B_2\,(\bar{B}_1\,B_0 + B_1\,\bar{B}_0)$$

$$= \overline{B}_2\ (\overline{B_1 \oplus B_0}) + B_2 (B_1 \oplus B_0)$$

$$\therefore P = B_2 \oplus (B_1 \oplus B_0)$$

Step 3 : Logic diagram :

– The logic diagram for an odd parity generator is shown in Fig. P. 2.19.1(b).



(C-1266) Fig. P. 2.19.1(b) : Logic diagram of an odd parity generator

Ex. 2.19.2 : Design 3-bit parity generator circuit using even parity bit.

Soln. :

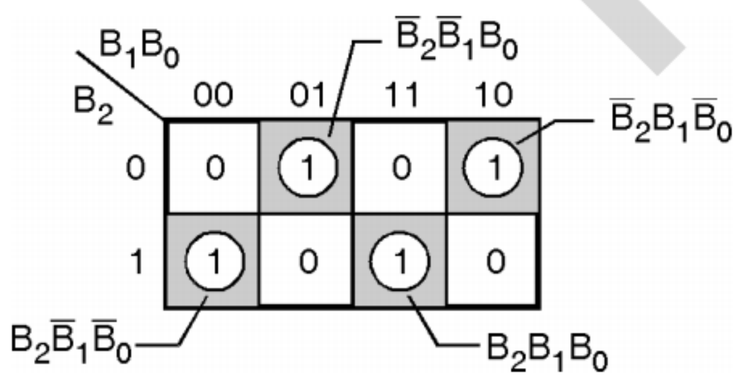
Step 1 : Write the truth table :

– Table P. 2.19.2 shows the required truth table. The parity bits make the total number of 1's even.

(C-8140) Table P. 2.19.2

Inputs			Output
B_2	B_1	B_0	Even parity bit P
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

Step 2 : K-maps and simplification :



(C-1334(a))

– From the K-map we get,

$$P = \overline{B}_2 \overline{B}_1 B_0 + \overline{B}_2 B_1 \overline{B}_0 + B_2 \overline{B}_1 \overline{B}_0 + B_2 B_1 B_0$$

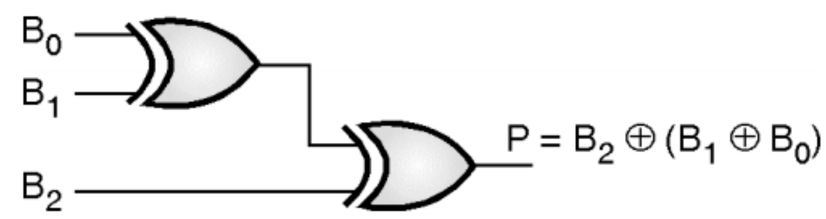
$$= \overline{B}_2 (\overline{B}_1 B_0 + B_1 \overline{B}_0) + B_2 (\overline{B}_1 \overline{B}_0 + B_1 B_0)$$

$$= \overline{B}_2 (B_1 \oplus B_0) + B_2 (\overline{B_1 \oplus B_0})$$

$$= B_2 \oplus (B_1 \oplus B_0)$$

Step 3 : Logic diagram :

– The logic diagram for 3-bit parity generator circuit using even parity bit :



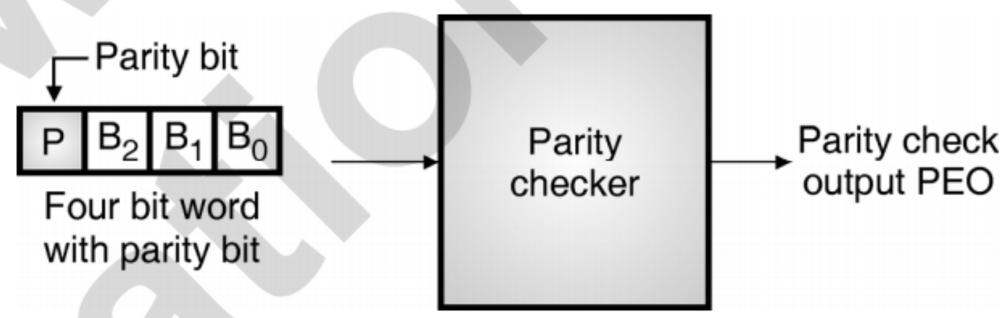
(C-1334) Fig. P. 2.19.2

2.19.2 Parity Checker :

Definition :

– Parity checker is a logic circuit which checks the parity of the logical word applied at its input. The parity checker circuit is at the receiver.

– The block diagram of parity checker is shown in Fig. 2.19.3.



(C-507) Fig. 2.19.3 : Block diagram of parity checker

– The received word with a parity bit is applied to the parity checker.

– The parity checker will check the parity of the received word and produce its output.

– Since the parity of transmitted words is "odd", the parity checker will treat the received words with odd parity as the correct words and those with even parity as the incorrect words.

– The output of parity checker circuit is denoted by parity error output PEO.

PEO = 1	..... If error is present i.e. if the received four bit word has an even parity.
PEO = 0	..... If there is no error i.e. if the received four bit word has an odd parity.

Truth table of parity checker :

– According to the conditions stated above, the truth table is prepared as follows :

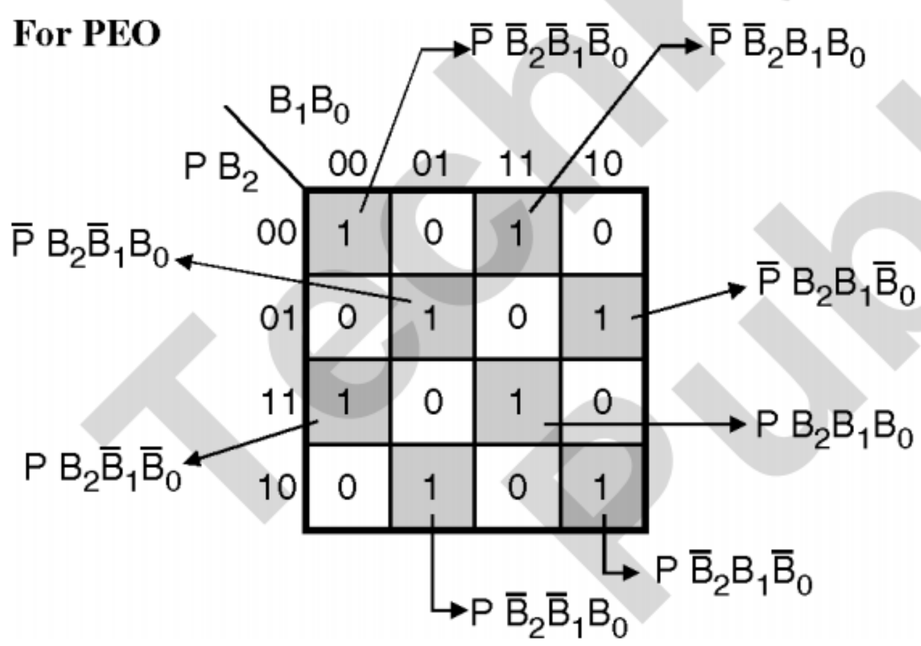
TechKnowledge Publications

(C-508) Table 2.19.1 : Truth table of a parity checker

Inputs				Outputs
P	B_2	B_1	B_0	PEO
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

1 = error (even parity)
0 = No error (odd parity)

K-map and simplifications :



(C-508) Fig. 2.19.4 : K-map for a parity checker output

Simplification :

$$\text{PEO} = \bar{P}\,\bar{B}_2\,\bar{B}_1\,\bar{B}_0 + \bar{P}\,\bar{B}_2\,B_1\,B_0 + \bar{P}\,B_2\,\bar{B}_1\,B_0$$

$$+ \bar{P}\,B_2\,B_1\bar{B}_0 + P\,B_2\,\bar{B}_1\,\bar{B}_0 + P\,B_2\,B_1\,B_0$$

$$+ P\,\bar{B}_2\,\bar{B}_1\,B_0 + P\,\bar{B}_2\,B_1\bar{B}_0$$

$$= \bar{B}_1\,\bar{B}_0\,(\bar{P}\,\bar{B}_2 + P\,B_2) + B_1\,B_0\,(\bar{P}\,\bar{B}_2 + PB_2)$$

$$+ \bar{B}_1\,B_0\,(\bar{P}\,B_2 + P\,\bar{B}_2) + B_1\,\bar{B}_0\,(\bar{P}\,B_2 + P\,\bar{B}_2)$$

$$\therefore \text{PEO} = (\bar{P}\,\bar{B}_2 + P\,B_2)\,(\bar{B}_1\,\bar{B}_0 + B_1\,B_0)$$

$$+ (\bar{P}\,B_2 + P\,\bar{B}_2)\,(\bar{B}_1\,B_0 + B_1\bar{B}_0)$$

$$= (P \odot B_2)\,(B_1 \odot B_0) + (P \oplus B_2)\,(B_1 \oplus B_0)$$

$$= \overline{(P \oplus B_2)}\,\overline{(B_1 \oplus B_0)} + (P \oplus B_2)\,(B_1 \oplus B_0)$$

$$\text{Let } X = P \oplus B_2$$

$$\text{And } Y = (B_1 \oplus B_0)$$

$$\therefore \text{PEO} = \bar{X}\,\bar{Y} + X\,Y$$

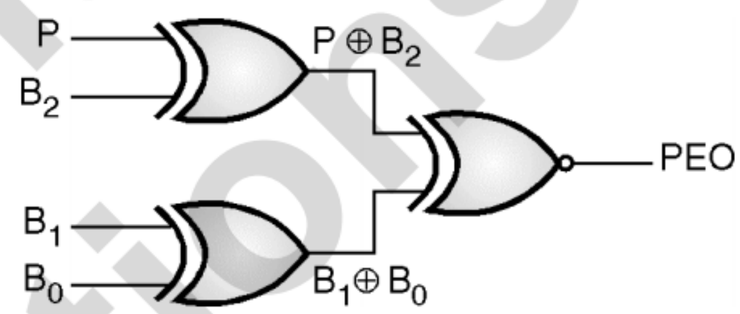
$$= X \odot Y = \overline{X \oplus Y}$$

– Substituting the values we get,

$$\text{PEO} = \overline{(P \oplus B_2) \oplus (B_1 \oplus B_0)}$$

Logic diagram :

– Logic diagram is as shown in Fig. 2.19.5.



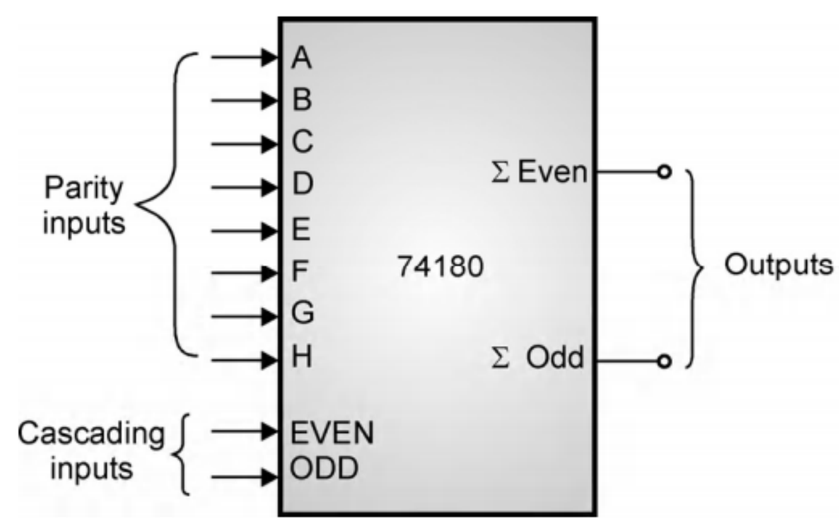
(C-509) Fig. 2.19.5

2.19.3 IC 74180 : A Parity Generator / Checker : SPPU : Dec. 11

University Questions

Q. 1 How does IC 74180 work ? **(Dec. 11, 4 Marks)**

– Fig. 2.19.6(a) shows the block diagram of 74180, which is a parity generator/checker IC.

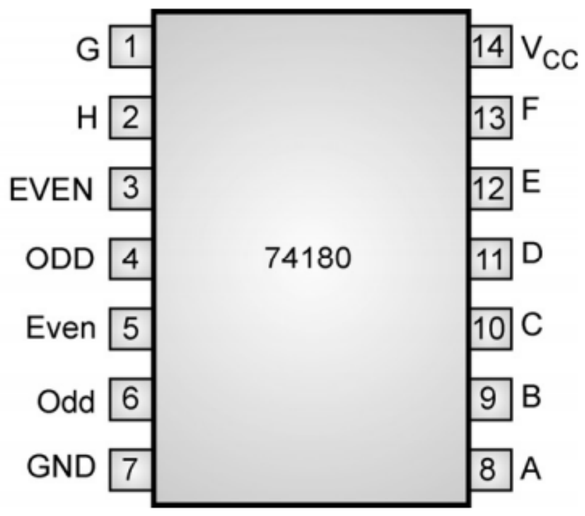


(C-1094) Fig. 2.19.6(a) : Block diagram of 74180

– There are 8 – parity inputs A to H and two cascading inputs. There are two outputs Σ even and Σ odd.

– And the two cascading inputs are EVEN and ODD.

– The pin configuration of IC 74180 is shown in Fig. 2.19.6(b).



(C-1094) Fig. 2.19.6(b) : Pin configuration of 74180

– Table 2.19.2 shows the functional table for 74180.

(C-8141) Table 2.19.2 : Functional table for 74180 (Use it for parity checker)

Row no.	Parity of inputs	Cascading inputs		Outputs	
	(A to H)	EVEN	ODD	Σ Even	Σ odd
1	Even	1	0	1	0
2	Odd	1	0	0	1
3	Even	0	1	0	1
4	Odd	0	1	1	0
5	X	1	1	0	0
6	X	0	0	1	1

– It is possible to use 74180 as a parity generator as well as a parity checker.

– The function table of 74180 explains its use as a parity checker. This will be clear by solving the following example.

Ex. 2.19.3 : Draw the circuit diagram for a 9 bit odd parity checker using IC 74180. **Dec. 11, 4 Marks**

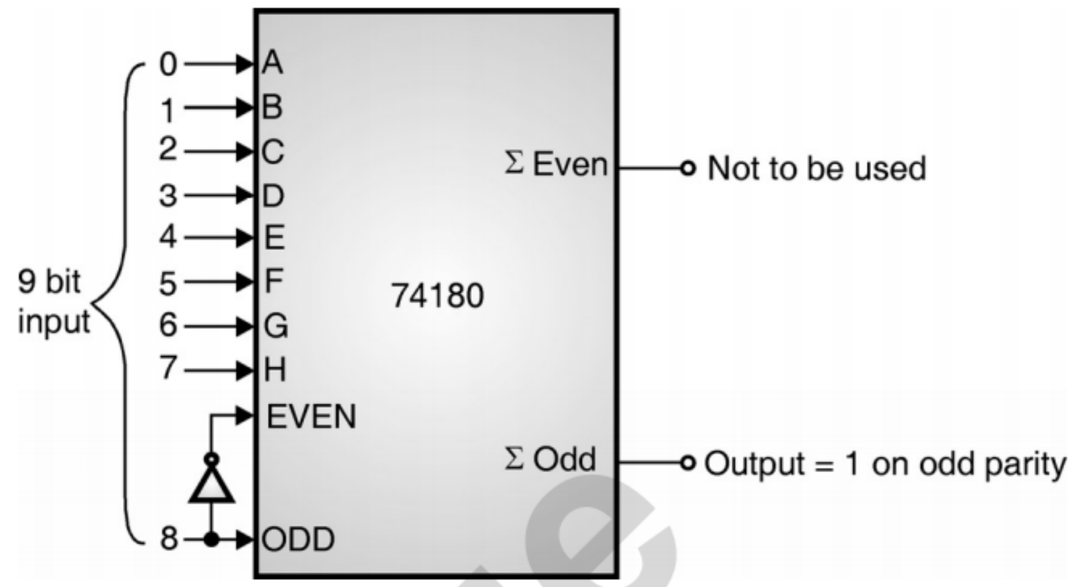
Soln. :

– Since the odd parity checker is to be drawn, we have to use the Σ odd output of 74180.

– Since 9 bit input is to be applied, we should apply 8 of the 9 bits to the input A – H, whereas the 9th bit is applied to the ODD input as shown in Fig. P. 2.19.3.

– Output Σ odd will be high if the parity of the 9 bit input is odd.

– The function table of 74180 (refer Table 2.19.2) explains the operation of this circuit.



(C-1095) Fig. P. 2.19.3 : A 9 bit parity checker

2.19.4 74180 as Parity Generator : SPPU : Dec. 11

University Questions

Q. 1 What do you mean by parity ? How does IC 74180 work ? **(Dec. 11, 8 Marks)**

– Table 2.19.3 shows the function table of 74180 as a 9 bit parity generator.

(C-8142) Table 2.19.3 : Function table of 74180 as 9 bit parity generator

Parity of inputs	Cascading inputs		Parity of	Parity of
(A to H)	EVEN	ODD	A to H and Σ EVEN	A to H and Σ ODD
Odd	1	0	Odd	Even
Even	1	0	Odd	Even
Odd	0	1	Even	Odd
Even	0	1	Even	Odd

– The cascading inputs should never be equal. The unused parity inputs must be connected to ground.

– Example 2.19.4 illustrates the use of 74180 as parity generator.

Ex. 2.19.4 : Draw the logic diagram to illustrate the use of IC 74180 as the 10 bit even parity generator.

Soln. :

– The logic diagram for IC 74180 used as a 10 bit even parity generator is shown in Fig. P. 2.19.4.

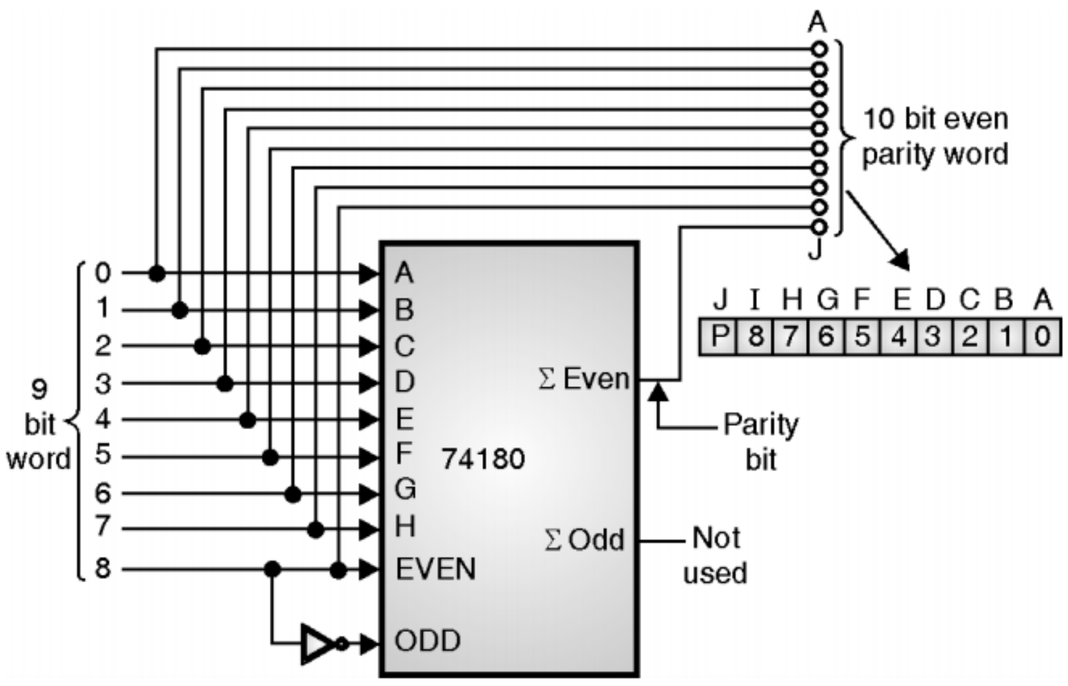
– Note that the first eight input bits 0 to 7 have been applied to the eight inputs A to H.

– The ninth input is applied to the EVEN input because we want even parity.

– The parity bit for this 9 bit input is produced at Σ Even output.

Tech Knowledge Publications

– The nine input bits and Σ even output together form the 10 bit even parity word.

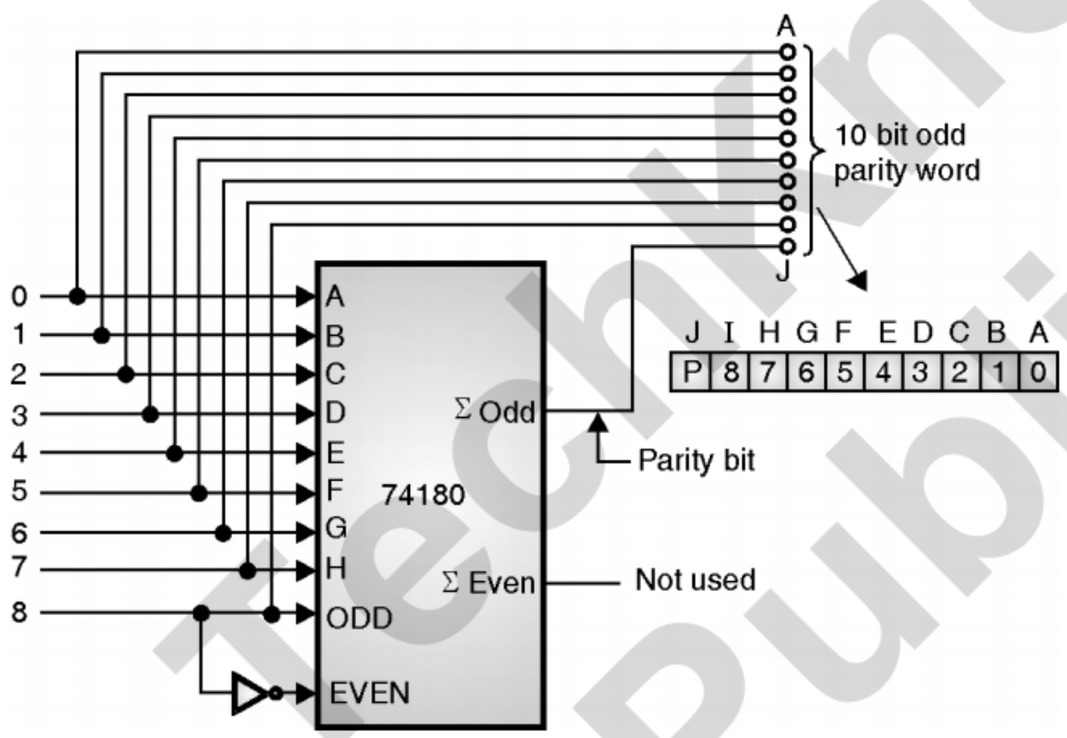


(C-1096) **Fig. P. 2.19.4 : A 10 bit even parity generator**

Ex. 2.19.5 : Draw the logic diagram to illustrate the use of IC 74180 as a 10 bit odd parity generator.

Soln. :

– Refer Fig. P. 2.19.5 for answer.



(C-1097) **Fig. P. 2.19.5 : A 10 bit odd parity generator using 74180**

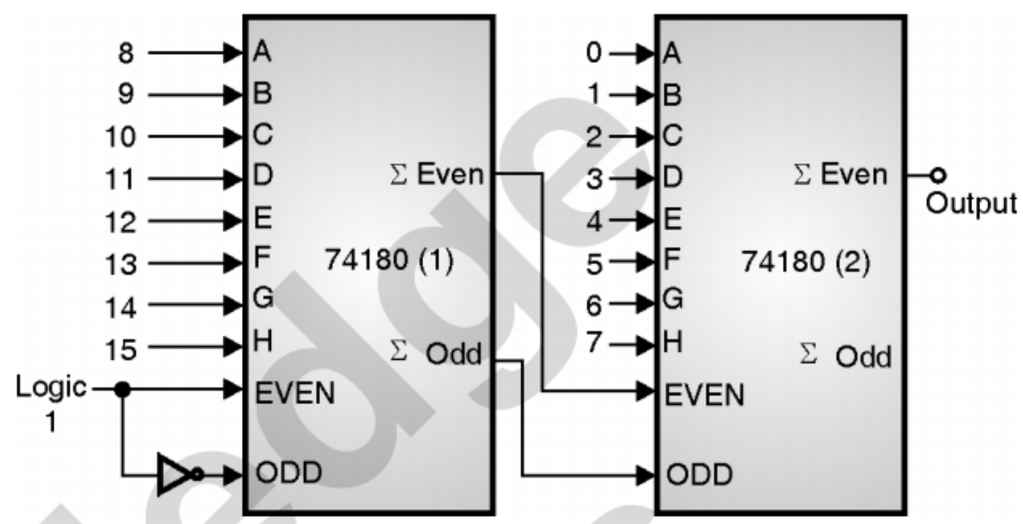
2.19.5 Cascading of 74180 :

– A 16-bit even parity checker using IC 74180 is shown in Fig. 2.19.7.

– Note that the EVEN input of the first 74180 is connected to a logic 1, and due to an inverter connected between the EVEN and ODD inputs, the ODD input is at logic 0 level.

– The Σ even and Σ odd outputs of first 74180 are connected to the EVEN and ODD inputs of the second 74180.

– The Σ even output of second IC 74180 produces a logic 1 if the 16 bit input word has an even parity.



(C-1098) **Fig. 2.19.7 : 16 bit even parity checker using 74180**

2.20 Examples for Practice :

Ex. 2.20.1 : Draw neat circuit diagrams of the following : 32 : 1 multiplexer tree using 8 : 1 multiplexer ICs. **Dec. 04, 3 Marks**

Soln. : Solve it yourself.

Ex. 2.20.2 : Design comparator having two 2-bit inputs (A and B) by using two 3 : 8 decoders and logic gates to give outputs : A > B, A = B and A < B. **Dec. 05, 10 Marks**

Soln. : Solve it yourself.

Ex. 2.20.3 : Implement the following function using 4 : 1 MUX and logic gates : F (A, B, C, D) = Σ(0, 2, 5, 8 10, 15). **Dec. 06, 6 Marks**

Soln. : Solve it yourself.

Ex. 2.20.4 : Implement the following function using 3 : 8 decoder and some logic gates : F(A, B, C, D) = Σm (1, 3, 4, 6, 8, 10, 12, 15), Where A = MSB. **May 08, 8 Marks**

Soln. : Solve it yourself.

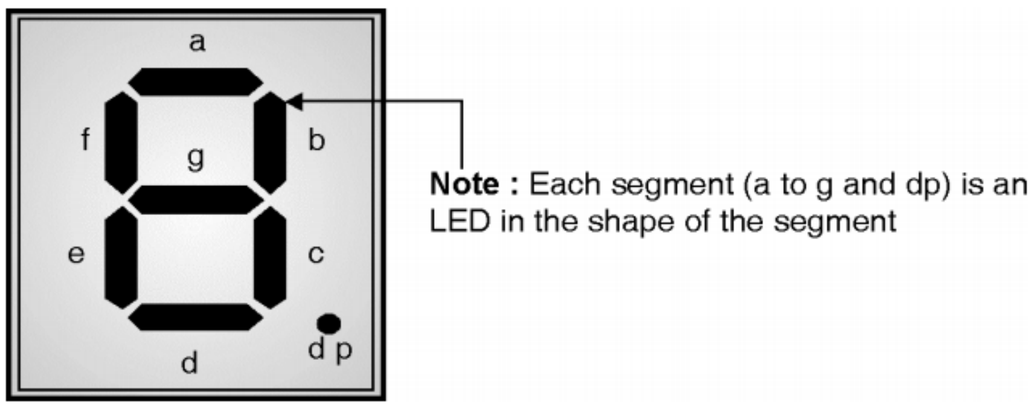
TechKnowledge Publications

2.21 Case Study : Combinational Logic Design of BCD to 7 Segment Display Controller :

2.21.1 Seven Segment LED Display :

– All of us know what a seven segment display is.

– Fig. 2.21.1 shows the construction of a seven segment display.



(B-2342) Fig. 2.21.1 : Standard form of seven segment display

– We can display any number from 0 to 9 by turning on various combinations of the segments as shown in Table 2.21.1.

(B-2343(a)) Table 2.21.1

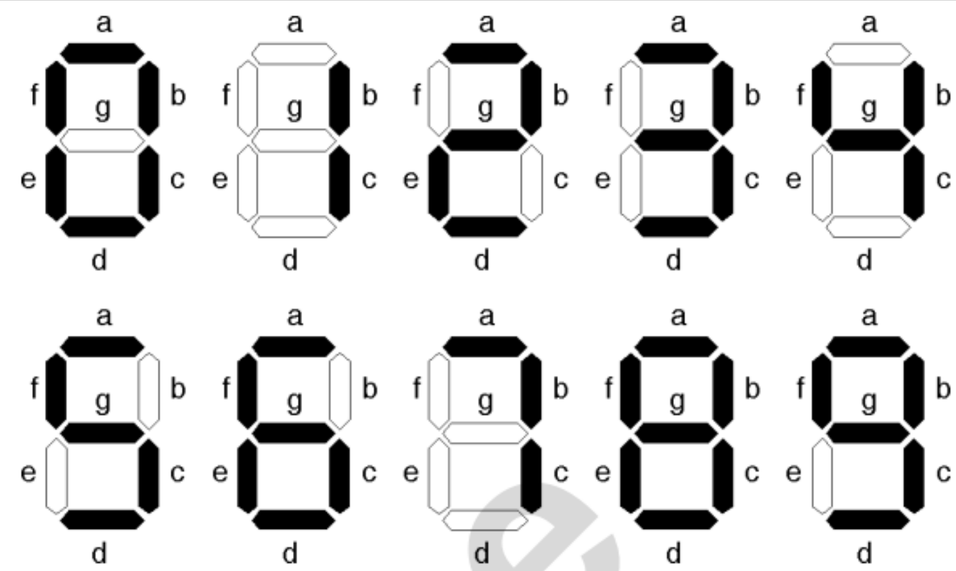
Segments							Number	Display
a	b	c	d	e	f	g		
ON	ON	ON	ON	ON	ON	—	(0)	0
—	ON	ON	—	—	—	—	(1)	1
ON	ON	—	ON	ON	—	ON	(2)	2
ON	ON	ON	ON	—	—	ON	(3)	3
—	ON	ON	—	—	ON	ON	(4)	4
ON	—	ON	ON	—	ON	ON	(5)	5
ON	—	ON	ON	ON	ON	ON	(6)	6
ON	ON	ON	—	—	—	—	(7)	7
ON	ON	ON	ON	ON	ON	ON	(8)	8
ON	ON	ON	ON	—	ON	ON	(9)	9

– Actually each segment (a to f) is nothing but an LED in a segment form.

– Their connection leads are brought out, and by applying a forward voltage to the segments to be turned on we can display any number between 0 and 9.

– For example if we want to display the number 7 then the segments a, b and c should be turned on and all other segments should be off.

– Similarly the other numbers can be displayed as shown in Fig. 2.21.2.



(B-2343) Fig. 2.21.2 : Various digits displayed with seven segment display

2.21.2 Types of Seven Segment Displays :

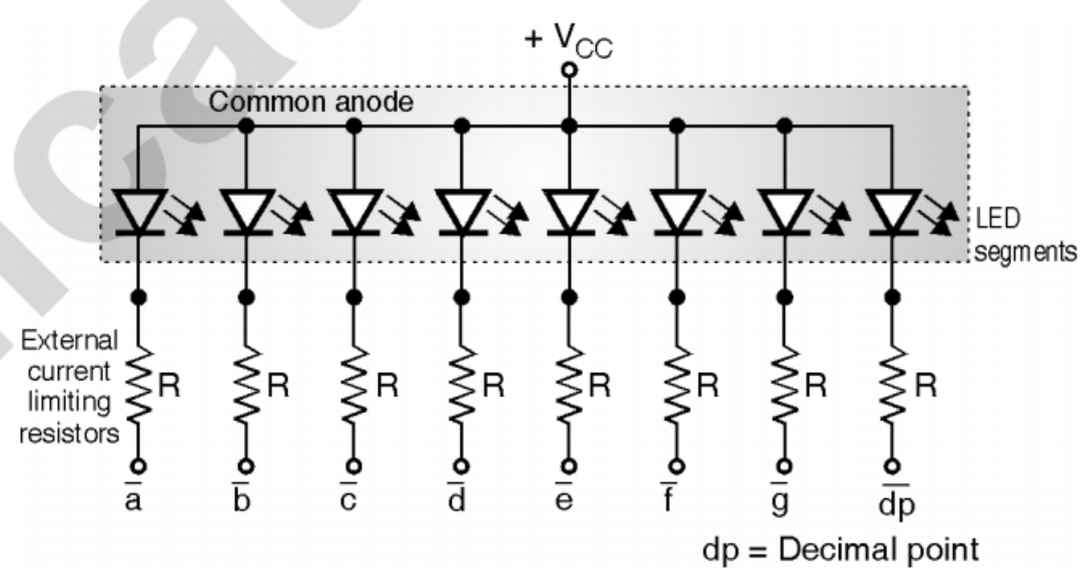
There are two types of seven segment LED displays :

1. Common anode 2. Common cathode

2.21.3 Common Anode Display :

– A common anode seven segment display is as shown in Fig. 2.21.3.

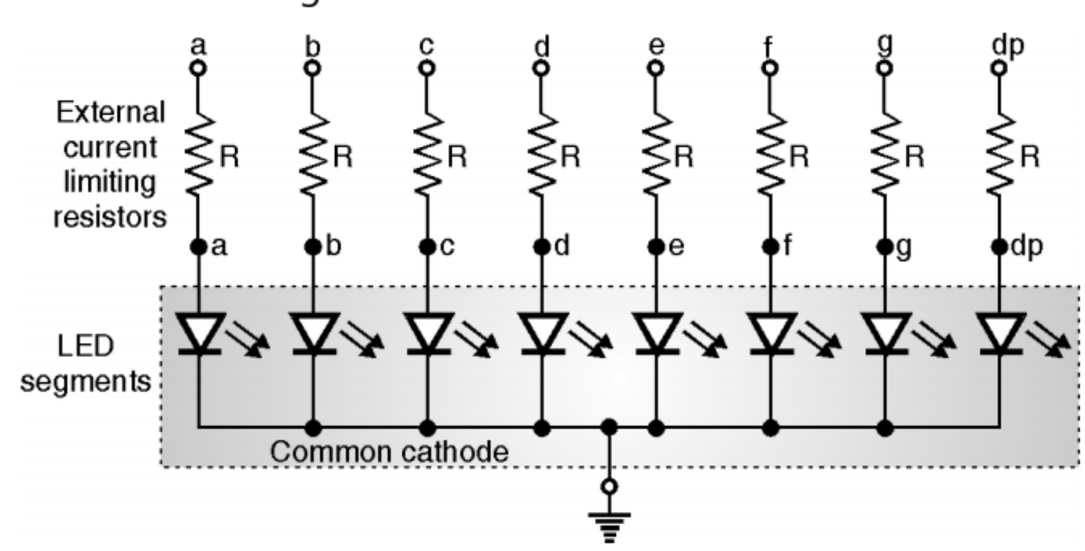
– Here as the name indicates the anode terminals of all LED segments are connected together.



(C-495) Fig. 2.21.3 : Common anode LED display

2.21.4 Common Cathode Display :

– A common cathode seven segment LED display is as shown in Fig. 2.21.4.



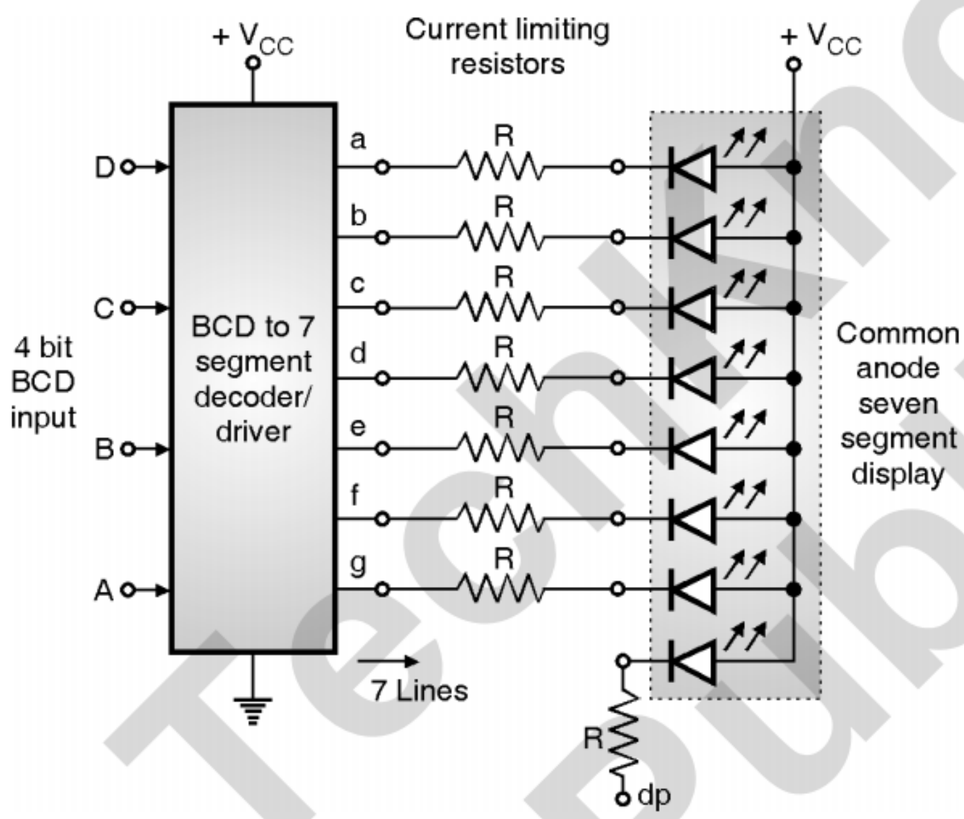
(C-496) Fig. 2.21.4 : Common cathode LED display

TechKnowledge Publications

- As the name indicates, the cathode terminal is made common and all the anode terminals are brought out separately.
- A current limiting resistor is externally connected in series with each segment.
- The common cathode point is connected to ground and the anodes of the segments to be illuminated are connected to positive supply voltage V_{CC}.

2.21.5 Use of a Decoder for Driving the Seven Segment Display :

- The use of a BCD to seven segment decoder / driver for driving the seven segment display is shown in Fig. 2.21.5.



(C-497) Fig. 2.21.5 : Driving the segment common anode display

- Many times the seven segment LED displays are connected at the output of digital ICs such as counters.
- Now as the counter output is in the BCD (binary coded decimal) form, which has only four lines, it cannot drive the seven segment display directly.
- Therefore we have to use a BCD to seven segment decoder / driver IC (integrated circuit) between the counter output and seven segment display.
- The common anode type display is being used here.
- The decoder accepts a four bit BCD count from a counter, converts it to a seven bit code suitable for the seven segment display ($\overline{a}$ to $\overline{g}$) and drives the display.
- To turn on a segment the corresponding decoder output goes low, and sinks current (for common anode display).
- A current limiting resistor is connected in series with each segment.

2.21.6 BCD to Seven Segment Display Driver (Common Anode Display) :

- Let us assume that the type of display is common anode.
- Hence the output of the converter should be "0" if a display segment is to be turned on.

Truth table :

- The truth table is as shown in Table 2.21.2.

(C-499(a)) Table 2.21.2 : Truth table for BCD to 7 segment decoder

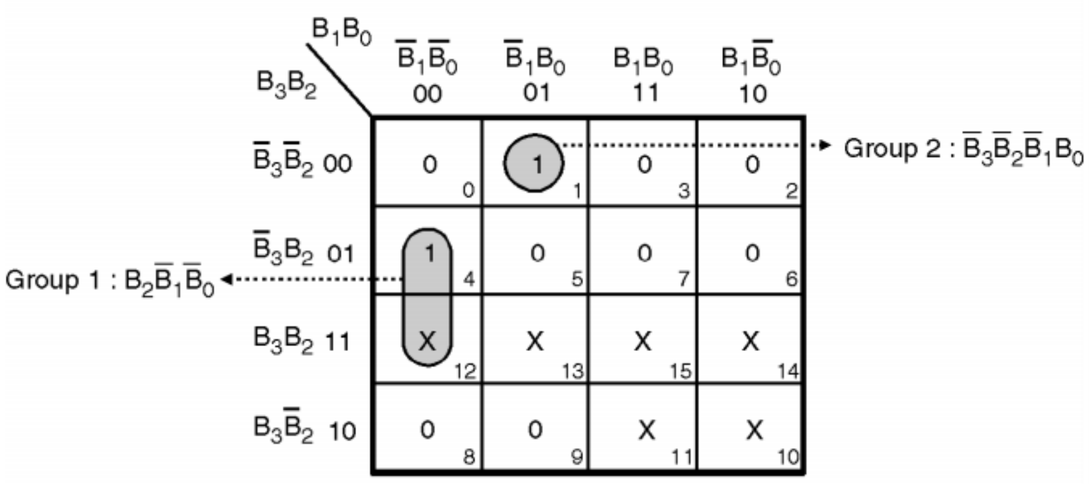
Decimal	Inputs				Outputs						
	B_3	B_2	B_1	B_0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

K-maps and simplifications :

- The K maps for the seven outputs are as shown in Fig. 2.21.6.

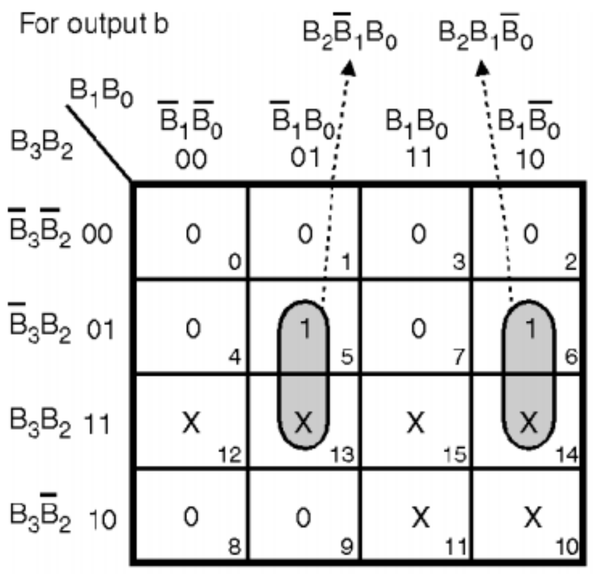
TechKnowledge Publications

For output a



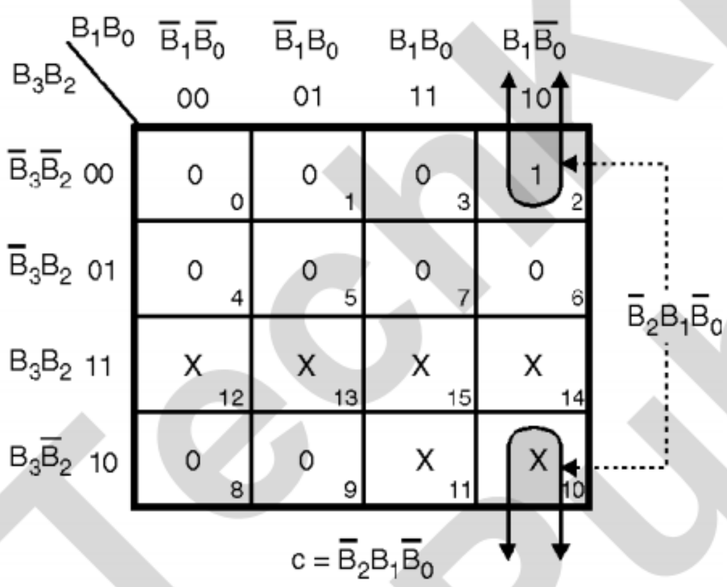
$a = B_2\overline{B}_1\overline{B}_0 + \overline{B}_3\overline{B}_2\overline{B}_1B_0$

For output b



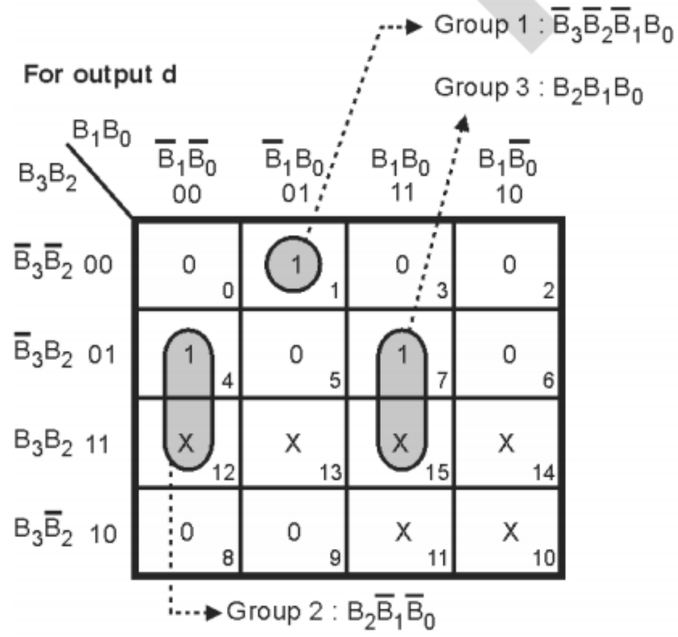
$b = B_2\overline{B}_1B_0 + B_2B_1\overline{B}_0$

For output c



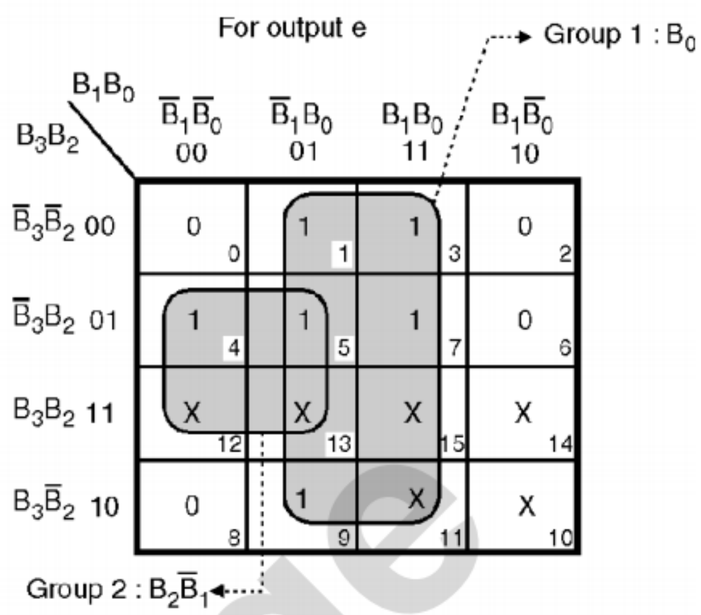
$c = \overline{B}_2B_1\overline{B}_0$

For output d



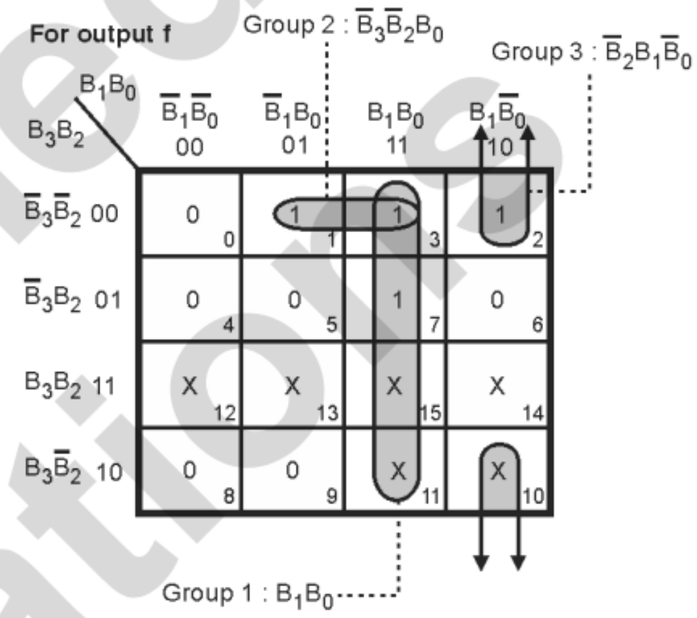
$d = \overline{B}_3\overline{B}_2\overline{B}_1B_0 + B_2\overline{B}_1\overline{B}_0 + B_2B_1B_0$

For output e



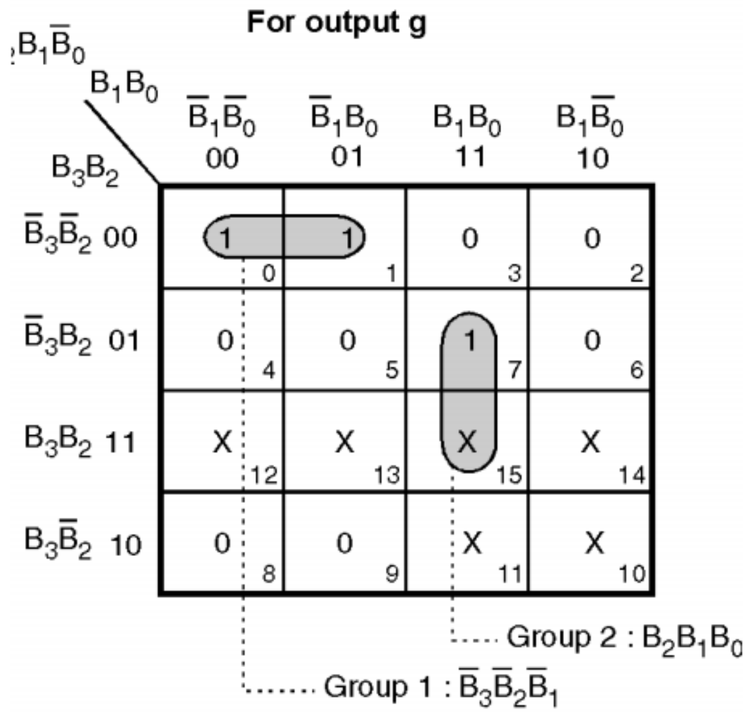
$e = B_0 + B_2\overline{B}_1$

For output f



$f = B_1B_0 + \overline{B}_3\overline{B}_2B_0 + \overline{B}_2B_1\overline{B}_0$

For output g



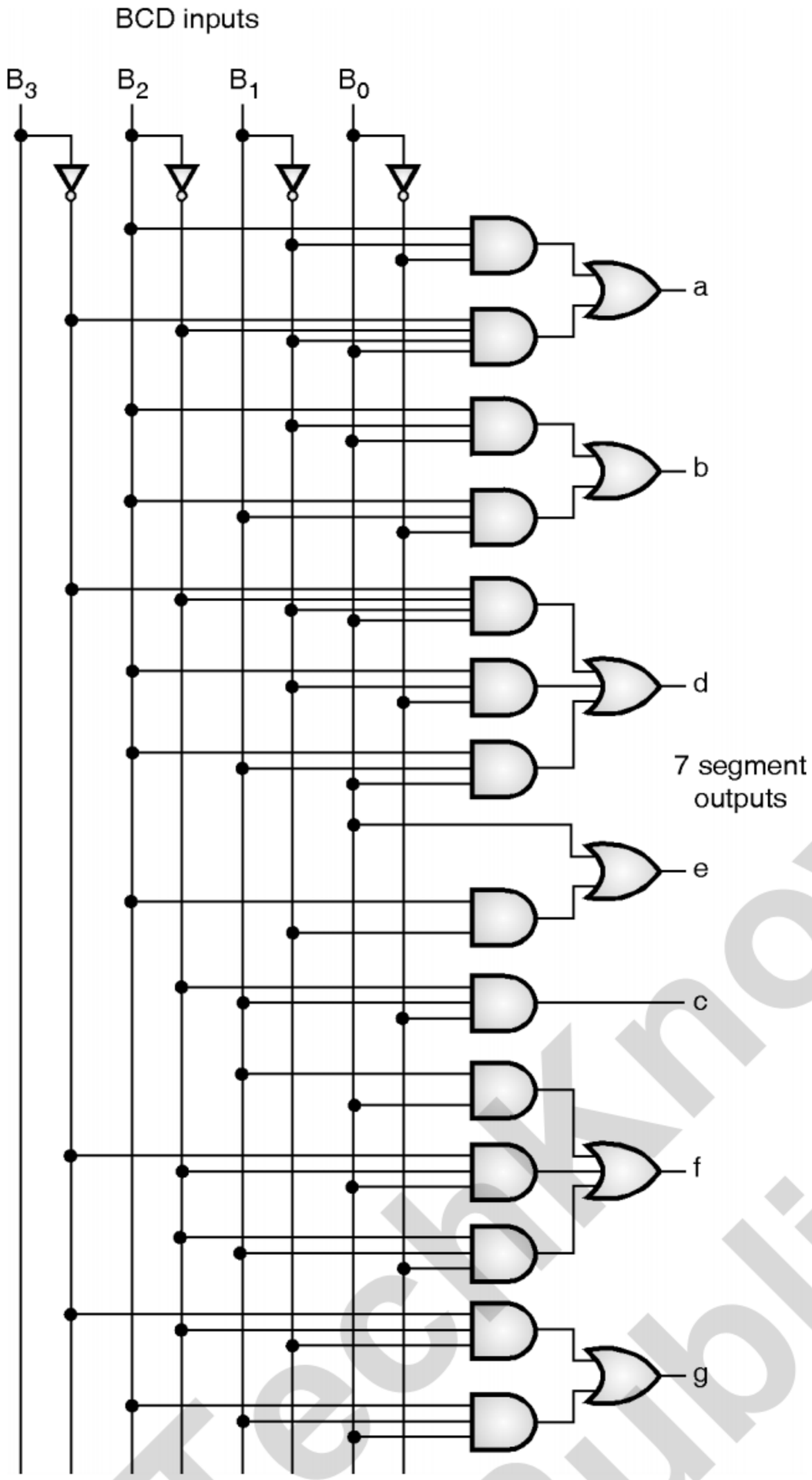
$g = \overline{B}_3\overline{B}_2\overline{B}_1 + B_2B_1B_0$

(C-499) Fig. 2.21.6 : K maps for the outputs of BCD to seven segment decoder

Realization :

– The logic diagram for BCD to seven segment decoder is shown in Fig. 2.21.7.

TechKnowledge Publications



(C-501) **Fig. 2.21.7 : BCD to seven segment decoder**

Review Questions

Q. 1 Explain the working of a half adder ? Draw its logic diagram.

Q. 2 Draw the logic diagram of full adder and its truth table.

Q. 3 Give the circuit diagram of a full adder using NAND gate and explain. Give its truth table.

Q. 4 What is similar between a half adder and a half subtractor ?

Q. 5 Build a full adder from half adder circuits.

Q. 6 With suitable block diagram explain the operation of n-bit serial adder.

Q. 7 Write a note on a 4 bit parallel binary adder.

Q. 8 Draw pin diagram of IC 7483 and explain its operation as 4-bit binary adder.

Q. 9 What is half subtractor ? Draw the logic diagram and truth table of half subtractor.

Q. 10 What is meant by a full subtractor ? Draw a full subtractor circuit.

Q. 11 Draw the circuit diagram of single digit BCD adder using IC 7483.

Q. 12 Implement a full subtractor circuit using only NAND gates.

Q. 13 In detail explain the working of 4-bit binary adder/subtractor using IC 7483.

Q. 14 For one digit BCD adder :

1. Draw circuit diagram.
2. Explain its working.
3. Can $(11)_{10}$ and $(9)_{10}$ be added by this circuit ? Justify.

Q. 15 Draw half subtractor circuit. Use NAND gates only. Explain its working.

Q. 16 Draw 4 bit adder/subtractor circuit using IC 7483 and IC 7486. Explain its working.

Q. 17 Draw the circuit of 4 bit binary parallel adder. Explain its working.

Q. 18 What is meant by a multiplexer ? Explain with block diagram the principle of multiplexing.

Q. 19 Explain with diagram and truth table the operation of 4 : 1 Mux.

Q. 20 Explain briefly with pin-diagram the following ICs IC-74157, IC-74151, IC-74150.

Q. 21 What is a 'Multiplexer tree' ?

Tech Knowledge Publications

Q. 22 Explain with diagram 64 to 1 multiplexer using four 16 to 1 multiplexer.

Q. 23 Explain with diagram 32 : 1 multiplexer using tree of 8 : 1 multiplexer.

Q. 24 Explain with diagram 32 : 1 multiplexer using two 16 : 1 multiplexer.

Q. 25 Give applications of multiplexers.

Q. 26 Explain with diagram the working of 1 to 8 demultiplexer.

Q. 27 Explain with diagram the working of 1 to 16 demultiplexer.

Q. 28 Explain with pin-diagram the following ICs : IC74151, IC 74138.

Q. 29 Give the circuit diagram for 1 : 64 demux using tree of 1 : 16 demux.

Q. 30 Explain how demultiplexer can be used as a decoder.

Q. 31 What is the necessity of multiplexer ?

Q. 32 Draw the circuit diagram of 1 to 4 line demultiplexer and explain the working.

Q. 33 Draw circuit diagrams for the following :

1. 2 : 1 multiplexer.
2. 1 : 2 demultiplexer.

Q. 34 With a neat block diagram explain the function of an encoder.

Q. 35 What is meant by a priority encoder ? Give example.

Q. 36 What do you mean by a 'Decoder' ? Give its applications.

Q. 37 What is a parity generator ?

Q. 38 What is a parity checker ?

Q. 39 Explain the use of IC 74180 as a parity generator/checker.

❑❑❑

TechKnowledge Publications

Unit III

Chapter 3

Flip Flops

TechKnowledge Publications

Syllabus

SR, JK, D, T, Preset and clear, Master slave JK flip flops, Truth tables and excitation tables, Conversion from one type to another type of flip flop.

Chapter Contents

3.1	Introduction	3.10	Edge Triggered J-K Flip Flop
3.2	S-R Latch using NOR Gates	3.11	Toggle Flip Flop (T Flip Flop)
3.3	Triggering Methods	3.12	Master Slave (MS) JK Flip Flop
3.4	Gated Latches	3.13	Preset and Clear Inputs
3.5	The Gated S-R Latch (Level Triggered S-R Flip Flop)	3.14	Various Representations of Flip Flops
3.6	The Gated D Latch (Clocked D Flip Flop)	3.15	Excitation Table of Flip-Flop
3.7	Gated JK Latch (Level Triggered JK Flip Flop)	3.16	Conversion of Flip Flops
3.8	Edge Triggered Flip Flops	3.17	Applications of Flip Flops
3.9	Edge Triggered D Flip Flop	3.18	Analysis of Clocked Sequential Circuits

3.1 Introduction :

Combinational circuits :

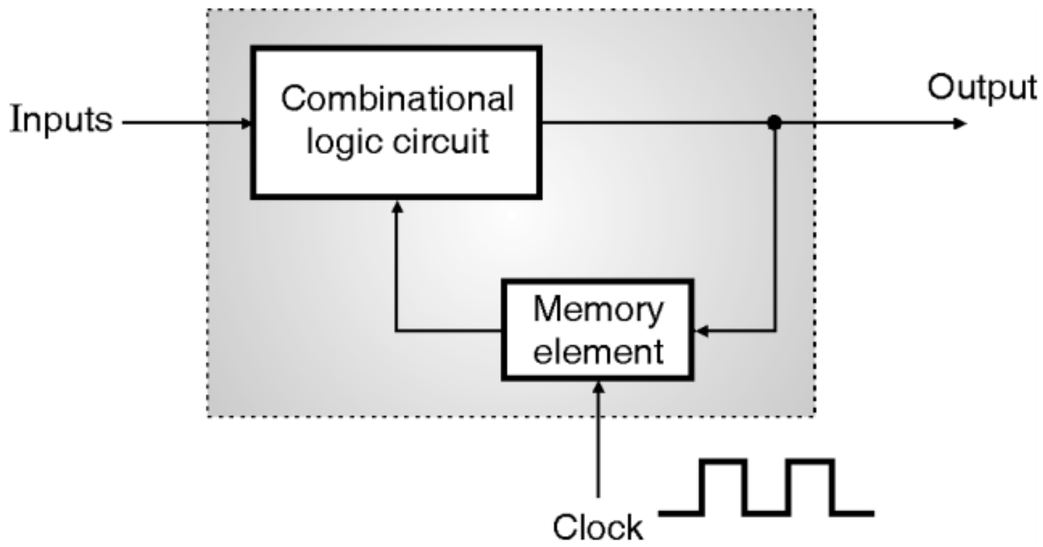
Definition :

- A combinational circuit is a logic circuit the output of which depends only on the combination of the inputs. The output does not depend on the past value of inputs or outputs.
- Hence combinational circuits do not require any memory (to store the past values of inputs or outputs).
- Till now we have discussed only the combinational circuits.
- The output of combinational circuit at any instant of time, depends only on the levels present at input terminals. It does not depend on the past status of inputs.
- The combinational circuits do not use any memory. Therefore the previous states of input does not have any effect on the present state of the circuit.
- Also the sequence in which the inputs are being applied has no effect on the output of a combinational circuit.
- We do not have to use any timing and synchronization signal such as clock signal in a combinational circuit.

Sequential circuits :

Definition :

- In the sequential circuit, the **timing** parameter also needs to be taken into consideration.
- The output of a sequential circuit depends on the present time inputs, the previous output (past) and the sequence in which the inputs are applied.
- In order to provide the previous input or output a memory element is required to be used. Thus a sequential circuit needs to use a memory element as shown in Fig. 3.1.1.



(C-560) **Fig. 3.1.1 : Block diagram of a sequential circuit**

- Fig. 3.1.1 shows the block diagram of a sequential circuit which includes the memory element in the feedback path.

Present state of sequential circuit :

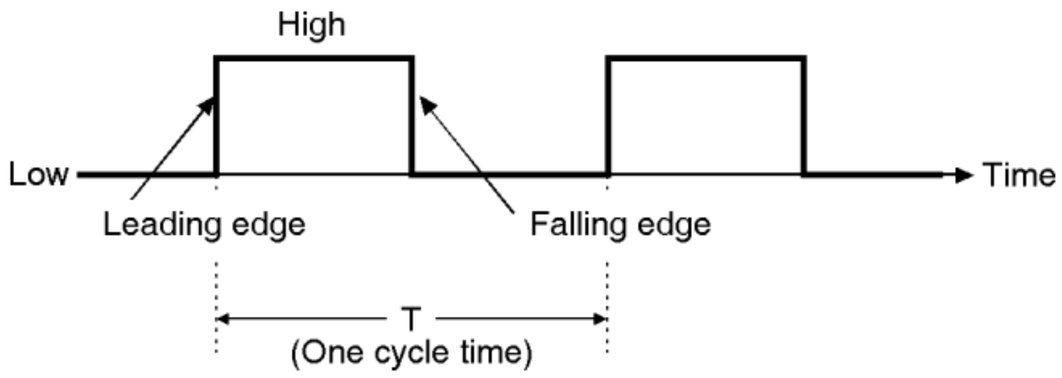
- The data stored by the memory element at any given instant of time is called as the present state of the sequential circuit.

Next state :

- The combinational circuit shown in Fig. 3.1.1 operates on the external inputs and the present state to produce new outputs.
- Some of these new outputs are stored in the memory element and called as the next state of the sequential circuit.
- The most important part of the sequential circuit seems to be the memory element. The memory element of Fig. 3.1.1 is known as Flip Flop (FF). It is the basic memory element.

3.1.1 Clock Signal :

- The clock signal shown in Fig. 3.1.2 is a timing signal. Every sequential signal will have this timing signal applied as an input signal as an input signal.
- Clock is a rectangular signal as shown in Fig. 3.1.2, with a duty cycle equal to 50%. That means its on time is equal to its off time.
- The clock signal repeats itself after every T seconds. Hence the clock frequency is f = 1/T.



(C-561) **Fig. 3.1.2 : Clock signal**

3.1.2 Comparison of Combinational and Sequential Circuits :

SPPU : Dec. 09, May 17, May 19

University Questions

Q. 1 Explain the difference between combinational and sequential circuits. **(Dec. 09, 4 Marks, May 17, 2 Marks)**

Q. 2 Write difference between sequential and combinational circuit. **(May 19, 4 Marks)**

TechKnowledge Publications

Sr. No.	Parameter	Combinational circuits	Sequential circuits
1.	Output depends on	Inputs present at that instant of time.	Present inputs and past inputs/outputs.
2.	Memory	Not necessary	Necessary
3.	Clock input	Not necessary	Necessary
4.	Examples	Adders, subtractors, code converters	Flip flops, shift registers, counters

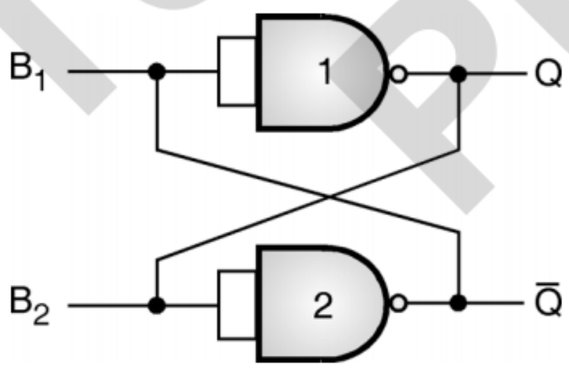
3.1.3 1-Bit Memory Cell (Basic Bistable Element) :

Definition :

- Flip-flop is also known as the basic digital memory circuit.
- It has two stable states namely logic 1 state and logic 0 state. We can design it either using NOR gates or NAND gates.

Circuit diagram :

- A flip-flop can be designed by using the fundamental circuit shown in Fig. 3.1.3.
- NAND gates 1 and 2 are basically acting as inverters. Hence this circuit is called as a cross coupled inverter.
- Output of gate 1 is connected to the input of gate-2 and output of gate 2 is connected to input of gate-1 as shown in Fig. 3.1.3.



(C-562) **Fig. 3.1.3 : A cross coupled inverter as memory element**

Operation :

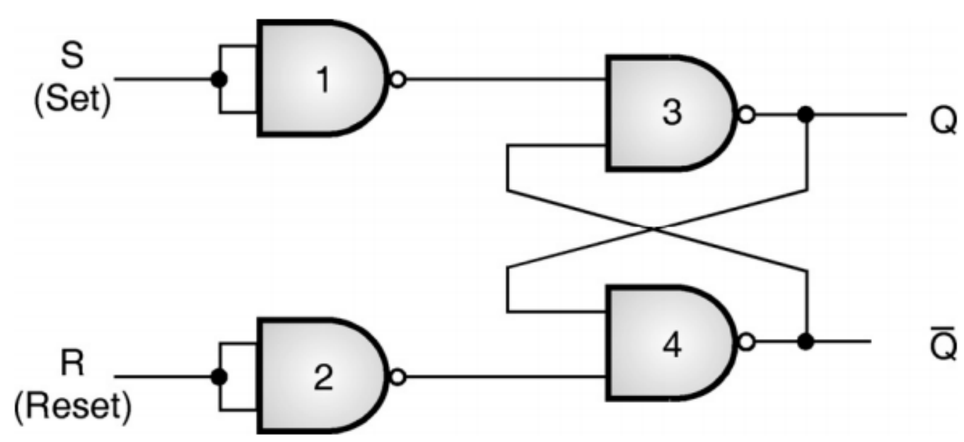
- Assume that output of gate-1 i.e. Q = 1. Hence B_2 = 1.
- As B_2 = 1, output of gate-2 i.e. $\overline{Q}$ = 0. This makes B_1 = 0. Hence Q continues to be equal to 1.
- Similarly we can demonstrate that if we start with Q = 0, then we end up obtaining Q = 0 and $\overline{Q}$ = 1.

Conclusions :

- From the above discussion we can draw the following conclusions :
- The outputs of the circuit (Q and $\overline{Q}$) will always be complementary. That means if Q = 0 then $\overline{Q}$ = 1 and vice versa. They will never be equal Q = $\overline{Q}$ = 0 or 1 is an invalid state.
- This circuit has two stable states.
- One of them corresponds to Q = 1, $\overline{Q}$ = 0 and it is called as 1 state or **set state.** Whereas the other state corresponds to Q = 0, $\overline{Q}$ = 1 and it is called as 0 state or **reset state.**
- If the circuit is in the reset state (Q = 0, $\overline{Q}$ = 1), then it will continue to be in the reset state and if it is in the set state (Q = 1, $\overline{Q}$ = 0) then it will continue to remain in the set state.
- This property of the circuit shows that it can store 1 bit of digital information. Therefore it is called as a **1-bit memory cell.**

3.1.4 Latch :

- The cross coupled inverter of Fig. 3.1.3 is capable of locking or latching the information. Hence this circuit is also called as a latch.
- The disadvantage of the cross coupled inverter circuit is that we cannot enter the desired digital data into it.
- This disadvantage can be overcome by modifying the circuit as shown in Fig. 3.1.4.
- This modification will allow us to enter the desired digital data into the circuit.



(C-563) **Fig. 3.1.4 : Modified memory cell**

Operation :

Case I : S = R = 0 (No change) :

- The outputs of gates 1 and 2 will become 1.
- Let Q = 0 and $\overline{Q}$ = 1 initially. Hence both the inputs to gate 3 are 1 and the inputs to gate-4 are (01).

TechKnowledge Publications

– So gate-3 output i.e. Q = 0 and gate-4 output i.e. $\overline{Q}$ = 1.

– Thus with S = R = 0, there is no change in the state of outputs.

Case II : S = 1, R = 0 (Set) :

– Since S = 1 and R = 0, one of the inputs to gate-3 will be 0. This will force Q output to 1.

– Hence both the inputs to gate-4 will be 1. This forces $\overline{Q}$ to 0.

– Hence for S = 1, R = 0, the outputs are Q = 1 and $\overline{Q}$ = 0. This is set state.

Case III : S = 0, R = 1 (Reset) :

– If S = 0, R = 1 then one of the inputs to gate-4 will be 0. This will force the $\overline{Q}$ output to 1.

– Hence both the inputs to gate 3 will be 1. This forces Q to 0.

– Thus for S = 0, R = 1, the outputs are Q = 0, $\overline{Q}$ = 1. This is the reset state or clear state.

Case IV : S = R = 1 (Race : Prohibited) :

– If S = R = 1 then outputs of gates 1 and 2 will be zero.

– Hence one of the inputs to gates 3 and 4 will be 0.

– So both the outputs Q and $\overline{Q}$ will try to become 1. It is not allowed as Q and $\overline{Q}$ should be complementary. Hence S = R = 1 condition is prohibited.

3.2 S-R Latch using NOR Gates :

Logic diagram :

– Latch is a bistable circuit which has two stable states.

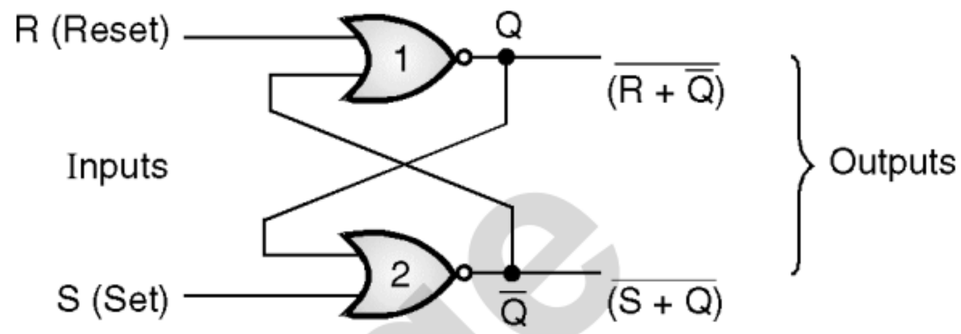
– It has two outputs Q and $\overline{Q}$ which are complements of each other. The two input terminals to this latch are set (S) and reset (R).

– Latch is a sequential logic circuit which monitors all its inputs continuously and will change its output as soon as the input changes. It does not wait for the clock signal.

– Generally an enable input signal is given for a latch. When the enable signal is active, the output will change in response to a change in input.

– S-R (set – reset) latch is the simplest type of latch and Fig. 3.2.1 shows S-R latch constructed using the NOR gates.

– Two NOR gates are cross coupled. That means the output of NOR gate 1 is connected to one of the inputs of the NOR gate 2. And output of NOR gate 2 is connected to one of the inputs of NOR gate 1.



(C-564) **Fig. 3.2.1 : S-R latch using NOR gates**

– The outputs Q and $\overline{Q}$ can be expressed using Boolean equations as follows :

$$Q = \overline{R + \overline{Q}} \text{ and } \overline{Q} = \overline{S + Q}$$

3.2.1 Operation of S-R Latch :

– Remember that the NOR gate output goes to "0" when anyone its inputs becomes HIGH (1).

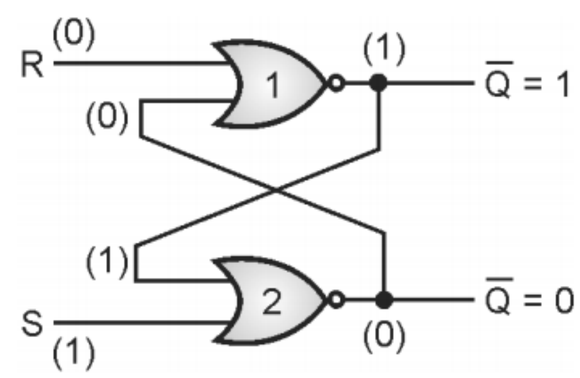
– If both the inputs are high (1), then the output is "0".

– Let us understand the operation under four different conditions :

1. Case I : S = 1, R = 0
2. Case II : S = 0, R = 1
3. Case III : S = 0, R = 0
4. Case IV : S = 1, R = 1

– For the operation refer Figs. 3.2.2(a), (b), (c) and (d)

Case I : S = 1, R = 0 (Set)



(C-565) **Fig. 3.2.2(a) : Operation with S = 1, R = 0**

– As S = 1, output of NOR gate 2 i.e. $\overline{Q}$ becomes 0. R is already 0. Hence both inputs of NOR – 1 are 0.

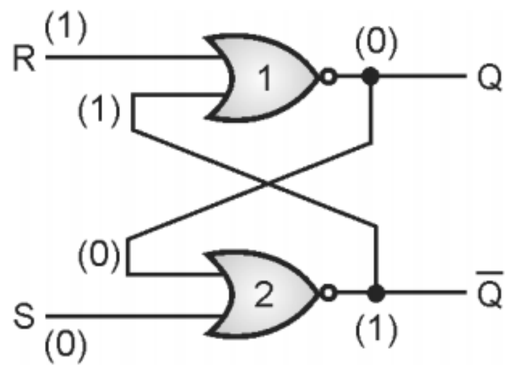
– Therefore Q = 1.

– Hence both inputs to NOR – 2 are 1.

– Hence $\overline{Q}$ = 0. Thus a stable output state of Q = 1 and $\overline{Q}$ = 0 is reached.

∴ For S = 1, R = 0, Q = 1 and $\overline{Q}$ = 0. This is called as the "Set" mode.

Case II : S = 0, R = 1 (Reset)



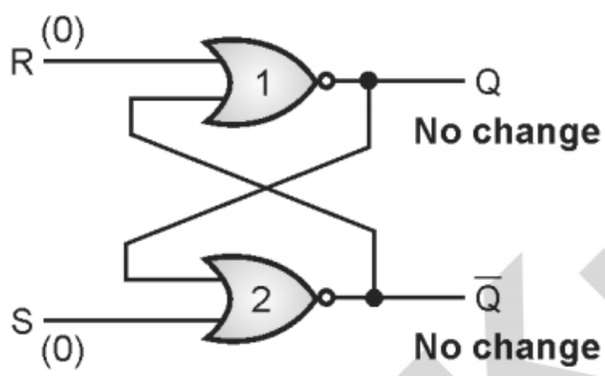
(C-566) Fig. 3.2.2(b) : Operation with S = 0 and R = 1

– As R = 1 output of NOR-1 i.e. Q becomes 0. S is already 0. Hence both inputs to NOR-2 are 0.

– Hence output of NOR-2 i.e. $\overline{Q}$ = 1. Thus the circuit reaches a stable state where Q = 0 and $\overline{Q}$ = 1.

∴ For S = 0 and R = 1, Q = 0 and $\overline{Q}$ = 1. This is known as the reset mode of operation.

Case III : S = 0, R = 0 :



(C-567) Fig. 3.2.2(c) : Operation with S = R = 0

– We know that $\overline{Q} = \overline{S + Q}$ and $Q = \overline{R + \overline{Q}}$

– Substitute S = R = 0

$\therefore \overline{Q} = \overline{0 + Q}$ and $Q = \overline{0 + \overline{Q}}$

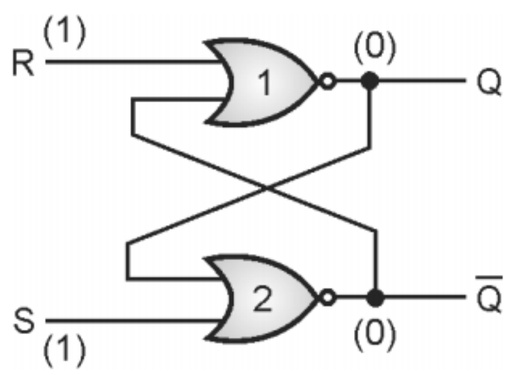
$\therefore \overline{Q} = \overline{0} \cdot \overline{Q}$ and $Q = \overline{0} \cdot Q$

according to De-Morgans theorems.

$\therefore \overline{Q} = 1 \cdot \overline{Q} = \overline{Q}$ and $Q = 1 \cdot Q = Q$

∴ If S = R = 0, then Q and $\overline{Q}$ do not change their states.

Case IV : S = 1, R = 1 :



(C-568) Fig. 3.2.2(d) : Operation with S = R = 1

– If S = R = 1, then one input each of NOR-1 and NOR-2 is at logic 1. So outputs of both these gates will try to become 0.

– Thus both the outputs Q and $\overline{Q}$ try to become 0 at the same time. This is not acceptable because Q and $\overline{Q}$ should be complements of each other.

– Therefore this state is an indeterminate state and therefore should be avoided.

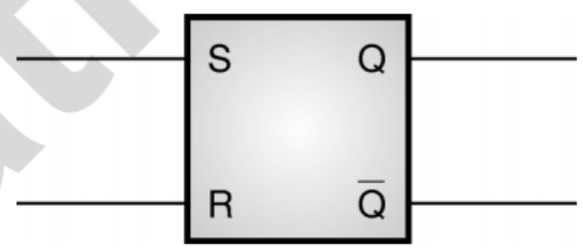
– This state should not be used because it violates the basic requirement of a latch i.e. Q and $\overline{Q}$ should be complements of each other.

3.2.2 Symbol and Truth Table of S-R Latch :

– The symbol and truth table of S-R latch using NOR gates are as shown in Figs. 3.2.3(a) and (b) respectively.

– In the truth table Q_n and $\overline{Q}_n$ represent the present states of outputs i.e. these are the outputs before applying a new set of inputs.

– Q_{n+1} and $\overline{Q}_{n+1}$ represent the next states of outputs i.e. the outputs after apply a new set of inputs.



Q_n and $\overline{Q}_n$: Present states Q_{n+1} and $\overline{Q}_{n+1}$: Next states

(C-569) (a) Symbol

Inputs		Outputs				Comment
S	R	Q_n	$\overline{Q}_n$	Q_{n+1}	$\overline{Q}_{n+1}$	
0	0	0	1	0	1	No change (NC)
0	0	1	0	1	0	
0	1	0	1	0	1	Reset
0	1	1	0	0	1	
1	0	0	1	1	0	Set
1	0	1	0	1	0	
1	1	0	1	X	X	Prohibited state
1	1	1	0	X	X	

(C-8074) (b) Truth table of S-R latch

Fig. 3.2.3

Summary of operation of S-R latch :

The summary of operation of an S-R latch is as follows :

– For S = R = 0 the latch output does not change.

– S = 0, R = 1 is called as the "Reset" condition as Q = 0 and $\overline{Q}$ = 1.

Tech Knowledge Publications

- S = 1, R = 0 is called as the "Set" condition as Q = 1 and $\overline{Q}$ = 0.
- S = R = 1 is the prohibited state. The output is unpredictable. This condition should therefore be avoided.

Race condition :

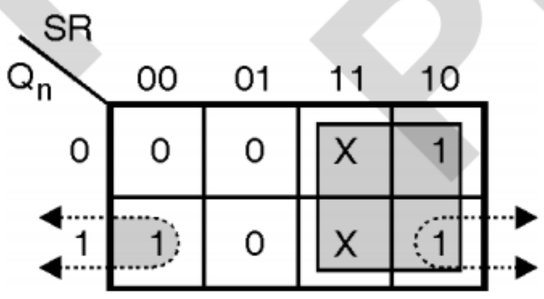
- The condition S = R = 1 is called as "Race" condition.
- When any one input to a NOR gate is 1, its output becomes 0. Thus both the outputs will try to become 0. This is called as the RACE condition.

3.2.3 Characteristic Equation :

- One way of explaining the behaviour of a latch or flip flop is to use its truth table.
- The truth table is also called as **excitation table.**
- Another way of doing it is to use special type of equations called **characteristic equations.**
- The characteristic equation of a flip-flop is the equation which relates the next state of the flip flop or latch Q_{n+1} or $\overline{Q}_{n+1}$ to the current state and inputs (Q_n, S and R).
- Characteristic equation is actually obtained from the truth table of the flip flop or latch, using the K-map.

Characteristic equation of SR latch :

- Refer to the truth table of SR latch and write the K-map for the next state of output i.e. Q_{n+1} as shown in Fig. 3.2.4.



(C-570) Fig. 3.2.4 : K-map for next state Q_{n+1} of SR latch

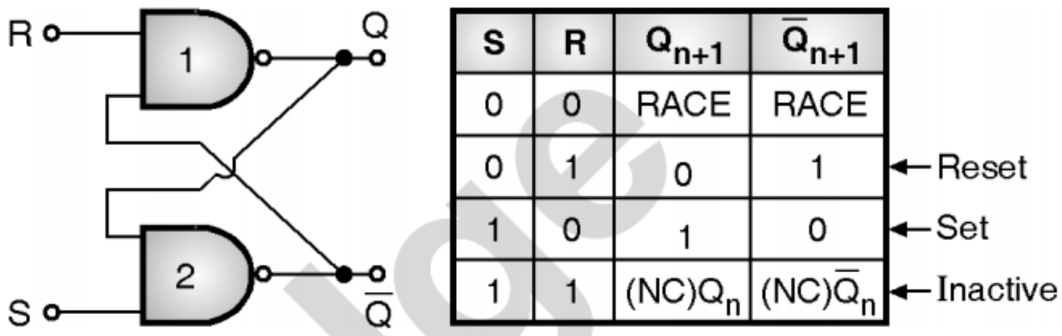
- After simplification, the characteristic equation of SR latch is given by,

$$Q_{n+1} = S + \overline{R}\,Q_n \qquad ...(3.2.1)$$

3.2.4 S-R Latch using NAND Gates :

Logic diagram :

- We can construct a S-R latch with NAND gates as shown in Fig. 3.2.5(a).
- Note that the outputs of two NAND gates are cross connected, in an identical manner as that in the NOR latch.
- Fig. 3.2.5(a) shows the NAND latch and Fig. 3.2.5(b) shows its truth table.



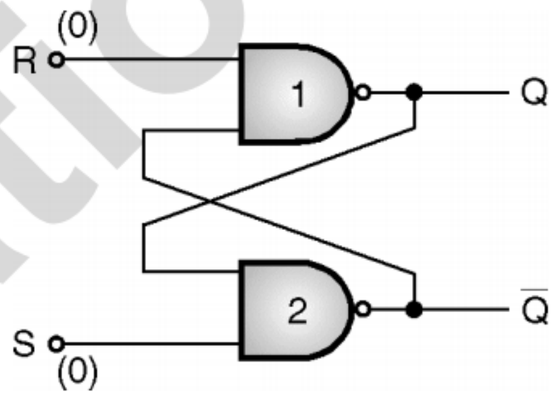
(a) NAND latch **(b) Truth table**

(C-571) Fig. 3.2.5

Operation :

- The operation of S-R NAND latch is summarised in Fig. 3.2.6.

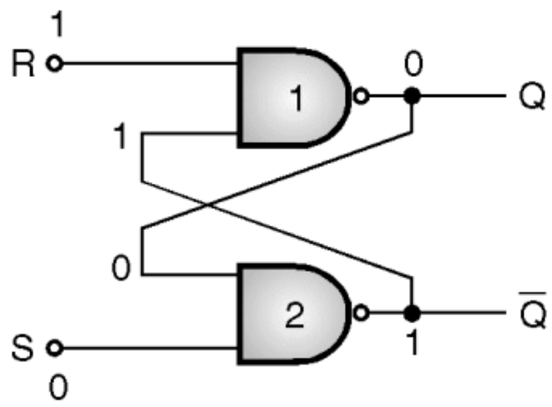
Case I : S = 0, R = 0 : Race



(C-572) Fig. 3.2.6(a) : S = 0, R = 0

- When any one input of a NAND gate becomes 0, its output is forced to 1.
- Here S = R = 0. ∴ Q and $\overline{Q}$ both will be forced to be equal to 1.
- This is an undeterminate state and hence should be avoided.
- This is also called as Race condition.

Case II : S = 0, R = 1 : Reset



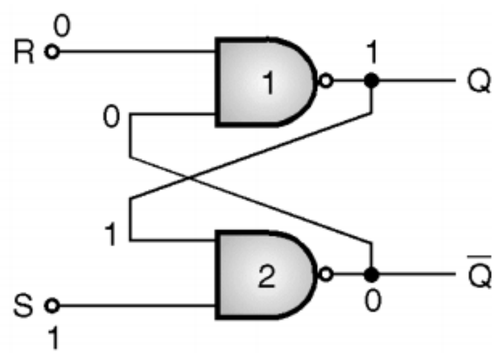
(C-573) Fig. 3.2.6(b)

- Since S = 0, it forces $\overline{Q}$ to be 1.
- Hence both inputs to NAND-1 are 1.
- Hence Q = 0.

– Thus with S = 0 and R = 1 the outputs are Q = 0 and $\overline{Q}$ = 1.

– This is the reset condition.

Case III : S = 1, R = 0 : Set



(C-573) **Fig. 3.2.6(c)**

– Since R = 0, Q is forced to 1.

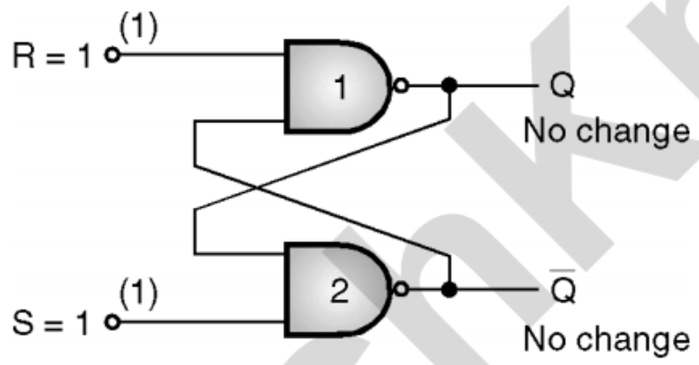
– Hence both inputs to NAND-2 are 1.

– Hence $\overline{Q}$ = 0.

– Thus with S = 1 and R = 0 the outputs are Q = 1 and $\overline{Q}$ = 0.

– This is the set condition.

Case IV : S = 1, R = 1 : No change



(C-574) **Fig. 3.2.6(d)**

$$Q_{n+1} = \overline{R \cdot \overline{Q}_n} \text{ and } \overline{Q}_{n+1} = \overline{S \cdot Q_n}$$

– Using De-Morgan's theorem,

$$Q_{n+1} = \overline{R} + Q_n \text{ and } \overline{Q}_{n+1} = \overline{S} + \overline{Q}_n.$$

– Substitute $\overline{R}$ = 0 and $\overline{S}$ = 0 to get,

$$Q_{n+1} = 0 + Q_n = Q_n \text{ and } \overline{Q}_{n+1} = 0 + \overline{Q}_n = \overline{Q}_n.$$

– Thus there is no change in the outputs if S = R = 1.

– The symbol for S-R latch is shown in Fig. 3.2.7 alongwith the summary of operation.

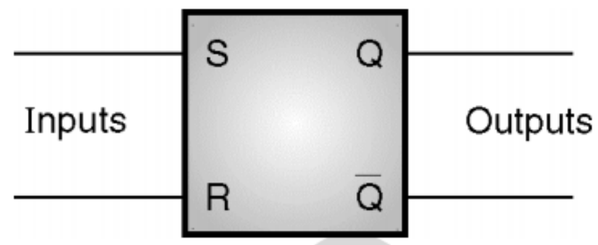
Circuit symbol and summary of operation of S-R latch :

– For S = 0, R = 0, both the outputs will be forced to become 1. This is RACE condition and should be available.

– For S = 0, R = 1, the outputs are Q = 0, $\overline{Q}$ = 1 and it is called as the Reset condition.

– For S = 1, R = 0, the outputs are Q = 1, $\overline{Q}$ = 0 and it is called as the set condition.

– For S = R = 1, there is no change in the output state.



(C-575) **Fig. 3.2.7 : Symbol of S-R latch**

3.3 Triggering Methods :

– In the latches and flip-flops, we use the additional signal called clock signal.

– Depending on which portion of the clock signal the latch or flip-flop responds to, we can classify them into two types :

1. Level triggered circuits.
2. Edge triggered circuits.

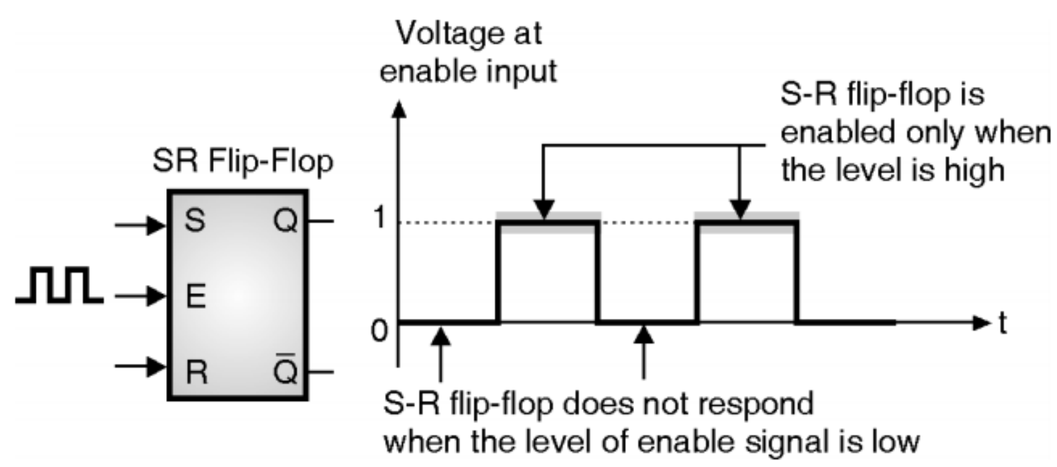
3.3.1 Concept of Level Triggering :

Definition :

– The latch or flip-flop circuits which respond to change in their inputs, only if their enable input (E) held at an active level which may be either HIGH or LOW level are called as **level triggered** latches or flip-flops.

– Thus these circuits do not respond at the rising or falling edges of clock. They only respond to the steady HIGH or LOW levels of the clock signal.

– Fig. 3.3.1(a) shows the symbol of a level triggered SR flip flop and Fig. 3.3.1(b) shows the clock signal applied at its input.



(a) **(b)**

(C-576) **Fig. 3.3.1 : Concept of level triggering**

3.3.2 Types of Level Triggered Flip-flops :

– There are two types of level triggered flip-flops :

1. Positive level triggered.
2. Negative level triggered.

Positive level triggered :

- If the outputs of a flip-flop respond to the input changes, only when its clock inputs at HIGH (1) level, then it is called as the positive level triggered flip-flop.
- The block diagram shown in Fig. 3.3.1(a) is a positive level triggered S-R FF.

Negative level triggered FF :

If the outputs of a flip-flop respond to the input changes, only when its clock input is at LOW (0) level, then it is called as the negative level triggered flip-flop.

Note : The level triggering is not used practically, due to some of its disadvantages.

3.3.3 Concept of Edge Triggering :

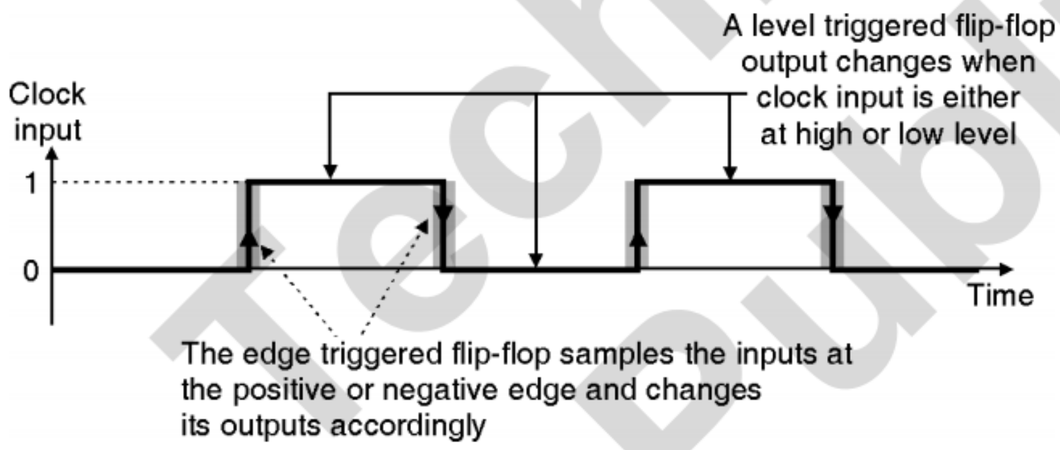
SPPU : May 11

University Questions

Q. 1 What is edge triggering ? **(May 11, 2 Marks)**

Definition :

- The flipflops which change their outputs only corresponding to the positive (rising) or negative (falling) edge of the clock input are called as **edge triggered flipflops.**
- These flip-flops are therefore said to be edge sensitive or edge triggered rather and not level triggered.



(C-577) Fig. 3.3.2

- The rectangular signal applied to the clock input of a flip-flop is shown in Fig. 3.3.2.
- If the same signal is applied as the clock signal to an edge triggered flip flop, then its outputs will change only at either rising (positive) edge or at the falling (negative) edge of the clock.
- The edge triggered flip-flops do not respond to the steady state high or low level in the clock signal at all.

3.3.4 Types of Edge Triggered Flip Flops :

- There are two types of edge triggered flip flops :
  1. Positive edge triggered flip flops.
  2. Negative edge triggered flip flops.
- Positive edge triggered flip flops will allow its outputs to change in response to its inputs only at the instants corresponding to the rising edges of clock (or positive spikes).
- Its outputs will not respond to change in inputs at any other instant of time.
- Negative edge triggered flip flops will respond only to the negative going edges (or spikes) of the clock.

3.4 Gated Latches :

- We have discussed the RS latches using the NAND and NOR gates.
- Now 2 more NAND gates are added to the basic SR latch and one more input called enable (E) is added, in order to obtain a gated SR latch or a level triggered SR flip-flop.
- A level voltage (0 or 1) or a clock signal can be applied to the enable (E) input.
- These flipflops will respond to the inputs if and only if we apply an **active level** at the enable input. This active level can be either 0 or 1 depending on the type of flip-flop.
- Such flipflops are called as level triggered flipflops or gated latches or clocked flipflops.

3.4.1 Types of Level Triggered (Clocked) Flip Flops :

- There are two types of level triggered latches :
  1. Positive level triggered.
  2. Negative level triggered.

3.5 The Gated S-R Latch (Level Triggered S-R Flip Flop) :

3.5.1 Positive Level Triggered SR Flip-flop :

Logic diagram :

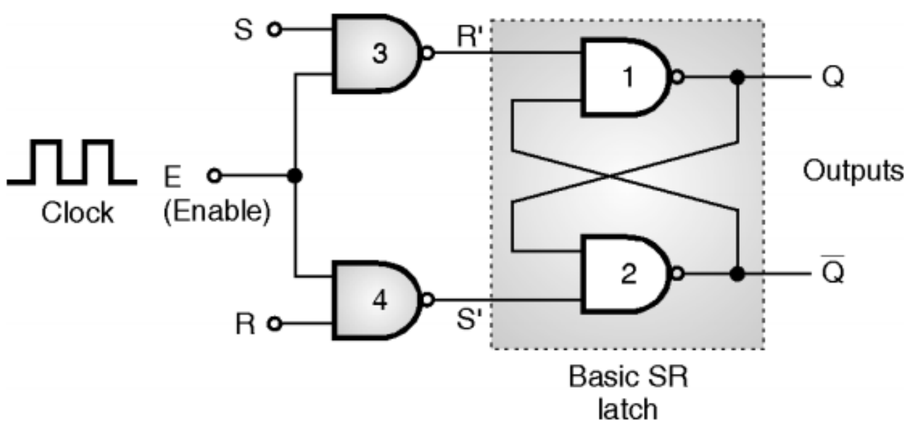
- The gated S-R latch is shown in Fig. 3.5.1. It is also called as clocked SR flip flop.
- It is basically the S-R latch using NAND gates with an additional "enable" (E) input. It is also called as **level triggered** S-R FF.
- The outputs of basic S-R latch used to change instantly in response to any change made at the input. But this does not happen with the gated S-R latch.

TechKnowledge Publications

– For this circuit, the change in output will take place if and only if the enable input (E) is made active.

– This circuit being positive level triggered, will respond to changes in input only if the enable input is held at logic 1 level.

– In short this circuit will operate as an S-R latch if E = 1 (Enable input is active) but there is no change in the outputs if E = 0 (Enable input is inactive).



(C-578) **Fig. 3.5.1 : Gated S - R latch**

Operation :

Case I : S = X, R = X, E = 0 (No change)

– Since enable E = 0, the outputs of NAND gates 3 and 4 will be forced to be 1 irrespective of the values of S and R.

– That means R′ = S′ = 1. These are the inputs of the basic S-R latch enclosed in the dotted box in Fig. 3.5.1.

– Hence the outputs of NAND latch i.e. Q and $\overline{Q}$ will not change. Thus if E = 0, then there is no change in the output of the gated S-R latch.

Case II : S = R = 0 E = 1 : No change

– If S = R = 0 then outputs of NAND gates 3 and 4 are forced to become 1.

– Hence R′ and S′ both will be equal to 1. Since S′ and R′ are the inputs of the basic S-R latch using NAND gates, there will be no change in the state of outputs.

– Thus for S = R = 0 the output state of this flip-flop remains unchanged.

Case III : S = 0, R = 1, E = 1 (Reset)

– Since S = 0, output of NAND-3 i.e. R′ = 1. And as R = 1 and E = 1 the output of NAND-4 i.e. S′ = 0.

– Hence Q_{n+1} = 0 and $\overline{Q}_{n+1}$ = 1. This is the reset condition.

Case IV : S = 1, R = 0, E = 1 (Set)

– Output of NAND 3 i.e. R′ = 0 and output of NAND 4 i.e. S′ = 1.

– Hence output of S-R NAND latch is Q_{n+1} = 1 and $\overline{Q}_{n+1}$ = 0.

– This is the reset condition.

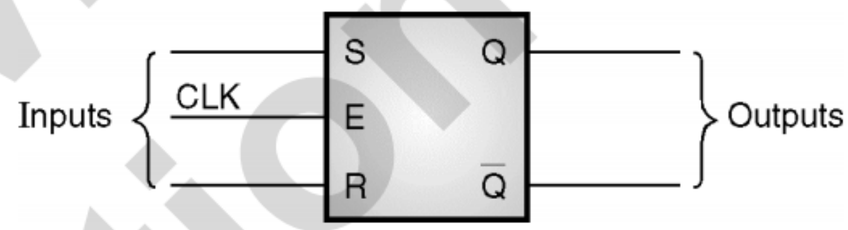
Case V : S = 1, R = 1, E = 1 (RACE)

– As S = 1, R = 1 and E = 1, the outputs of NAND gates 3 and 4 both are 0. i.e. S′ = R′ = 0.

– Hence the "Race" condition will occur in the basic NAND latch. This operation should be avoided as both Q and $\overline{Q}$ will try to become 1 at the same time as discussed earlier.

Symbol and truth table :

– The symbol and truth table of the gates S-R latch are as shown in Fig. 3.5.2(a) and Fig. 3.5.2(b) respectively.



(C-579) **Fig. 3.5.2(a) : Symbol for S - R latch**

(C-7827) **Fig. 3.5.2(b) : Truth table of gated S - R latch**

Inputs				Outputs		Comments
Case	Enable E	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
I	0	×	×	Q_n	$\overline{Q}_n$	No change as E = 0
II	1	0	0	Q_n	$\overline{Q}_n$	No change (NC)
III	1	0	1	0	1	Reset condition
IV	1	1	0	1	0	Set condition
V	1	1	1	RACE (Indeterminate)		Avoid this condition

3.5.2 Negative Level Triggered SR Flip Flop :

Logic diagram :

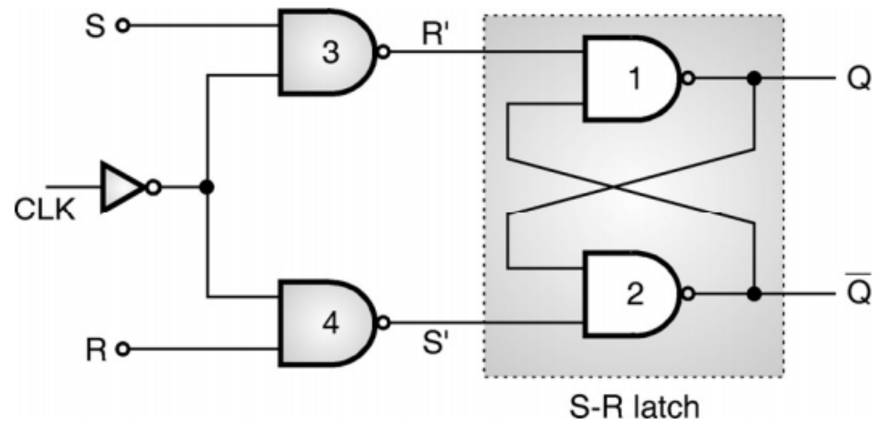
– Fig. 3.5.3(a) shows the circuit diagram of a negative level triggered SR flip-flop.

– It is the same circuit that we discussed in the previous section with only one additional inverter.

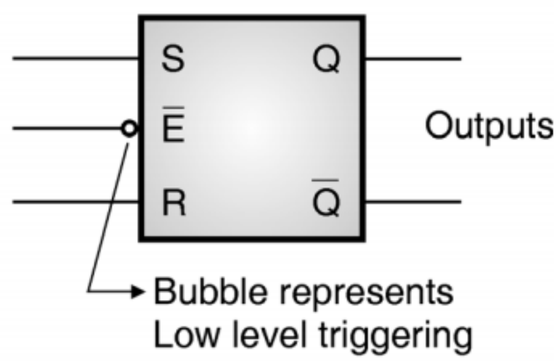
– Due to the additional inverter connected to the enable terminal, this circuit becomes sensitive to the low level (0) applied to the enable input. Hence it will enable the outputs if E = 0.

– If a square wave is applied to the enable input then the latch output will respond to input changes only when the square input is at its low (0) level.

TechKnowledge Publications



(a)



(b) Symbol

(C-580) Fig. 3.5.3 : Negative level triggered S - R latch

– Fig. 3.5.3(b) shows the symbol of negative level triggered S-R latch. Note that there is a bubble added to the enable input which is active low input.

3.5.3 Disadvantage of S-R latch :

– From the truth tables of the S-R latch using NOR and NAND gates we can draw the following important conclusion :

1. When S = R = 0 or S = R = 1, the outputs Q and $\overline{Q}$ either don't change (NC) or they are indeterminate (invalid) due to race condition.
2. This disadvantage of S-R latch can be overcome by using the gated D latch discussed in the next section.

3.5.4 Application of S-R latch :

– It is used as electronic timer.

3.6 The Gated D Latch (Clocked D Flip Flop) :

– In some applications, the S and R inputs will always be complementary. i.e. when S = 0, R = 1 and when S =1, R = 0. That means S = $\overline{R}$.

– For such applications we can use the gated D latch.

Logic diagram :

– The circuit diagram of the gated D latch is shown in Fig. 3.6.1(a) and its logic symbol is shown in Fig. 3.6.1(b).

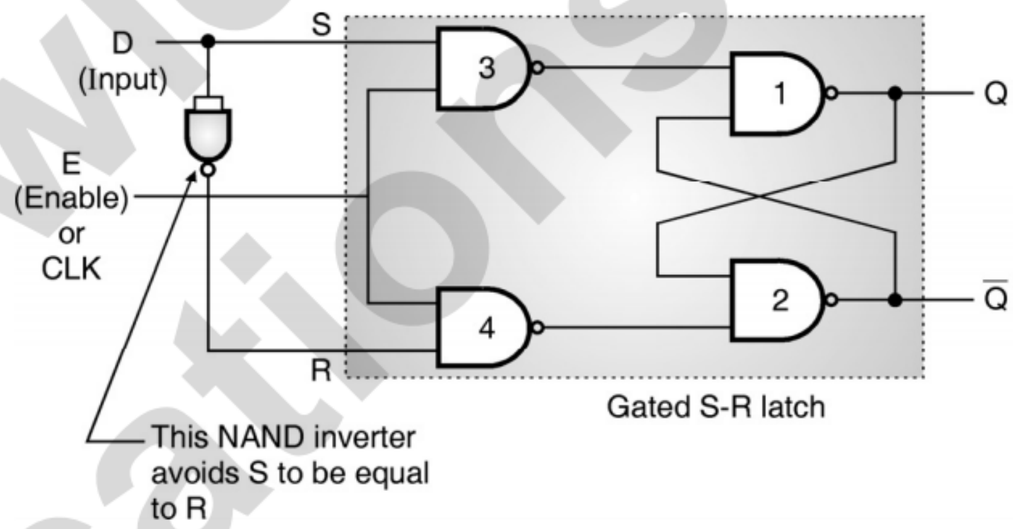
This is also called as level triggered D flip-flop or clocked D flip-flop.

– Note that D latch is the simple gated S-R latch with a small modification. A NAND inverter is connected between its S and R inputs as shown in Fig. 3.6.1(a).

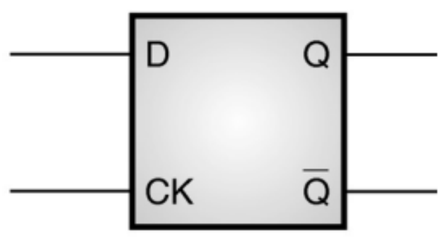
– This latch has only one input denoted by D.

– Due to the NAND inverter, S and R inputs will always be the complements of each other. Hence the input conditions such as S = R = 0 or S = R = 1, will never appear.

– This will avoid the problems associated with SR = 00 and SR = 11 conditions of SR flip-flop.



(a) Gated D latch



(b) Logic symbol of gated D latch

(C-3419) Fig. 3.6.1

Truth table :

– Truth table for the gated D latch is given in Table 3.6.1.

(C-8061) Table 3.6.1 : Truth table for the gated D latch

Inputs		Outputs		Comment
E	D	Q_{n+1}	$\overline{Q}_{n+1}$	
0	X	Q_n	$\overline{Q}_n$	No change (NC)
1	0	0	1	Reset condition
1	1	1	0	Set condition

Operation :

– If E = 0 then the latch is disabled. Hence there is no change in output.

Tech Knowledge Publications

– If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state the, next state is $Q_{n+1} = 0$ and $\overline{Q}_{n+1} = 1$. This is the reset condition.

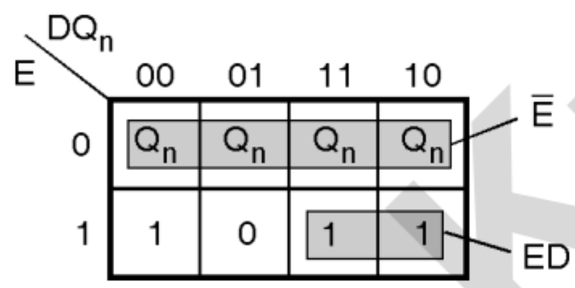
– On the other hand if E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and $Q_{n+1} = 1$, $\overline{Q}_{n+1} = 0$ irrespective of the present state.

From the truth table it is evident that Q output is same as D input, in otherwords Q output follows the D input. If D = 0 then Q = 0 and if D = 1 then Q = 1. However output follows input after some propagation delay hence the other name of the D flip-flop is delay flip-flop.

Characteristic equation for D latch :

– Refer to the truth table of D latch and write the K-map for Q_{n+1} as shown in Fig. 3.6.2.

For Q_{n+1}



(C-584) **Fig. 3.6.2 : K-map for D latch**

– After simplification we get the characteristic equation of D latch as,

$$Q_{n+1} = E \cdot D + \overline{E}$$

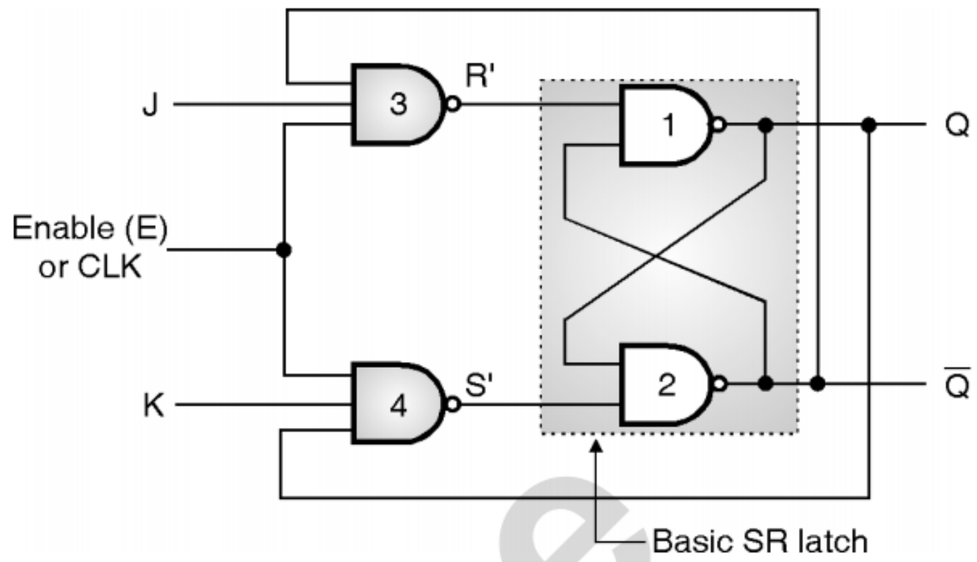
3.7 Gated JK Latch (Level Triggered JK Flip Flop) :

Logic diagram :

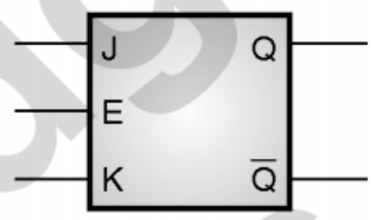
– The JK latch using NAND gates is shown in Fig. 3.7.1(a). It consists of the basic SR latch and an enable input. It is also called as level triggered JK flip-flop.

– Note the outputs Q and $\overline{Q}$ have been fed back and connected to the inputs of NAND gates 4 and 3 respectively.

– The JK latch of Fig. 3.7.1(a) responds, to the input changes if a positive level is applied at the enable (E) input. Hence it is a positive level triggered latch.



(a) A level triggered JK flip-flop



(b) Symbol of the gated JK latch

(C-3420) **Fig. 3.7.1**

Operation :

– The operation of NAND JK latch is exactly identical to that of the positive edge triggered JK flip flop discussed in section 3.10.1.

– The only difference between them is that, this circuit is level triggered.

– The operation of NAND JK latch has been summarized in Table 3.7.1.

(C-3420(a)) **Table 3.7.1 : Truth table of positive level triggered JK latch**

Case No.	Inputs			Outputs		State
	E	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
Case I	0	×	×	Q_n	$\overline{Q}_n$	No change
Case II	1	0	0	Q_n	$\overline{Q}_n$	No change
Case III	1	0	1	0	1	Reset
Case IV	1	1	0	1	0	Set
Case V	1	1	1	$\overline{Q}_n$	Q_n	Toggle

3.7.1 Race Around Condition in JK Latch :

SPPU : May 05, May 07

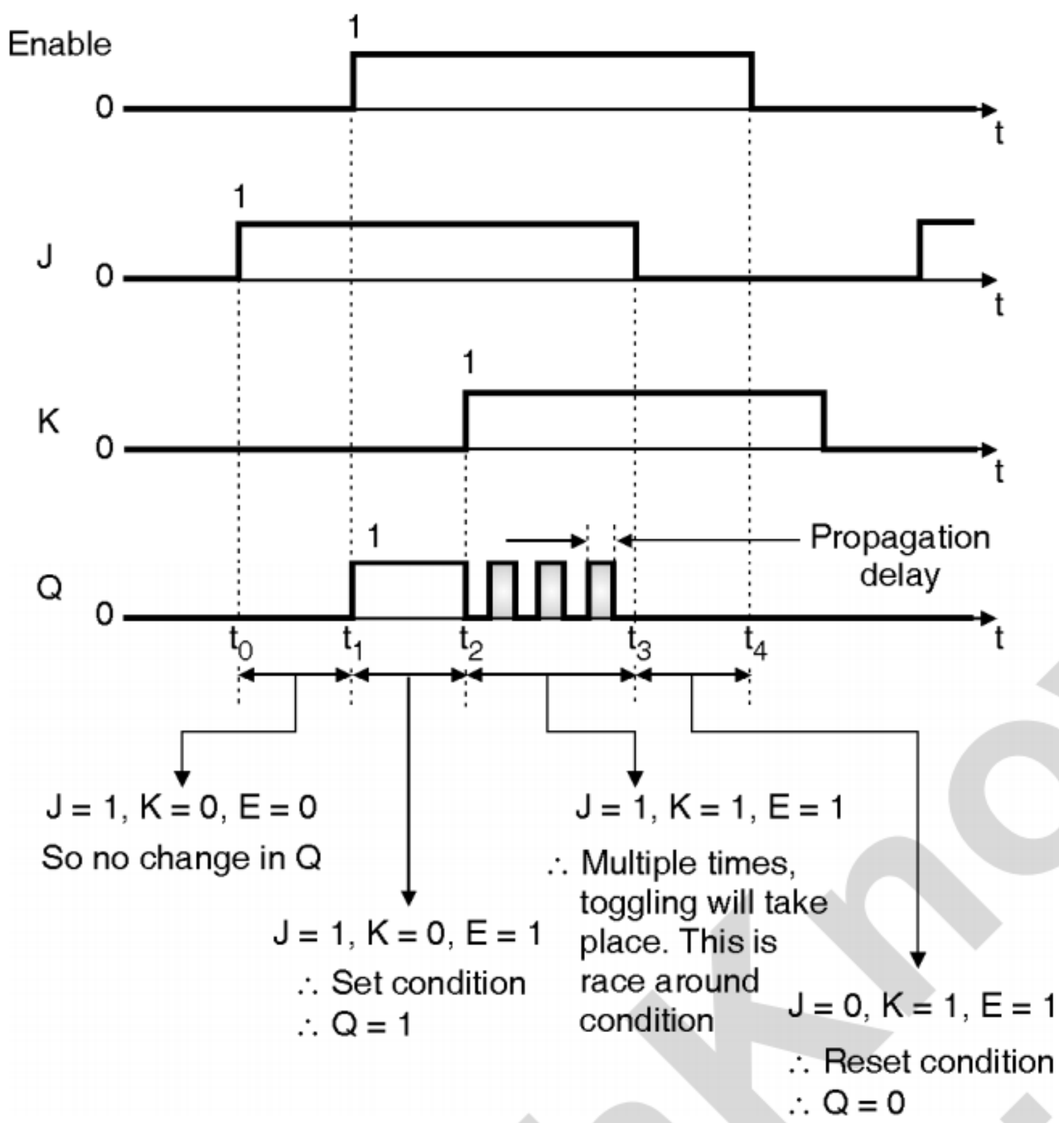
University Questions

Q. 1 What is race-around condition ? Explain with the help of timing diagram. How is it removed in basic flip-flop circuit ? **(May 05, 8 Marks)**

Q. 2 What is race-around condition ? How will you eliminate it ? Explain with the help of necessary circuit diagram and timing diagram. **(May 07, 8 Marks)**

TechKnowledge Publications

- The "Race Around Condition" that we are going to explain occurs when J = K = 1 i.e. when the latch is in the toggle mode.
- Refer Fig. 3.7.2 which shows the waveforms for the various modes, when a rectangular waveform is applied to the "Enable" input.



(C-586) Fig. 3.7.2 : Waveforms for various modes of a JK latch

Interval t_0 - t_1 :

- During this interval J = 1, K = 0 and E = 0.
- Hence the latch is disabled and there is no change in Q.

Interval t_1 - t_2 :

- During this interval J = 1, K = 0 and E = 1.
- Hence this is a set condition and Q becomes 1.

Interval t_2 - t_3 : Race around

- At instant t_2, J = K = 1 and E = 1 Hence the JK latch is in the toggle mode and Q becomes low (0) and $\overline{Q}$ = 1.
- These changed outputs get applied at the inputs of NAND gates 3 and 4 of the JK latch. Thus the new inputs to Gates 3 and 4 are :

  NAND - 3 : J = 1, E = 1, $\overline{Q}$ = 1,

  NAND - 4 : K = 1, E = 1, Q = 0.
- Hence R′ will become 0 and S′ will become 1.
- Therefore after a time period corresponding to the propagation delay, the Q and $\overline{Q}$ outputs will change to, Q = 1 and $\overline{Q}$ = 0.
- These changed output again get applied to the inputs of NAND-3 and 4 and the outputs will toggle again.
- Thus as long as J = K = 1 and E = 1, the outputs will keep toggling indefinitely as shown in Fig. 3.7.2. This multiple toggling in the J-K latch is called as Race Around condition. It must be avoided.

Interval t_3 - t_4 :

- During this interval J = 0, K = 1 and E = 1.
- Hence it is the reset condition. So Q becomes zero.

How to avoid race around condition ?

- The race around condition in JK latch can be avoided by,
  1. Using the edge triggered JK flip flop.
  2. Using the master slave JK flip flop.

3.7.2 Difference between Latch and Flip-flop :

Latches :

- Latches and flip flops both are basically the bistable elements.
- As discussed earlier a latch has got an enable input. As long as it is active, the latch output will keep changing according to the changes in its input.
- In other words latch is a level triggered flip flop.

Flip-flop :

- But flip flop is a sequential circuit which generally samples its inputs and changes its outputs only at particular instants of time and not continuously.
- The flip-flops are therefore said to be edge sensitive or edge triggered rather than being level triggered like latches.

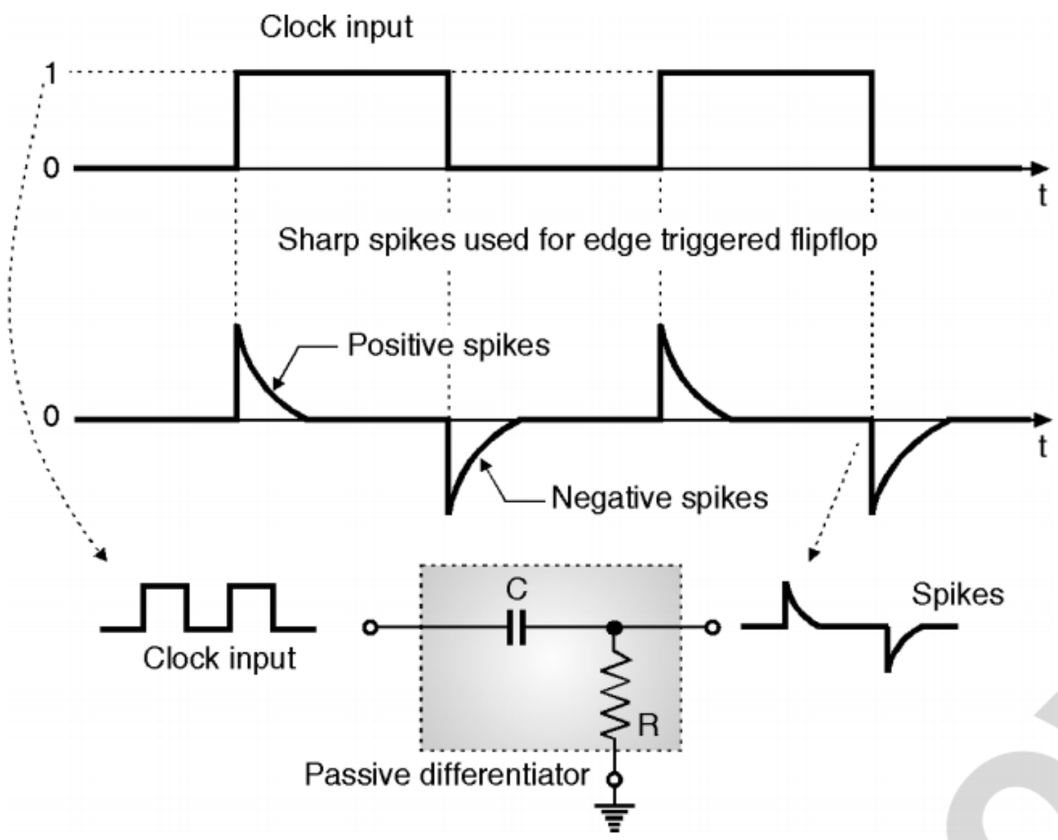
3.8 Edge Triggered Flip Flops :

- We have already discussed the concept of edge triggering.
- For the edge triggered flip flops, it is necessary to apply the clock signal (timing signal) in the form of sharp positive and negative spikes instead of in the form of a rectangular pulse train.

Tech Knowledge Publications

– Such sharp spikes are shown in Fig. 3.8.1. These spikes can be derived from the rectangular clock pulses with the help of a passive differentiator circuit shown in Fig. 3.8.1.

– Thus the passive differentiator acts as a pulse shaping circuit.



(C-587) **Fig. 3.8.1 : Use of differentiator to obtain sharp edges**

3.8.1 Positive Edge Triggered S-R Flip Flop :

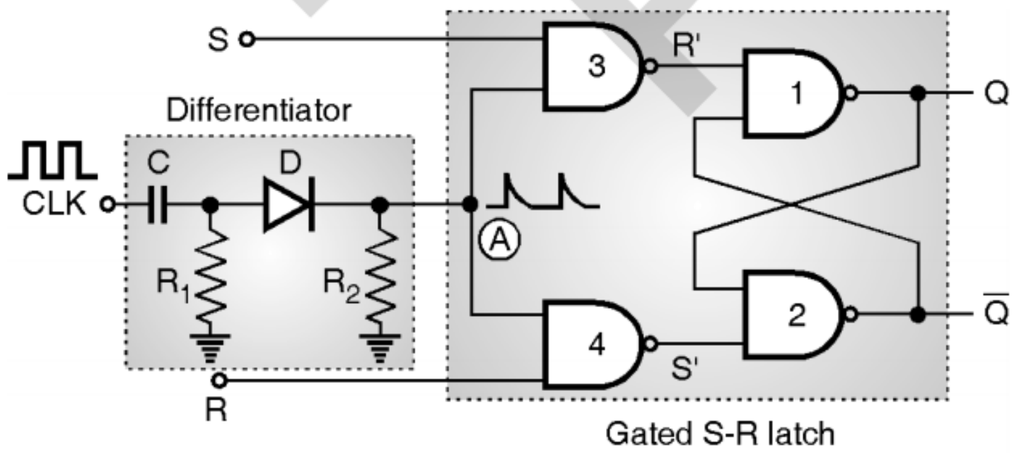
SPPU : Dec. 11

University Questions

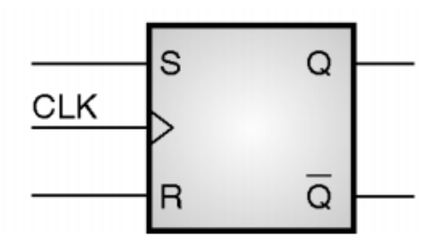
Q. 1 Draw and explain SR flip-flop using NAND gates. **(Dec. 11, 8 Marks)**

Circuit diagram :

– Fig. 3.8.2(a) shows the circuit diagram of a positive edge triggered S-R flip flop and Fig. 3.8.2(b) shows the logic symbol for it.



(a) Positive edge triggered S-R flip flop



(b) Logic symbol

(C-588) **Fig. 3.8.2**

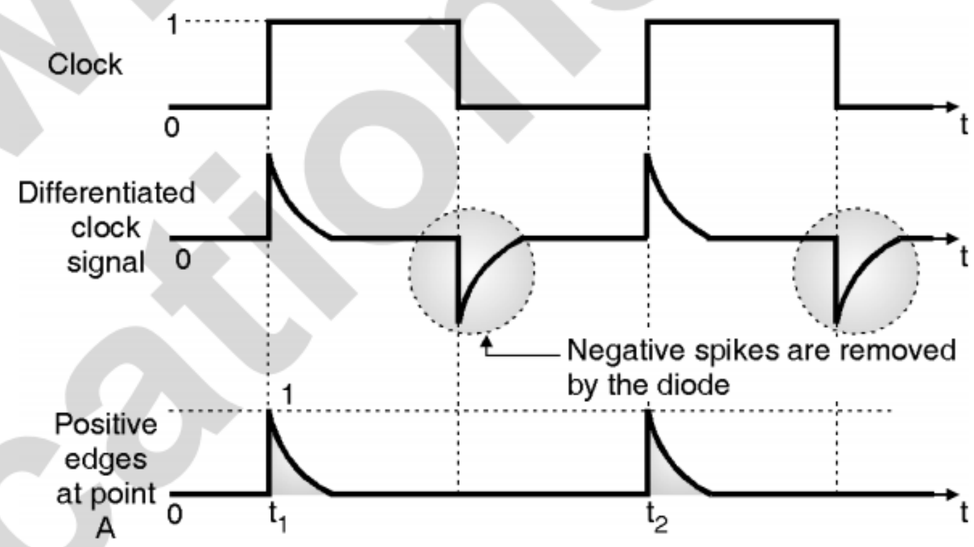
Operation :

– Note that the SR flip flop of Fig. 3.8.2(a) consists of a differentiator circuit and a gated S-R latch (level triggered SR flip-flop) which we have already discussed.

– C - R_1 acts as a differentiator and converts the rectangular clock pulses into positive and negative spikes.

– The diode acts as rectifying diode and allows only the positive spikes to pass through it while, blocking the negative spikes.

– Thus we get only the positive spikes, corresponding to the leading edges of the clock signal, as shown in Fig. 3.8.3 at point A which is enable input of the gated S-R latch.



(C-589) **Fig. 3.8.3 : Generation of positive triggering pulses**

– Due to these sharp positive spikes applied at point "A", the gated S-R latch in the S-R flip flop will be enabled only for a short duration of time when the positive spike is present at A. (at instants t_1, t_2 ... in Fig. 3.8.3)

– At these instants, the flip-flop behaviour is exactly identical to that of the enabled gated S-R latch (enabled level triggered SR flip-flop).

– The truth table of positive edge triggered S-R flip flop will also be identical to that of the gated S-R latch with only one change.

– The "enable" input will now be replaced by clock input. And the outputs will change only if a positive edge is applied to the clock input. The positive clock edge is denoted by an arrow (↑) in Table 3.8.1.

Truth Table :

– The truth table of a positive edge triggered SR flip flop is shown in Table 3.8.1.

TechKnowledge Publications

(C-6260) Table 3.8.1 : Truth table of a positive edge triggered SR flip flop

	Inputs			Outputs		Remark
	CLK	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
FF is disabled	0	×	×	Q_n	$\overline{Q}_n$	No change (NC)
	1	×	×	Q_n	$\overline{Q}_n$	No change (NC)
	↓	×	×	Q_n	$\overline{Q}_n$	No change (NC)
FF responds only to the positive edges of clock	↑	0	0	Q_n	$\overline{Q}_n$	No change (NC)
	↑	0	1	0	1	Reset
	↑	1	0	1	0	Set
	↑	1	1	Race	Race	Avoid

↓ = Negative edge of clock, ↑ = Positive edge of clock

- Note that this flip-flop does not respond to the positive or negative levels in the clock signal.
- Similarly it does not respond to the negative edges of the clock.
- This flip flop will respond only to the positive edges of clock signal.
- With positive edge of the clock, the SR flip flop behaves in the following way.

S = R = 0 → No change in output

S = 0, R = 1 → $Q_{n+1} = 0, \overline{Q}_{n+1} = 1$ Reset condition

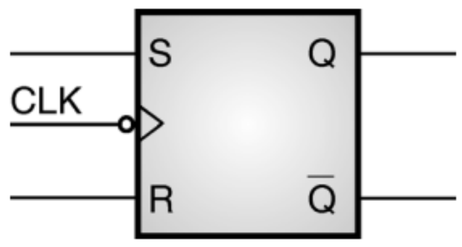
S = 1, R = 0 → $Q_{n+1} = 1, \overline{Q}_{n+1} = 0$ Set condition

S = R = 1 → Race condition.

3.8.2 Negative Edge Triggered S - R Flip Flop :

Symbol and Truth table :

- The internal circuit (with NAND gates) of the negative edge triggered S-R flip flop is exactly same as that for the positive edge triggered SR flip-flop discussed earlier.
- The differentiator circuit is slightly modified in order to make this flip flop respond to the negative (falling) edges of the clock input.
- Fig. 3.8.4 shows the circuit symbol of the negative edge triggered S-R flip flop and Table 3.8.2 shows its truth table.



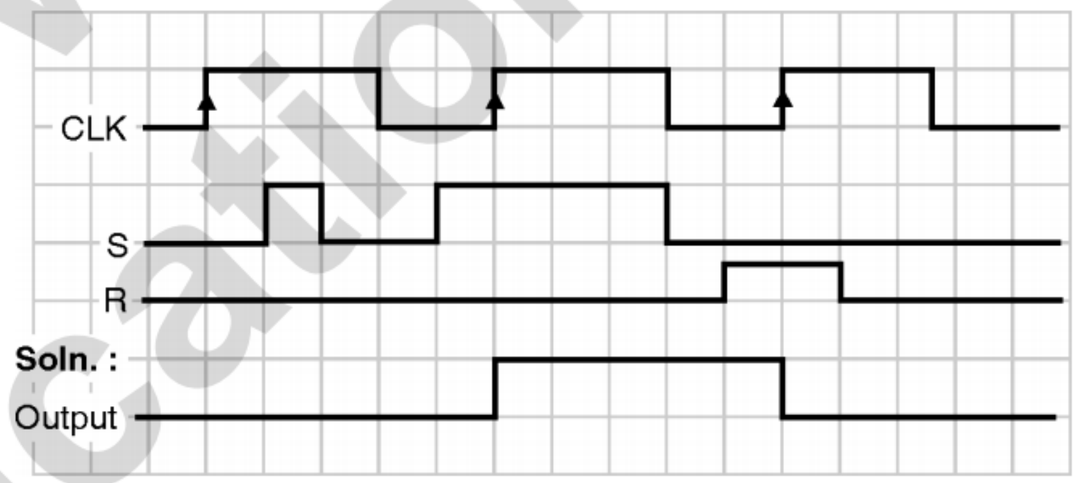
(C-590(a)) Fig. 3.8.4 : Circuit symbol of negative edge triggered SR FF

(C-6207) Table 3.8.2 : Truth table of negative edge triggered S-R FF

	Inputs			Outputs		State
	CLK	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	
FF is disabled	0	×	×	Q_n	$\overline{Q}_n$	No change (NC)
	1	×	×	Q_n	$\overline{Q}_n$	No change (NC)
	↑	×	×	Q_n	$\overline{Q}_n$	No change (NC)
FF responds only to negative edges	↓	0	0	Q_n	$\overline{Q}_n$	No change (NC)
	↓	0	1	0	1	Reset
	↓	1	0	1	0	Set
	↓	1	1	Race	Race	Avoid

↑ = Positive edge of clock, ↓ = Negative edge of clock

Ex. 3.8.1 : Draw the waveform for the Q output for the input waveforms shown in Fig. P. 3.8.1 for the positive edge triggered SR flip-flop.



(C-591) Fig. P. 3.8.1

3.9 Edge Triggered D Flip Flop :

- The edge triggered D flip flops can be of two types :
  1. Positive edge triggered D flip flop.
  2. Negative edge triggered D flip flop.
- These flip flops can be derived from the level triggered D latch which we have discussed in section 3.6.

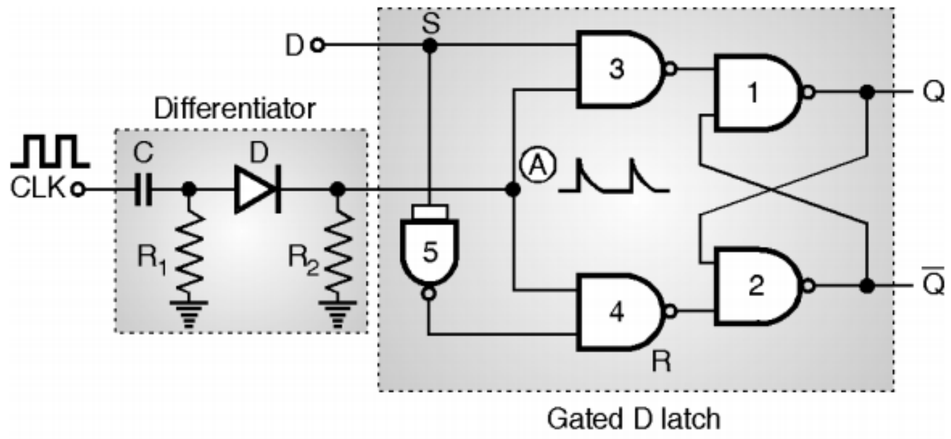
3.9.1 Positive Edge Triggered D Flip Flop :

Logic diagram :

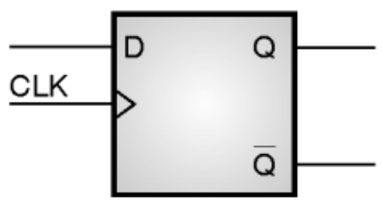
- Fig. 3.9.1(a) shows the positive edge triggered D flip flop. It consists of a gated D latch and a differentiator circuit. The symbol is as shown in Fig. 3.9.1(b).
- The clock pulses are applied to the circuit through a differentiator formed by R_1C and a rectifier circuit consisting of diode D and R_2.
- The NAND gates 1 through 5 form a D latch.
- The differentiator converts the clock pulses into positive and negative spikes and the combination of D and R_2

Tech Knowledge Publications

will allow only the positive spikes to pass through to point "A", by blocking the negative spikes.

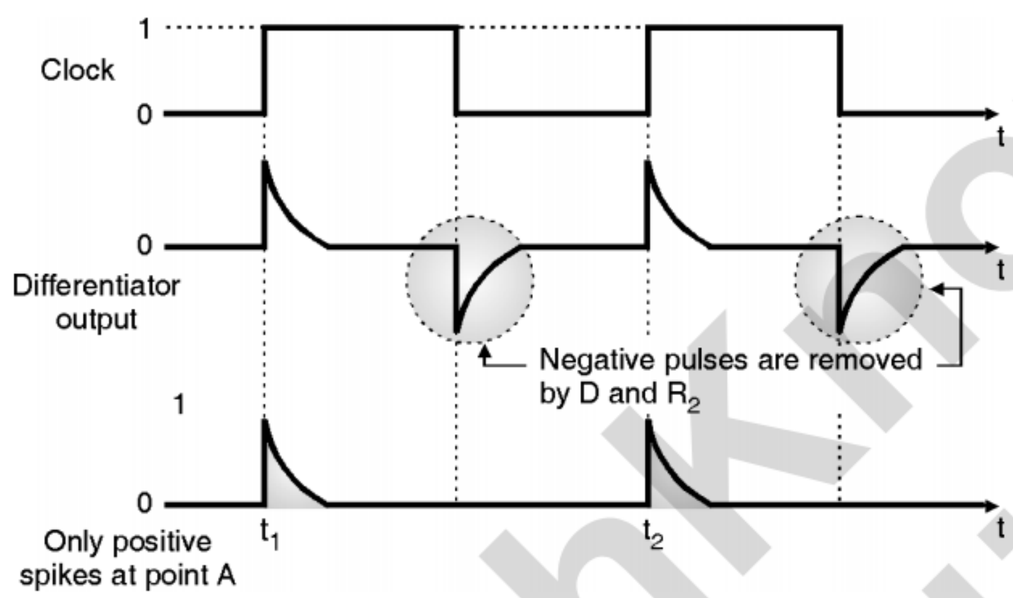


(a) Positive edge triggered D-flip flop



(b) Symbol

(C-592(a)) Fig. 3.9.1



(c) Waveform

(C-592(a)) Fig. 3.9.1

Operation :

- The operation of edge triggered D FF is exactly same as that of the gated D latch discussed earlier with only one difference.
- The D latch had an enable terminal whereas the D flip flop has a clock input.
- The gated D latch of Fig. 3.9.1 is enabled by the positive spikes obtained at point A.
- Hence the outputs will change based on the inputs at these particular instants only. Thus the edge triggered D flip-flop responds only to the positive (leading) edges of the clock pulses.
- At any other instants of time, the D flip flop will not respond to the changes in input.

Truth table :

- Table 3.9.1 shows the truth table of a positive edge triggered D flip flop.

(C-6208) Table 3.9.1 : Truth table of a positive edge triggered D flip flop

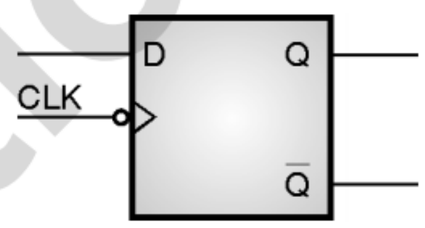
	Inputs		Outputs		State
	CLK	D	Q_{n+1}	$\overline{Q}_{n+1}$	
FF disabled	0	×	Q_n	$\overline{Q}_n$	No change
	1	×	Q_n	$\overline{Q}_n$	No change
FF responds only to the positive edges	↓	×	Q_n	$\overline{Q}_n$	No change
	↑	0	0	1	Q follows D input
	↑	1	1	0	

From the truth table it is clear that the Q output of the flip flop follows the D input.

3.9.2 Negative Edge Triggered D Flip Flop :

Symbol :

- The symbol for negative edge triggered D flip flop is shown in Fig. 3.9.2.



(C-593) Fig. 3.9.2 : Symbol of negative edge triggered D flip flop

- This F/F responds only to the negative edges of the clock pulses. This is how it is different from the positive edge triggered flip flop.
- Otherwise, the operation of the negative edge triggered D flip flop is exactly same as that of the positive edge triggered D flip flop.

Truth Table :

- The truth table of negative edge triggered D flip flop is shown in Table 3.9.2.

(C-6209) Table 3.9.2 : Truth table of negative edge triggered D flipflop

	Inputs		Outputs		Comment
	CLK	D	Q_{n+1}	$\overline{Q}_{n+1}$	
FF disabled	0	×	Q_n	$\overline{Q}_n$	No change
	1	×	Q_n	$\overline{Q}_n$	No change
FF responds only to the negative edges	↑	×	Q_n	$\overline{Q}_n$	No change
	↓	0	0	1	Q output follows D input
	↓	1	1	0	

TechKnowledge Publications

3.9.3 Applications of D Flipflop :

1. As a delay element.
2. In the digital latches.

Transparent latch :

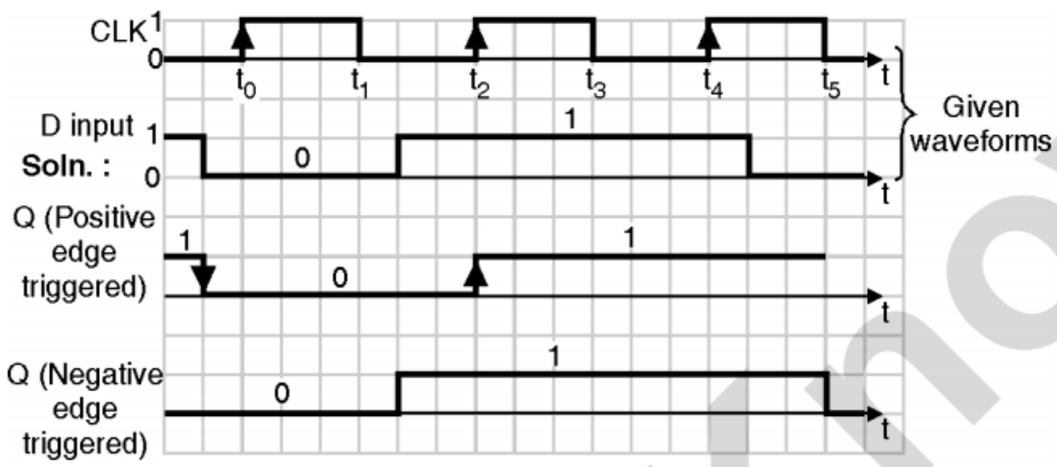
– In level triggered D FF (D latch) the Q output always follows the changes in D input.

– Hence D latch is sometimes called as transparent latch.

Ex. 3.9.1 : Draw the output waveform for a positive and negative edge triggered D flip flop, if the clock and D input waveforms are as shown in Fig. P. 3.9.1.

Soln. :

– The required waveforms are as shown in Fig. P. 3.9.1.



(C-3421) **Fig. P. 3.9.1**

3.10 Edge Triggered J-K Flip Flop :

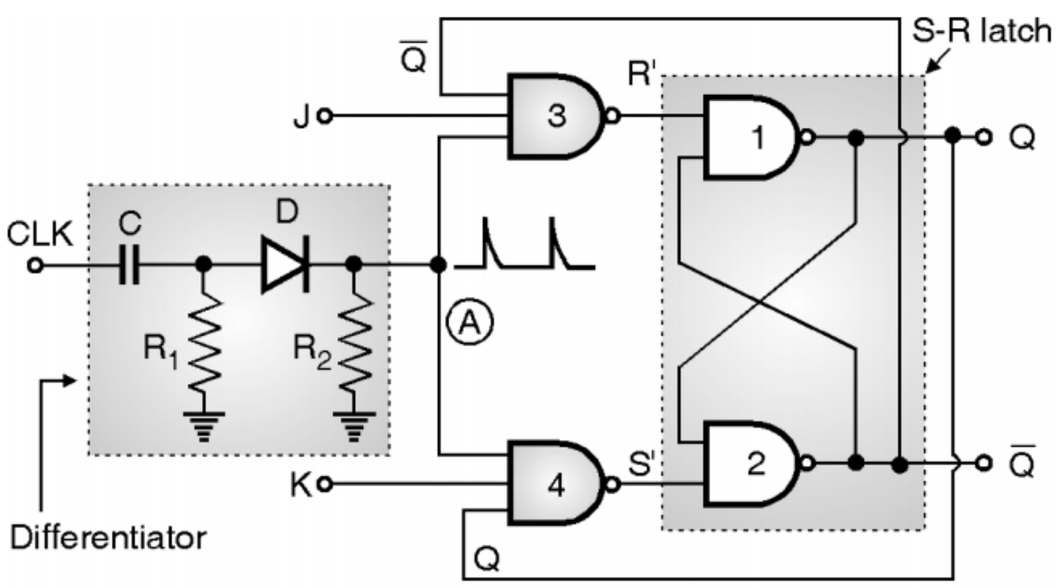
– Edge triggered J-K flip flops are of two types :

1. Positive edge triggered JK flip flop
2. Negative edge triggered JK flip flop

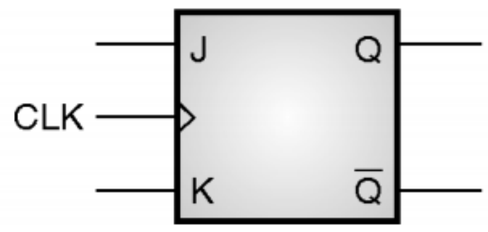
3.10.1 Positive Edge Triggered JK Flip Flop :

Logic diagram :

– The circuit diagram of the positive edge triggered JK flip flop is shown in Fig. 3.10.1(a) and its circuit symbol is shown in Fig. 3.10.1(b).



(C-596) **Fig. 3.10.1(a) : Positive edge triggered JK flip flop**



(C-596) **Fig. 3.10.1(b) : Symbol**

Operation :

– The clock signal is a train of rectangular or square waves.

– It is passed through a differentiator (C – R_1) and a rectifier (D and R_2), to obtain only the positive spikes at point A as already discussed.

– The negative spikes are blocked by the diode.

– NAND gates 1 and 2 form the basic S-R latch. The other two NAND gates (3 and 4) have three inputs each. The outputs Q and $\overline{Q}$ are fed back to the inputs of gates 4 and 3 respectively.

– Referring to Fig. 3.10.1(a) we can write the mathematical expressions for S′ and R′ as follows :

$$S' = \overline{K \cdot Q \cdot CLK} \text{ and } R' = \overline{J \cdot \overline{Q} \cdot CLK}$$

– Let us now understand the operation step by step.

Case I : CLK = 0 or 1 i.e. level

– If CLK = 0 or 1 i.e. level and no pulse this flip flop is disabled, because the output of differentiator is zero for any level input.

– Therefore Q and $\overline{Q}$ output do not change their state.

Case II : If clock edge is falling edge (↓)

– For the falling edge of the clock, the rectifier (D – R_2) will block the negative spike. Hence voltage at point A is logic 0.

– So one input to both the NAND gates 3 and 4 is 0.

– Hence both S′ and R′ will be forced to 1.

– For an SR latch using NAND gates, the outputs will remain unchanged if S = R = 1.

∴ Corresponding to the falling edge of clock, the outputs Q and $\overline{Q}$ will remain unchanged.

Case III : CLK = ↑, J = 0, K = 0 (No change)

– Since J = 0, R′ = 1 and as K = 0, S′ = 1.

– As S′ = R′ = 1 the outputs Q and $\overline{Q}$ will not change their state even though a positive edge of the clock pulse is being applied.

$$\therefore J = 0, K = 0, Q_{n+1} = Q \text{ and } \overline{Q}_{n+1} = \overline{Q}.$$

– No change in output.

TechKnowledge Publications

Case IV : CLK = ↑, J = 0, K = 1 (Reset)

– Recall the expressions for S′ and R′,

$$S' = \overline{K \cdot Q \cdot CLK} \text{ and } R' = \overline{J \cdot \overline{Q} \cdot CLK}$$

– If the previous state of Q and $\overline{Q}$ is Q = 1 and $\overline{Q}$ = 0 then

$$S' = \overline{1 \cdot 1 \cdot 1} = 0 \text{ and } R' = \overline{0 \cdot 0 \cdot 1} = 1$$

– Therefore according to the operation of S-R latch if S′ = 0, R′ = 1 then Q = 0, and $\overline{Q}$ = 1 ·

$$\therefore Q = 0 \text{ and } \overline{Q} = 1$$

– Thus with J = 0, K = 1 and positive going clock, the JK flip flop will **reset**.

– If Q = 0 and $\overline{Q}$ = 1 before the application of clock pulse, then there will not be any change in their state.

Case V : CLK = ↑, J = 1, K = 0 (Set)

– If the previous state of Q and $\overline{Q}$ is Q = 0 and $\overline{Q}$ = 1 then ,

$$S' = \overline{0 \cdot 0 \cdot 1} = 1 \text{ and } R' = \overline{1 \cdot 1 \cdot 1} = 0$$

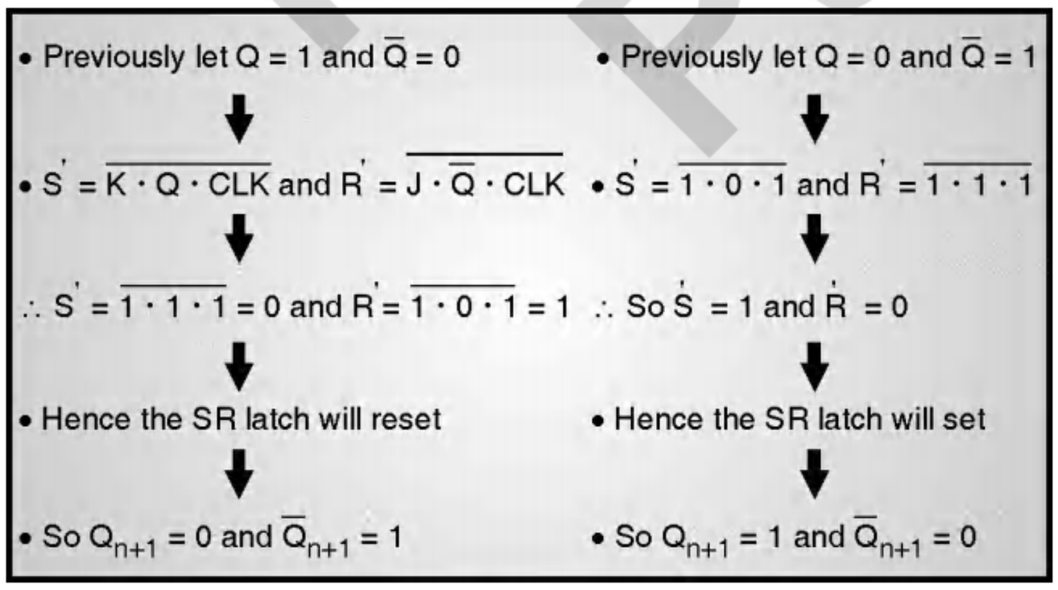
– Hence according to the operation of S-R latch, if S′ = 1 and R′ = 0 then,

$$Q = 1 \text{ and } \overline{Q} = 0 \text{.... i.e. the flip flop is } \textbf{set}.$$

– If Q and $\overline{Q}$ are already 1,0 then there will not be any change in the output state.

Case VI : CLK = ↑, J = 1, K = 1 Toggle mode

– We will discuss this case for two different previous cases.



(C-6399)

– From the operation discussed above, we conclude that when J = K = 1 and a positive clock edge is applied, then Q and $\overline{Q}$ outputs are inverted i.e. $Q_{n+1} = \overline{Q}_n$ and $\overline{Q}_{n+1} = Q_n$.

– This is called as the **TOGGLING** mode. This is a very important mode of operation.

Truth table of JK flip flop :

(C-6837) Table 3.10.1 : Truth table for positive edge triggered JK FF

Case No.	Inputs			Outputs		State
	CLK	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
Case I	0 or 1	x	x	No change	No change	Flip flop is disabled
Case II	↓	x	x	No change	No change	
Case III	↑	0	0	No change	No change	
Case IV	↑	0	1	0	1	Reset
Case V	↑	1	0	1	0	Set
Case VI	↑	1	1	$\overline{Q}_n$	Q_n	Toggle

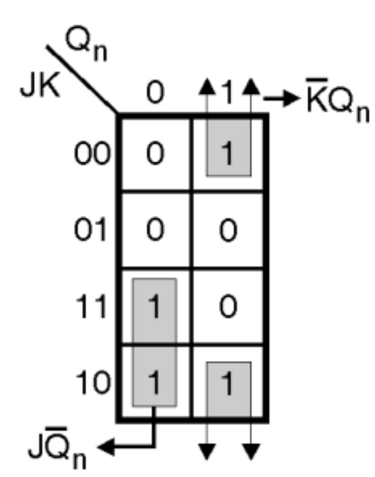
3.10.2 Characteristic Equation of JK Flip Flop :

– Refer to the truth table of JK flip flop (Table 3.10.2) and write the K-map for Q_{n+1} as shown in Fig. 3.10.2.

(C-8082) Table 3.10.2 : Truth table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

For Q_{n+1}



$$\therefore Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$$

(C-597) Fig. 3.10.2 : K-map for Q_{n+1}

∴ The characteristic equation is,

$$Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$$

3.10.3 How does an Edge Triggered JK FF Avoid Race Around Condition ?

SPPU : Dec. 07, Dec. 08, Dec. 09

University Questions

Q. 1 Discuss methods to avoid race around condition in JK flip-flop. **(Dec. 07, 4 Marks)**

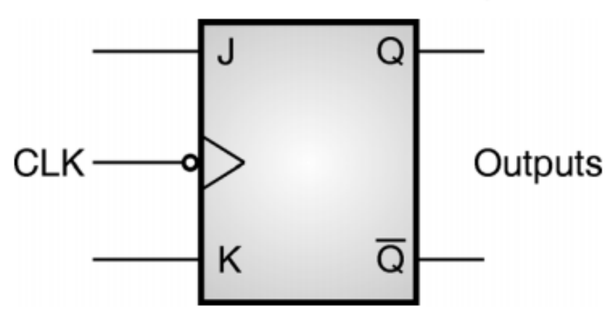
Q. 2 How is race around condition is avoided ? **(Dec. 08, Dec. 09, 3 Marks)**

- For the race around to take place, it is necessary to have the enable input high along with J = K = 1.
- As the enable input remains high for a long time in a JK latch, the problem of multiple toggling arises which is known as Race condition.
- But in edge triggered JK flip flop, the positive clock pulse is present only for a very short time because it is in the form of a narrow differentiated spike.
- Hence by the time the toggled outputs (Q and $\overline{Q}$) return back to the inputs of NAND gates 3 and 4, the positive clock spike has died down to zero. Hence the multiple toggling cannot take place.
- Thus the edge triggering avoids the race around condition.

3.10.4 Negative Edge Triggered JK Flip-Flop :

Symbol :

- The symbol for negative edge triggered JK flip-flop is shown in Fig. 3.10.3.



(C-598) **Fig. 3.10.3 : Symbol of negative edge triggered JK FF**

- This FF responds only to the negative edges of the clock pulses.
- The rest of the operation of this FF is exactly same as that of the positive edge triggered JK FF.

Truth table :

- The truth table of negative edge triggered JK FF is as shown in Table 3.10.3.

(C-6838) **Table 3.10.3 : Truth table of negative edge triggered JK FF**

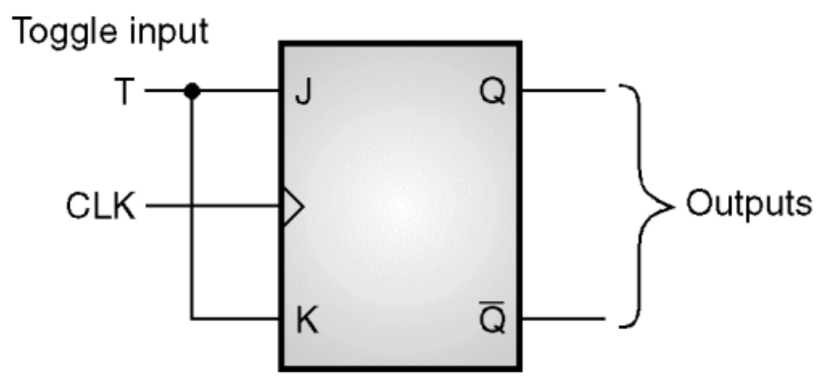
Inputs			Outputs		State
CLK	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
0 or 1	x	x	Q_n	$\overline{Q}_n$	NC (FF is disabled)
↑	x	x	Q_n	$\overline{Q}_n$	
↓	0	0	Q_n	$\overline{Q}_n$	No change
↓	0	1	0	1	Reset
↓	1	0	1	0	Set
↓	1	1	$\overline{Q}_n$	Q_n	Toggle

3.11 Toggle Flip Flop (T Flip Flop) :

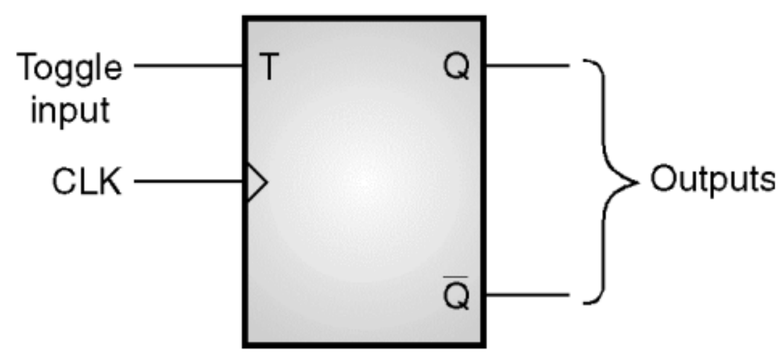
3.11.1 Positive Edge Triggered T-FF :

Symbol :

- Toggle flip flop is basically a JK flip flop with J and K terminals permanently connected together. It has only one input denoted by "T", as shown in Fig. 3.11.1(a).
- The symbol for positive edge triggered T flip flop is shown in Fig. 3.11.1(b) and Table 3.11.1 shows its truth table.



(a) JK FF is converted into T flip flop



(b) Logic symbol of positive edge triggered T flip flop

(C-599) **Fig. 3.11.1**

Truth table :

(C-6839) Table 3.11.1 : Truth table of a Toggle FF (positive edge triggered)

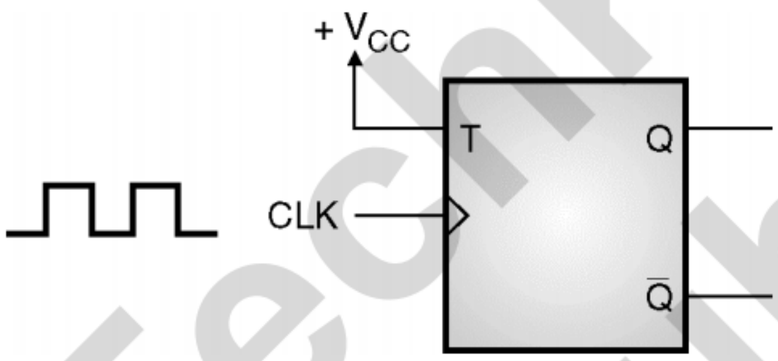
Inputs		Outputs		State
CLK	T	Q_{n+1}	$\bar{Q}_{n+1}$	
↑	0	Q_n	$\bar{Q}_n$	No change
↓	x	Q_n	$\bar{Q}_n$	No change
1	x	Q_n	$\bar{Q}_n$	No change
0	x	Q_n	$\bar{Q}_n$	No change
↑	1	$\bar{Q}_n$	Q_n	Toggle

Operation :

- When T = 0, J = K = 0. Hence the outputs Q and $\bar{Q}$ won't change.
- But if T = 1, then J = K = 1 and the outputs will toggle corresponding to every leading edge of clock signal.
- This has been illustrated in Ex. 3.11.1.

Ex. 3.11.1 : Determine the output Q for the set up shown in Fig. P. 3.11.1(a).

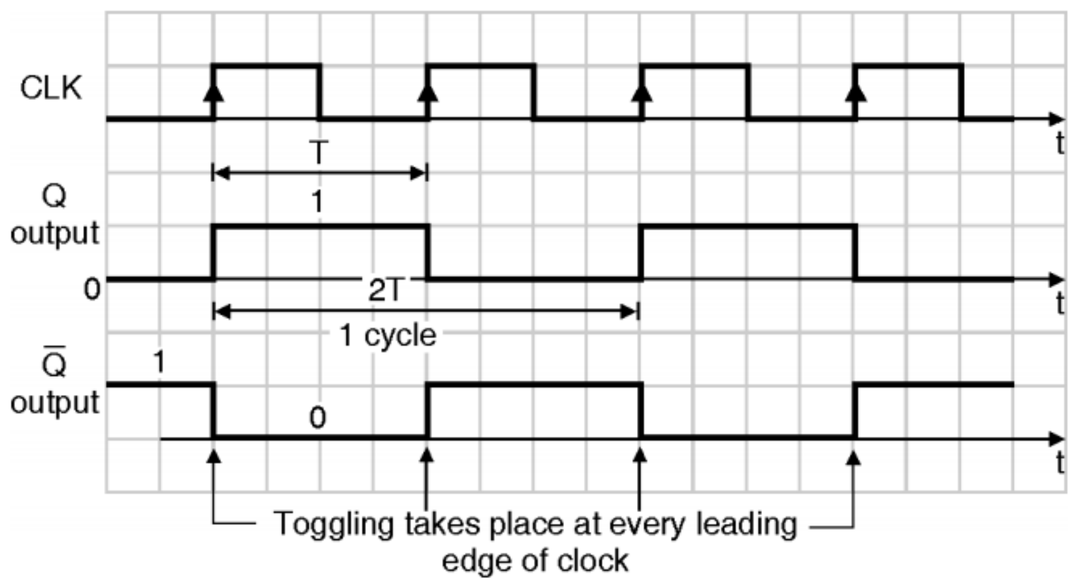
What is the relation between the frequency of clock and that of Q output ?



(C-600) Fig. P. 3.11.1(a) : Given set up

Soln. :

- The flip flop of Fig. P. 3.11.1(a) is a toggle flip flop with T = 1. It is a positive edge triggered flip flop.
- Hence the outputs will toggle in response to each rising edge of the clock, as shown in Fig. P. 3.11.1(b).



(C-601) Fig. P. 3.11.1(b) : Waveforms for a T F/F

Relation between input and output frequency :

- Referring to Fig. P. 3.11.1(b),

1 cycle period of CLK signal = T,

$\therefore$ Clock frequency $f_{CLK} = \frac{1}{T}$

1 cycle period of Q output = 2 T,

$\therefore$ Output frequency $f_o = \frac{1}{2T}$

But $(\frac{1}{T}) = f_{CLK}$

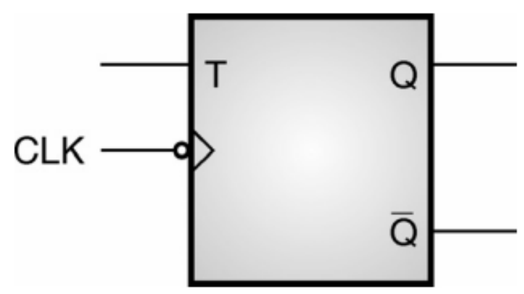
$\therefore$ Output frequency $f_o = \frac{f_{CLK}}{2}$ **...Ans.**

> The T flip flop divides the clock frequency by 2. Hence a T flip flop can be used as a frequency divider.

3.11.2 Negative Edge Triggered T Flip Flop :

Symbol :

- Fig. 3.11.2 shows the logic symbol of a negative edge triggered toggle flip flop and Table 3.11.2 gives its truth table.



(C-602) Fig. 3.11.2 : Logic symbol of a negative edge triggered toggle flip flop

Truth table :

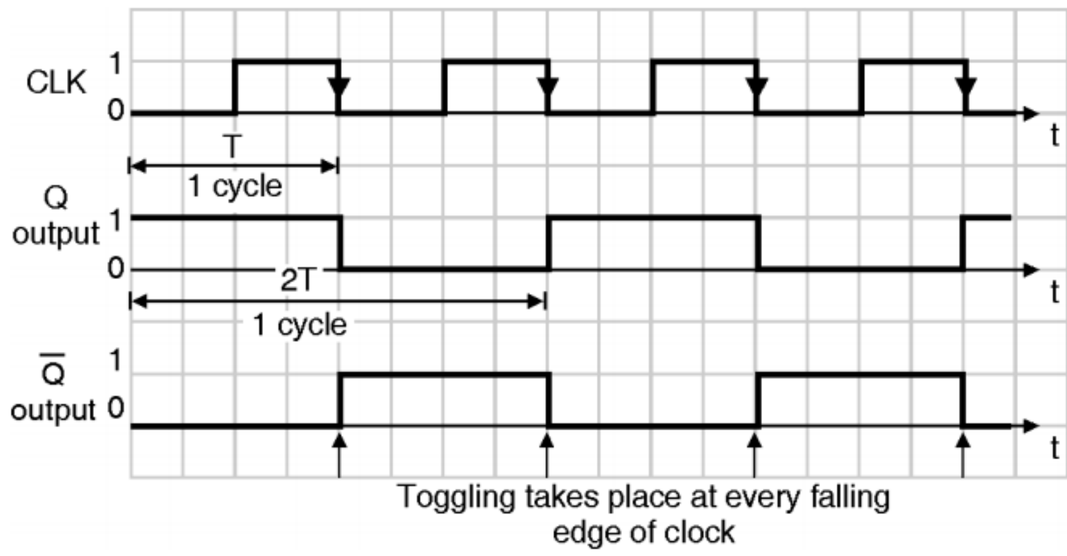
(C-6840) Table 3.11.2 : Truth table

Inputs		Outputs		State
T	CLK	Q_{n+1}	$\bar{Q}_{n+1}$	
0	x	Q_n	$\bar{Q}_n$	No change
1	↑	Q_n	$\bar{Q}_n$	No change
1	0	Q_n	$\bar{Q}_n$	No change
1	1	Q_n	$\bar{Q}_n$	No change
1	↓	$\bar{Q}_n$	Q_n	Toggle

Waveforms :

- The waveforms for negative edge triggered toggle FF are shown in Fig. 3.11.3.

TechKnowledge Publications



(C-603) Fig. 3.11.3 : Waveforms for negative edge triggered T FF

- The toggle input T = 1. The outputs Q and $\overline{Q}$ toggle corresponding to every falling edge of clock.
- This F/F also, acts as a frequency divider. The frequency of Q and $\overline{Q}$ outputs is half the frequency of clock input.

3.11.3 Application of T F/F :

- The T F/F acts as the basic building block of a ripple counter.

3.12 Master Slave (MS) JK Flip Flop :

SPPU : Dec. 06, Dec. 10

University Questions

Q. 1 What do you mean by Master-Slave JK Flip-flop ? Explain the advantage of this Flip-flop. Draw suitable circuit diagram and timing diagram.

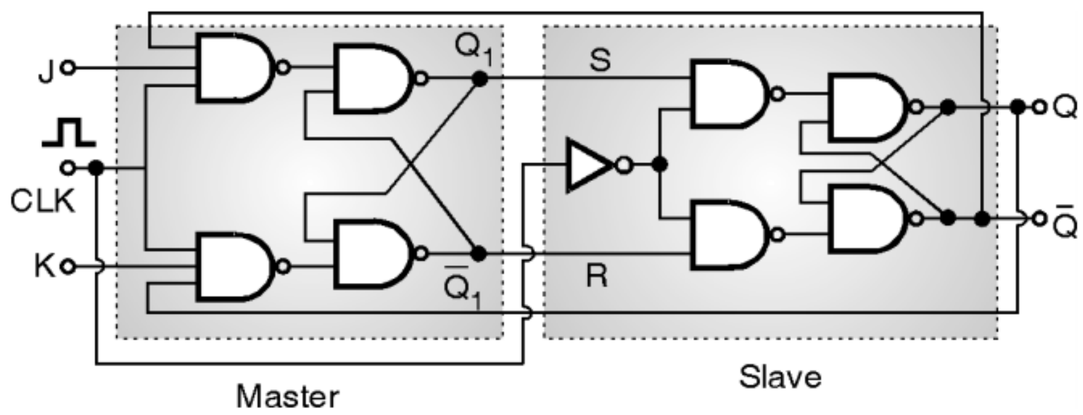
(Dec. 06, 10 Marks)

Q. 2 What is the advantage of M-S flip-flop ? Explain working of MS J-K flip-flop in detail.

(Dec. 10, 8 Marks)

Logic diagram :

- Fig. 3.12.1 shows the master slave JK flip flop.
- It is a combination of a clocked JK latch (level triggered JK FF) and clocked SR latch (level triggered SRFF).
- The clocked JK latch acts as the master and the clocked SR latch acts as the slave.



(C-606) Fig. 3.12.1 : Master slave JK FF

- Master is positive level triggered. But due to the presence of the inverter in the clock line, the slave will respond to the negative level.
- Thus both master and slave circuits are level triggered circuits.
- Hence when the clock = 1 (positive level) the master is active and the slave is inactive. Whereas when clock = 0 (low level) the slave is active and the master is inactive.

Truth table :

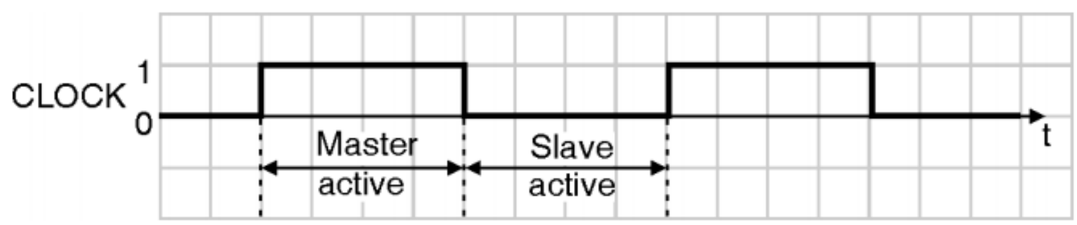
- Table 3.12.1 gives truth table of master slave JK flip flop.

(C-6267) Table 3.12.1 : Truth table of master slave JK FF

Case	Inputs			Outputs		Remark
	CLK	J	K	Q_{n+1}	$\overline{Q}_{n+1}$	
I	X	0	0	Q_n	$\overline{Q}_n$	No change
II	⎍(1)	0	0	Q_n	$\overline{Q}_n$	No change
III	⎍(1)	0	1	0	1	Reset
IV	⎍(1)	1	0	1	0	Set
V	⎍(1)	1	1	$\overline{Q}_n$	Q_n	Toggle

Operation :

- We will discuss the operation of the master slave JK FF with reference to its truth table.
- We must always remember one important thing that in the positive half cycle of the clock, the master is active and in the negative half cycle, the slave is active.
- This is shown in Fig. 3.12.2.



(C-607) Fig. 3.12.2

Case I : Clock = ×, J = K = 0 (No change)

- For clock = 1, the master is active, slave inactive. As J = K = 0. ∴ Outputs of master i.e. Q_1 and $\overline{Q}_1$ will not change. Hence the S and R inputs to the slave will remain unchanged.
- As soon as clock = 0, the slave becomes active and master is inactive.

TechKnowledge Publications

– But since the S and R inputs have not changed, the slave outputs will also remain unchanged.

∴ The outputs will not change if J = K = 0.

Case II : Clock = ⎍, J = K = 0 (No change) **(C-6395)**

This condition has been already discussed in case I.

Case III : Clock = ⎍, J = 0 and K = 1 (Reset) **(C-6395(a))**

– Clock = 1 : Master active, slave inactive.

∴ Outputs of the master become $Q_1 = 0$ and $\overline{Q}_1 = 1$. That means S = 0 and R = 1.

– Clock = 0 : Slave active, master inactive

∴ Outputs of the slave become Q = 0 and $\overline{Q} = 1$. This is the RESET operation.

– Again if clock = 1 : Master active, slave inactive.

∴ Even with the changed outputs Q = 0 and $\overline{Q} = 1$ fed back to master, its outputs will be $Q_1 = 0$ and $\overline{Q}_1 = 1$. That means S = 0 and R = 1.

– Hence with clock = 0 and slave becoming active, the outputs of slave will remain Q = 0 and $\overline{Q} = 1$.

– Thus we get a stable output from the Master slave.

Case IV : Clock = ⎍, J = 1 and K = 0 (Set) **(C-6395(b))**

– Clock = 1 : Master active, slave inactive.

∴ Outputs of master become $Q_1 = 1$ and $\overline{Q}_1 = 0$ i.e. S = 1, R = 0.

– Clock = 0 : Master inactive, slave active.

∴ Outputs of slave become Q = 1 and $\overline{Q} = 0$.

– Again if clock = 1 then it can be shown that the outputs of the slave are stabilized to Q = 1 and $\overline{Q} = 0$.

Case V : Clock = ⎍, J = 1 and K = 1 (Toggle without race) **(C-6395(c))**

– Clock = 1 : Master active, slave inactive.

∴ Outputs of master will toggle. So S and R also will be inverted.

– Clock = 0 : Master inactive, slave active.

∴ Outputs of the slave will toggle.

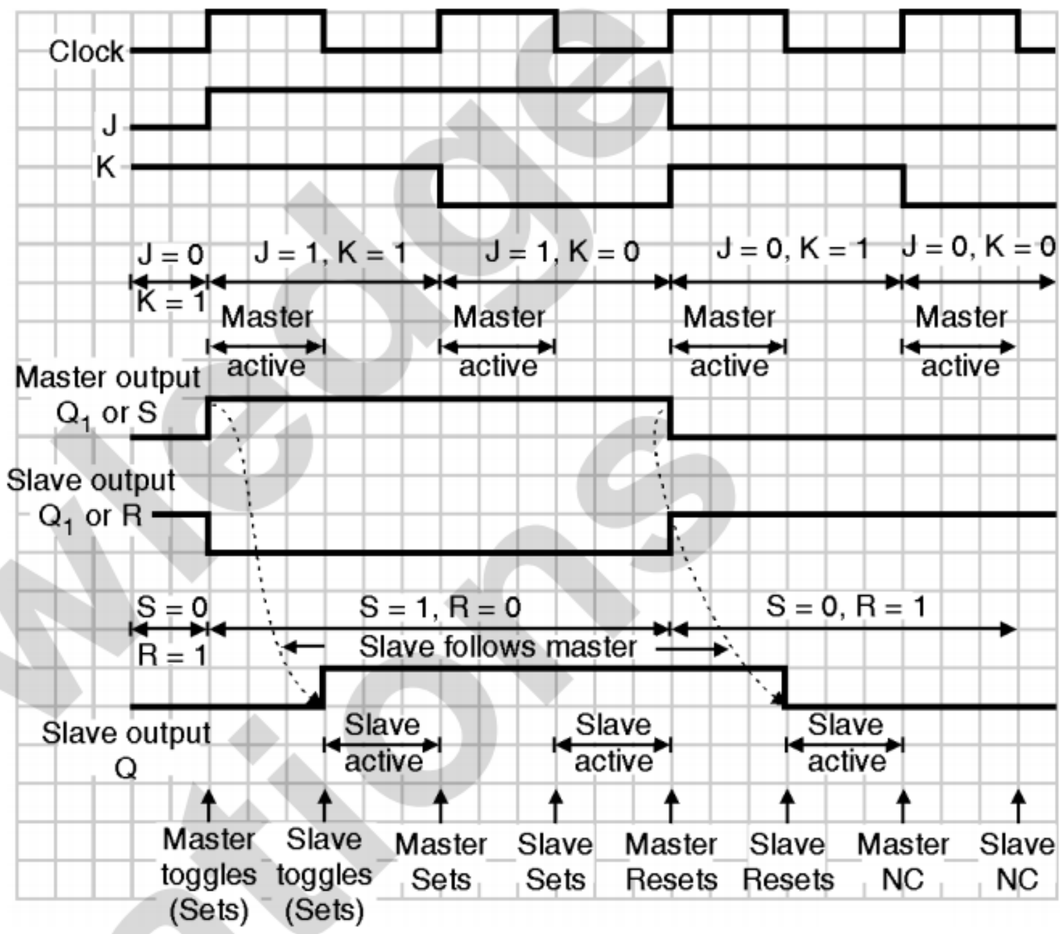
– These changed output are returned back to the master inputs.

– But since clock = 0, the master is still inactive. So it does not respond to these changed outputs.

– This avoids the multiple toggling which leads to the race around condition.

– Thus the master slave flip flop will avoid the race around condition.

– The waveforms for the master slave flipflop are shown in Fig. 3.12.3.



(C-608) **Fig. 3.12.3 : Waveforms of master slave JK flip flop**

Observations from the waveforms :

– We can make the following important observations from the waveforms of the master slave JK FF :

1. The slave always follows the master, after a delay of half clock cycle period.
2. The multiple toggling or the race around condition is successfully avoided.

3.13 Preset and Clear Inputs :

– In the flip flops discussed so far when the power switch is turned on, the state of outputs Q and $\overline{Q}$ can be anything that means either Q = 0, $\overline{Q} = 1$ or Q = 1, $\overline{Q} = 0$. It is not predictable.

– But this uncertainty cannot be tolerated in certain applications.

– In some applications it is necessary to initially set or reset the flip flops precisely.

– This can be practically achieved by adding two more inputs to a flip flop, called **preset (PR)** and **clear (CLR)** inputs.

Tech Knowledge Publications

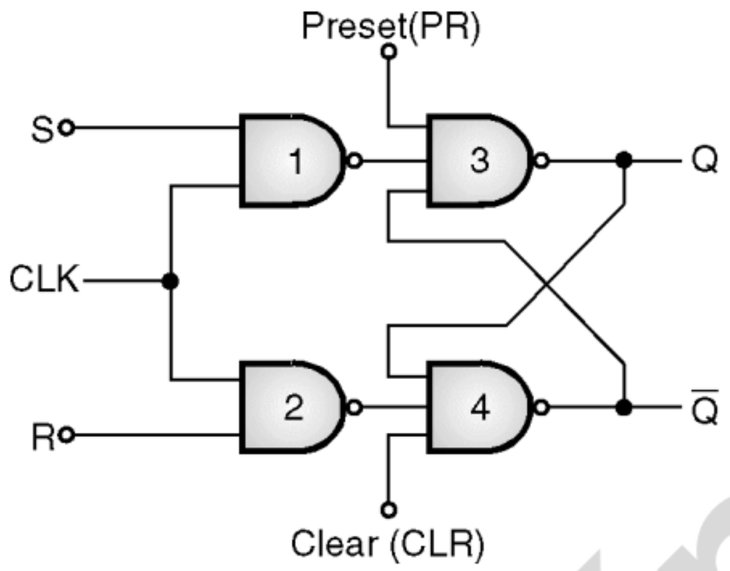
– These inputs are called as **direct** or **asynchronous inputs** because we can apply them any time between clock pulses without thinking about their synchronization with the clock.

– That means applying PR or CLR inputs is not related to the clock in any way.

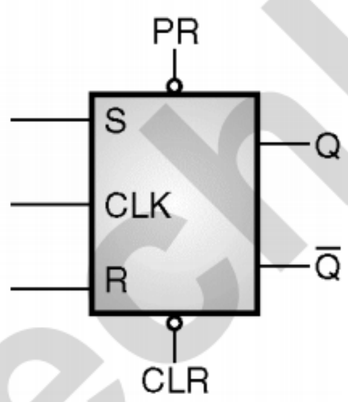
3.13.1 S-R Flip-Flop with Preset and Clear Inputs :

Logic diagram and symbol :

– The S-R flipflop with preset and clear is shown in Fig. 3.13.1(a) and its symbol is shown in Fig. 3.13.1(b).



(a) S-R FF with preset and clear



(b) Symbol

(C-609) Fig. 3.13.1

– Note that both these inputs are active low inputs. This is indicated by the bubbles on these inputs in Fig. 3.13.1(b).

Operation :

Case I : PR = CLR = 1

If PR = CLR = 1 then both are inactive and the S-R FF operates as per the truth table of the conventional SR flip-flop.

Case II : PR = 0 and CLR = 1 (Preset condition)

– As PR = 0, the output of gate 3 of Fig. 3.13.1(a) will be 1.

– That means Q = 1.

– Therefore all the three inputs to the gate 4 will be 1 and $\overline{Q}$ will become 0.

– Thus making PR = 0 will set the flip flop. Note that with PR = 0 and CLR = 1 the flipflop is set irrespective of the status of S, R or clock inputs.

– Therefore it is said that the PR input has higher priority than S, R or CLK inputs.

Case III : PR = 1, CLR = 0 (Clear or Reset condition)

– If PR = 1 and CLR = 0, then the output of gate-4 i.e. $\overline{Q}$ = 1.

– Therefore all the three inputs to gate-3 will be 1 and hence Q = 0.

– Thus making CLR = 0 will reset the flip flop. Note that with CLR = 0 and PR = 1, the FF is reset irrespective of the status of S, R or clock inputs.

– Therefore it is said that CLR has higher priority than S, R or CLK inputs.

Case IV : PR = 0, CLR = 0

– This condition should never be used, because it leads to an uncertain state.

– The operation of SR flip flop with preset and clear inputs is summarized in Table 3.13.1.

– The don't care conditions (×) marked in the CLK column indicate that PR and CLR inputs have higher priority than clock input.

(C-609(a)) Table 3.13.1 : Summary of operation of SR FF

Inputs			Output	Operation performed
CLK	PR	CLR		
1	1	1	Q_{n+1}	Normal S-R flipflop
×	0	1	1	FF is set
×	1	0	0	FF is reset

3.13.2 Synchronous Preset and Clear Inputs :

– Preset and clear inputs can be of two types :

1. Asynchronous with clock.
2. Synchronous with clock.

– We have already discussed the asynchronous preset and clear inputs.

– The operation of synchronous preset and clear inputs can be understood from the truth table given in Table 3.13.2.

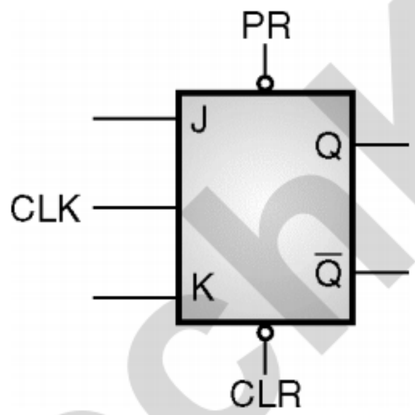
Tech Knowledge Publications

(C-8075) Table 3.13.2 : Truth table for S-R FF with synchronous preset and clear

Inputs					Outputs	
P_r	C_r	CLK	S	R	Q_{n+1}	$\overline{Q}_{n+1}$
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	1	NC	NC
1	1	1	0	1	0	1
1	1	1	1	0	1	0
1	1	1	1	1	RACE	RACE

3.13.3 JK Flip Flop with Preset and Clear Inputs :

- A JK flip flop with preset and clear inputs is shown in Fig. 3.13.2. These are the synchronous preset and clear terminals.
- Both these inputs are active low and have higher priority than all the other inputs.



(C-610) (a) Symbol of a JK FF with preset and clear inputs

Inputs			Output	Operation performed
CLK	PR	CLR		
1	1	1	Q_{n+1}	Normal JK flipflop
×	0	1	1	FF is set
×	1	0	0	FF is reset

(C-610(a))(b) Summary of operation of JK FF

Fig. 3.13.2

3.13.4 Applications of JK Flip Flop :

1. Shift register.
2. Counters.

3.14 Various Representations of Flip Flops :

- There are different ways in which a flip flop can be represented.
- Each representation is suitable for a different application.
- Various representations of flip flops are :
  1. Characteristic equations.
  2. Flip flop as Finite State Machine (FSM).
  3. Flip flop excitation tables.

3.14.1 Characteristic Equations :

- We have already introduced the concept of characteristic equations earlier in this chapter and we have written the characteristic equations for various flip flops.

3.15 Excitation Table of Flip-Flop :

- Till now we have written the truth tables of various flip flops.
- The truth tables are also known as the **characteristic tables.**
- But while designing the sequential circuits, sometimes the present and next state of a circuit are given and we are expected to find the corresponding input condition.
- We need to use the **excitation tables** of flipflops to do this. These tables are different from the characteristic tables.
- Present state is the state prior to application of clock pulse and the next state means the state after application of clock pulse.

3.15.1 Excitation Table of SR Flip Flops :

SPPU : Dec. 13

University Questions

Q. 1 Draw the excitation table of S-R flip-flop.

(Dec. 13, 2 Marks)

- For example the outputs of an S-R FF before clock pulse are $Q_n = 0$ and $\overline{Q}_n = 1$ and it is expected that these outputs should remain unchanged after application of clock. Then what must be the values of inputs S_n and R_n to achieve this ?
- Refer to the truth table of SR FF to answer this question.
- The answer is, for the following two conditions the outputs remain unchanged at Q = 0 and $\overline{Q}$ = 1.

TechKnowledge Publications

(C-6571) **Table 3.15.1 : Truth table of SR flip flop**

	CLK	S_n	R_n	Q_{n+1}	$\overline{Q}_{n+1}$
Row 1	↑	0	0	Q_n	$\overline{Q}_n$
Row 2	↑	0	1	0	1
	↑	1	0	1	0
	↑	1	1	Race	Race

Condition 1 :

S_n and R_n = 0 → Refer first row of Table 3.15.1.

Condition 2 :

S_n = 0 R_n = 1 → Refer second row of Table 3.15.1.

– From the two conditions mentioned above we conclude that S_n input should be equal to 0 and R_n input can be 0 or 1 (don't care) if we want to maintain Q = 0 and $\overline{Q}$ = 1 before and after clock.

– Similarly we can find the input conditions (Values of S and R inputs) for all the possible situations that may exist on the output side.

– The table containing all these output situations and the corresponding input conditions is called as the "excitation table" of a flip flop.

– The excitation table of SR flip flop is shown in Table 3.15.2.

C-6581) **Table 3.15.2 : Excitation table of SR flip flop**

	Present state of Q output	Next state of Q output	S_n input	R_n input
Case I	0	0	0	X
Case II	0	1	1	0
Case III	1	0	0	1
Case IV	1	1	X	0

Description of excitation table of SR FF :

We have already discussed case I.

Case II : Q should change from 0 to 1

– This is nothing but the set condition.

– Hence S_n = 1 and R_n = 0 should be the inputs.

Case III : Q should change from 1 to 0

– This is nothing but the reset condition.

– Hence S_n should be 0 and R_n should be 1.

Case IV : Q should be 1. No change

To satisfy this requirement, we have two possible input conditions :

Condition 1 : S_n = R_n = 0 i.e. No change in output.

Condition 2 : S_n = 1 and R_n = 0.

– From these conditions we conclude that S_n can be either 0 or 1 i.e. don't care and R_n = 0.

– Hence the inputs corresponding to this case is S_n = × and R_n = 0.

– Similarly we can write the excitation tables for the other flip flops.

3.15.2 Excitation Table of D Flip Flop :

– Excitation table of a D flip flop is given by Table 3.15.3.

(C-7828) **Table 3.15.3 : Excitation table of a D flip flop**

Q output		Input
Present state	Next state	D_n
0	0	0
0	1	1
1	0	0
1	1	1

3.15.3 Excitation Table of JK Flip Flop :

SPPU : May 15

University Questions

Q. 1 Draw the excitation table of J-K flip flop.

(May 15, 2 Marks)

– Excitation table of a JK flip flop is given by Table 3.15.4.

(C-6211) **Table 3.15.4 : Excitation table of JK flip flop**

	Q output		Inputs	
	Present state	Next state	J_n	K_n
Case I	0	0	0	×
Case II	0	1	1	×
Case III	1	0	×	1
Case IV	1	1	×	0

Don't care conditions result due to toggling

– Refer to Case II of Table 3.15.4. For the change in Q output from 0 to 1 we have the following two input conditions :

Condition I :

J_n = 1 and K_n = 0 i.e. set condition.

Condition II :

J_n = 1 and K_n = 1 i.e. toggle.

Tech Knowledge Publications

– Hence J_n = 1 and K_n = × (0 or 1) corresponding to case II.

– Similarly for case III also, the don't care condition corresponds to toggle mode.

3.15.4 Excitation Table of T Flip Flop :

– Excitation table of T flip flop is shown in Table 3.15.5.

(C-7829) Table 3.15.5 : Excitation table of T flip-flop

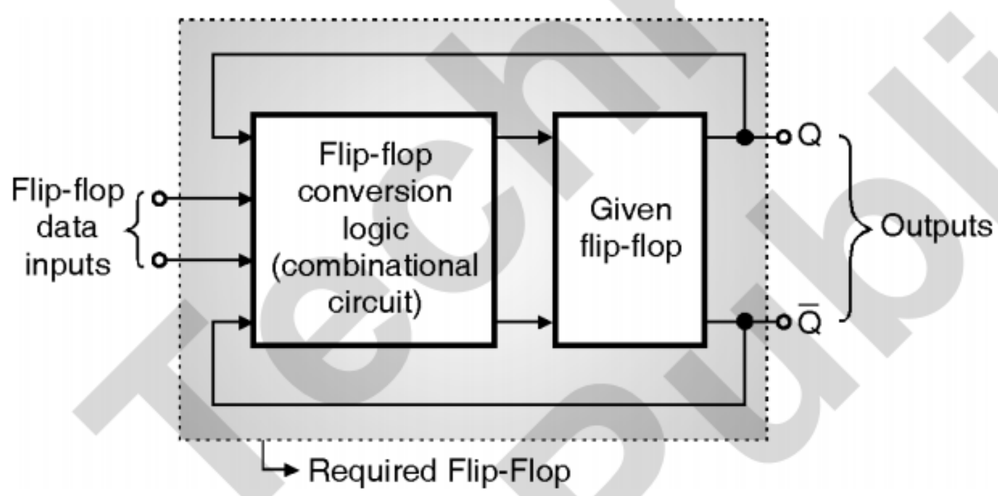
Q output		Input
Present state	Next state	T_n
0	0	0
0	1	1
1	0	1
1	1	0

3.16 Conversion of Flip Flops :

Concept :

– The conversion from one type of flip flop to the other (say SR FF to JKFF), needs a systematic approach using the excitation tables and K map simplifications.

– Fig. 3.16.1 shows a generalized model for conversion from one flip flop to the other.



(C-611) Fig. 3.16.1 : General model used to convert one type of FF to the other

– As shown in Fig. 3.16.1 the required flip flop is actually a combination of the given flip flip and a combinational logic circuit using gates. (Such a combinational circuit is called as the flip flop conversion logic).

– The inputs to FF conversion logic are, the flip flop data inputs and the outputs of given flip-flop i.e. the given FF and the desired FF.

– The conversion logic is designed by combining the excitation tables of both the flip flops.

– The truth table of the conversion logic has data inputs and Q and $\overline{Q}$ outputs of the given FF as inputs whereas the inputs of the given FF are the outputs of the truth table.

– Then we draw the K map for each output and obtain the simplified expressions. The conversion logic is then implemented using gates.

3.16.1 Conversion from S-R Flip Flop to D Flip Flop : SPPU : May 12, Dec. 16

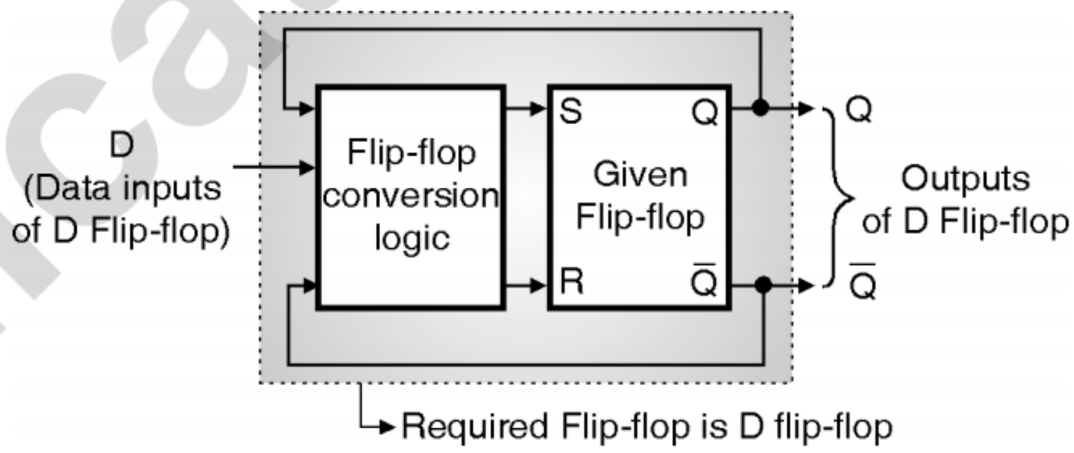
University Questions

Q. 1 Convert the basic SR-Flip-Flop (SR-FF) into D FF. **(May 12, Dec. 16, 2 Marks)**

– Refer Fig. 3.16.2. Here the given FF is SR FF and the required FF is D FF.

– The truth table for the conversion logic is shown in Table 3.16.1. The inputs are D and Q whereas outputs are S and R.

– The truth table is prepared by combining the excitation tables of D F/F and SR FF.



(C-612) Fig. 3.16.2

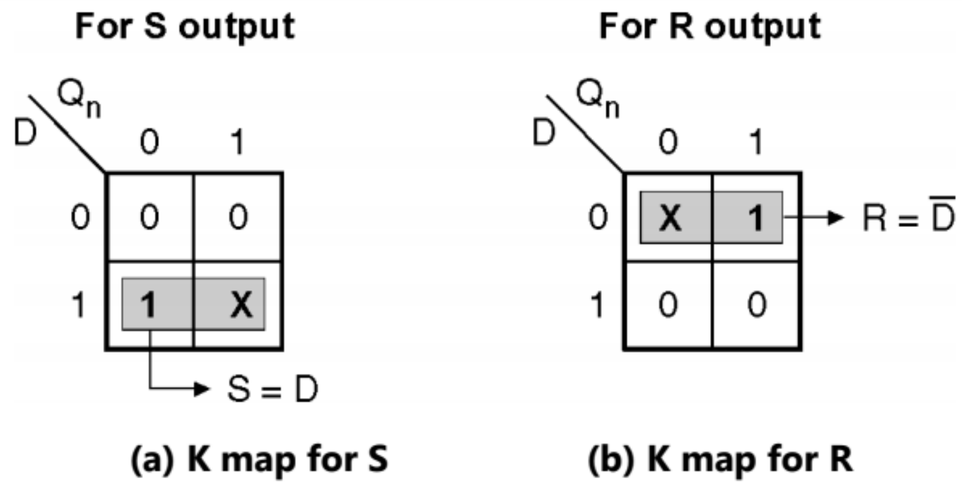
(C-6387) Table 3.16.1 : Truth table for SR to D FF conversion

Inputs			Outputs	
D	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	X
1	0	1	1	0
0	1	0	0	1
1	1	1	X	0

← Entries from excitation table of D FF →

← Entries from excitation table of SR FF →

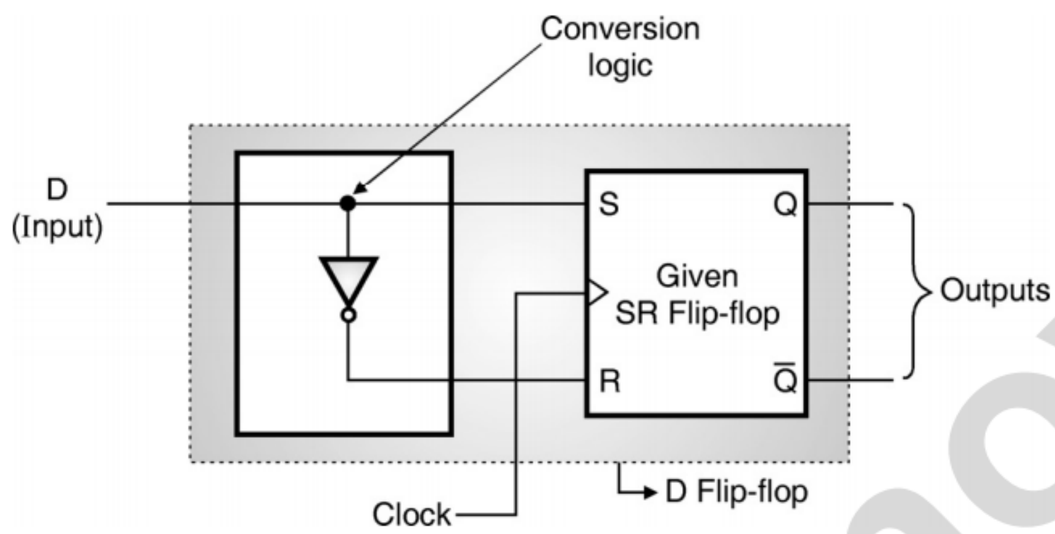
– Now write the K maps for the S and R outputs as shown in Figs. 3.16.3(a) and (b).



(C-613) **Fig. 3.16.3**

Logic diagram :

– The logic diagram for SR FF to D FF is shown in Fig. 3.16.4.



(C-614) **Fig. 3.16.4 : SR flip flop to D flip flop**

3.16.2 Conversion of JK FF to T FF :

SPPU : May 06, Dec. 09, Dec. 16, Dec. 19

University Questions

Q. 1 Convert J-K flip-flop into T-FF. Show the truth table.

(May 06, Dec. 09, Dec. 16, Dec. 19, 4 Marks)

Step 1 : Write the truth table for conversion :

– The required truth table is obtained from the excitation tables of JK and T flip flops as follows :

(C-6388) **Table 3.16.2 : Truth table for JK to T FF conversion**

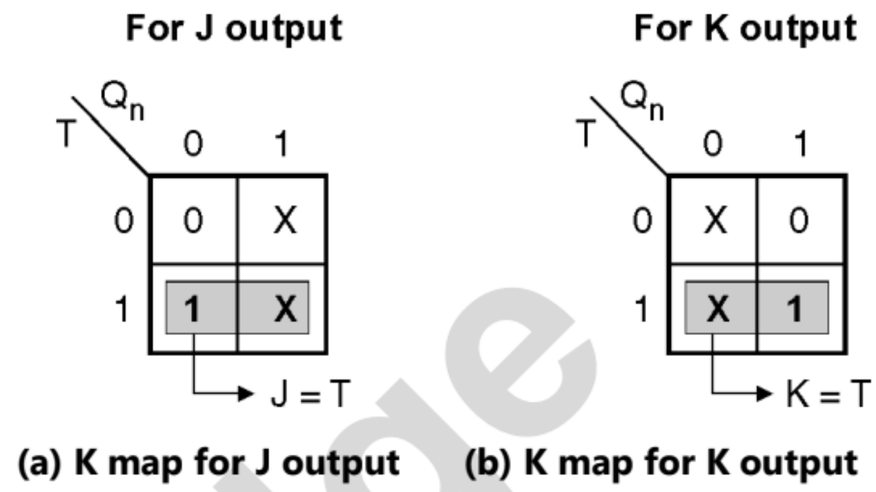
Inputs			Outputs	
T	Present state of Q	Next state of Q	J	K
0	0	0	0	X
1	0	1	1	X
1	1	0	X	1
0	1	1	X	0

← Excitation table of T FF →

← Excitation table of JK FF →

Step 2 : K maps and simplification :

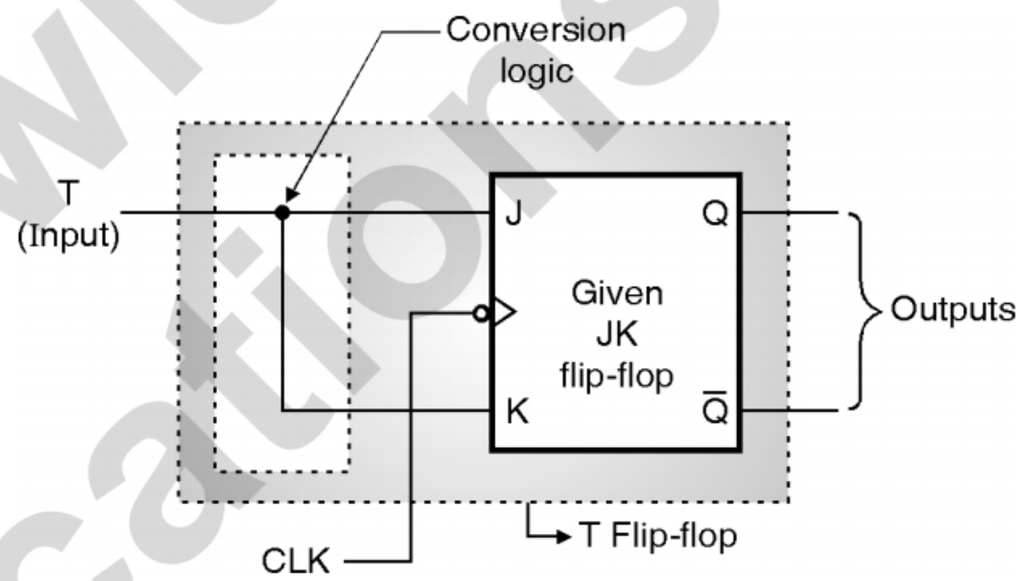
– The K maps for outputs J and K are shown in Fig. 3.16.5.



(C-615) **Fig. 3.16.5**

Step 3 : Draw the logic diagram :

– The logic diagram is shown in Fig. 3.16.6.



(C-616) **Fig. 3.16.6 : Logic diagram for conversion of JK FF to T FF**

3.16.3 SR Flip Flop to T Flip Flop :

SPPU : May 12

University Questions

Q. 1 Convert the basic SR-Flip-Flop (SR-FF) into T-FF.

(May 12, 2 Marks)

– The stepwise conversion process is as follows :

Step 1 : Write the truth table :

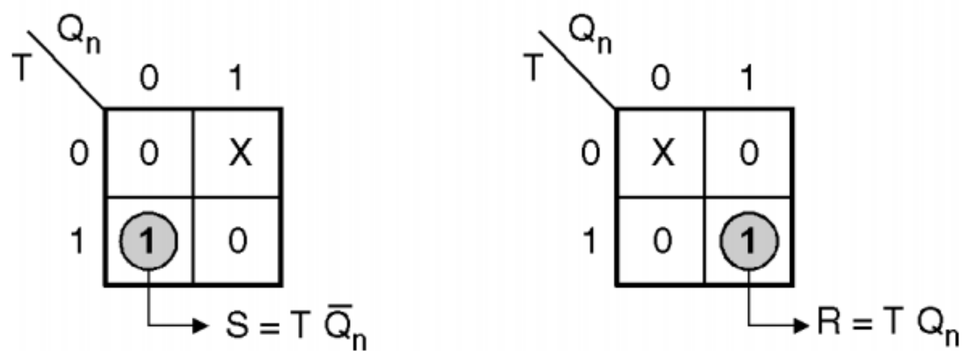
(C-8069) **Table 3.16.3 : Truth table for SR FF to T FF**

Inputs			Outputs	
T	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	X
1	0	1	1	0
1	1	0	0	1
0	1	1	X	0

Tech Knowledge Publications

Step 2 : Write the K maps and obtain the expressions for S and R :

For S output **For R output**

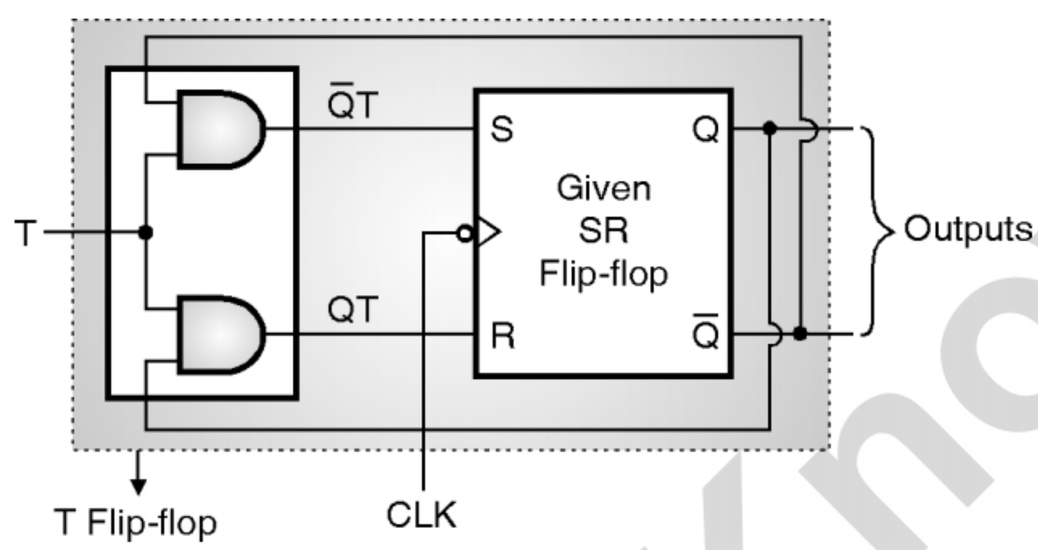


(a) K map for S **(b) K map for R**

(C-617) **Fig. 3.16.7**

Step 3 : Draw the logic diagram :

– The logic diagram is shown in Fig. 3.16.8.



(C-618) **Fig. 3.16.8 : Conversion from SR flip flop to T flip flop**

3.16.4 SR Flip Flop to JK Flip Flop :

SPPU : May 12

University Questions

Q. 1 Convert the basic SR-Flip-Flop (SR-FF) into JK-FF. **(May 12, 2 Marks)**

Step 1 : Write the truth table for SR to JK :

– The truth table for SR to JK flip flop conversion is shown in Table 3.16.4.

(C-6389) **Table 3.16.4 : Truth table for SR to JK FF conversion**

Inputs				Outputs	
J	K	Present state Q_n	Next state Q_{n+1}	S	R
0	0	0	0	0	X
0	1	0	0	0	X
1	0	0	1	1	0
1	1	0	1	1	0
0	1	1	0	0	1
1	1	1	0	0	1
0	0	1	1	X	0
1	0	1	1	X	0

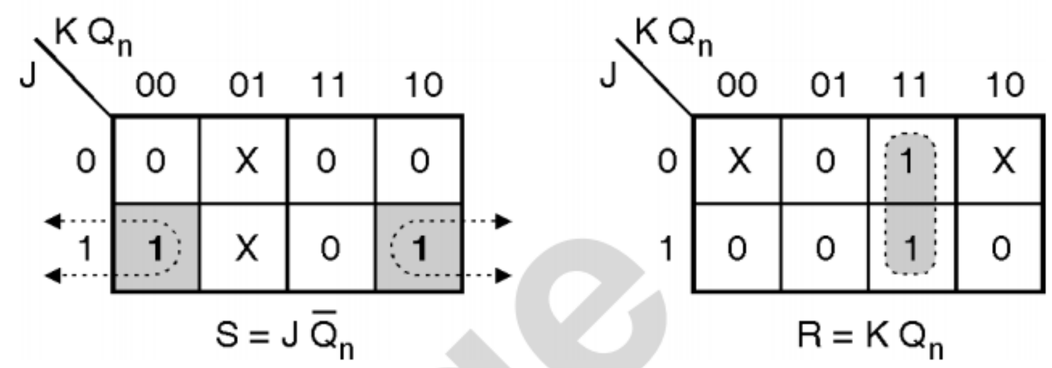
← Excitation table of JK FF →

← Excitation table of SR FF →

Step 2 : K maps and simplification :

– K maps for S and R outputs are shown in Fig. 3.16.9.

For output S **For output R**

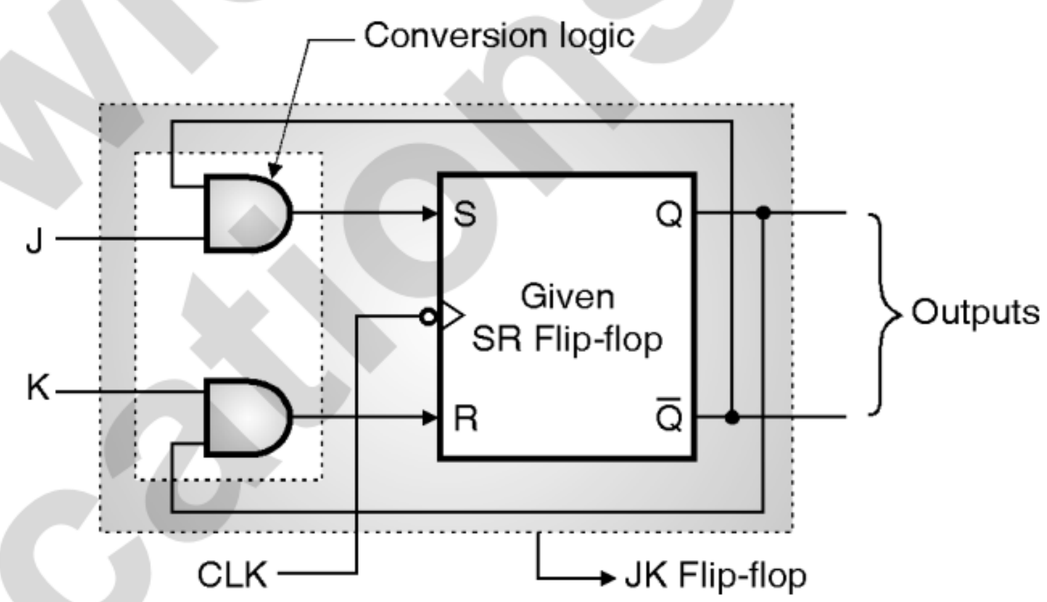


(a) K-map for S output **(b) K-map for R output**

(C-619) **Fig. 3.16.9**

Step 3 : Logic diagram :

– The logic diagram of SR to JK flip flop is given in Fig. 3.16.10.



(C-620) **Fig. 3.16.10 : SR to JK flip flop conversion**

3.16.5 Conversion of D Flip Flop to T Flip Flop :

SPPU : Dec. 17

University Questions

Q. 1 Convert the following flip-flop : D flip-flop to T flip-flop. **(Dec. 17, 2 Marks)**

Step 1 : Write the truth table for D to T FF conversion :

– The truth table is as follows :

(C-8044) **Table 3.16.5 : Truth table for D to T FF conversion**

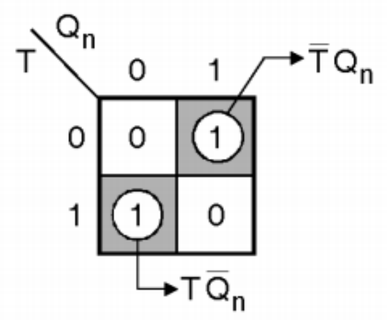
Inputs			Outputs
T	Present state Q_n	Next state Q_{n+1}	D
0	0	0	0
1	0	1	1
1	1	0	0
0	1	1	1

Step 2 : K map and simplification and logic diagram :

– K map is shown in Fig. 3.16.11(a) and logic diagram is shown in Fig. 3.16.11(b).

TechKnowledge Publications

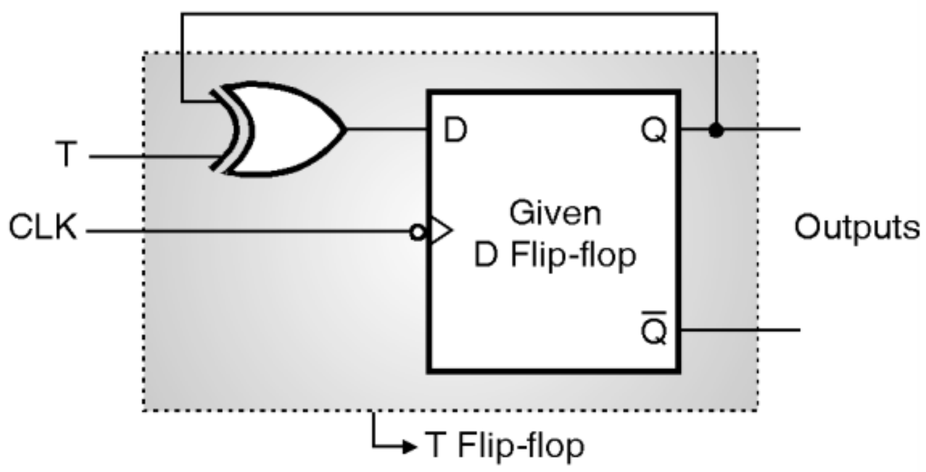
For output D



$\therefore D = T\overline{Q}_n + \overline{T}Q_n$

$\therefore D = T \oplus Q_n$

(a) K map and simplification



(b) Logic diagram

(C-621) **Fig. 3.16.11 : D flip flop to T flip flop**

3.16.6 T Flip Flop to D Flip Flop Conversion :

Step 1 : Write the truth table :

– The truth table is as given in Table 3.16.6.

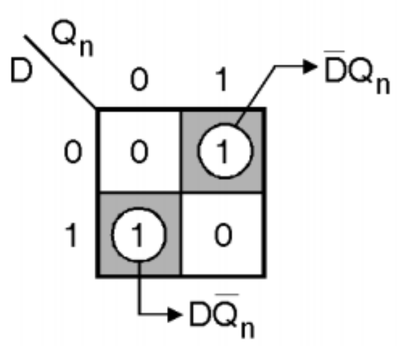
(C-7831) **Table 3.16.6 : Truth table for T FF to D FF conversion**

Inputs			Outputs
D	Previous state Q_n	Next state Q_{n+1}	T
0	0	0	0
1	0	1	1
0	1	0	1
1	1	1	0

Step 2 : K maps simplification and logic diagram :

– The K map is shown in Fig. 3.16.12(a) and the logic diagram is given in Fig. 3.16.12(b).

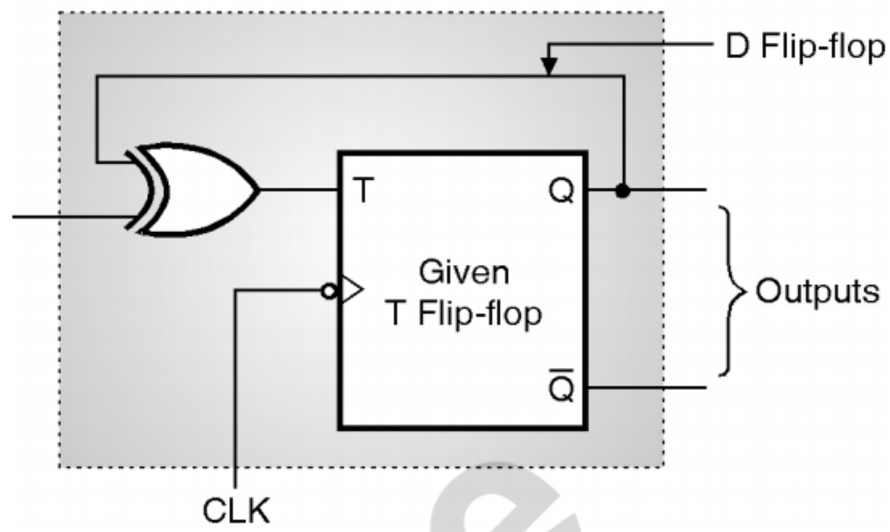
For output D



$\therefore T = D\overline{Q}_n + \overline{D}Q_n$

$\therefore T = D \oplus Q_n$

(C-622) **Fig. 3.16.12(a) : K map for D output**



(C-622) **Fig. 3.16.12(b) : Logic diagram of T to D flip flop conversion**

3.16.7 JK Flip Flop to D Flip Flop Conversion :

SPPU : May 06, Dec. 09

University Questions

Q. 1 How will you convert JK flip-flop into D flip-flop ? Explain application of D flip-flop in sequential circuits. **(May 06, 4 Marks)**

Q. 2 Convert J-K flip-flop into D-FF. Show the truth table. **(Dec. 09, 4 Marks)**

Step 1 : Write the truth table for JK to D conversion :

– The truth table is as follows :

(C-6390) **Table 3.16.7 : Truth table for JK to D flip flop conversion**

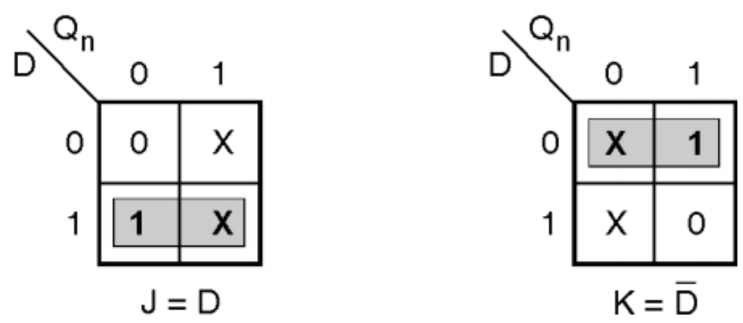
Inputs			Outputs	
D	Previous state Q_n	Next state Q_{n+1}	J	K
0	0	0	0	X
1	0	1	1	X
0	1	0	X	1
1	1	1	X	0

← Excitation table of D FF →

← Excitation table of JK FF →

Step 2 : K maps and simplification :

For J output **For K output**

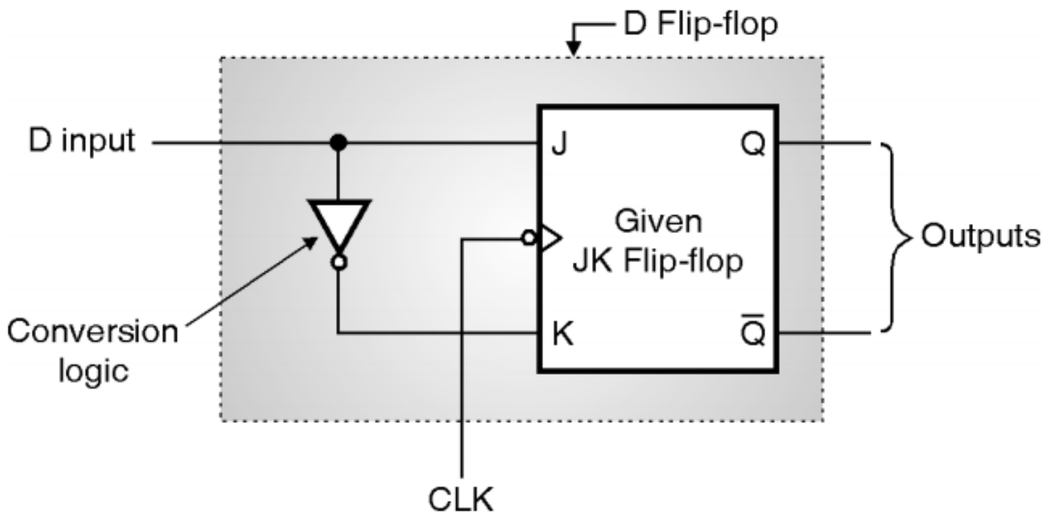


(a) K map for J output **(b) K map for K output**

(C-623) **Fig. 3.16.13**

Step 3 : Draw the logic diagram :

– The logic diagram for JK flip flop to D flip flop is shown in Fig. 3.16.14.



(C-624) **Fig. 3.16.14 : Logic diagram for conversion from JK FF to D FF**

3.16.8 JK Flip Flop to SR Flip Flop Conversion : SPPU : May 10

University Questions

Q. 1 Design SR flip-flop using JK flip-flop.

(May 10, 4 Marks)

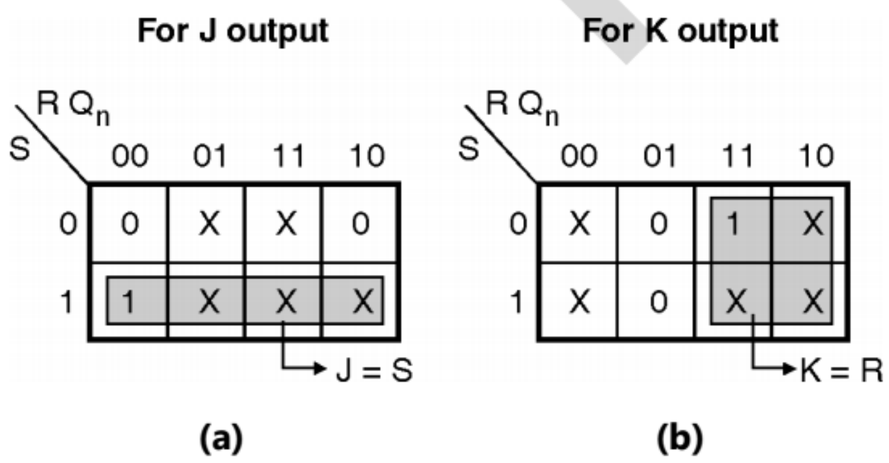
Step 1 : Write the truth table for JK to SR :

– The truth table is shown in Table 3.16.8.

(C-8045) **Table 3.16.8 : Truth table for JK to SR conversion**

Inputs				Outputs	
S	R	Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	0	0	X
0	1	0	0	0	X
1	0	0	1	1	X
1	0	0	1	1	X
0	1	1	0	X	1
0	1	1	0	X	1
0	0	1	1	X	0
1	0	1	1	X	0

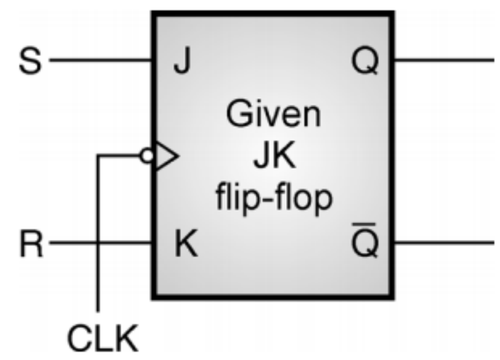
Step 2 : K-maps and simplifications :



(C-625) **Fig. 3.16.15**

Step 3 : Logic diagram :

– The logic diagram for JK to SR FF is shown in Fig. 3.16.15(c).



(C-626) **Fig. 3.16.15(c) : JK to SR FF conversion**

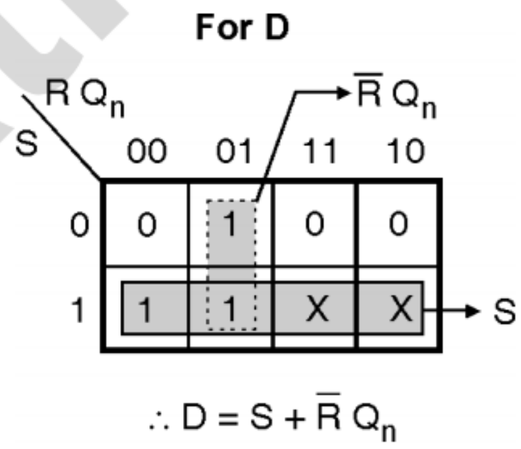
3.16.9 D FF to SR FF Conversion :

Step 1 : Write the truth table for D to S-R conversion :

(C-7832) **Table 3.16.9 : Truth table for D to S-R conversion**

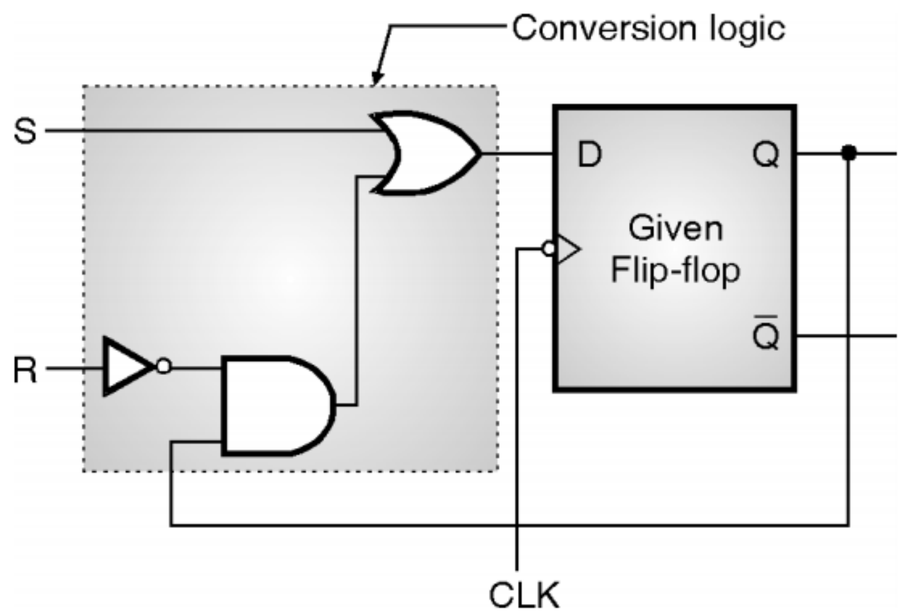
Inputs				Output
S	R	Present state Q_n	Next state Q_{n+1}	D
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
0	1	1	0	0
0	0	1	1	1
1	0	1	1	1

Step 2 : K maps and simplification :



(C-627) **Fig. 3.16.16 : K-map and simplification**

Step 3 : Logic diagram :



(C-628) **Fig. 3.16.17 : Logic diagram for D to S-R FF conversion**

TechKnowledge Publications

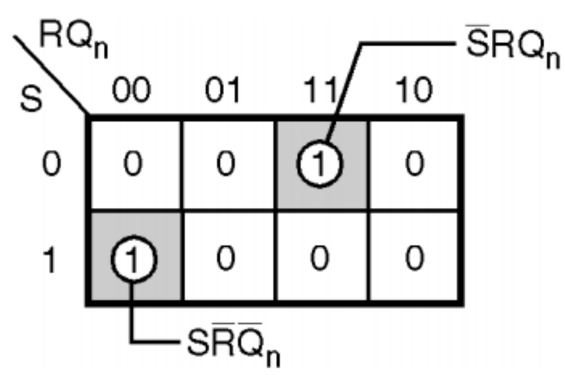
3.16.10 T FF to SR FF Conversion :

Step 1 : Write the truth table for T to S-R conversion :

(C-8046) Table 3.16.10 : Truth table for T to S-R conversion

Inputs				Output T
S	R	Present state Q_n	Next state Q_{n+1}	
0	X	0	0	0
1	0	0	1	1
0	1	1	0	1
X	0	1	1	0

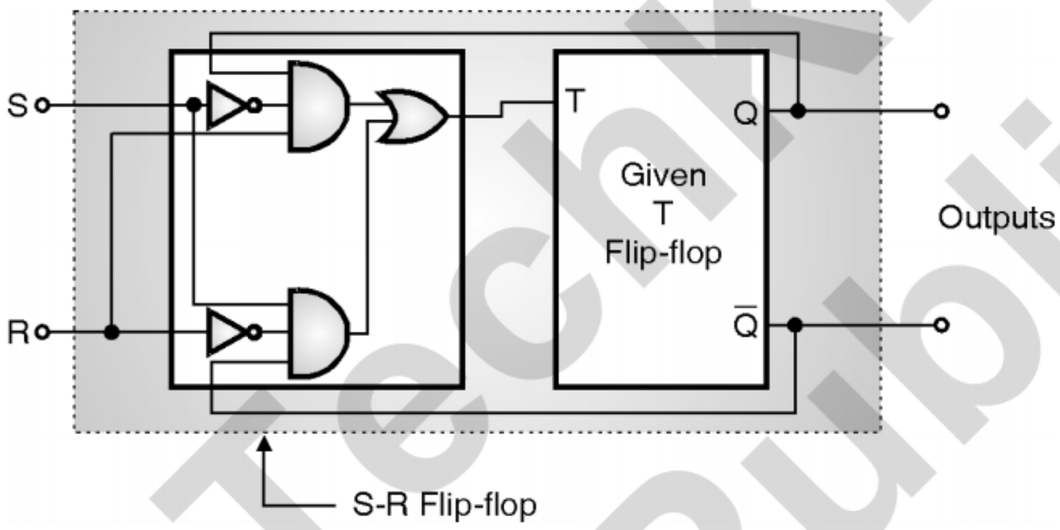
Step 2 : K maps and simplification for T output :



(C-629) Fig. 3.16.18 : K-map and simplification

$$\therefore\ T = S\bar{R}\,\bar{Q}_n + \bar{S}\,R\,Q_n$$

Step 3 : Logic diagram :



(C-630) Fig. 3.16.19 : Logic diagram for T to S-R FF conversion

3.16.11 Conversion from D FF to JK FF :

Step 1 : Write the truth table :

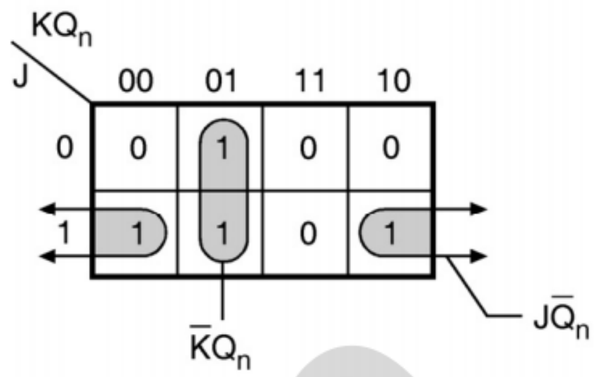
– Table 3.16.11 shows the truth table obtained from the excitation tables of D and JK FFs.

(C-8047) Table 3.16.11 : Truth table for D to JK conversion

Inputs				Output D
J	K	Present state Q_n	Next state Q_{n+1}	
0	X	0	0	0
1	X	0	1	1
X	1	1	0	0
X	0	1	1	1

Step 2 : K maps and simplification :

For output D

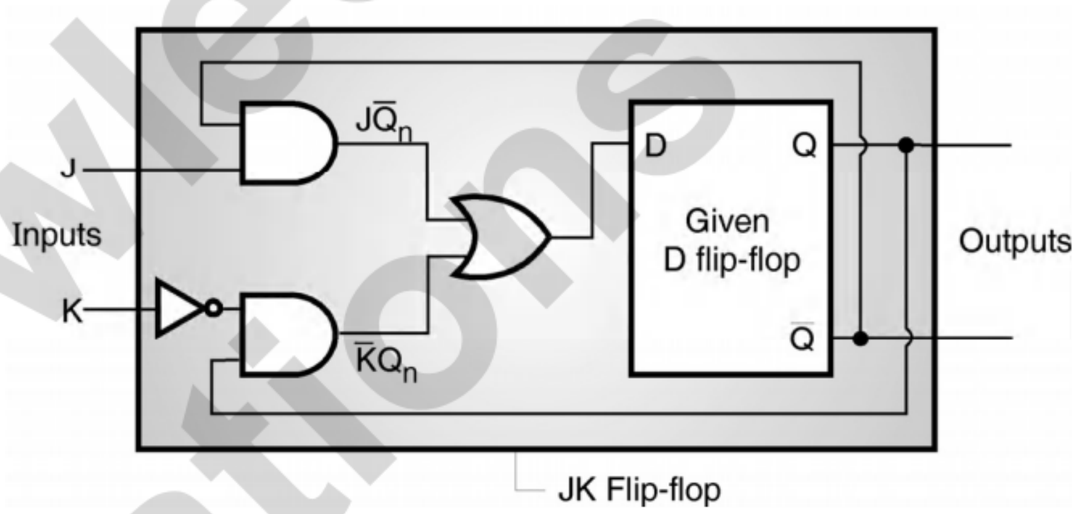


$$\therefore\ D = \bar{K}Q_n + J\bar{Q}_n$$

(C-1342(a)) Fig. 3.16.20

Step 3 : Logic diagram :

– The logic diagram for D FF to JK FF is shown in Fig. 3.16.21.



(C-1342(b)) Fig. 3.16.21 : D FF to JK FF

3.17 Applications of Flip Flops :

– Some of the important applications of the flip flops are :

1. Elimination of keyboard debounce.
2. As a memory element.
3. In various types of registers.
4. In counters/timers.
5. As a delay element.

3.17.1 Application of SR Latch for Elimination of Keyboard Debounce :

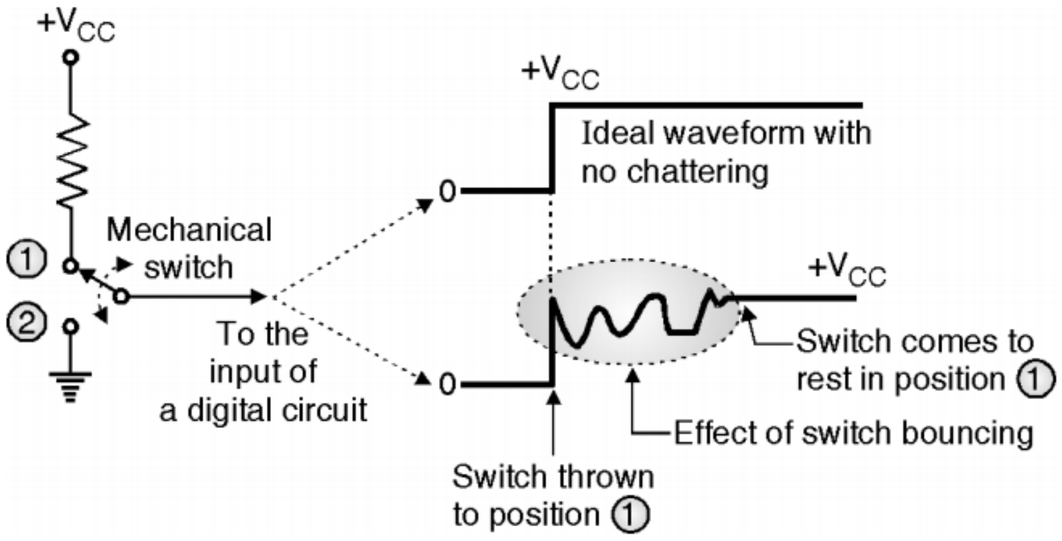
– Sometimes we need to use mechanical switches as input devices in order to enter digital information (0 or 1) into a digital system.

– **Switch bouncing or chattering** is a very serious problem associated with these switches.

– When the pole of a switch (also called as the arm) is thrown from say position 2 to 1. (see Fig. 3.17.1(a)), it chatters or bounces a number of times before finally coming to rest in position 1.

– The effect of this unwanted bouncing has been illustrated in Fig. 3.17.1(b). The input to the logic circuit will oscillate due to switch chattering.

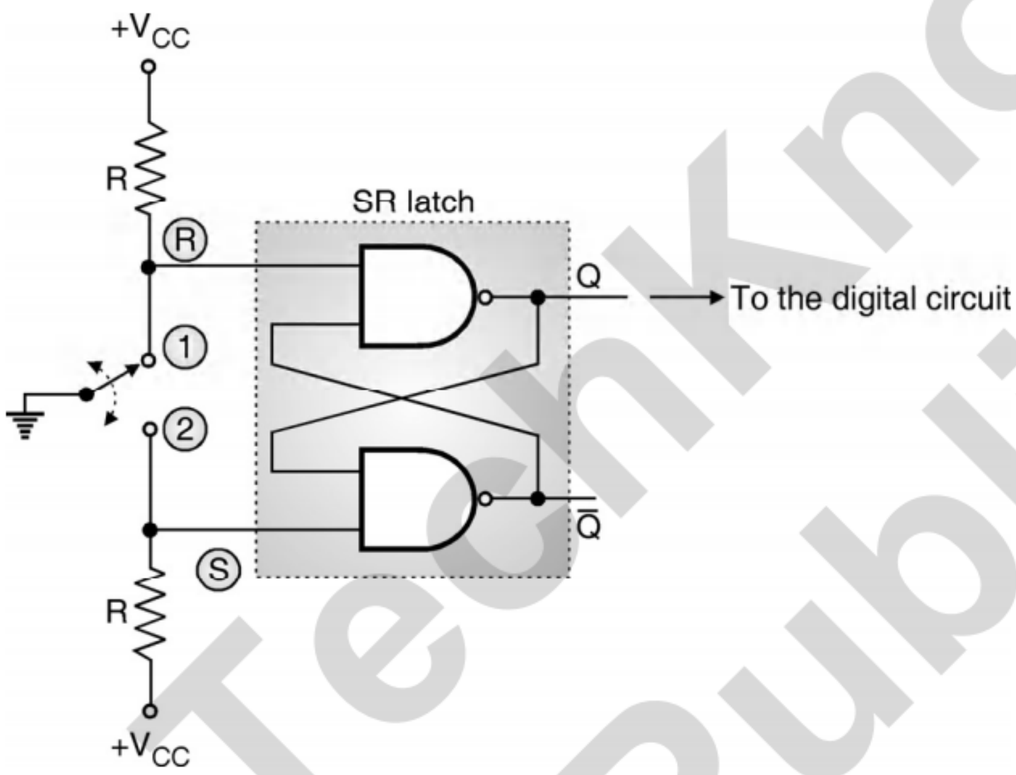
Tech Knowledge Publications



(a) Mechanical switch **(b) Effect of switch bouncing**

(C-641) Fig. 3.17.1

- This produces an oscillatory output from a sequential circuit and creates problems in the operation of the system.
- This problem can be sorted out by using bounce elimination switch as shown in Fig. 3.17.1(a).
- Fig. 3.17.2 shows how we can use the SR latch for creating a bounce elimination switch.

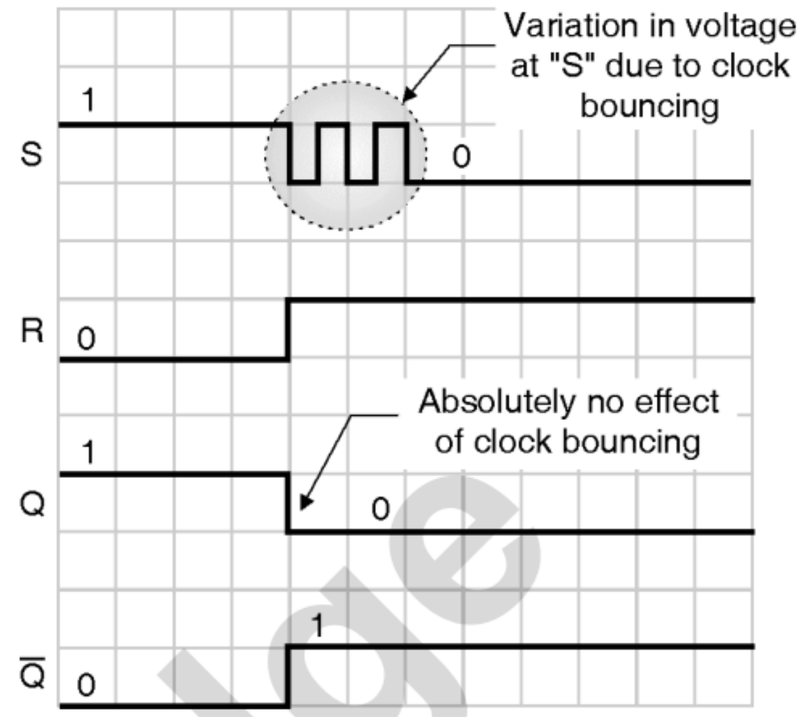


(C-642) Fig. 3.17.2 : A bounce elimination switch

Operation :

(C-643)

- Initially let the switch be in position 1
- Therefore S = 1 and R = 0. Hence Q = 1
- Now the switch is thrown to position 2
- Therefore S = 0 and R = 1 as the switch first
  ∴ Q = 0
- Now if the switch bounces off the contact 2, then S = R = 1
- Hence Q output will remain unchanged. Hence Q = 0
- After some bounces, the switch comes to rest at position 2



(C-644) Fig. 3.17.3 : Waveforms of bounce elimination clock

- Thus at the Q output of SR latch we get a clean waveform without any oscillations as shown in Fig. 3.17.3.

Note : The other applications of Flip Flops such as registers and counters have been discussed in chapters 4 and 5 of this book.

3.18 Analysis of Clocked Sequential Circuits :

- The behaviour of a clocked sequential circuit is dependent on the inputs, the outputs and the state of its flip-flops.
- The outputs as well as the next state both are function of inputs and present state.
- The analysis of the given clocked sequential circuit includes writing the state table and drawing the state diagram for the given circuit.
- In this section, we will introduce the concepts such as state diagram, state table, state equation and input equations and the step by step analysis procedure.

3.18.1 State Table :

- We use the truth table for explaining the relation between its inputs and outputs of a combinational circuit.
- But it is not suitable to use the truth table to describe the input output relation of a sequential circuit.
- Instead we use the state table to describe the input output relation of the sequential circuits.

TechKnowledge Publications

- The basic building block of a sequential circuit is a flip flop. The outputs Q_A, Q_B ... etc. of such flip flops will be used as **state inputs** to a sequential circuit. They are also called as **state variable.**
- In addition to this "x" represents an external input and Y represents the output of the sequential circuit.
- Y will be dependent on the state variables (Q_A, Q_B, ...) and the external input x.
- The general format of a state table is shown in Table 3.18.1.

(C-7833) Table 3.18.1 : General format of state table

Present state		Next state				Output Y	
		x = 0		x = 1		x = 0	x = 1
Q_A	Q_B	Q_A	Q_B	Q_A	Q_B		
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	1	0

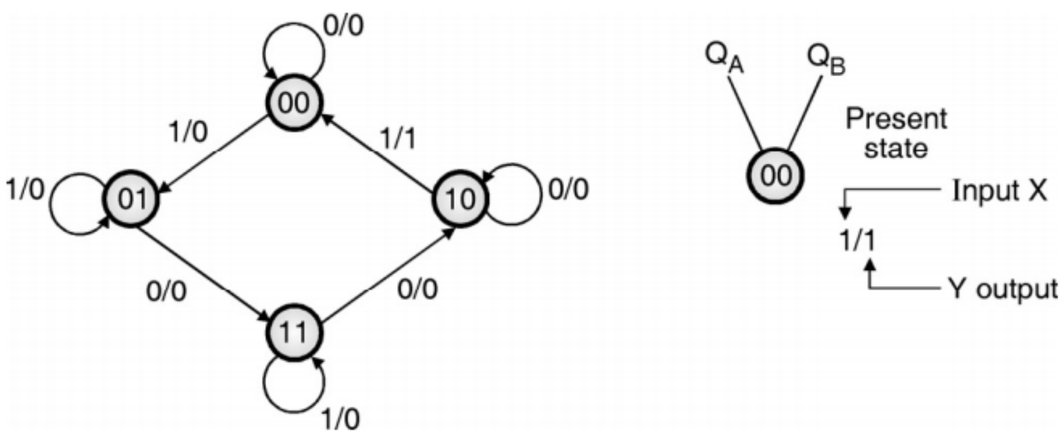
Definition :

- The state table is defined as the table that tells us about the relation between the present state, next state, external inputs and output of a sequential circuit.

3.18.2 State Diagram :

Definition :

- A state diagram is the graphical representation of a sequential circuit, which consists of states, state transitions and actions.
- State diagrams depict the permitted states and transitions as well as the events that effect these transition.
- The information available in the state table is represented graphically using the state diagram.
- The state diagram is drawn by using the state table as a reference. Such a state diagram is shown in Fig. 3.18.1.

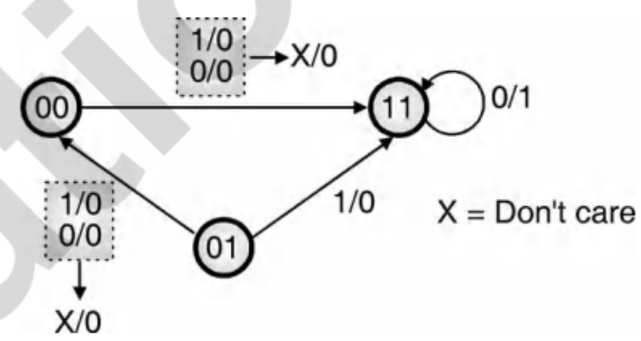


(C-653) Fig. 3.18.1 : State diagram

- The circle represents the present state. The arrows between the circles define the state transition say from 00 to 01 or 01 to 11.
- If there is directed line connecting the same circle then it means that the next state is same as the present state.
- The lines joining the circles are labeled with a pair of binary numbers with a " / " in between. For example the line joining 00 and 01 is labeled as 1/0.
- Note that 00 to 01 transition takes place when x = 1 and Y = 0 (see row-1 of the state table). Hence 1 in 1/0 corresponds to x and 0 corresponds to y.

Don't care condition in the state diagram :

- Sometimes the same next state is reached for more than one present states.
- This is called as don't care condition in the state diagram, as shown in Fig. 3.18.2.



(C-654) Fig. 3.18.2 : Don't care condition in state diagram

3.18.3 State Equation :

Definition :

- State equation is an algebraic equation. The left side of this equation represents the next state of the flip flops.
- And the right hand side of this equation specifies the present state conditions which make the next state equal to 1.
- For example consider the following state expression.

(C-6235)

$$Q_{A(n+1)} = (\bar{Q}_A Q_B + Q_A \bar{Q}_B + Q_A Q_B)\,x + Q_A \bar{Q}_B x \ldots\ldots$$

Next state

Combinations of present state and inputs which make the next state equal to 1.

- State equation is also called as **application equation**.
- We can derive the state equation from the state table using the K maps. The state equation is the Boolean function with time included into it.
- If the right hand side part is zero and we apply a clock pulse, then the next state will be equal to zero.

Ex. 3.18.1 : For the clocked D FF write the state table, draw the state diagram and write the state equation.

Soln. :

Step 1 : Write the truth table :

– Table P. 3.18.1(a) represents the truth table for a clocked D flip flop.

(C-6212) Table P. 3.18.1(a) : Truth table of a clocked D FF

Inputs		Outputs		
CLK	D	Q_{n+1}	$\overline{Q}_{n+1}$	
0	X	Q_n	$\overline{Q}_n$	No change in output
1	X	Q_n	$\overline{Q}_n$	
↓	X	Q_n	$\overline{Q}_n$	
↑	0	0	1	Q output follows
↑	1	1	0	

Step 2 : Write the state table :

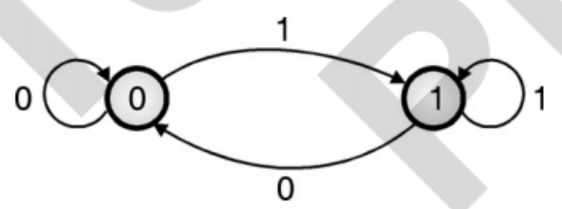
– Table P. 3.18.1(b) represents the state table for the clocked D FF.

(C-7834) Table P. 3.18.1(b) : State table of a clocked D FF

Present state Q_n	Next state Q_{n+1}	
	D = 0	D = 1
0	0	1
1	0	1

Step 3 : State diagram :

– State diagram of clocked D FF is shown in Fig. P. 3.18.1.



(C-659(a)) Fig. P. 3.18.1 : State diagram of a clocked D FF

Step 4 : Write excitation table :

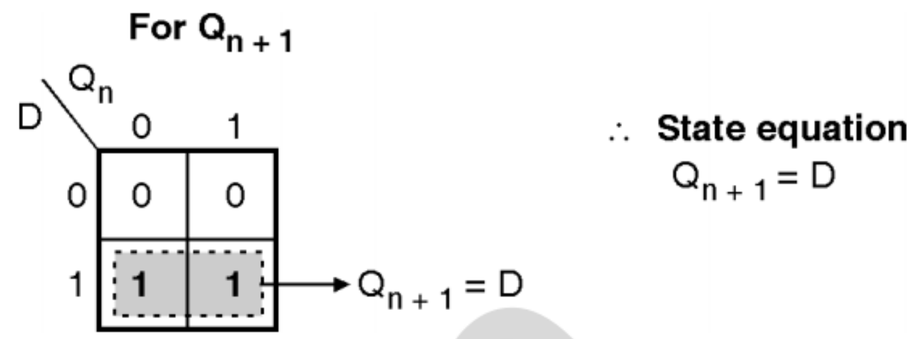
– The excitation table for D FF is given in Table P. 3.18.1(c).

(C-7835) Table P. 3.18.1(c) : Excitation table for D FF

Present state Q_n	Next state Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 5 : Write the state equation :

– The K-map for output Q_{n+1} is shown in Fig. P. 3.18.1(a).



(C-656) Fig. P. 3.18.1(a) : K-map and state equation

Ex. 3.18.2 : For the clocked JK FF write the state table, draw the state diagram and write the state equation.

Soln. :

Step 1 : Write the truth table :

– Table P. 3.18.2(a) represents the truth table for a clocked JK flip flop.

(C-6213) Table P. 3.18.2(a) : Truth table of JK FF

Inputs			Output	
CLK	J	K	Q_{n+1}	
0	X	X	Q_n (NC)	No change in output
1	X	X	Q_n (NC)	
↓	X	X	Q_n (NC)	
↑	0	0	Q_n (NC)	
↑	0	1	0	
↑	1	0	1	
↑	1	1	$\overline{Q}_n$	

Step 2 : Write the state table :

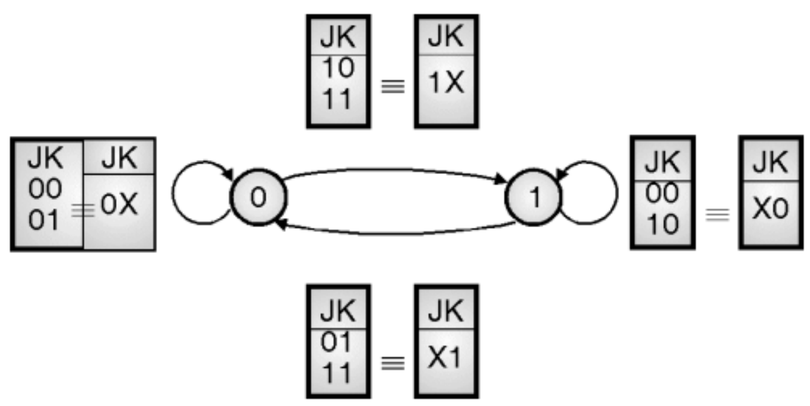
– Table P. 3.18.2(b) represents the state table for a clocked JK FF.

(C-8070) Table P. 3.18.2(b) : State table of clocked JK FF

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

Step 3 : Draw the state diagram :

– State diagram is as shown in Fig. P. 3.18.2(a).



(C-657) Fig. P. 3.18.2(a) : State diagram of a JK flip flop

Step 4 : Write the excitation table :

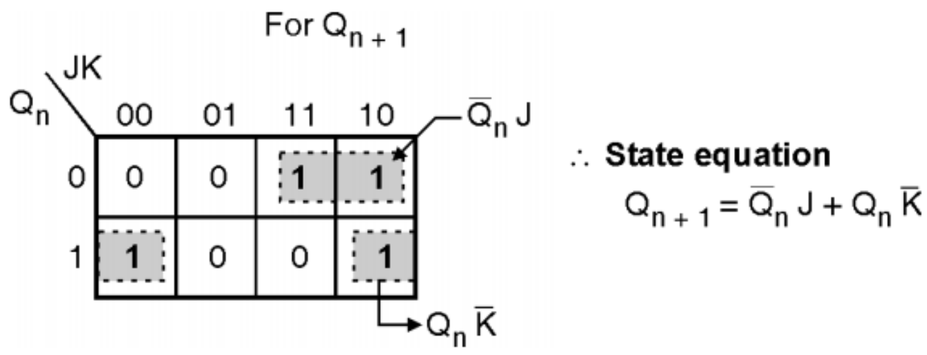
– The excitation table of JK FF is as shown in Table P. 3.18.2(c).

(C-7840) Table P. 3.18.2(c) : Excitation table for JK FF

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Step 5 : Write the state equation :

– The K-map for output Q_{n+1} is shown in Fig. P. 3.18.2(b).



(C-658) Fig. P. 3.18.2(b) : K-map state equation

Ex. 3.18.3 : For a toggle FF write the state table, draw the state diagram and write the state equation.

Soln. :

Step 1 : Write the truth table :

– Table P. 3.18.3(a) gives the truth table for a T FF.

(C-8071) Table P. 3.18.3(a) : Truth table for a T FF

CLK	T	Q_{n+1}
0	X	Q_n
1	X	Q_n
↑	X	Q_n
↓	0	Q_n
↓	1	$\overline{Q}_n$

Step 2 : Write the state table :

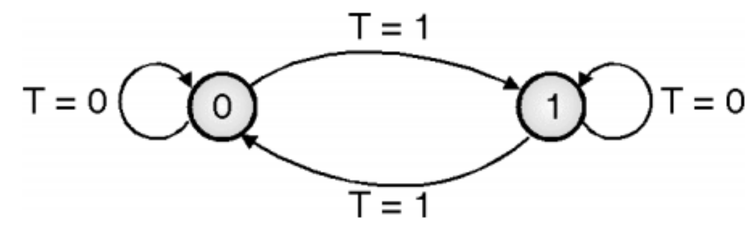
– The state table is shown in Table P. 3.18.3(b).

(C-8072) Table P. 3.18.3(b) : State table for a T FF

Present state Q_n	Next state Q_{n+1}	
	T = 0	T = 1
0	0	1
1	1	0

Step 3 : Draw the state diagram :

– The state diagram is shown in Fig. P. 3.18.3(a).



(C-659) Fig. P. 3.18.3(a) : State diagram of a T FF

Step 4 : Write the excitation table :

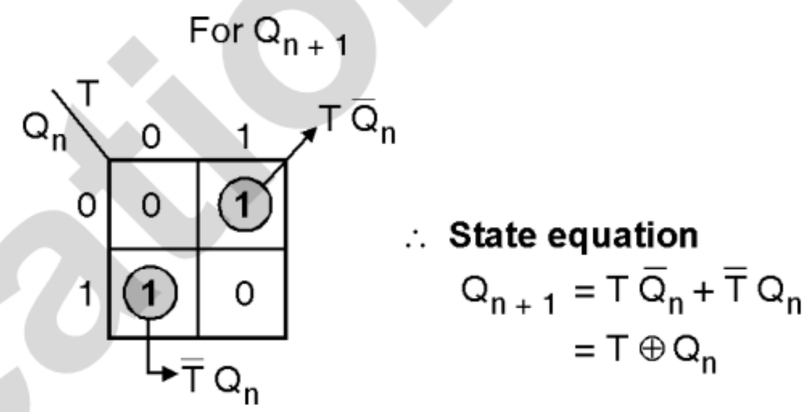
– Table P. 3.18.3(c) represents the excitation table of T FF.

(C-7842) Table P. 3.18.3(c) : Excitation table for T FF

Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 5 : Write the state equation :

– The K-map for Q_{n+1} output is shown in Fig. P. 3.18.3(b). The state equation can be obtained from this K-map.



(C-660) Fig. P. 3.18.3(b)

3.18.4 Flip Flop Input Equations :

– The logic diagram of a sequential circuit consists of flip flops and gates. Fig. 3.18.3 shows an example circuit.

– The interconnected gates form a combinational circuit and we can specify it by using the Boolean expressions.

– If the types of flip flops are known and all the Boolean expressions also are known to us, then it is possible for us to draw the logic diagram of a sequential circuit.

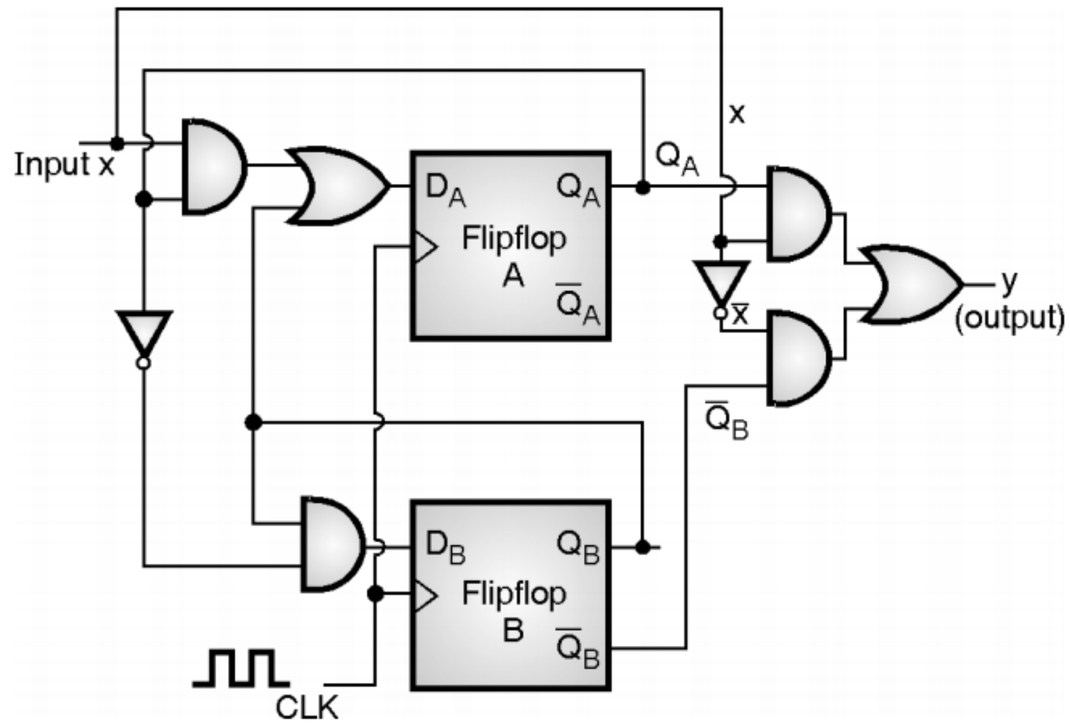
Output equations :

– We can express the part of combinational circuit that generates the external outputs by the Boolean expressions called as the output equations.

– The output equation for the sequential circuit of Fig. 3.18.3 is,

$$= x Q_A + \overline{x}\,\overline{Q}_B \qquad ...(3.18.1)$$

TechKnowledge Publications



(C-661) **Fig. 3.18.3 : A sequential circuit**

Input equations :

- The part of the combinational circuit that generates the inputs to flip flops is expressed by Boolean equations called as flip flop input equations.
- The input equations for the sequential circuit of Fig. 3.18.3 are,

$$D_A = x \cdot Q_A + Q_B \quad ...(3.18.2)$$

$$\text{and} \quad D_B = \overline{Q}_A \cdot Q_B \quad ...(3.18.3)$$

- The input equations are also called as **excitation equations.**
- With the input and output equations we can draw the logic diagram of a sequential circuit.

Review Questions

Q. 1 Mention the types of digital systems. Explain with block diagram their operating principle.

Q. 2 What is a flip-flop ?

Q. 3 State and explain the triggering methods used for flip-flops.

Q. 4 What is the function of preset and clear inputs in flip-flop ?

Q. 5 Explain with truth table the working of clocked RS flip-flop.

Q. 6 State the disadvantages of RS flip-flop. How can they be avoided ?

Q. 7 Explain with diagram the working of D type flip-flop. Give its truth table.

Q. 8 Give reason why D flip-flop is called as data latch ?

Q. 9 Draw the circuit using logic gates of a T-type flip-flop. Draw its symbol and write its truth table.

Q. 10 Explain S-R flip flop using NOR gates.

Q. 11 Describe how two cross-coupled NAND gates form a R-S flip-flop ? Write its truth table.

Q. 12 What are the various types of flip-flops ?

Q. 13 Draw the circuit of SR flip-flop using NAND gate.

Q. 14 Draw the schematic diagram of JK flip-flop and describe its working. Write down its truth table.

Q. 15 What is race around condition ?

Q. 16 Draw the circuit of J-K flip-flop using NAND gate.

Q. 17 Draw a neat diagram of master slave J-K flip-flop. Explain how race around condition is avoided using master slave J-K flip-flop ?

Q. 18 Explain the working of the master slave JK flip-flop.

Q. 19 Can a flip flop be used as a memory ? If so how many bits can be stored by R-S flip flop ?

Q. 20 Explain T flip-flop.

Q. 21 Explain the following flip-flops :

1. Clocked SR.
2. JK with preset and clear.
3. Master Slave JK.
4. D type and T type.

Q. 22 Design a conversion logic to convert a JK flip-flop to a D flip-flop.

Q. 23 Write a short note on race around condition in JK flip-flop.

Q. 24 Draw neat circuit diagram of clocked JK flip-flop using NAND gates. Give its truth table and explain race around condition.

Q. 25 What is race around condition ? How does it gets eliminated in master slave JK FF ? Explain.

TechKnowledge Publications

Q. 26 Explain how JK FF is converted into :

1. D FF 2. T FF

Q. 27 Carry out the following flip flop conversions (Conversion tables and K-maps expected) :

1. S-R to D
2. D to S-R
3. J-K to S-R

Q. 28 If $\overline{Q}$ output of a D-type Flip-Flop is connected to D input, it acts as a toggle switch. State whether true or false ? Justify your answer.

Q. 29 What is the basic difference between pulse-triggered and edge-triggered flip-flops ?

❑❑❑

TechKnowledge Publications

Unit III

Chapter 4

Registers

TechKnowledge Publications

Syllabus

SISO, SIPO, PISO, PIPO, Shift registers, Bidirectional register, Universal shift register.

Chapter Contents

4.1	Introduction	4.7	Parallel In Serial Out Mode (PISO)
4.2	Data Formats	4.8	Parallel In Parallel Out (PIPO)
4.3	Classification of Registers	4.9	Bidirectional Shift Register
4.4	Buffer Registers	4.10	Universal Shift Register
4.5	Shift Registers	4.11	Applications of Shift Registers
4.6	Serial In Parallel Out (SIPO)		

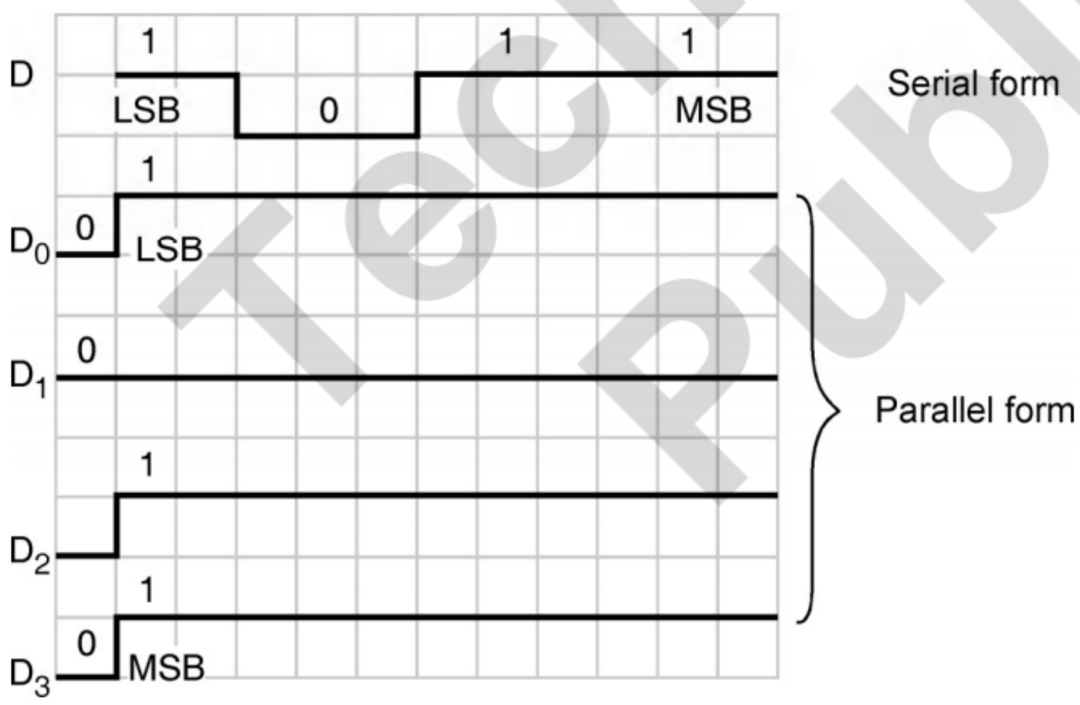
4.1 Introduction :

Definition :

- In chapter 3, we have stated various applications of flip-flops. One of them was "registers".
- Flip-flop is a 1 bit memory cell which can be used for storing the digital data.
- To increase the storage capacity in terms of number of bits, we have to use a group of flip-flops. Such a group of flip-flops is known as **a register.**
- Thus register is a group of flip-flops. The "n-bit" register will consist of "n" number of flip-flops and it is capable of storing an "n bit" word.

4.2 Data Formats :

- The data can be entered in **serial** (one bit at a time) manner or in the **parallel** form (all the bits at the same time) into a register.
- And the stored data can be retrieved in the serial or parallel form.
- A four bit digital data 1 0 1 1 is represented in the serial and parallel forms in Fig. 4.2.1.



(C-729(h)) Fig. 4.2.1 : Data representation in serial and parallel forms

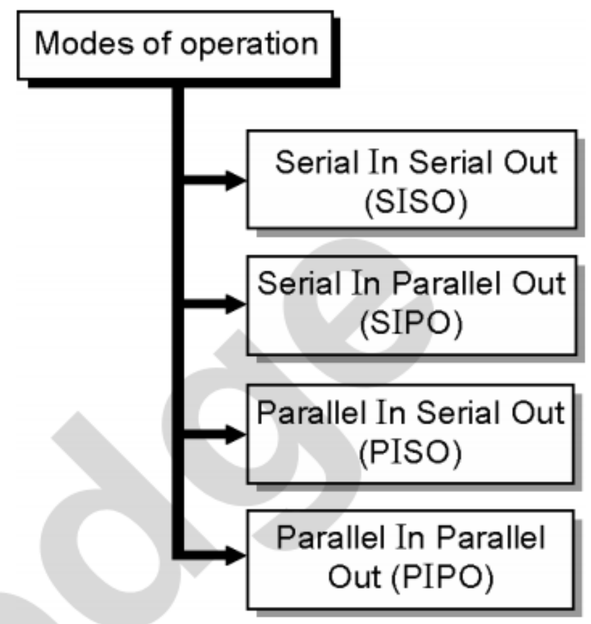
4.3 Classification of Registers :

SPPU : Dec. 04

University Questions

Q. 1 Explain different modes of universal shift register with application of each. **(Dec. 04, 6 Marks)**

- Registers are classified on the basis of how the data bits are entered and taken out from a register. There are four possible modes as follows :



(C-729(i)) Fig. 4.3.1 : Modes of operation of registers

- We can design all these registers using discrete flip-flops such as SR, JK or D flip-flops.
- But the registers are also available in the IC form.
- Some of the registers available in 54/74 TTL series as listed in Table 4.3.1.

Table 4.3.1 : Shift registers available in 74/54 series

IC number	Description
7491, 7491A	8 bit serial in, serial out
7494	4 bit parallel in, serial out
7495	4 bit serial/parallel in, parallel out (shift right shift left)
7496	5 bit parallel in / parallel out, serial in / serial out.

4.4 Buffer Registers :

Logic diagram :

- The simplest type of register constructed using four D flip-flops is shown in Fig. 4.4.1.
- This is a 4 bit register, but we can construct an n-bit register by following the same principle.
- This register is also called as the **buffer register.**
- Each D-flip-flop is negative edge triggered and all the flip-flops are connected to a common clock signal. Hence all of them are triggered at the same instant of time, i.e. all the flip flops will change their state at the same instant of time.
- Buffer registers are used for temporary storage of digital words.

TechKnowledge Publications

Operation :

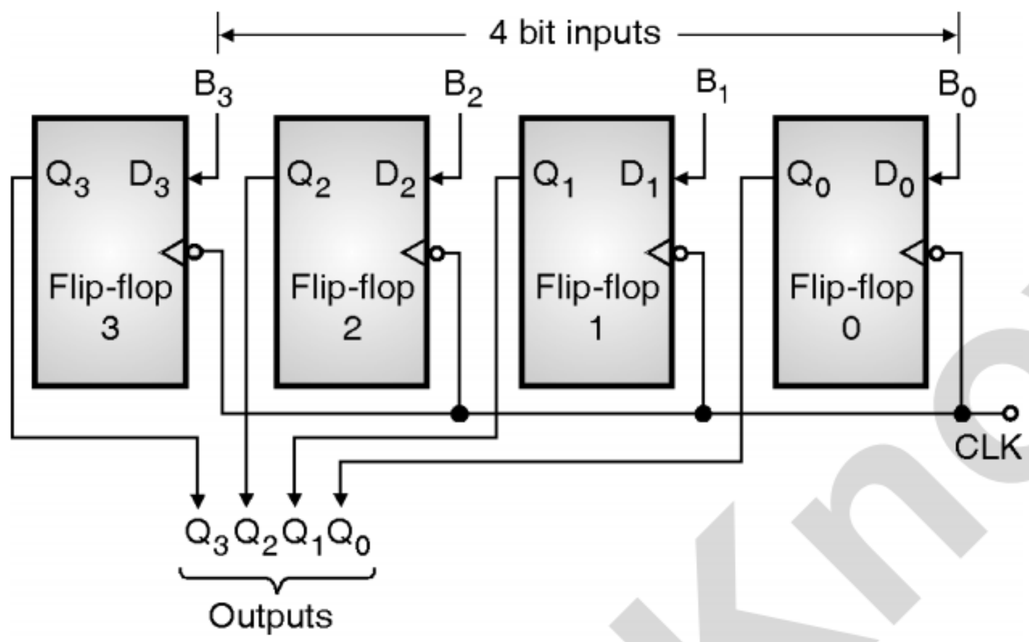
– Let us assume that the word to be stored is $B_3\ B_2\ B_1\ B_0$ = 1 0 1 0.

– These bits are connected to the D inputs of the four D flip-flops as shown in Fig. 4.4.1.

– Then the clock pulse is applied.

– Corresponding to the first negative edge of the clock pulse, the outputs of all the D flip-flops will follow their respective inputs.

$\therefore \quad Q_3\ Q_2\ Q_1\ Q_0 = B_3\ B_2\ B_1\ B_0 = 1\ 0\ 1\ 0$



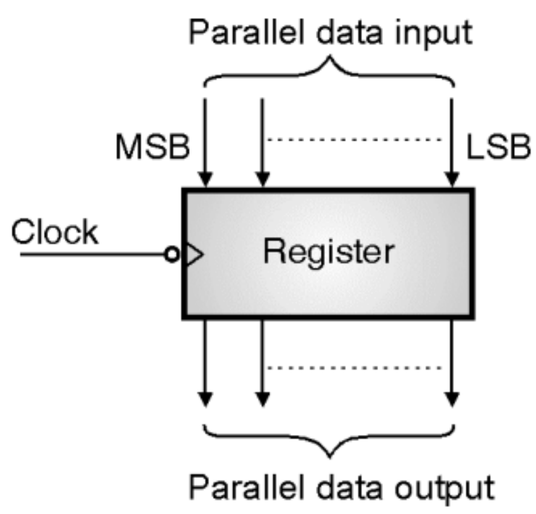
(C-729(j)) **Fig. 4.4.1 : A four bit buffer register using D flip-flops**

– Even if the inputs are now changed, the output remains latched to 1 0 1 0 till the next negative edge of the clock arrives at the input.

– Thus the buffer register is capable of storing the digital data.

Schematic diagram :

– The schematic diagram of a buffer register is as shown in Fig. 4.4.2.



(C-730) **Fig. 4.4.2 : Schematic diagram of buffer register**

Conclusions :

– Some of the important conclusions from the discussion till now are as follows :

1. There must be one-FF for each bit to be stored. Therefore to store a 4 bit number we need four flip-flops.
2. Note that all the four input bits B3 B2 B1 B0 are loaded into the buffer register simultaneously, i.e. at the same instant of time.
3. Therefore this way of applying the input and taking the output is called as Parallel Input Parallel Output (PIPO) and the mode of operation is called as **parallel shifting.**

4.5 Shift Registers :

Definition :

– The binary data in a register can be moved within the register from one flip-flop to the other or outside it with application of clock pulses.

– The registers that allow such data transfers are called as shift registers.

– Shift registers are used for data storage, data transfer and certain arithmetic and logic operations.

Modes of operation of a shift register :

– The various modes in which a shift register can operate are as follows :

1. Serial Input Serial Output. (SISO).
2. Serial Input Parallel Output. (SIPO).
3. Parallel In Serial Out. (PISO).
4. Parallel In Parallel Out. (PIPO).

– These modes are explained in brief in Table 4.5.1.

Table 4.5.1 : Brief explanation of various modes of shift register

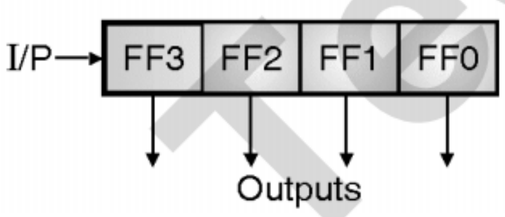
Sr. No.	Mode	Illustrative diagram	Comments
1.	Serial input serial output (serial shift right)	Refer Fig. A	Data bits shift from left to right by 1 position per clock cycle.

TechKnowledge Publications

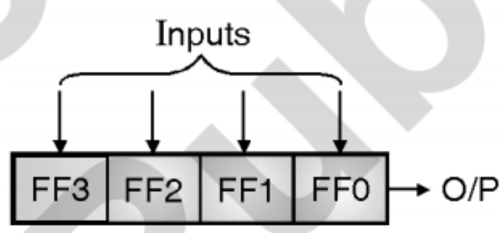
Sr. No.	Mode	Illustrative diagram	Comments
2.	Serial input serial output (serial shift left)	Refer Fig. B	Data bits shift from right to left by 1 position per clock.
3.	Serial input parallel output	Refer Fig. C	All output bits are made available simultaneously after 4-clock pulses.
4.	Parallel input serial output	Refer Fig. D	All inputs are loaded simultaneously but output bit by bit.
5.	Parallel input parallel output	Refer Fig. E	All inputs are loaded simultaneously and are available at the output simultaneously.

(C-731) **Fig. A**

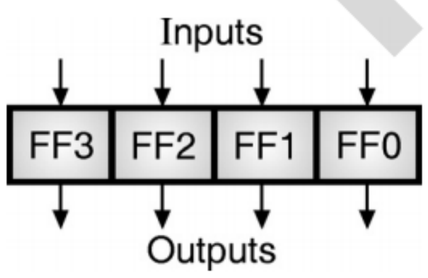
(C-731) **Fig. B**



(C-731) **Fig. C**



(C-731) **Fig. D**



(C-3167) **Fig. E**

4.5.1 Serial Input Serial Output (Shift Left Mode) : SPPU : May 05

University Questions

Q. 1 Draw and explain the circuit diagram of 3-bit register with Serial Left Shift. **(May 05, 4 Marks)**

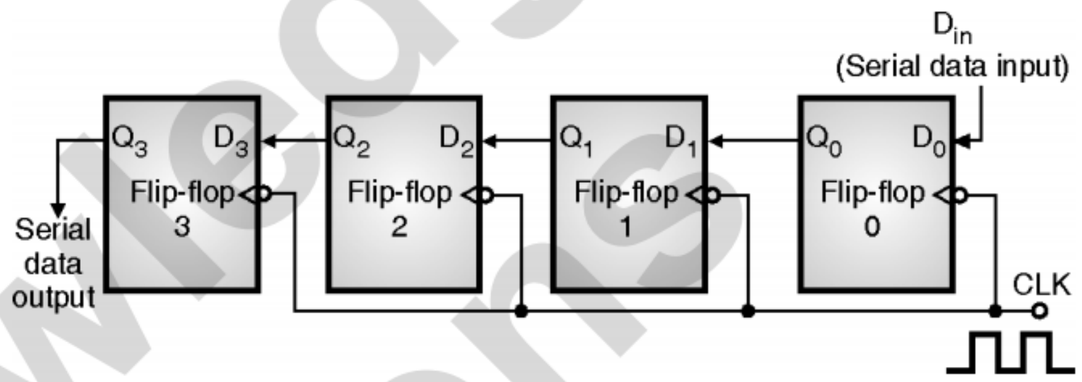
Principle :

– Data bits shift from right to left by 1 position per clock cycle as shown in Fig. 4.5.1(a).

(C-731) **Fig. 4.5.1(a) : Principle of Serial shift left register**

Logic diagram :

– The serial input serial output type shift register with shift left mode is shown in Fig. 4.5.1(b).



(C-732) **Fig. 4.5.1(b) : Serial shift left register**

– Let all the flip-flops be initially in the reset condition i.e. $Q_3 = Q_2 = Q_1 = Q_0 = 0$.

– We are going to illustrate the entry of a four bit binary number 1 1 1 1 into the register.

– When this is to be done, this number should be applied to "D_{in}" bit by bit with the MSB bit applied first.

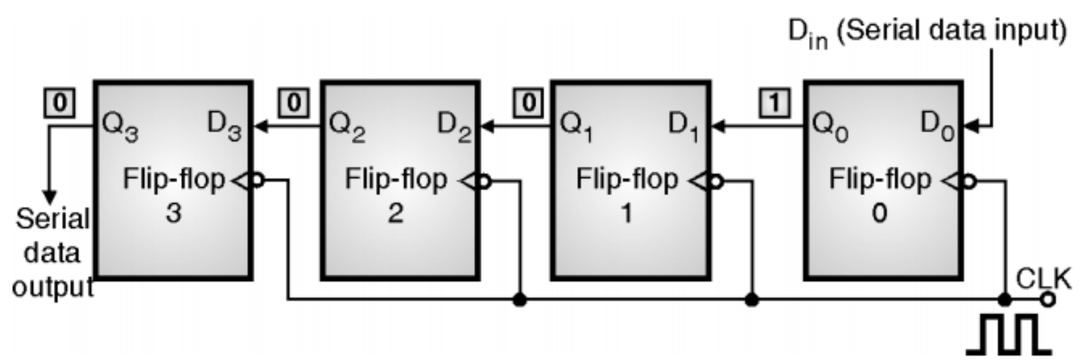
– The D input of FF-0 i.e. D_0 is connected to serial data input (D_{in}). Output of FF-0 i.e. Q_0 is connected to the input of the next flip-flop i.e. D_1 and so on.

Operation :

– Before application of clock signal let $Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 0 and apply MSB bit of the number to be entered to D_{in}. So $D_{in} = D_0 = 1$.

– Apply the clock. On the first falling edge of clock, the FF-0 is set and the stored word in the register is

$Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 1



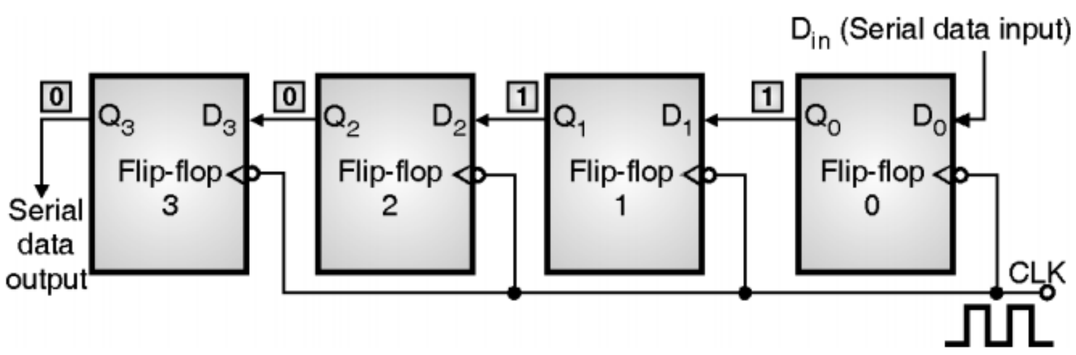
(C-733) **Fig. 4.5.2 : Shift register status after first falling clock edge**

TechKnowledge Publications

– Apply the next bit to D_{in}. So D_{in} = 1.

– As soon as the next negative edge of the clock hits, FF-1 will set and the stored word changes to,

$Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 1 1



(C-734) Fig. 4.5.3 : Shift register status after the second falling edge of clock

– Apply the next bit to be stored i.e. 1 to D_{in}.

– Apply the clock pulse. As soon as the third negative clock edge hits, FF-2 will be set and the output get modified to, $Q_3\ Q_2\ Q_1\ Q_0$ = 0 1 1 1.

– Similarly with D_{in}= 1, and with the fourth negative clock edge arriving, the stored word in the register is, $Q_3\ Q_2\ Q_1\ Q_0$ = 1 1 1 1.

(C-737) Table 4.5.2 : Summary of shift left operation

	CLK	Q_3	$Q_2 = D_3$	$Q_1 = D_2$	$Q_0 = D_1$	Serial input $D_{in} = D_0$
Initially		0	0	0	0	
1st	↓	0	0	0	1	1
2nd	↓	0	0	1	1	1
3rd	↓	0	1	1	1	1
4th	↓	1	1	1	1	1

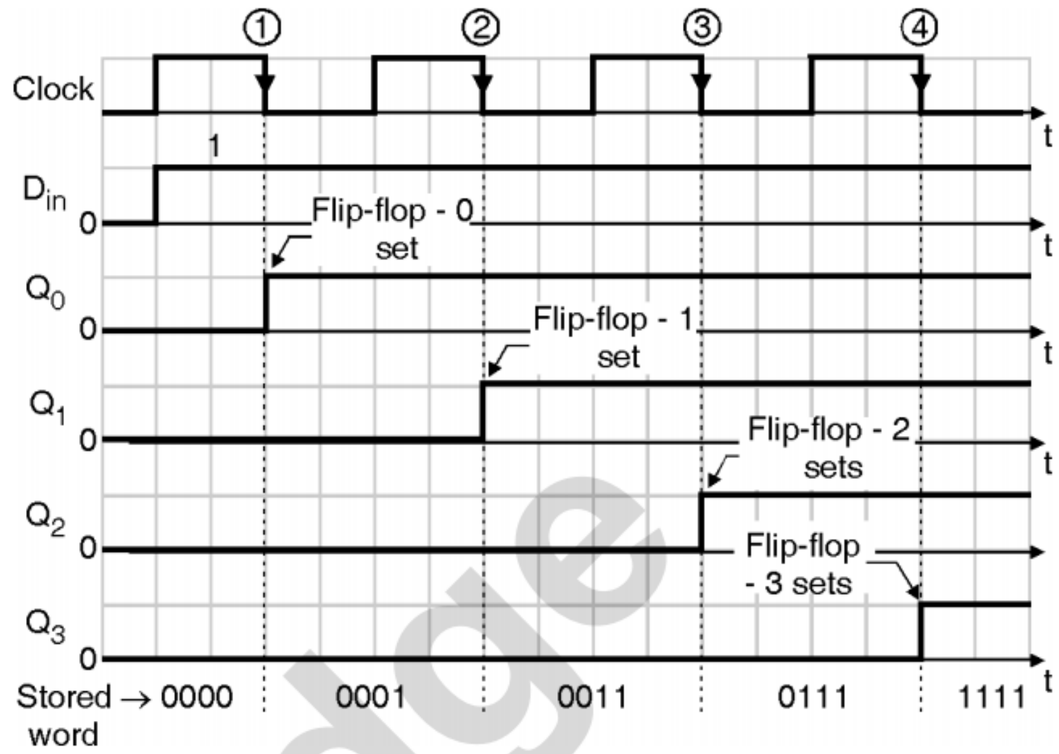
Direction of data travel ←

Waveforms for shift left operation :

– The waveforms for the shift left operation are shown in Fig. 4.5.4.

Important note :

Using the SISO mode, we needed 4-clock pulses to store a 4-bit word. So in general we can conclude that it requires n number of clock pulses to store an n bit word using SISO mode.



(C-738) Fig. 4.5.4 : Waveforms for the shift left operation

4.5.2 Serial In Serial Out (Shift Right Mode) :

SPPU : May 05, Dec. 14

University Questions

Q. 1 Explain with neat diagram working of serial-in serial-out 4-bit shift register. Draw necessary timing diagram. **(May 05, Dec. 14, 6 Marks)**

Principle :

– Data bits shift from left to right by 1 position per clock cycle as shown in Fig. 4.5.5(a).

(C-731) Fig. 4.5.5(a) : Principle of Serial shift right register

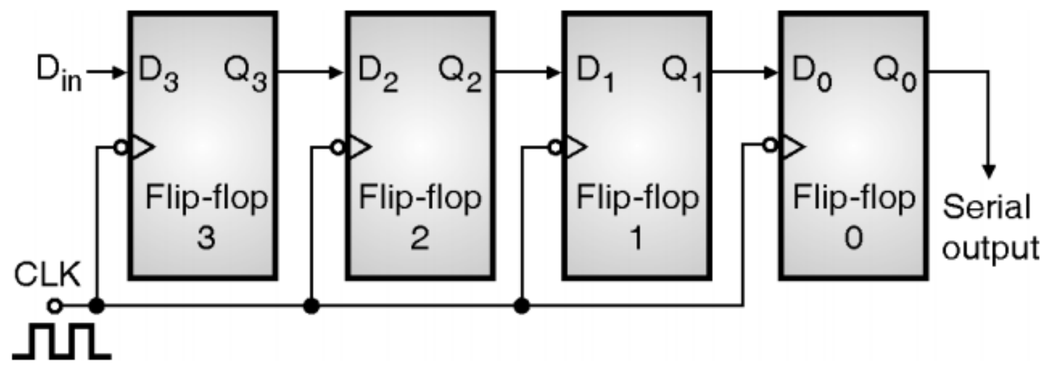
Logic diagram :

– The serial input serial output type shift register with shift right mode is shown in Fig. 4.5.5(b).

– Let all the flip-flops be initially in the reset condition i.e. $Q_3 = Q_2 = Q_1 = Q_0 = 0$.

– We are going to illustrate the entry of a four bit binary number 1 1 1 1 into the register.

– When this is to be done, this number should be applied to "D_{in}" bit by bit with the LSB bit applied first.



(C-739) Fig. 4.5.5(b) : Serial shift right register

TechKnowledge Publications

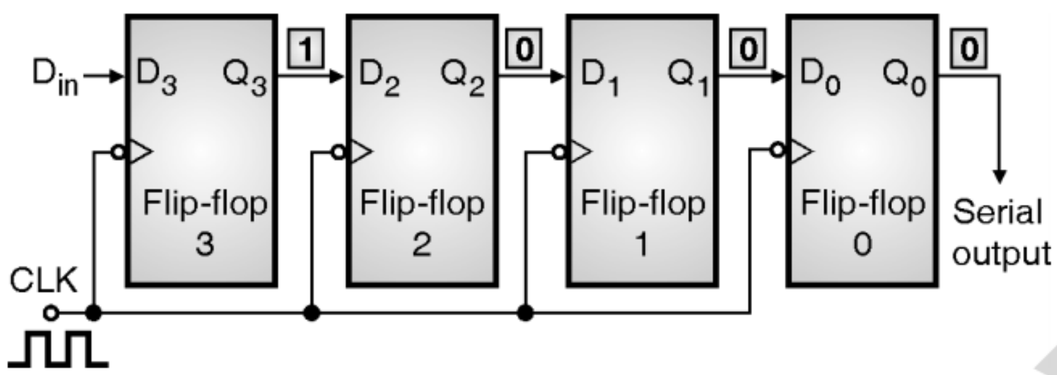
– The D input of FF-3 i.e. D_3 is connected to serial data input (D_{in}). Output of FF-3 i.e. Q_3 is connected to the input of the next flip-flop i.e. D_2 and so on.

Operation :

– Before application of clock signal let $Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 0 and apply LSB bit of the number to be entered to D_{in}. So $D_{in} = D_3 = 1$.

– Apply the clock. On the first falling edge of clock, the FF-3 is set, and the stored word in the register is,

$$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 0\ 0\ 0$$



(C-740) Fig. 4.5.6 : Shift register status after first falling clock edge

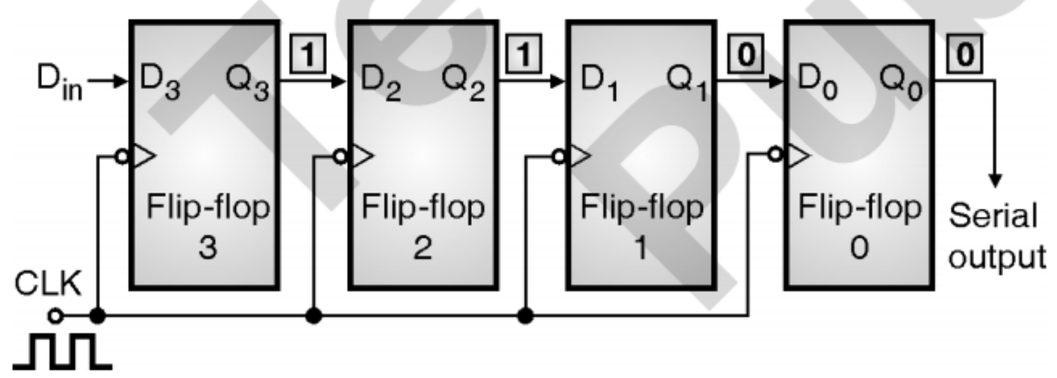
– The shift register after the application of the first clock pulse is as shown in Fig. 4.5.6.

– Apply the next bit to D_{in}. So $D_{in} = 1$.

– As soon as the next negative edge of the clock is applied, FF-2 will set and the stored word changes to,

$$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 1\ 0\ 0$$

– This is as shown in Fig. 4.5.7.



(C-741) Fig. 4.5.7 : Shift register status after the second falling edge of clock

– Apply the next bit to be stored i.e. 1 to D_{in}.

– Apply the clock pulse. As soon as the third negative clock edge gets applied, FF-1 will be set and the output get modified to,

$$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 1\ 1\ 0$$

– Similarly with D_{in} = 1 and with the fourth negative clock edge arriving, the stored word in the register is

$$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 1\ 1\ 1$$

– Table 4.5.3 summarizes the shift right operation.

(C-744) Table 4.5.3 : Summary of shift right operation

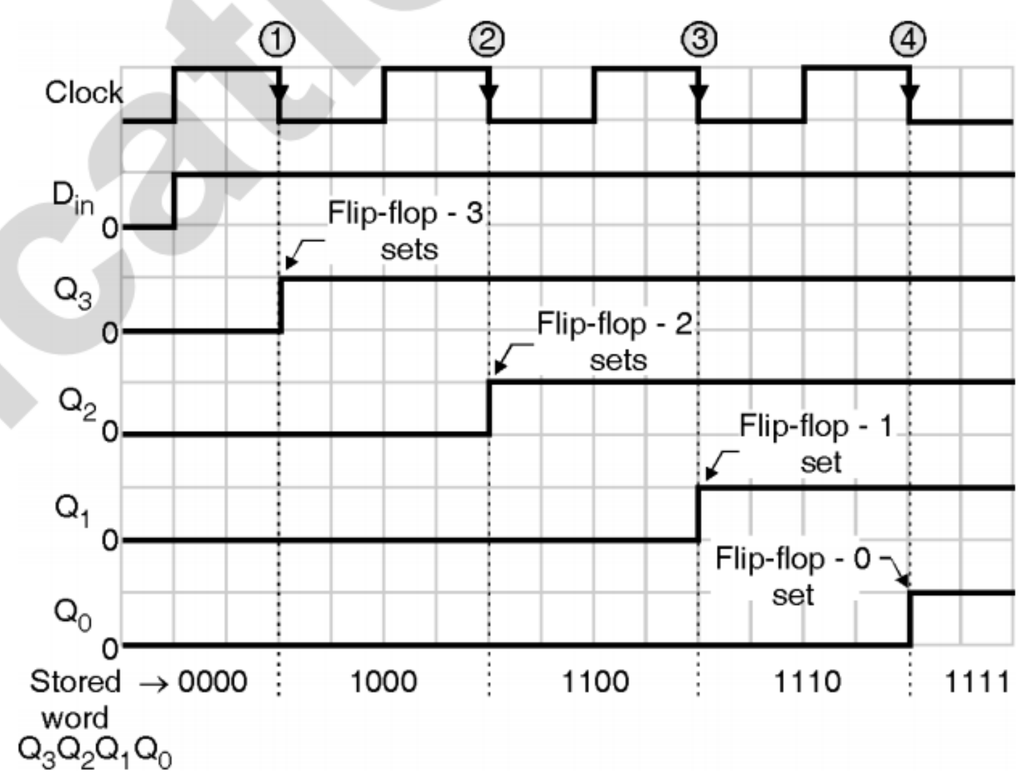
	CLK	$D_{in} = D_3$	$Q_3 = D_2$	$Q_2 = D_1$	$Q_1 = D_0$	Q_0
Initially			0	0	0	0
1st	↓	1	1	0	0	0
2nd	↓	1	1	1	0	0
3rd	↓	1	1	1	1	0
4th	↓	1	1	1	1	1

→ Direction of data travel

Waveforms for shift right operation :

– The waveforms for shift right operation are as shown in Fig. 4.5.8.

Important note : Using the SISO mode, we needed 4-clock pulses to store a 4-bit word. So in general we can conclude that it requires n number of clock pulses to store an n bit word using SISO mode.



(C-745) Fig. 4.5.8 : Waveforms for the shift right operation

4.5.3 Applications of Serial Operation :

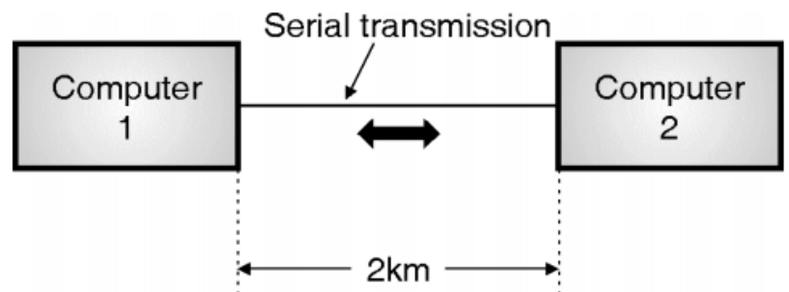
– The transmission of data from one place to the other takes place in serial manner as shown in Fig. 4.5.9. The serial shifting of data transmits one bit at a time per clock cycle.

– It takes a longer time for serial transmission, because the time required to transmit one bit is equal to the time corresponding to one clock cycle.

– The parallel transmission is much faster as it can transmit 8 bits per clock cycle. But it needs more wires for connecting a transmitter to receiver.

TechKnowledge Publications

– However for long distance communication where the distances are in kilometres, serial communication has an advantage that only one conductor is required to be used for the data transfer.

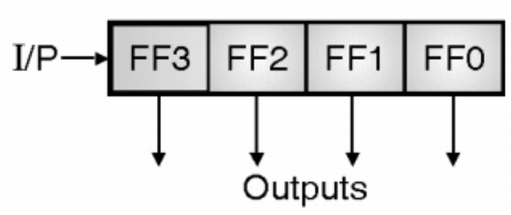


(C-746) **Fig. 4.5.9 : Application of serial operation**

4.6 Serial In Parallel Out (SIPO) :

Principle :

– In this operation the data is entered serially and taken out in parallel as shown in the following figure.



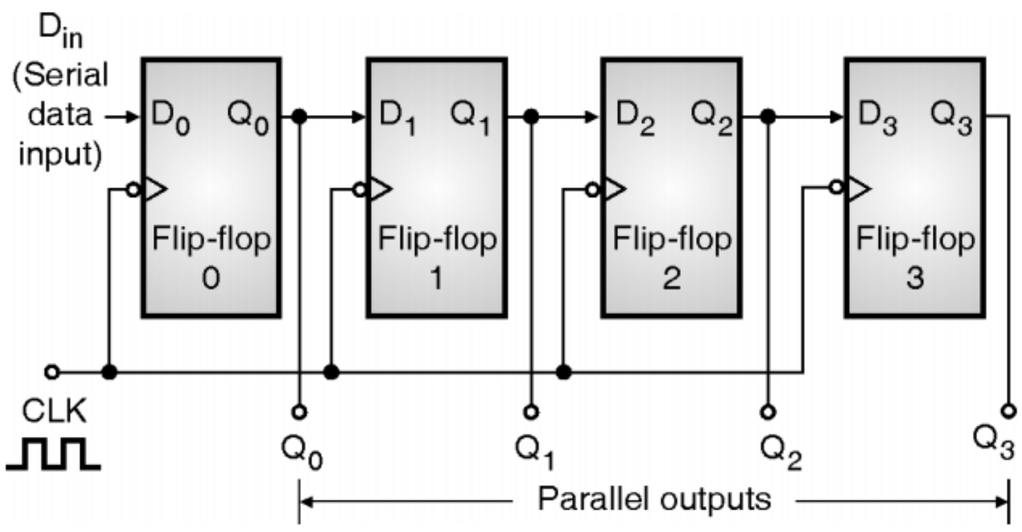
(C-731)

– In this operation the data is entered serially and taken out in parallel.

– That means first the data is loaded bit by bit. The outputs are disabled as long as the loading is taking place.

– As soon as the loading is complete, and all the flip-flops contain their required data, the outputs are enabled so that all the loaded data is made available over all the output lines simultaneously.

– Number of clock cycles required to load a four bit word is 4. Hence the speed of operation of SIPO mode is same as that of SISO mode.



(C-747) **Fig. 4.6.1 : Serial input parallel output mode**

4.7 Parallel In Serial Out Mode (PISO) :

SPPU : May 05, Dec. 07, Dec. 08, May 10, May 14

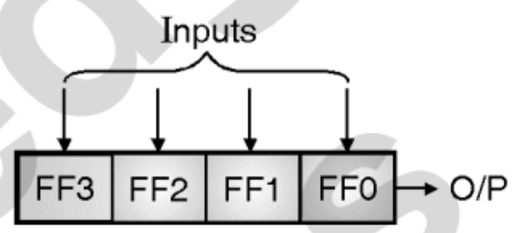
University Questions

Q. 1 Explain with neat diagram working of parallel in serial out 4-bit shift register.

(May 05, Dec. 07, Dec. 08, May 10, May 14, 6 Marks)

Principle :

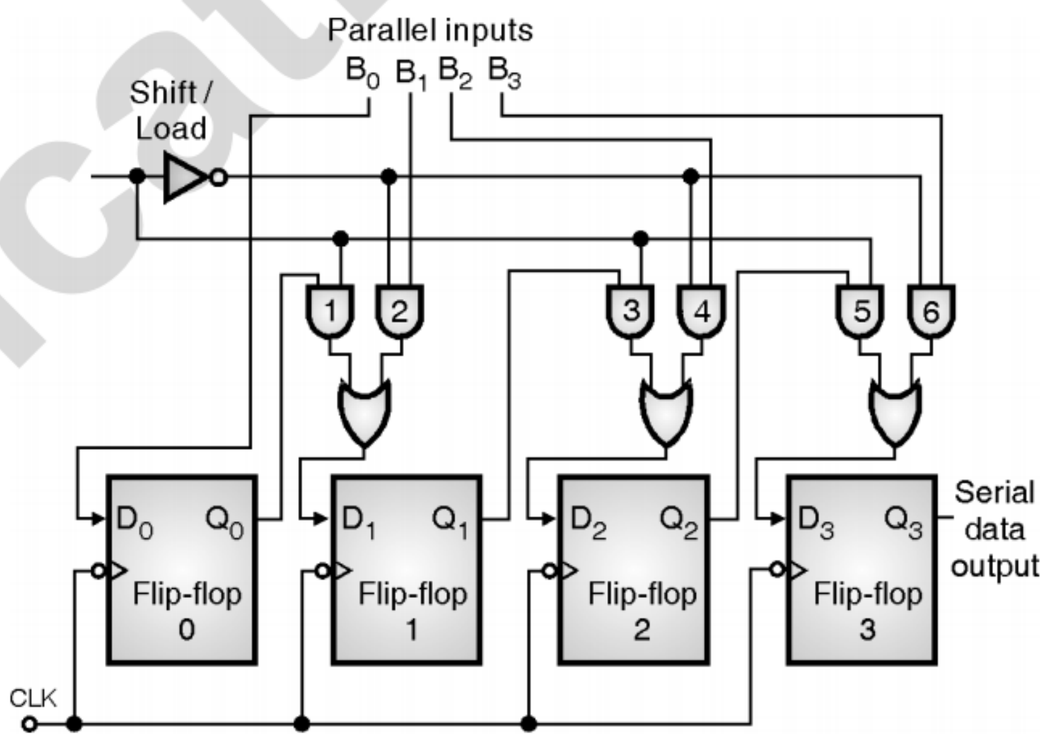
– In this mode, the bits are entered in parallel i.e. simultaneously into a shift register as shown below.



(C-731)

Logic diagram :

– The circuit shown in Fig. 4.7.1 is a four bit parallel input serial output register.



(C-748) **Fig. 4.7.1 : Parallel in serial out shift register**

– Output of previous FF is connected to the input of the next one via a combinational circuit.

– The binary input word B_0, B_1, B_2, B_3 is applied through the same combinational circuit.

– There are two modes in which this circuit can work namely shift mode or load mode.

Load mode :

– In order to load the word B_3 B_2 B_1 B_0 into the shift register we have to select the load mode by setting shift/ $\overline{\text{load}}$ input to 0.

TechKnowledge Publications

– When the shift / $\overline{\text{load}}$ line is low (0), the AND gates 2, 4 and 6 become active.

– They will pass B_1, B_2 and B_3 bits to the inputs D_1, D_2 and D_3 of the corresponding flip-flops. D_0 is directly connected to B_0.

– On the low going edge of clock, the binary inputs B_0 B_1 B_2 B_3 will get loaded into the corresponding flip-flops. Thus parallel loading takes place.

Shift mode :

– In order to operate the shift register in the shift mode we have to select the shift mode by applying a logic 1 to the shift/ $\overline{\text{load}}$ input.

– When the shift / $\overline{\text{load}}$ line is high (1), the AND gates 2, 4, 6 become inactive. Hence the parallel loading of the data becomes impossible.

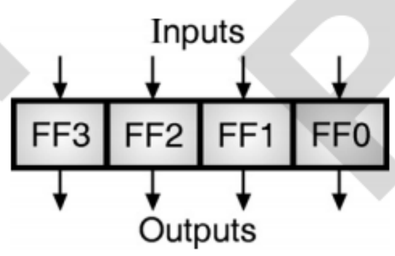
– But the AND gates 1, 3 and 5 become active. Therefore the shifting of data from left to right bit by bit on application of clock pulses becomes possible. D_0 acts as the data input terminal and at Q_3 we get the serial data output.

– Thus the parallel in serial out operation takes place.

4.8 Parallel In Parallel Out (PIPO) :

Principle :

– All inputs are loaded simultaneously and are available at the output simultaneously as shown below.



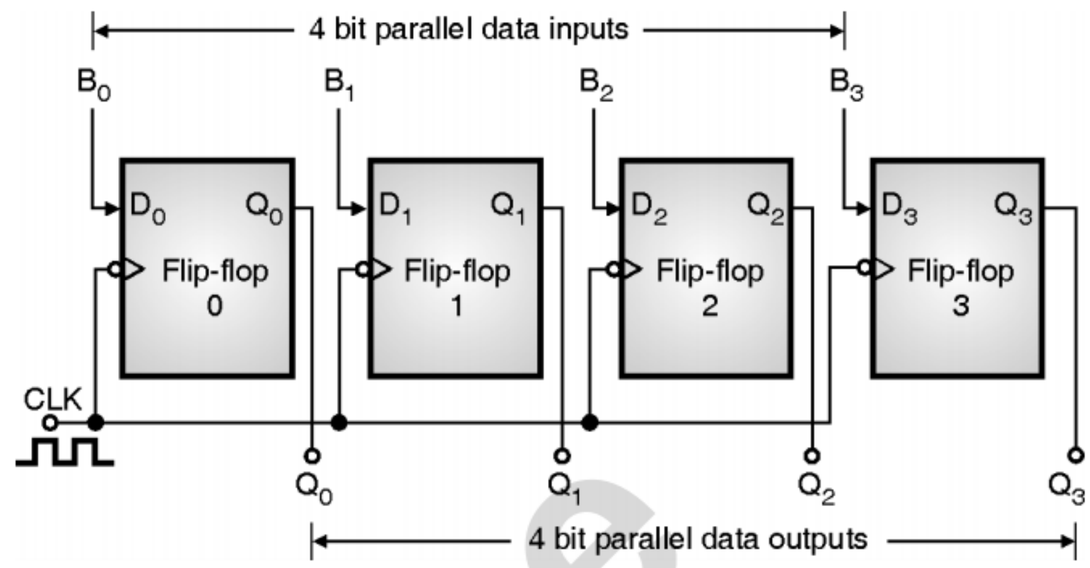
(C-3167) **Fig. 4.8.1(a) : Principle of PIPO register**

Logic diagram :

– Fig. 4.8.1(b) demonstrates the parallel in parallel out mode of operation.

– The 4 bit binary input B_0 B_1 B_2 B_3 is applied to the data inputs D_0, D_1, D_2 and D_3 respectively of the four flip-flops.

– As soon as a negative clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously.



(C-749) **Fig. 4.8.1(b) : Parallel in parallel out shift register**

– The loaded bits will appear simultaneously to the output side. Only one clock pulse is essential to load all the bits. Therefore PIPO mode is the fastest mode of operation.

4.9 Bidirectional Shift Register :

SPPU : May 06, Dec. 11

University Questions

Q. 1 Draw and explain 4-bit shift register having shift left and shift right facilities. Explain any one application of such register. **(May 06, 6 Marks)**

Q. 2 Draw and explain 4 bit bidirectional shift register. **(Dec. 11, 8 Marks)**

– If a binary number is shifted left by one position then it is equivalent to multiplying the original number by 2.

– On the other hand if a binary number is shifted right by one position then it is equivalent to dividing the original number by 2. This is illustrated below.

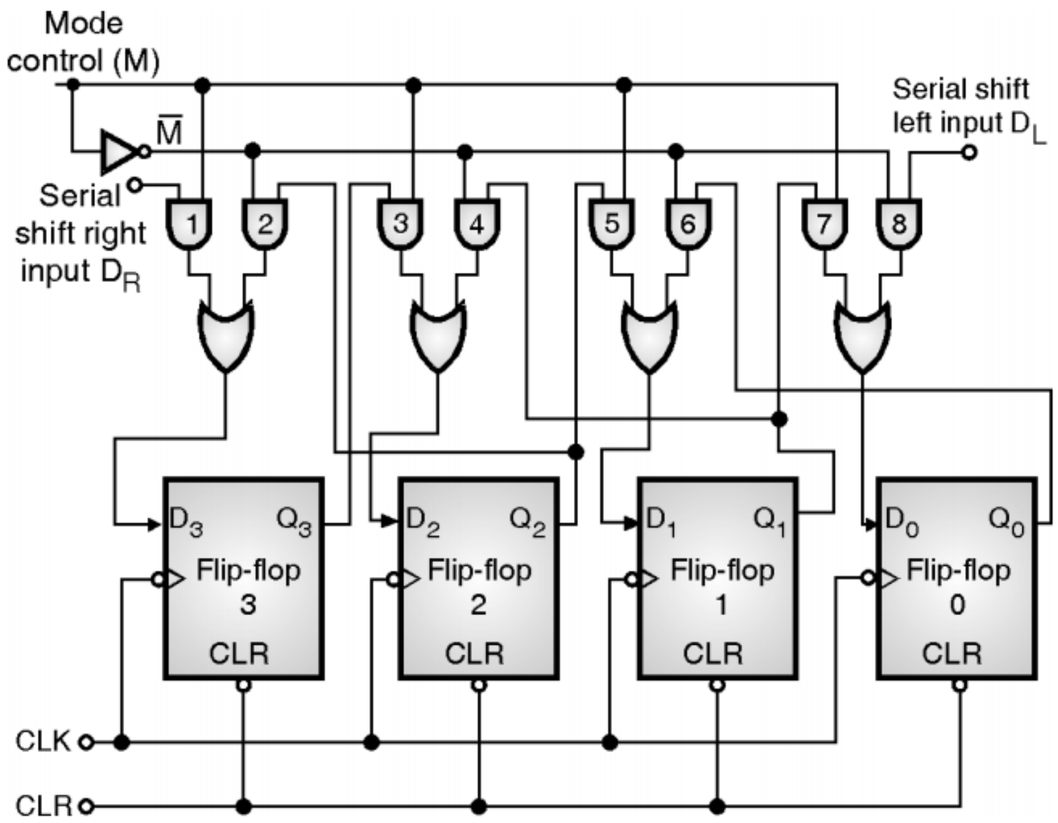
– Let a four bit number Q_3 Q_2 Q_1 Q_0 = 0010 = $(2)_{10}$ is existing in a shift register.

– Now with a 0 applied at the input, if we shift these contents by one position to left then we get Q_3 Q_2 Q_1 Q_0 = 0100 = $(4)_{10}$.

– Thus the shift left is equivalent to multiplying by 2. Now shift it right by one position to get Q_3 Q_2 Q_1 Q_0 = 0001 = $(1)_{10}$. Thus shifting right is equivalent to dividing by 2.

– Hence if we want to use the shift register to multiply and divide the given binary number, then we should be able to move the data in either left or right direction as and when we want.

TechKnowledge Publications



(C-750) Fig. 4.9.1 : A 4-bit bi-directional shift register

- Such a register is called as a bi-directional register. A four bit bi-directional shift register is shown in Fig. 4.9.1.
- There are two serial inputs namely the serial right shift data input D_R and the serial left shift data input D_L alongwith a Mode control input (M).

Operation :

With M = 1 : Shift right operation

- If M = 1, then the AND gates 1, 3, 5 and 7 are enabled whereas the remaining AND gates 2, 4, 6 and 8 will be disabled.
- Hence the data at D_R (shift right input) is shifted right bit by bit from FF-3 to FF-0 on the application of clock pulses.
- Thus with M = 1 we get the serial right shift operation.

With M = 0 : Shift left operation

- When the mode control M is connected to "0" then the AND gates 2, 4, 6 and 8 are enabled while 1, 3, 5 and 7 are disabled.
- Therefore the data at D_L (shift left input) is shifted left bit by bit from FF-0 to FF-3 on application of the clock pulses.
- Thus with M = 0 we get the serial left shift operation.
- Note that M should be changed only when CLK = 0, otherwise the data stored in the register may get changed in an undesirable manner.

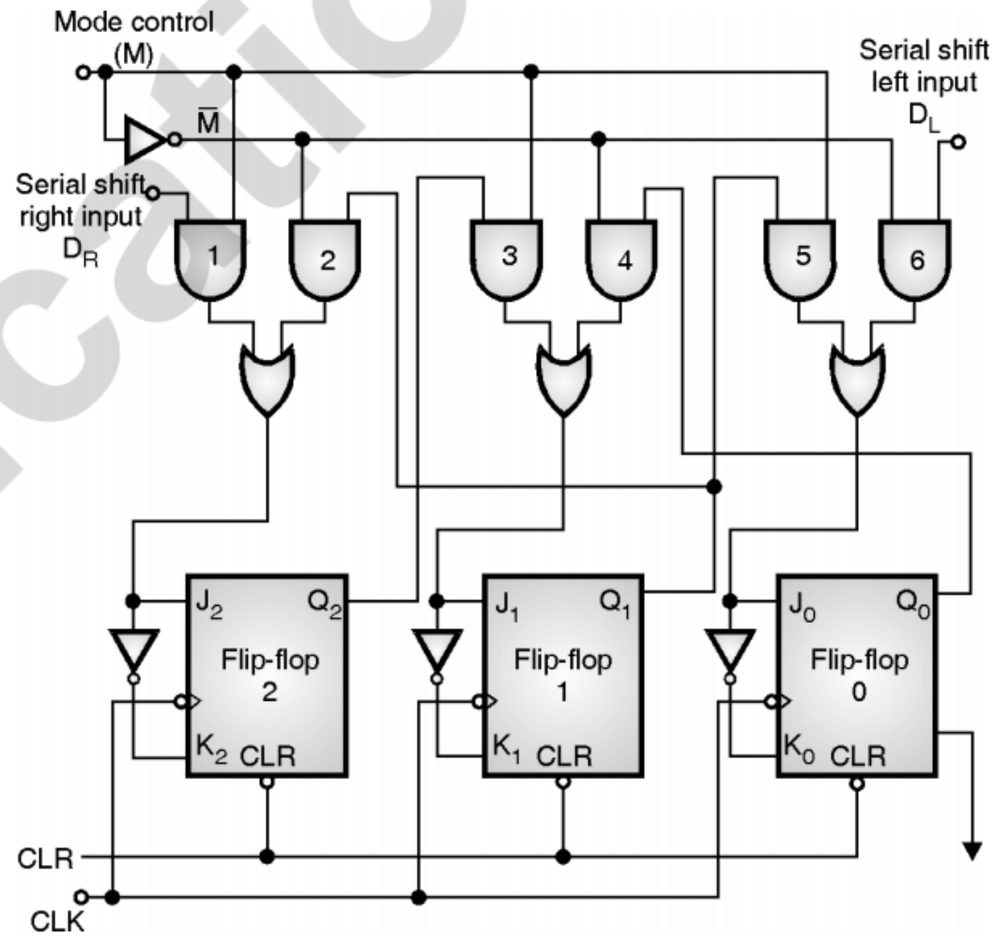
4.9.1 A 3-bit Bidirectional Register using the JK Flip Flops : SPPU : Dec. 05

University Questions

Q. 1 Draw the circuit diagram of 3-bit bi-directional shift register. **(Dec. 05, 6 Marks)**

Logic diagram :

- A 3-bit bi-directional shift register using JK flip flops is shown in Fig. 4.9.2.
- This circuit is same as the 4 bit bi-directional shift register discussed in the previous section.
- The only modification is the inclusion of an inverter between the J and K inputs of each flip flops.
- With M = 1 the shift right operation will take place as discussed in the previous section and with M = 0 the shift left operation is going to take place.



(C-751) Fig. 4.9.2 : A 3 bit bi-directional shift register using JK flip flops

4.10 Universal Shift Register :

SPPU : May 08

University Questions

Q. 1 Draw and explain 5-bit shift register having parallel in and serial in (left to right) facilities. Explain any one application of such register. **(May 08, 8 Marks)**

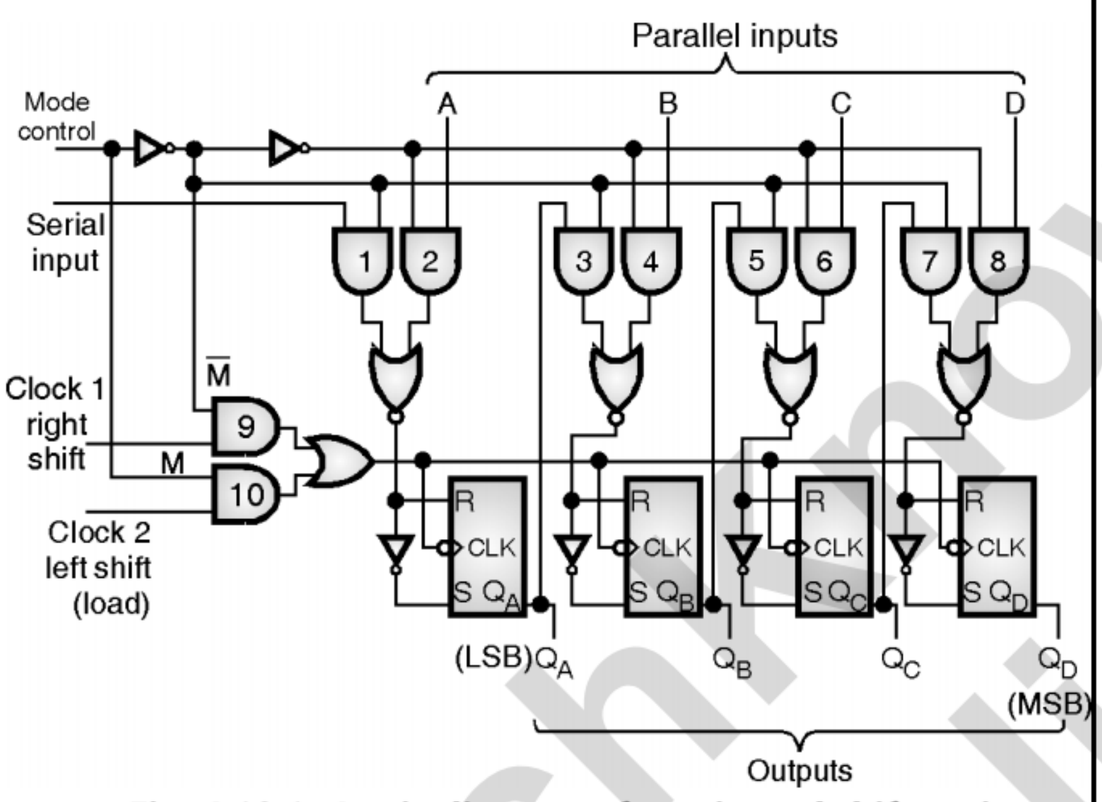
TechKnowledge Publications

Definition :

- A shift register which can shift the data in only one direction is called as a unidirectional shift register.
- A shift register which can shift the data in both the directions is called as a bi-directional shift register.
- Applying the same logic, a shift register which can shift the data in both the directions (shift right or left) as well as load it parallely, then it is called as a **universal shift register.**

Logic diagram :

- Fig. 4.10.1 shows the logic diagram of a universal shift register.



(C-752) **Fig. 4.10.1 : Logic diagram of a universal shift register**

- This shift register is capable of performing the following operations :

1. Parallel loading (parallel input parallel output).
2. Left shifting.
3. Right shifting.

- The mode control input is connected to Logic 1 for parallel loading operation whereas it is connected to 0 for serial shifting.
- With mode control pin connected to ground, the universal shift register acts as a bi-directional register.
- For serial left operation, the input is applied to the serial input which goes to AND gate-1 in Fig. 4.10.1.
- Whereas for the shift right operation, the serial input is applied to D input (input of AND gate 8).
- The well known example of universal shift register in the IC form is IC7495.

4.10.1 Universal Shift Register using Multiplexers and D-flip flops :

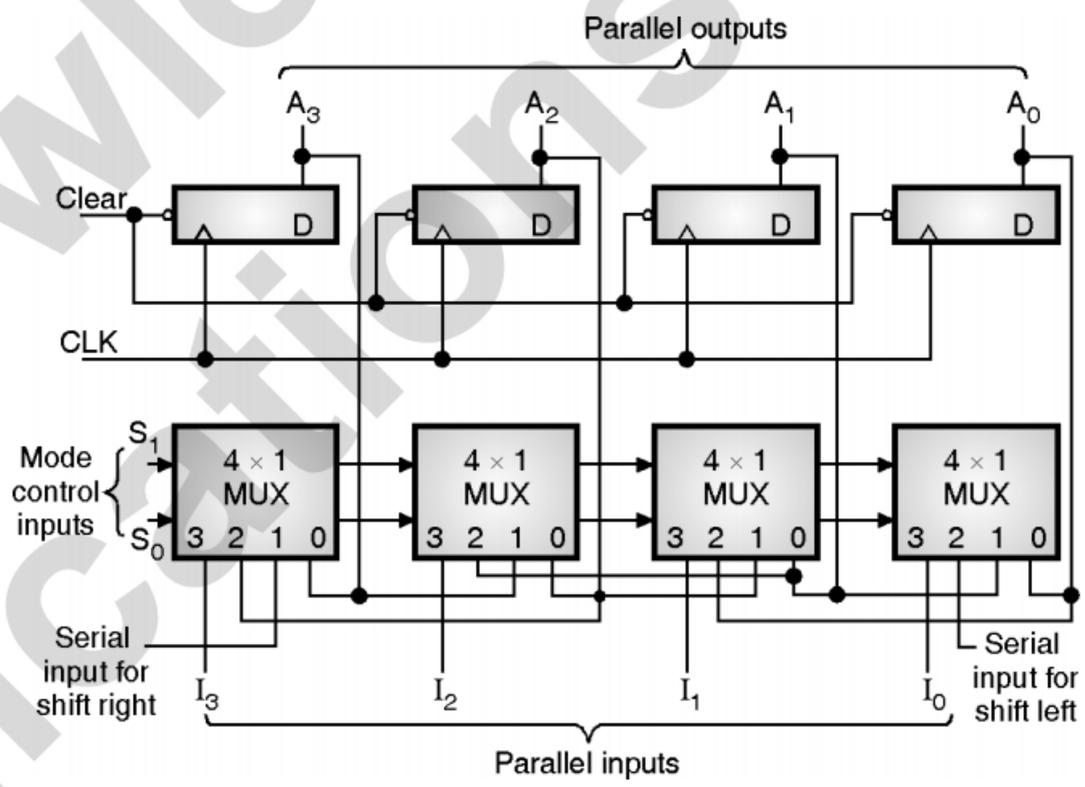
SPPU : Dec. 06

University Questions

Q. 1 Draw and explain 4-bit shift register having shift left, shift right and parallel in facilities. Use suitable multiplexers in your circuit. Explain any one application of such register. **(Dec. 06, 8 Marks)**

Logic diagram :

- Fig. 4.10.2 shows the diagram of a 4 bit universal shift register. It consists of four D-flip flops and four multiplexers.



(C-753) **Fig. 4.10.2 : 4-Bit universal shift register**

Operation :

- The S_1 and S_0 inputs are the select lines of all the four multiplexers and they are connected together. These lines act as the **mode control inputs.**
- The mode of operation is dependent on the status of the S_1 and S_0 lines as shown in Table 4.10.1.

(C-8076) **Table 4.10.1 : Function table of universal shift register**

Mode control		Register
S_1	S_0	Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

TechKnowledge Publications

1. Operation for S_1 S_0 = 00 (No change) :

– When S_1 S_0 = 00 input 0 of each multiplexer is selected. As the output of each FF is connected to input 0 of the corresponding MUX, the present value of the register is applied to D inputs of the flip flops.

– On the next positive clock edge, this present value gets transferred to the output. That means, the next state will be same as the present state. Thus there is **no change** in state for S_1 S_0 = 00.

2. Operation for S_1 S_0 = 01 (Right shift) :

– When S_1 S_0 = 01, terminal 1 of multiplexer input has a path to flip flop input D. This will cause the shift right operation.

3. Operation for S_1 S_0 = 10 (Left shift) :

– When S_1 S_0 = 10, the input 2 of each multiplexer get connected to the D input of corresponding flip flop.

– This will result in shift left operation.

4. Operation for S_1 S_0 = 11 (Parallel load) :

– When S_1 S_0 = 11, the input 3 of each multiplexer gets connected to the D input of corresponding flip flop.

– This will result in the parallel load operation.

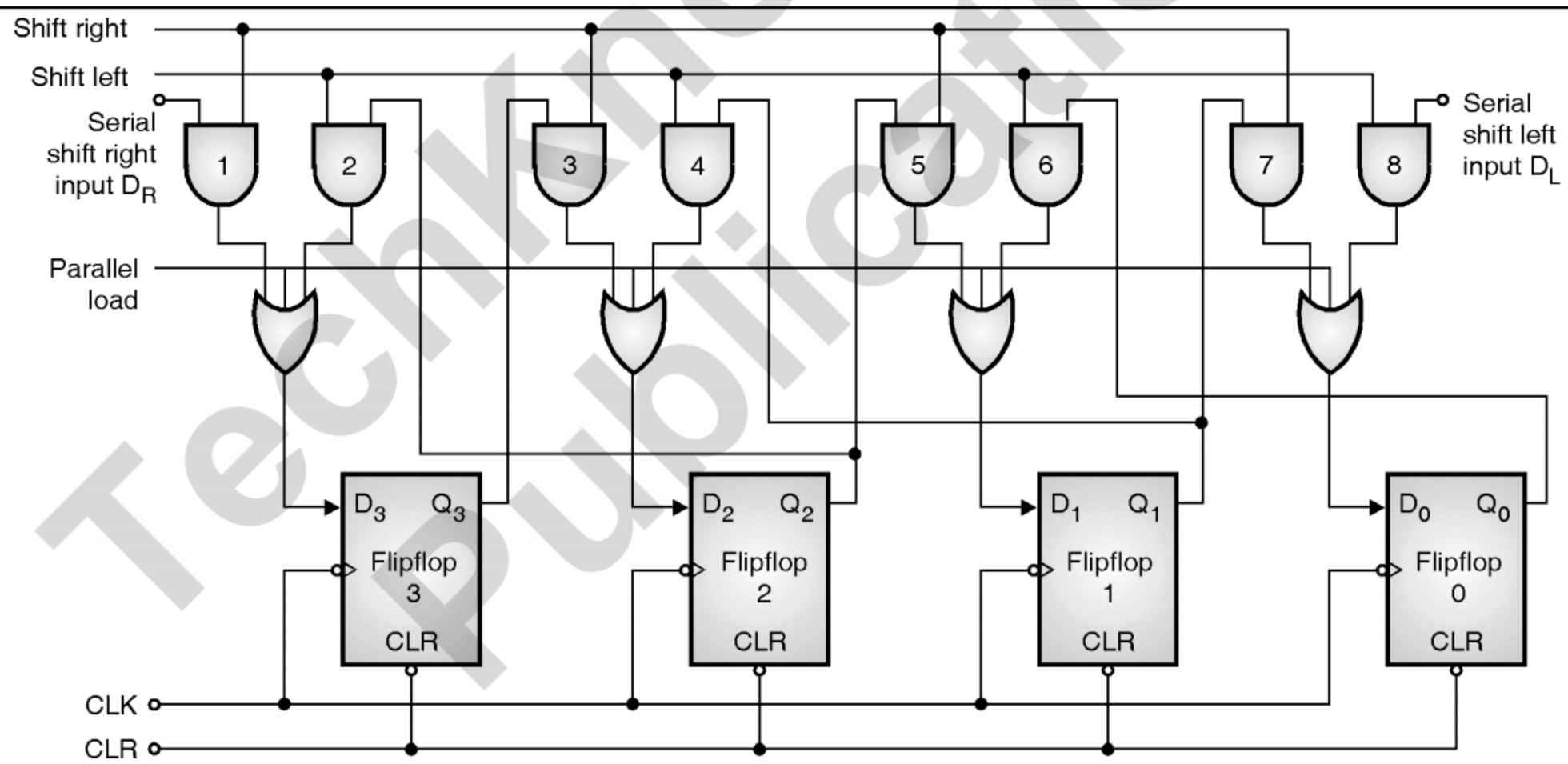
Ex. 4.10.1 : Design 4-bit shift register with the following facilities :

1. Shift-left (if L1 = 0 and L0 = 0)
2. Shift-right (if L1 = 0 and L0 = 1)
3. Parallel-in (if L1 = 1 and L0 = 1)

Control signals 'L1 L0' will select one of the possible operation. **May 07, 10 Marks**

Soln. :

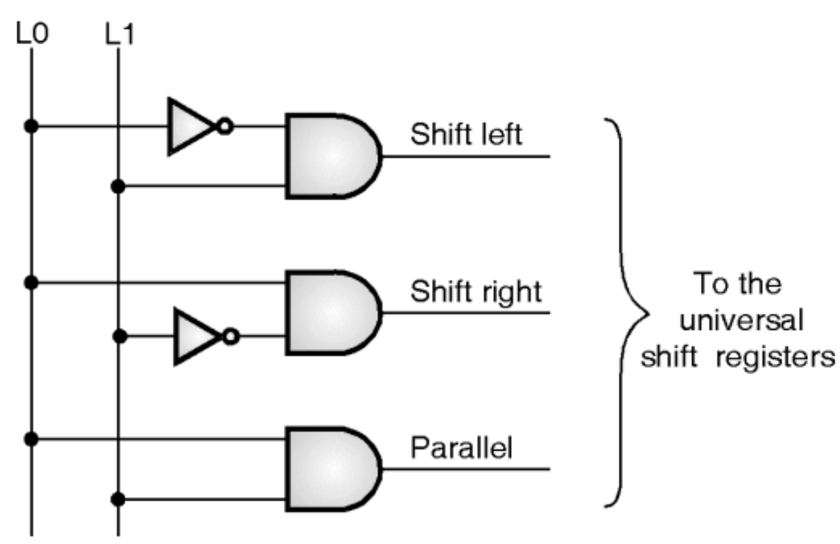
– The required universal shift register is shown in Fig. P. 4.10.1(a), whereas the combinational circuit required for the generation of operation selecting signals is shown in Fig. P. 4.10.1(b).



(C-3560) **Fig. P. 4.10.1(a) : Universal 4-bit shift register**

(C-8077) **Table P. 4.10.1 : Control signals**

Inputs		Outputs		
L0	L1	Shift left	Shift right	Parallel
0	0	–	–	–
0	1	1	0	0
1	0	0	1	0
1	0	0	0	1



(C-3561) **Fig. P. 4.10.1(b) : Control logic**

4.11 Applications of Shift Registers :

SPPU : Dec. 04, May 06, May 10

University Questions

Q. 1 Explain the applications of universal shift register. **(Dec. 04, 6 Marks)**

Q. 2 Draw and explain 4-bit shift register having shift left and shift right facilities. Explain any one application of 4-bit shift register. **(May 06, 6 Marks)**

Q. 3 Give any four applications of shift registers. **(May 10, 4 Marks)**

- Some of the common applications of a shift register are :

1. For temporary data storage.
2. For multiplication and division.
3. As a delay line.
4. Serial to parallel converter.
5. Parallel to serial converter.
6. Ring counter.
7. Twisted ring counter or Johnson counter.
8. Serial data transmission.

- We have already seen the use of shift register for temporary storage of data and for multiplication or division.

4.11.1 Serial to Parallel Converter :

SPPU : Dec. 08

University Questions

Q. 1 Explain how shift registers are used as Serial to parallel converters ? **(Dec. 08, 3 Marks)**

- In many applications, the data in serial form needs to be converted in the parallel form.
- The Serial Input Parallel Output Mode (SIPO) of the shift register is used for such applications.

4.11.2 Parallel to Serial Converter :

SPPU : Dec. 08

University Questions

Q. 1 Explain how shift registers are used as Parallel to serial converter ? **(Dec. 08, 3 Marks)**

- The data communication between two computers takes place in the serial transmission form.
- But internally the data processing takes place in the parallel form.
- Hence we need to use a parallel to serial converter to convert the internal parallel data into a serial data suitable for transmission.
- We can use the parallel to serial mode of the shift register for this operation.

4.11.3 Ring Counter :

- Refer Chapter 5 for this topic.

4.11.4 Johnson's Counter (Twisted / Switch Tail Ring Counter) :

- Refer Chapter 5 for this topic.

4.11.5 Sequence Generator :

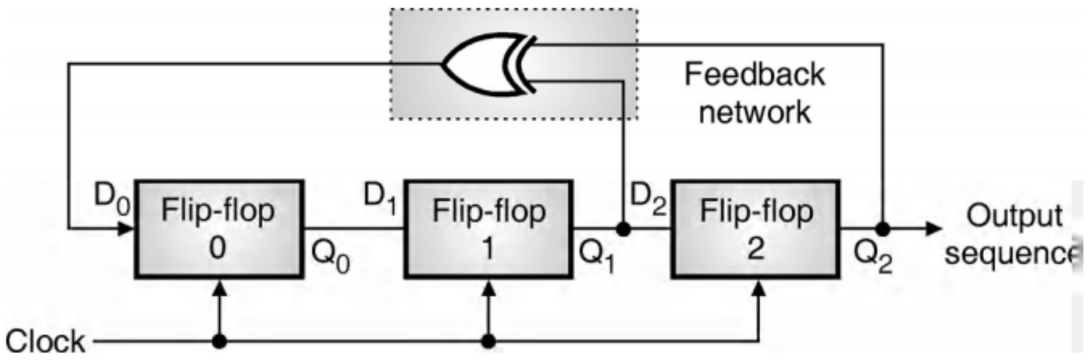
- A sequence generator is a sequential circuit which generates a desired sequence at its output. The output sequence is in synchronization with the clock input.
- It is possible to design a sequence generator using counters or using the shift registers.

4.11.6 Pseudo Random Binary Sequence (PRBS) Generator :

- Another important application of a shift register is the pseudo random generator. It is used for generating the random sequences.
- The PRBS generator consists of a number of flip-flops and a combinational circuit which provides a suitable feedback.
- The feedback circuit can be as simple as an EX-OR gate as shown in Fig. 4.11.1.
- Fig. 4.11.1 shows a 3-bit PRBS generator. It is called as a **random generator** because the outputs Q_2 Q_1 Q_0 do not follow the normal binary sequence from 000 to 111 or vice versa.
- The sequence followed by them depends on the feedback network and for this particular circuit, the sequence is as shown in Table 4.11.1.

TechKnowledge Publications

- The meaning of **Pseudo** is **fake** or "which appears to be". So the pseudo term in PRBS indicates that the generator appears to be random but in reality it is not random.
- This is because the output sequence cycles through all the possible combinations once every $2^n - 1$ clock cycles where n is the number of flip flops.
- The PRBS generator is as shown in Fig. 4.11.1. This is basically a shift register.



(C-1731) Fig. 4.11.1 : PRBS sequence generator

- Type D flip-flops are being connected such that the D input to a flip-flop is connected to the Q output of the previous flip-flop.
- The input D_0 of the first D flip-flop has been connected to the output of the feedback network.
- A feedback network generally consists of exclusive - OR gates.
- Therefore the output of the feedback network is equal to zero when an even number of its inputs Q_0, Q_1, ... are at zero level.
- And its output is equal to 1 when an odd number of its inputs are at logic 1 level.
- The inputs to the feedback network are the outputs from the flip-flops i.e. Q_0, Q_1, ... Q_m.
- However it is not necessary to connect all the Q outputs to the input of parity generator.

Output of feedback network	Status of Inputs
Logic "0"	Even number of inputs are at logic "0"
Logic "1"	Odd number of inputs are at logic "1"

- The character generated by a PRBS generator depends on the number of flip-flops used (m) and on the selection of which flip-flop outputs are connected to the inputs of feedback network.
- The state of each flip-flop changes and gets shifted to the next flip-flop corresponding to each pulse of the clock.
- A shift register of "m" flip-flops will have 2^m number of states; i.e. Q_0, Q_1 ... Q_{m-1} = 000 ... 0 to Q_0, Q_1 .. Q_{m-1} = 111 ... 1. Thus the output sequence will repeat itself after every 2^m bits.
- The PRBS generator of Fig. 4.11.1 cannot generate a truly random sequence because this structure is a deterministic structure. This is the reason why, the sequence repeats itself.
- In order to make the random sequence "look like" truly random, its length should be sufficiently large i.e. a large number of flip flops should be included. Typically upto 2000 flip flops are used.
- The maximum length of the sequence will be $2^m - 1$. This is because the state 000...0 is not to be considered.
- If all zero state is allowed to exist then the EX-OR gates used in the parity generator will produce a zero output all the time.
- To avoid this the all zero state should be excluded.

The Maximum Length Sequence :

- The maximum length sequence is a type of cyclic code which represents a commonly used periodic PRBS generator.
- Such a sequence has long periods and requires a linear feedback shift register for its generation. A PRBS generator with m-register linear feedback shift register produces a sequence of length $2^m - 1$ as discussed earlier.

Operation of PRBS generator :

- Assume that the initial state of the shift register is

  $Q_3\, Q_2\, Q_1 = 001$

- The outputs Q_2 and Q_3 are connected to a modulo-2 adder i.e. an EX-OR gate. Table 4.11.1 summarizes the operation of the PRBS generator.

TechKnowledge Publications

(E-574) **Table 4.11.1 : Operation of the PN sequence generator**

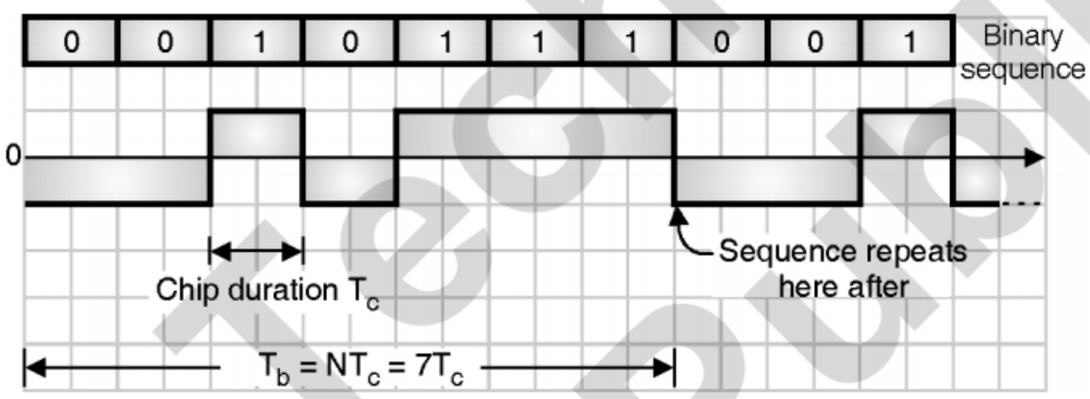
Clock Pulse Number	Shift register outputs Q_3	Q_2	Q_1	EX-OR gate output $Q_3 \oplus Q_2$	PN sequence Q_3
0	0	0	1	$0 \oplus 0 = 0$	0
1	0	1	0	$0 \oplus 1 = 1$	0
2	1	0	1	$1 \oplus 0 = 1$	1
3	0	1	1	$0 \oplus 1 = 1$	0
4	1	1	1	$1 \oplus 1 = 0$	1
5	1	1	0	$1 \oplus 1 = 0$	1
6	1	0	0	$1 \oplus 0 = 1$	1
7	0	0	1	$0 \oplus 0 = 0$	0
8	0	1	0	$0 \oplus 1 = 1$	0
9	1	0	1	$1 \oplus 0 = 1$	1
10	0	1	1	$0 \oplus 1 = 1$	0

The sequence repeats after this.

– The waveform obtained at the output of a PRBS generator is called as PRBS sequence. It is shown in Fig. 4.11.2. It is obtained at output Q_3.

Important observations :

– Some of the important observations from Fig. 4.11.2 are as follows :



(E-475) **Fig. 4.11.2 : PRBS sequence obtained at Q_3**

1. The length of the PRBS sequence obtained at the Q_3 output is equal to $2^m - 1$. In this case m = 3 therefore the maximum length of the sequence is $N = 2^3 - 1 = 7$
2. The PRBS sequence repeats itself after every 7 clock cycles.
3. The PRBS sequence is an NRZ type signal with logic 1 represented by + 1 and binary 0 is represented by – 1.
4. The duration of every bit is known as the **chip duration** T_c and the **chip rate** R_c is defined as the number of bits (chips) per second.

$$\therefore \quad T_c = \frac{1}{R_c} \text{ OR } R_c = \frac{1}{T_c}$$

5. The period of the PRBS sequence is given by,

$$T_b = N\,T_c$$

– Note that the selection of 100 as the initial state is random. Any other state of the remaining six can be treated as the initial state.

Applications of PRBS generator :

– Since the sequence produced is random, the PRBS generator is also called as a Pseudo Noise (PN) generator and the generated signal is called noise. This noise can be used to test the noise immunity of the system under test.

– PRBS generator is an important part of the data encryption system. Such a system is required to protect the data from data hackers.

Review Questions

Q. 1 What is the function of a shift register ? Give its applications.

Q. 2 State the types of shift registers.

Q. 3 With a neat diagram explain the operation of 4-bit left shift register. Give its truth table and timing diagram.

Q. 4 With a neat diagram explain the operation of 4-bit SISO (Serial-In-Serial-Out) register. Draw the timing diagram and give its truth table.

Q. 5 With a neat diagram explain the operation of 4-bit Serial-In-Parallel Out (SIPO) register. Give the truth table and timing diagram.

Q. 6 With a neat diagram explain the operation of 4-bit Parallel-In-Serial-Out (PISO) register. Give the truth table and timing waveforms.

Q. 7 What is meant by "Universal shift register" ?

Q. 8 What do you understand by a bi-directional shift register ? Explain its operation.

Q. 9 List any one shift register IC. Sketch its schematic diagram and pin configuration. Give its specifications.

Q. 10 Draw circuit diagram of 3 bit SIPO shift register. Use D flip flops. Explain its working.

Q. 11 Define bi-directional shift register. Draw 3 bit bi-directional shift register using D flip flop.

❑❑❑

TechKnowledge Publications

Unit III

Chapter 5

Counters

Technical Publications

Syllabus

Asynchronous counter, Synchronous counter, Ring counter, BCD counter, Johnson's counter, Modulus of the counter (IC 7490).

Chapter Contents

5.1	Introduction	5.9	Modulo – N Synchronous Counters
5.2	Asynchronous / Ripple Up Counters	5.10	UP / DOWN Synchronous Counter
5.3	Asynchronous Down Counters	5.11	Lock Out Condition
5.4	UP / DOWN Counters	5.12	Bush Diagram
5.5	Modulus of the Counter (MOD-N Counter)	5.13	Applications of Counters
5.6	Ripple Counter IC 7490 (Decade Counter)	5.14	Ring Counter
5.7	Problems Faced by Ripple Counters	5.15	Johnson's Counter (Twisted / Switch Tail Ring Counter)
5.8	Synchronous Counters		

5.1 Introduction :

Definition :

- The digital circuit used for counting pulses is known as counter. It is a sequential circuit.
- Counters is the most widely used application of flip-flops. It is a group of flip-flops with a clock signal applied.
- Counters count the number of clock pulses. Therefore with some modifications we can use them for measuring frequency or time period.

5.1.1 Types of Counters : SPPU : May 08

University Questions

Q. 1 What do you mean by binary ripple counter ?

(May 08, 2 Marks)

- Counters are basically of two types :

1. Asynchronous or ripple counters.
2. Synchronous counters.

1. Asynchronous or ripple counters :

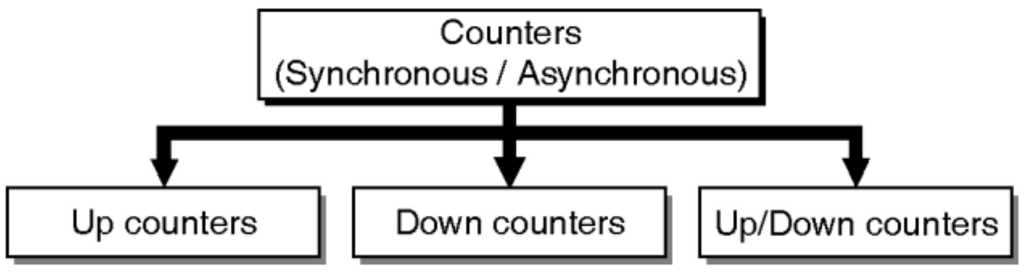
- For these counters the external clock signal is applied to one flip flop and then the output of preceding flip-flop is connected to the clock of next flip-flop.

2. Synchronous counters :

- In synchronous counters all the flip-flops receive the external clock pulse is applied to all the flip-flops simultaneously.
- Ring counter and Johnson counter are the examples of synchronous counters.

5.1.2 Classification of Counters :

- Depending on the way in which the counter outputs change, the synchronous or asynchronous counters are classified as follows :



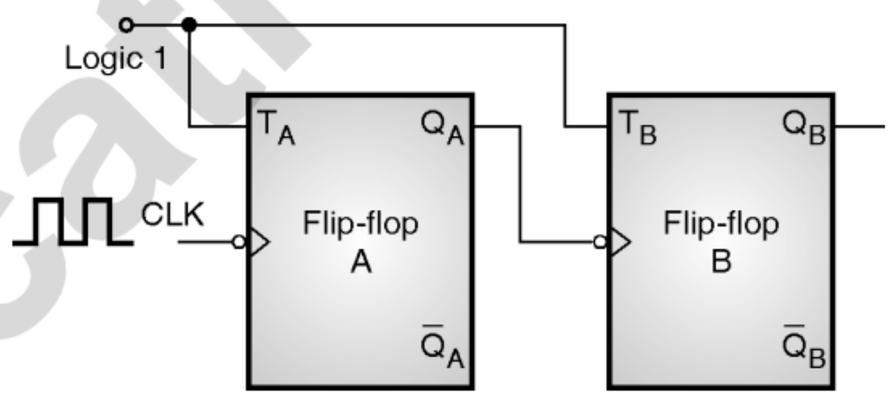
(C-6214)

- **Up counters** are the counters that count from small to big count. Their output goes on increasing as they receive clock pulses.
- For example, the output of an up counter will be 0-1-2-3...
- **Down counters** are the counters that count from large to small count. Their output goes on decreasing as they receive clock pulses.
- For example, the output of a down counter will be 7-6-5-4-3...
- **Up/Down counter** is the combination of up counter and down counter.

5.2 Asynchronous / Ripple Up Counters :

Logic diagram :

- Fig. 5.2.1 shows the logic diagram of a 2-bit ripple up counter.



(C-771) Fig. 5.2.1 : A two bit asynchronous binary up counter

- The number of flip-flops used is 2. Note that the number of bits will always be equal to the number of flip-flops. Thus a 4 bit counter will use four flip-flops.
- The toggle (T) flip-flops are being used. But we can use the JK flip-flops also with J and K connected permanently to logic 1.
- External clock is applied to the clock input of flip-flop A which is the LSB flip-flop and Q_A output is applied to the clock input of the next flip-flop i.e. FF-B.

Operation of the counter :

- Initially let both the flip-flops be in reset condition.

$$\therefore \quad Q_B Q_A = 00$$

On the first negative going clock edge :

- As soon as the first falling edge of the clock hits FF-A, it will toggle as $T_A = 1$. Hence Q_A will become equal to 1.

TechKnowledge Publications

- Q_A is connected to clock input of FF-B. Since Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B.
- There is no change in the status of Q_B because FF-B is a negative edge triggered FF.
- Therefore after the first clock pulse the counter outputs are

  $Q_B\,Q_A = 01$ ..... After the first CLK pulse

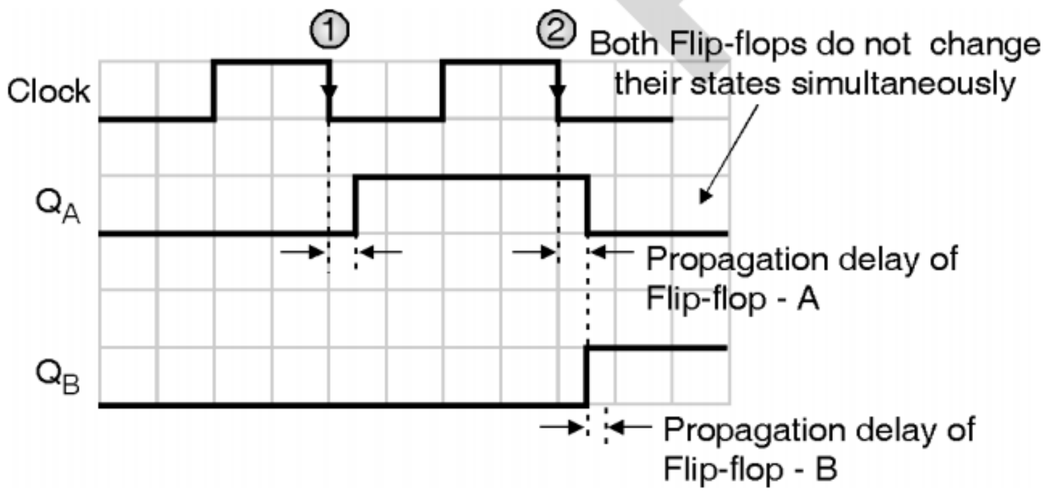
At the second falling edge of clock :

- On the arrival of second falling clock edge, FF-A toggles again and Q_A changes from 1 to 0.

  $\therefore Q_A = 0$ Corresponding to 2nd negative clock edge.
- This change in Q_A (from 1 to 0) acts as a negative clock edge for FF-B. So it will also toggle, and Q_B will change from 0 to 1.

  $\therefore Q_B = 1$
- Hence after the second clock pulse the counter outputs are

  $Q_B\,Q_A = 10$ ..... After the second CLK pulse
- Note that both the outputs are changing their state.
- But both the changes do not take place simultaneously. Q_A will change first from 1 to 0 and then Q_B will change from 0 to 1.
- This is due to the propagation delay of FF-A. So both flip-flops will never trigger at the same instant.
- Therefore the counter is called as an asynchronous counter. This is shown in Fig. 5.2.2.



(C-772) Fig. 5.2.2 : FFs do not change their state simultaneously

At the third falling edge of clock :

- On arrival of the third falling edge, FF-A toggles again and Q_A becomes 1 from 0.
- Since this is a positive going change, FF-B does not respond to it and remains inactive. So Q_B does not change and continues to be equal to 1.

  $\therefore Q_B\,Q_A = 11$ ..... After the third CLK pulse

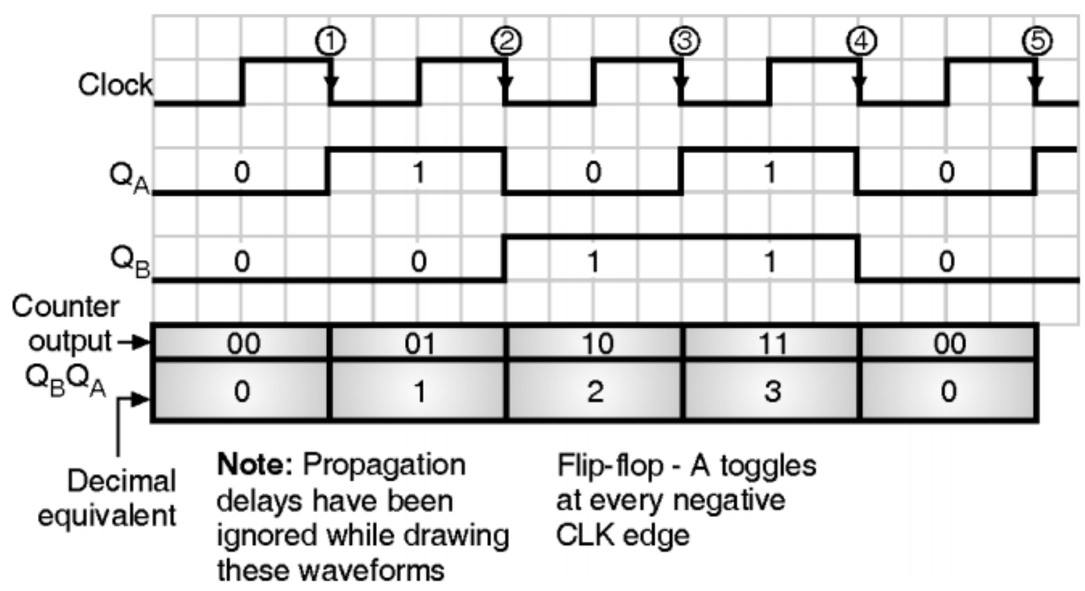
At the 4th negative clock edge :

- On the 4th falling clock edge, FF-A toggles and Q_A changes from 1 to 0.
- This negative going change in Q_A acts as a negative clock pulse for FF-B. Hence it toggles to change Q_B from 1 to 0.

  $\therefore Q_B\,Q_A = 00$ ..... After the fourth CLK pulse
- So the counter has reached the original state. The counter operation will now repeat.
- Table 5.2.1 summarizes the operation of the counter and Fig. 5.2.3 shows the timing waveforms.

(C-6843) Table 5.2.1 : Summary of operation of a 2-bit binary ripple up counter

Clock	Counter outputs		State number	Decimal equivalent of counter output
	Q_B (MSB)	Q_A (LSB)		
Initially	0	0	–	0
1st (↓)	0	1	1	1
2nd (↓)	1	0	2	2
3rd (↓)	1	1	3	3
4th (↓)	0	0	4	0



(C-773) Fig. 5.2.3 : Timing diagram for a 2 bit ripple up counter

Why is it called counter ?

- See Fig. 5.2.3. The decimal count corresponds to the number of clock pulses, which counter has received.
- Thus this circuit counts the clock pulses. Hence it is called as counter.

TechKnowledge Publications

Number of states :

– As seen from Table 5.2.1, this counter has four distinct states of output namely 00, 01, 10 and 11. In general the number of states = 2^n where n is equal to the number of flip-flops.

Maximum count :

– As seen from Table 5.2.1, the maximum count is 3 (decimal) i.e. 1 1 binary.

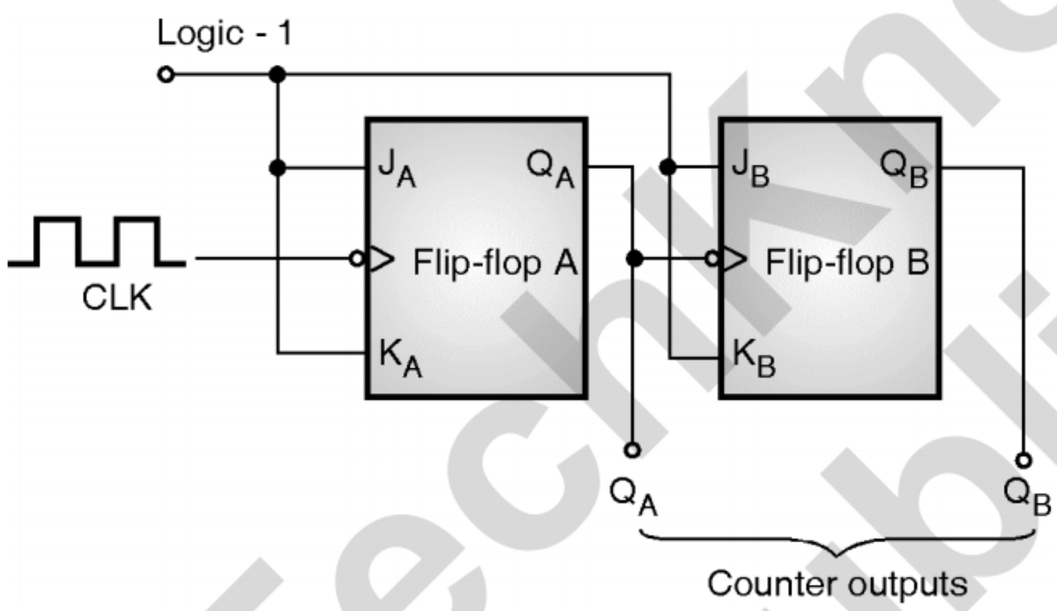
Maximum count = $3 = 2^2 - 1$

– In general the maximum count = $(2^n - 1)$, where n = Number of flip-flops.

5.2.1 Two Bit Asynchronous Up Counter using JK Flip-Flops :

Logic diagram :

– A 2 bit asynchronous counter up using JK flip-flops is shown in Fig. 5.2.4.



(C-774) Fig. 5.2.4 : Two bit ripple up counter using JK flip-flops

– Note that the J and K inputs of both the flip-flops are connected to logic 1 so actually JK flip-flops are converted into T flip-flops.

– The operation of this circuit is exactly same as that of the counter using the T flip-flops.

5.2.2 3 Bit Asynchronous Up Counter :

SPPU : May 07, Dec. 09, Dec. 12, Dec. 16

University Questions

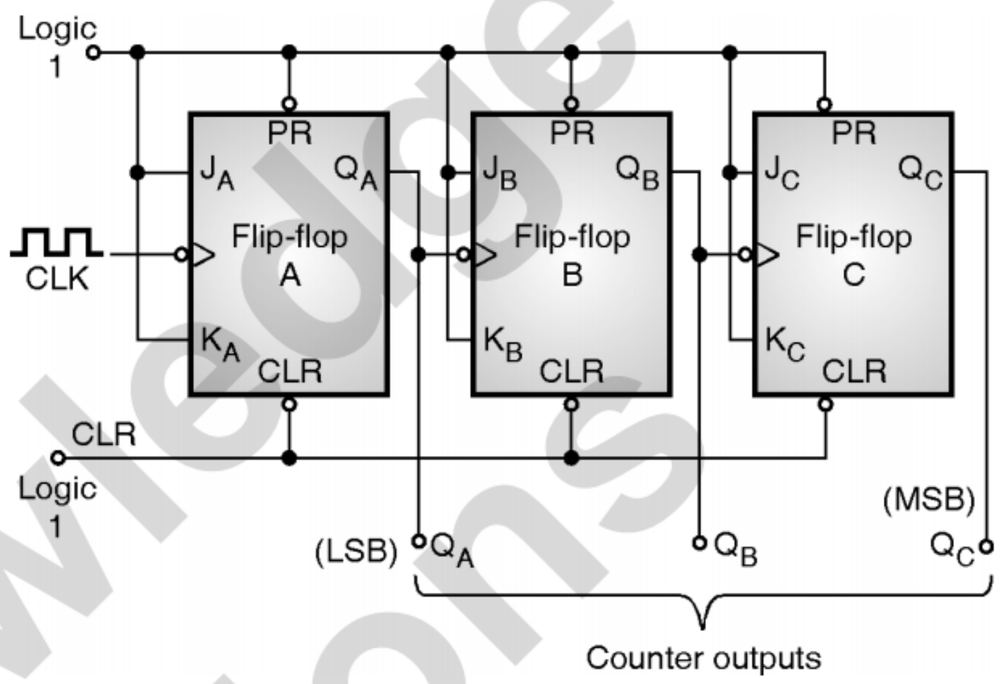
Q. 1 Draw 3-bit asynchronous counter. Explain timing diagram for the same.

(May 07, Dec. 09, Dec. 12, Dec. 16, 8 Marks)

Logic diagram :

– We can apply all the basic concepts which were introduced for the 2-bit ripple up counter to the 3-bit ripple up counter.

– Fig. 5.2.5 shows the logic diagram of a 3-bit ripple up counter. Since it is a 3-bit counter, we need to use 3-flip-flops.



(C-775) Fig. 5.2.5 : 3-bit ripple up counter

– Operation of the 3-bit ripple up counter takes place in exactly similar manner as that of a 2-bit counter.

Truth table :

– Table 5.2.2 summarizes the operation of the 3-bit asynchronous up counter.

(C-776(a)) Table 5.2.2 : Summary of operation of a 3-bit ripple up counter

Clock	Flip-flop outputs			State	Decimal equivalent
	Q_C (MSB)	Q_B	Q_A (LSB)		
Initially	0	0	0	1	0
1st (↓)	0	0	1	2	1
2nd (↓)	0	1	0	3	2
3rd (↓)	0	1	1	4	3
4th (↓)	1	0	0	5	4
5th (↓)	1	0	1	6	5
6th (↓)	1	1	0	7	6
7th (↓)	1	1	1	8	7
8th (↓)	0	0	0	1	0

– Note that the asynchronous preset and clear terminals are also being used.

– Both of them are active low inputs. Hence for the normal output of the counter preset and clear terminals should be connected to logic 1.

Number of states :

– Number of states = $2^n = 2^3 = 8$.

TechKnowledge Publications

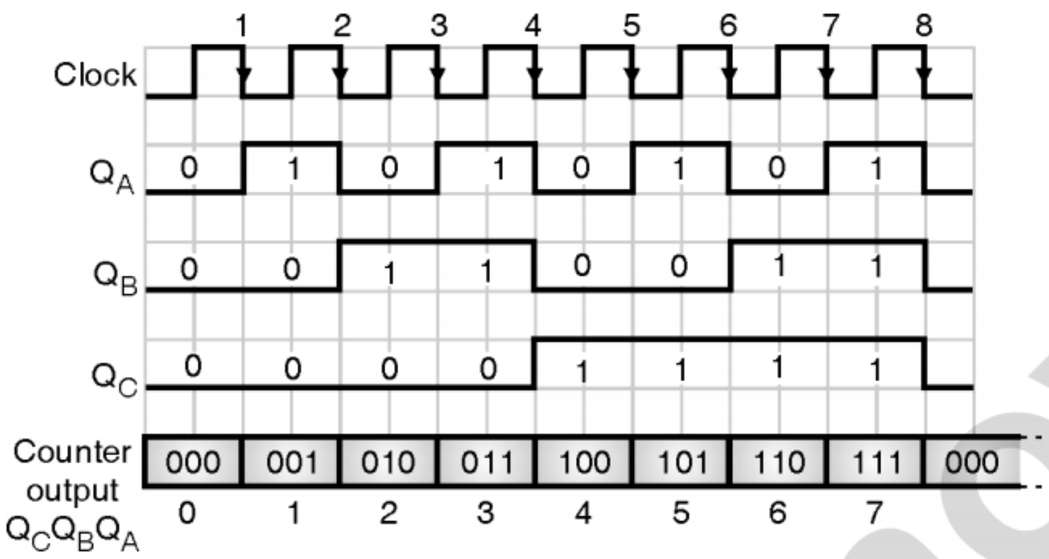
– The 3 bit ripple up counter can have 8 distinct states i.e. Q_C Q_B Q_A can take up values from 000, 001, 010, ...110, 111.

Maximum count :

– Maximum count = $2^n - 1 = 8 - 1 = 7$. Refer Table 5.2.2. The maximum count is Q_C Q_B Q_A = 1 1 1 i.e. decimal 7. Note that Q_C is treated as MSB and Q_A as LSB.

Timing diagram :

– The timing diagram of a 3-bit ripple up counter are as shown in Fig. 5.2.6.



(C-776) Fig. 5.2.6 : Timing diagram for a 3-bit ripple up counter

5.2.3 4 Bit Asynchronous up Counter :

SPPU : May 08, May 11

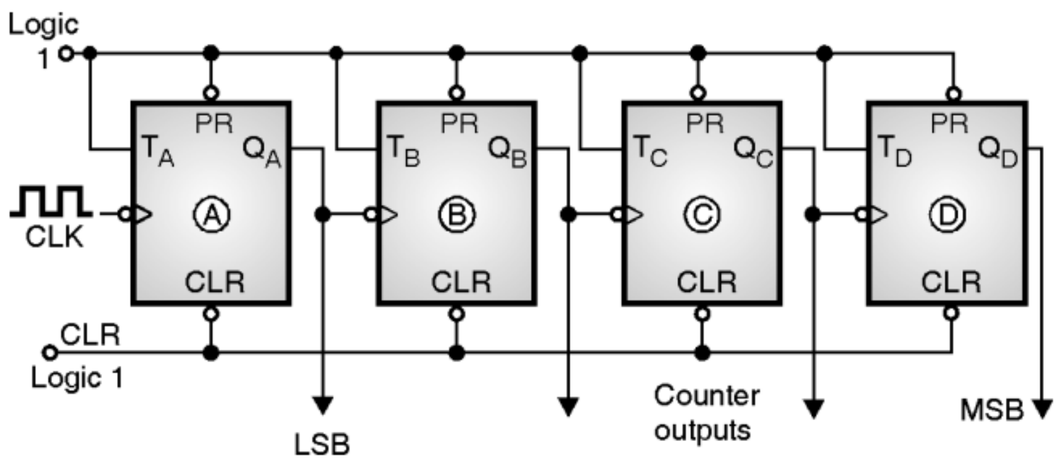
University Questions

Q. 1 Draw 4-bit asynchronous counter. Also explain timing diagram for the same.

(May 08, May 11, 8 Marks)

Logic diagram :

– Fig. 5.2.7(a) shows the circuit diagram of a 4 bit asynchronous counter using the T flip flops.



(C-777) Fig. 5.2.7(a) : 4 bit asynchronous up counter

– Since it is a 4 bit ripple up counter, we need to use four flip flops.

– Initially all the flipflops have zero output.

$\therefore$ $Q_DQ_CQ_BQ_A$ = 0000.

– All the flip flop are negative edge triggered. CLK signal is applied to the clock input of FF-A whereas Q outputs of every F/F is applied to the clock input of next F/F.

– For example Q_A to CLK of FF-B, Q_B to CLK of FF-C and so on.

Truth table :

– Table 5.2.3 shows the truth table for 4 bit asynchronous up counter. Its output passes through 16 states form 0000 i.e. $(0)_{10}$ to (1111) i.e. $(15)_{10}$.

– After 1111, the output again become 0000 and the operation repeats itself.

– Fig. 5.2.7(b) shows the timing diagram for the 4-bit asynchronous up counter.

– Q_D acts as MSB of the output whereas Q_A acts as the LSB.

(C-6844) Table 5.2.3 : Truth table for a 4-bit asynchronous up counter

Clock	FF outputs				Decimal
	Q_D	Q_C	Q_B	Q_A	
Initially	0	0	0	0	0
1st (↓)	0	0	0	1	1
2nd (↓)	0	0	1	0	2
3rd (↓)	0	0	1	1	3
:		:			:
:		:			:
14 (↓)	1	1	1	0	14
15 (↓)	1	1	1	1	15
16 (↓)	0	0	0	0	0

Number of states :

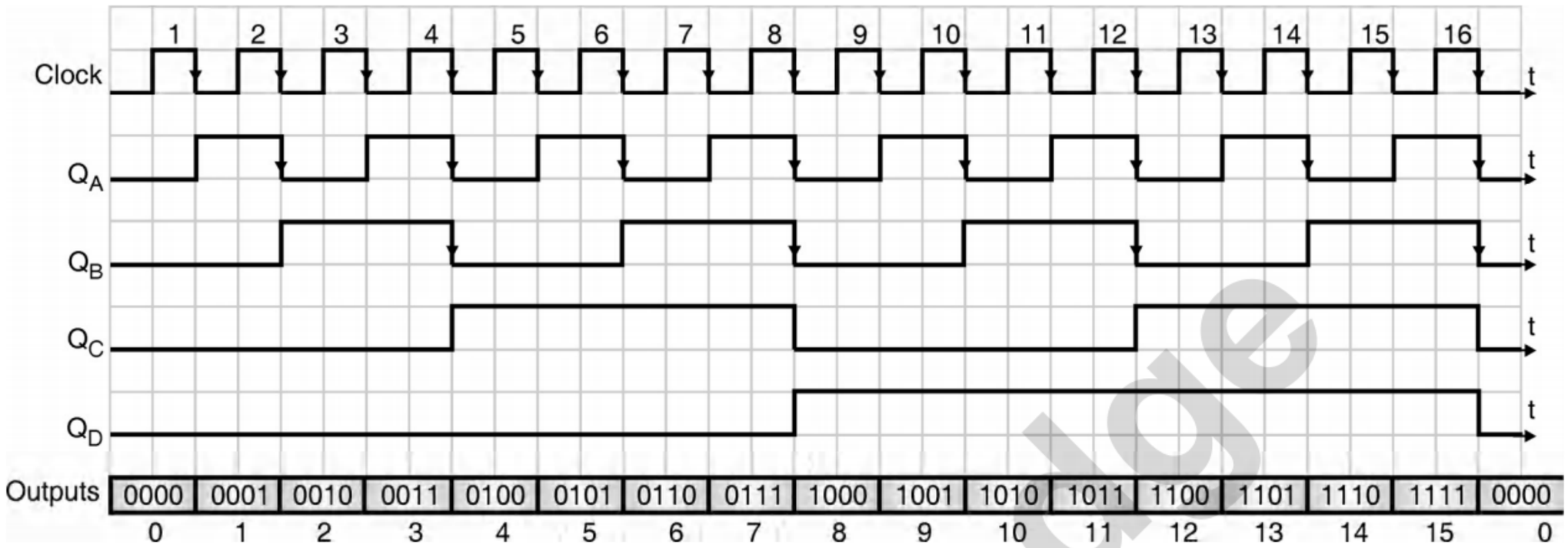
– The number of state through which the output of a 4 bit up counter passes is 16 (from 0 to 15).

Maximum count :

– The maximum count is 15 i.e. 1111. Thus a 4-bit ripple up counter will count from 0000 to 1111.

TechKnowledge Publications

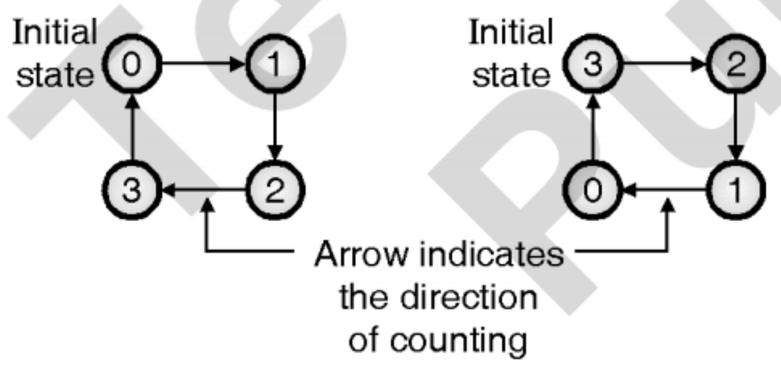
Timing diagram :



(C-778) **Fig. 5.2.7(b) : Waveforms of a 4 bit asynchronous up counter**

5.2.4 State Diagram of a Counter :

- The state diagram of a counter represents the states of a counter graphically.
- For example for a 2-bit up counter the state diagram is shown in Fig. 5.2.8(a) and for a 2-bit down counter the state diagram is shown in Fig. 5.2.8(b).
- The number written inside a circle represents the state number, whereas the arrow shows the direction of counter (up or down).
- Note that in the counters only the state is important. Hence in the state diagram we have not shown any input or output conditions.



(a) For a 2-bit up counter **(b) For a 2-bit down counter**

(C-779) **Fig. 5.2.8 : State diagram**

5.3 Asynchronous Down Counters :

5.3.1 3- Bit Asynchronous Down Counter :

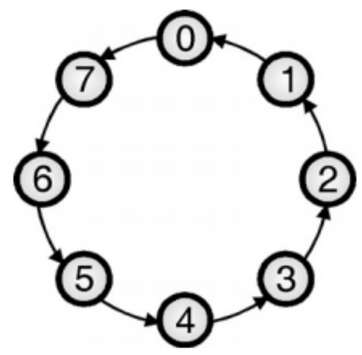
Truth table and state diagram :

- All the counters discussed so far have counted upwards from zero. So they can be called as **up counters.**
- But the counters which can count in the **downward** direction i.e. from the maximum count to zero are called **down counters.**
- The countdown sequence for a 3-bit asynchronous down counter is as follows :

(C-6970) **Table 5.3.1 : Truth table of 3 bit down counter**

	CLK	Output			Decimal
		Q_C	Q_B	Q_A	Count
Maximum count	–	0	0	0	0
	↓	1	1	1	7
Direction of counting	↓	1	1	0	6
	↓	1	0	1	5
	↓	1	0	0	4
	↓	0	1	1	3
	↓	0	1	0	2
	↓	0	0	1	1
	↓	0	0	0	0

Repeats



(C-2762) **Fig. 5.3.1 : State diagram**

- Thus counting takes place as follows :

  $Q_C\ Q_B\ Q_A$ = 111, 110, 101, 100, 011, 010, 001, 000.

- From this sequence it is evident that FF-A should toggle at every negative going clock edge but FF-B should change its state only at those instants when Q_A changes from LOW (0) to HIGH (1) and Q_C should changes only when Q_B changes from LOW to HIGH.

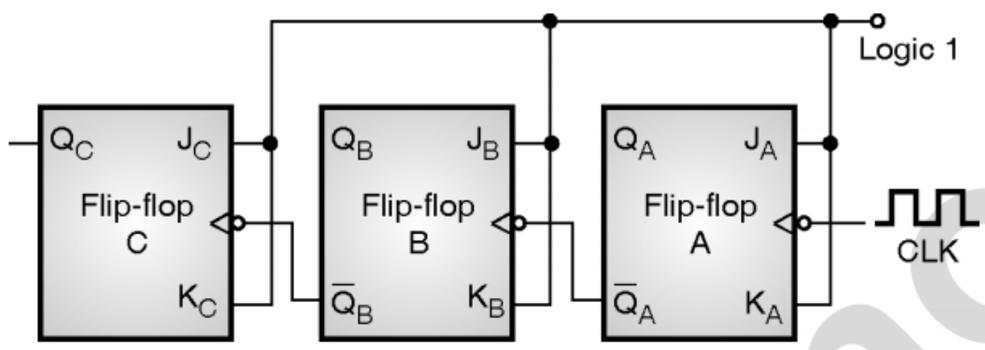
TechKnowledge Publications

– Thus in a down counter, each FF except the first one (FF-A) should toggle when the output of its preceding flip-flop changes from LOW to HIGH.

– If all the FFs are negative edge triggered i.e. responding to the negative CLK edge, then we can place an inverter in front of every CLK input or we can drive the CLK input of next FF from the $\overline{Q}$ output of the preceding FF and not from the Q outputs as shown in Fig. 5.3.2.

Logic diagram :

– A 3-bit asynchronous down counter is shown in Fig. 5.3.2. The clock input is applied directly to the clock input of FF-A. But $\overline{Q_A}$ is connected to clock of FF-B, $\overline{Q_B}$ to clock of FF-C and so on.



(C-781) **Fig. 5.3.2 : A 3-bit asynchronous down counter**

Operation :

– Initially let all the flip-flops be in the reset condition.

$\therefore \quad Q_C\, Q_B\, Q_A = 0\,0\,0$

– As soon as the first falling clock pulse arrives, FF-A toggles. So Q_A becomes 1 and $\overline{Q_A}$ changes from 1 to 0.

– The negative going change in $\overline{Q_A}$ acts as a clock to FF-B. Hence FF-B will change its state. So Q_B becomes 1 and $\overline{Q_B}$ changes from 1 to 0.

– This negative going change in $\overline{Q_B}$ acts as a clock to FF-C. Hence FF-C will change its state. So Q_C becomes 1 and $\overline{Q_C}$ becomes 0.

– Thus after the first clock pulse the output of counter are,

$Q_C\, Q_B\, Q_A = 1\,1\,1$ ..... After the 1st CLK pulse

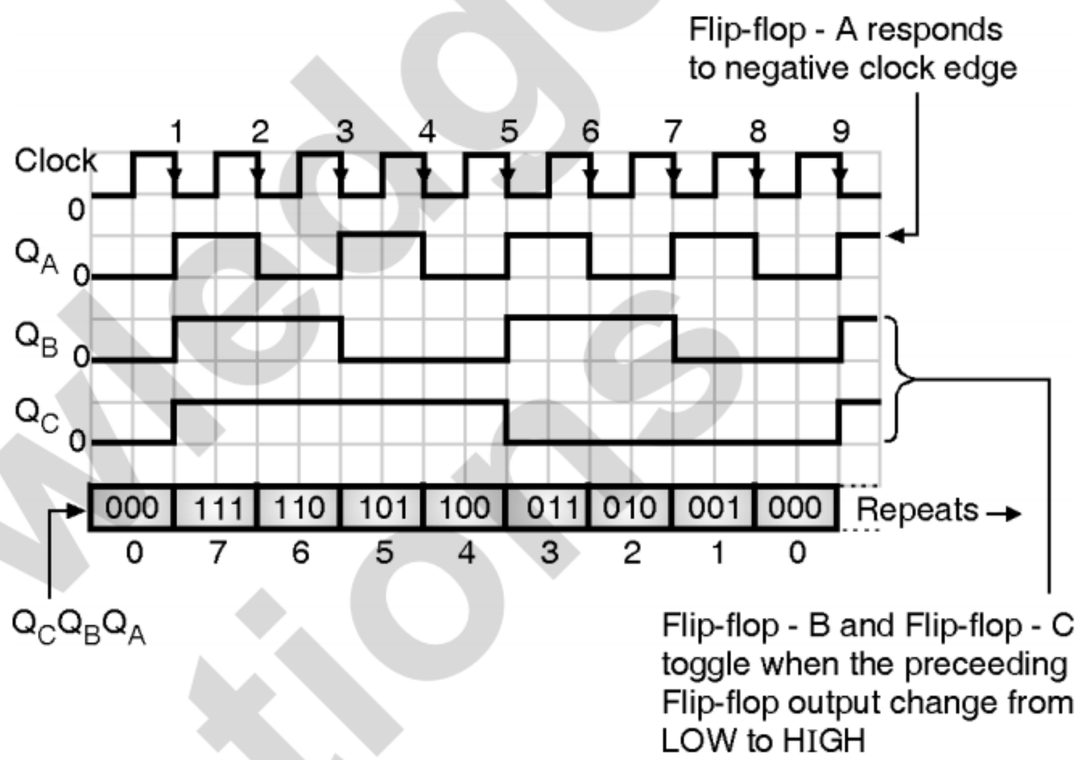
– Corresponding to the second falling clock edge, FF-A toggles. Q_A becomes 0 and $\overline{Q_A}$ becomes 1. This positive going change in $\overline{Q_A}$ does not alter the state of FF-B. So Q_B remains 1 and $\overline{Q_B}$ remains 0. So there is no change in the state of FF-C. Hence after the second clock pulse the counter outputs are,

$Q_C\, Q_B\, Q_A = 1\,1\,0$ ..... After the 2nd CLK pulse

– The down counting will thus take place. Similarly the counter will count down to pass through the states 101, 100, 011, 010, 001 and 000. The operation repeats itself thereafter.

Timing diagram :

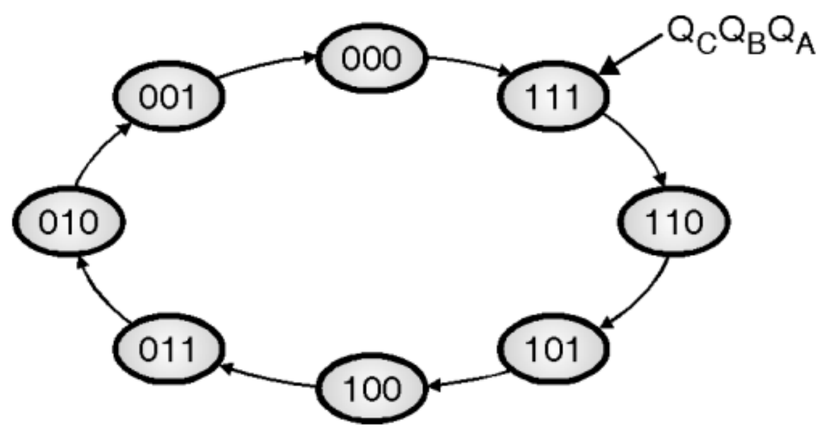
– The timing diagram for the down counter is shown in Fig. 5.3.3.



(C-782) **Fig. 5.3.3 : Timing diagram of a 3-bit down counter**

State diagram :

– The state diagram of the 3-bit down counter is shown in Fig. 5.3.4.



(C-783) **Fig. 5.3.4 : State diagram of a 3 bit ripple down counter**

Note : The numbers inside the circles correspond to the state of $Q_C\, Q_B\, Q_A$.

5.4 UP / DOWN Counters :

– We have designed the up counters and the down counters separately.

– But in practice both these modes are generally combined together and an UP/DOWN counter is formed.

– A mode control (M) input is also provided to select either up count or down count mode of operation.

– A combinational circuit is required to be designed and used between each pair of flip-flops in order to achieve the up/down operation.

Types of up/down counters :

The up/down counters are of two types :

1. UP/DOWN ripple counters.
2. UP/DOWN synchronous counters.

5.4.1 UP/DOWN Ripple Counters :

– In the up/down ripple counter all the FFs operate in the toggle mode. So either T flip-flops or JK flip-flops are to be used.

– The LSB flip-flop receives clock directly. But the clock to every other FF is obtained from Q or $\overline{Q}$ output of the previous FF.

UP counting mode (M = 0) :

– The CLK signal is applied directly to the clock input of the LSB flip-flop.

– For the remaining flip-flops, the Q output of the preceding FF is connected to the clock of the next stage if up counting is to be achieved. For this mode, the mode select input M is at logic 0 (M = 0).

DOWN counting mode (M = 1) :

– The clock signal is applied directly to the clock input of the LSB flip-flop. For the remaining flip-flop, the $\overline{Q}$ output of the preceding FF is connected to the clock of the next FF.

– This will operate the counter in the down counting mode. For down counting mode the mode select input M is kept at logic 1 (M = 1).

5.4.2 3-bit Up Down Ripple Counters :

– The design of 3-bit up down ripple counters has been illustrated in the following examples.

Ex. 5.4.1 : Design a 3-bit binary up/down ripple counter. Draw the timing diagram.

Soln. :

– The requirements of counter are :

1. 3-bit : Hence three FFs are required.
2. UP/DOWN : So a mode control input is essential.

– We know that for a ripple up counter, the Q output of preceding FF is connected to the clock input of the next one.

– And for a ripple down counter, the $\overline{Q}$ output of the preceding FF is connected to the clock input of the next one.

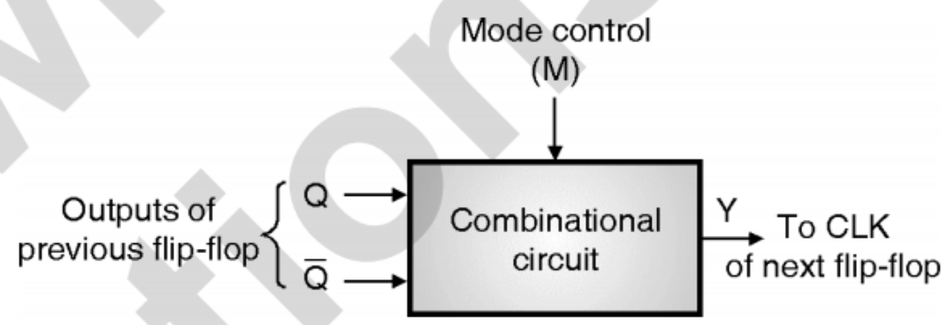
– Let the selection of Q or $\overline{Q}$ output of the preceding FF be controlled by the mode control input M such that,

If M = 0 UP counting. So connect Q to CLK

If M = 1 DOWN counting. So connect $\overline{Q}$ to CLK

– Let us design a combinational logic to satisfy all the requirements stated above.

Block diagram :



(C-786) Fig. P. 5.4.1(a) : Block diagram of combinational circuit

Truth table of combinational circuit :

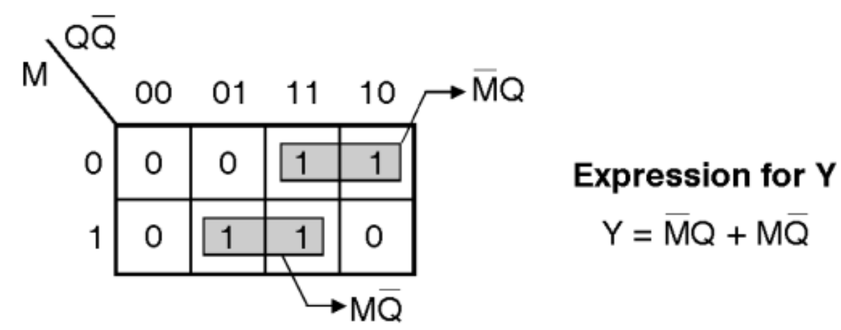
– The truth table of such a combinational circuit is shown in Table P. 5.4.1.

(C-6215) Table P. 5.4.1 : Truth table

Inputs			Output	
M	Q	$\overline{Q}$	Y	
0	0	0	0	Y = Q for up counting
0	0	1	0	
0	1	0	1	
0	1	1	1	
1	0	0	0	Y = $\overline{Q}$ for down counting
1	0	1	1	
1	1	0	0	
1	1	1	1	

K-map and simplified expression for output :

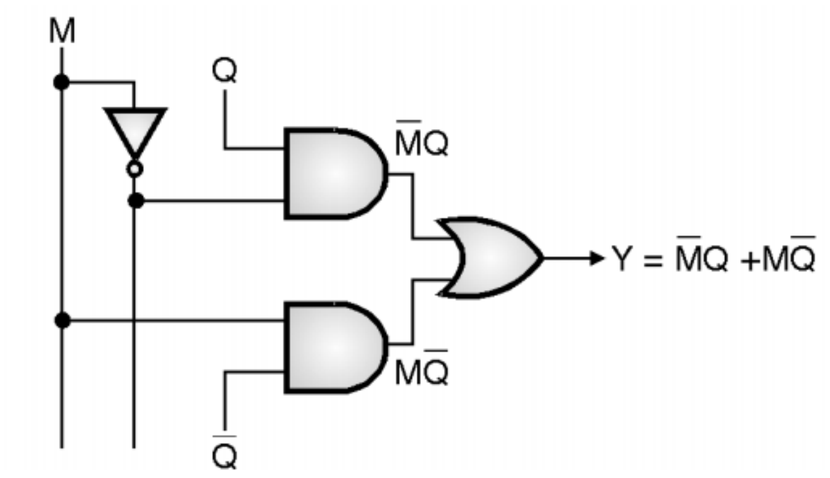
– Fig. P. 5.4.1(b) shows the K-map and simplified expression for Y.



(C-787) Fig. P. 5.4.1(b) : K-map for Y

Logic diagram for combinational circuit :

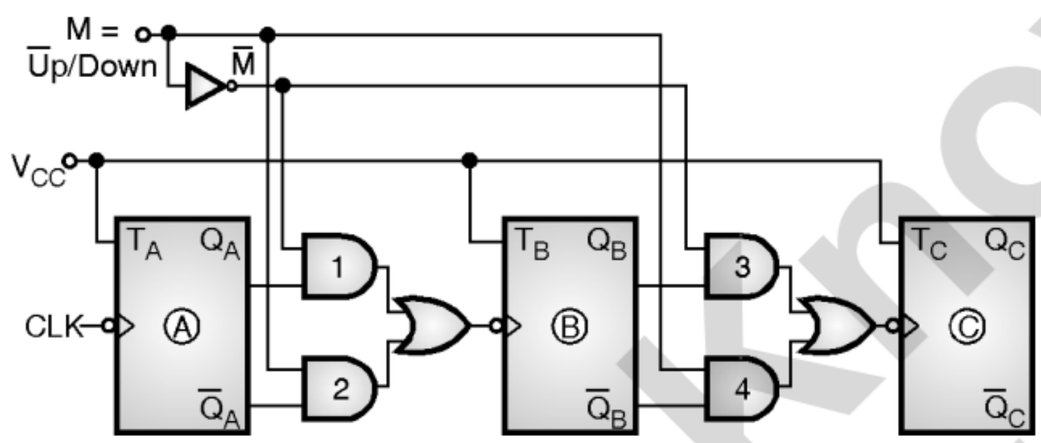
– The logic diagram for the combinational circuit is shown in Fig. P. 5.4.1(c).



(C-787) Fig. P. 5.4.1(c)

Logic diagram of a 3-bit up/down counter :

– The combinational circuit is connected between every pair of flip-flops to obtain the 3-bit up/down counter as shown in Fig. P. 5.4.1(d).



(C-788) Fig. P. 5.4.1(d) : A 3-bit up / down ripple counter

Operation of 3-bit up down ripple counter :

1. With M = 0 (Up counting mode) :

– If M = 0 and $\overline{M}$ = 1, then the AND gates 1 and 3 in Fig. P. 5.4.1(d) will be enabled whereas the AND gates 2 and 4 will be disabled.

– Hence Q_A gets connected to the clock input of FF-B and Q_B gets connected to the clock input of FF-C.

– These connections are same as those for the normal up counter.

– Thus with M = 0 the circuit works as an up counter.

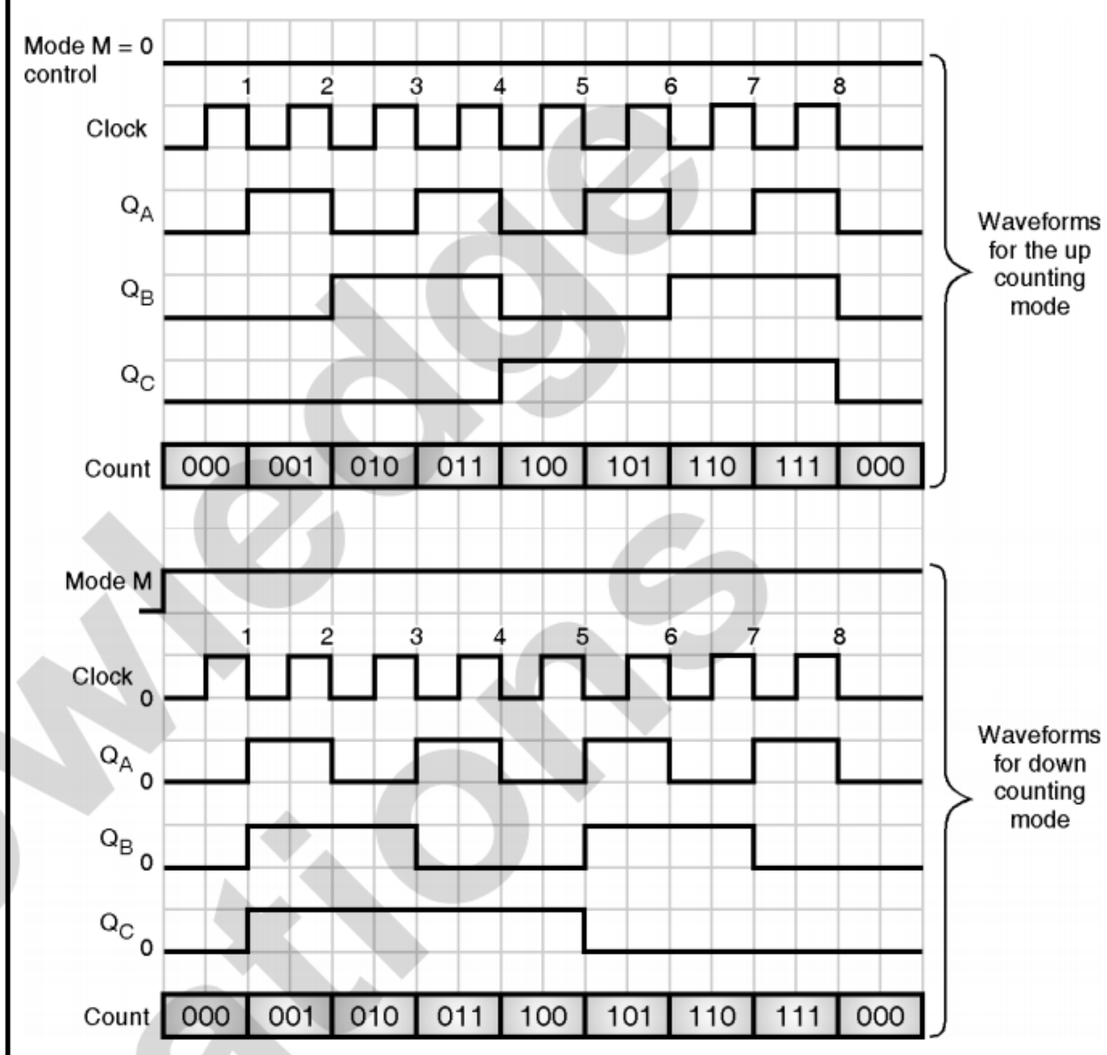
2. With M = 1 (Down counting mode) :

– If M = 1, then AND gates 2 and 4 in Fig. P. 5.4.1(d) are enabled whereas the AND gates 1 and 3 are disabled.

– Hence $\overline{Q_A}$ gets connected to the clock input of FF-B and $\overline{Q_B}$ gets connected to the clock input of FF-C. As discussed earlier, these connections will produce a down counter. Thus with M = 1 the circuit works as a down counter.

Timing diagram for a 3-bit up/down ripple counter :

– The timing diagram for a 3-bit up/down ripple counter is shown in Fig. P. 5.4.1(e).



(C-2676(a)) Fig. P. 5.4.1(e) : Timing diagram for 3-bit up/down ripple counter

5.5 Modulus of the Counter (MOD-N Counter) : SPPU : Dec. 09, Dec. 10

University Questions

Q. 1 What is MOD counter ? **(Dec. 09, 2 Marks)**

Q. 2 What is the advantage of MOD counter ? **(Dec. 10, 2 Marks)**

Definition :

– Modulus (MOD) of a counter represents the number of states through which the counter progresses during its operation. It is denoted by N.

– Thus MOD-N counter means the counter progresses through N states.

– A MOD-4 counter will have 4 states. A MOD-6 counter will have 6 states etc.

– Thus a 3 bit counter which has 8 states is a MOD 8 counter, and a 4 bit counter is a MOD-16 counter.

– In general m number of flip-flops are required to construct mod-n counter, where $N \leq 2^m$.

TechKnowledge Publications

– We can design a modulo counter with the help of the basic ripple counter structure and a combinational logic called reset logic.

– The 2-bit ripple counter is called as MOD-4 counter and 3-bit ripple counter is called as MOD-8 counter. So in general, an n-bit ripple counter is called as modulo-N counter where

MOD number = 2^n

– So we can conclude that **modulus** of a counter represents the **number of states** through which the counter progresses during its operation.

– Can we have a modulo-5 counter using a 3-bit ripple counter ? That means can we restrict the number of states to only 5 instead of 8 under normal conditions ?

– The answer is yes. We can design such modulo counters with the help of the basic ripple counter structure and a combinational logic called reset logic.

– Table 5.5.1 shows the relation between 2, 3 and 4 bit counters and their modulus.

(C-8049)Table 5.5.1

Counter type	Modulus
2 bit up or down	MOD-4
3 bit up or down	MOD-8
4 bit up or down	MOD-16

5.5.1 Design of Asynchronous MOD Counters :

Ex. 5.5.1 : Design a Modulo-5 ripple counter using a 3-bit ripple counter.

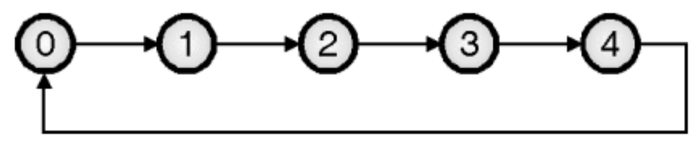
OR

Design MOD-5 asynchronous counter, and also draw the waveforms.

Soln. :

Step 1 : Draw the state diagram :

– The state diagram of MOD-5 ripple counter is as shown in Fig. P. 5.5.1(a).



(C-794) Fig. P. 5.5.1(a) : State diagram of a MOD-5 ripple counter

Step 2 : Write truth table for the reset logic :

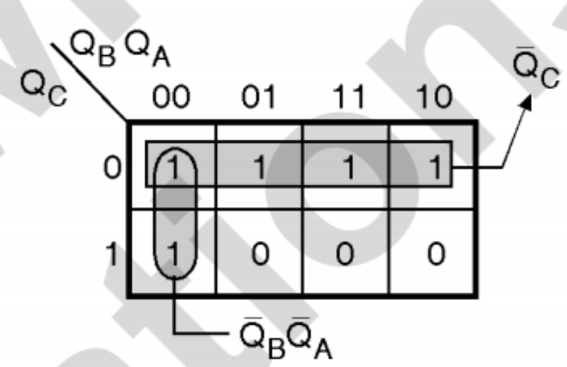
– Table P. 5.5.1 shows the truth table for the reset logic.

(C-6216) Table P. 5.5.1 : Truth table for the reset logic

State	Flip-flop outputs			Output Y of reset logic	
	Q_C	Q_B	Q_A		
0	0	0	0	1	Valid states
1	0	0	1	1	
2	0	1	0	1	
3	0	1	1	1	
4	1	0	0	1	
5	1	0	1	0	Invalid states
6	1	1	0	0	
7	1	1	1	0	

Step 3 : K-map :

For output Y



Expression for Y

$Y = \bar{Q}_C + \bar{Q}_B \bar{Q}_A$

(C-795) Fig. P. 5.5.1(b) : K-map and simplification

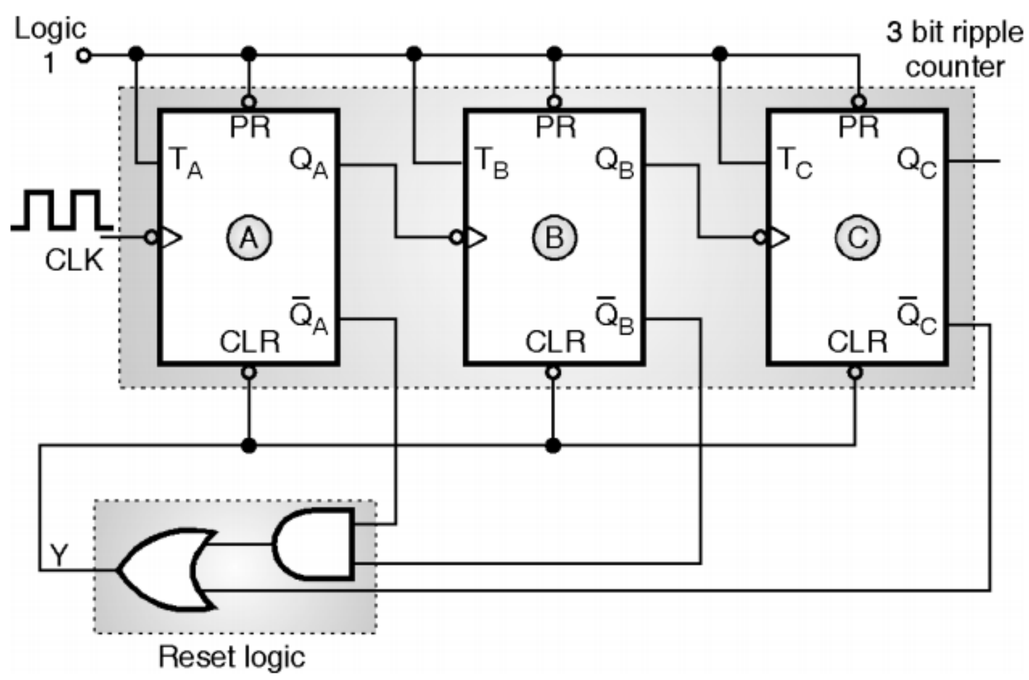
– The K map is as shown in Fig. P. 5.5.1(b).

– The states 0 through 4 are valid states and the output Y of reset logic (Y) is inactive (1) for them.

– The states 5, 6 and 7 are invalid states. If counter enters into any one of these states that Y = 0 (active) and will reset all the flip-flops.

Step 4 : Logic diagram :

– The logic diagram of a MOD-5 ripple counter is shown in Fig. P. 5.5.1(c).

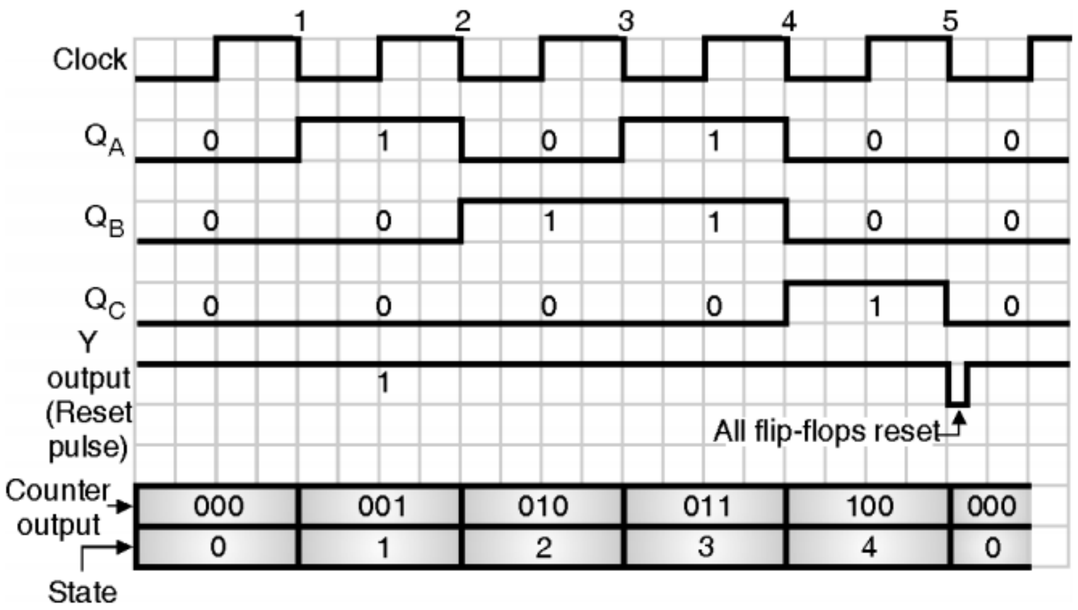


(C-796) Fig. P. 5.5.1(c) : Logic diagram of MOD-5 ripple counter

TechKnowledge Publications

Step 5 : Timing diagram :

– The timing diagram is as shown in Fig. P. 5.5.1(d).



(C-797) Fig. P. 5.5.1(d) : Timing diagram of MOD-5 ripple counter

Ex. 5.5.2 : For MOD-11 asynchronous up counter :

1. Draw circuit diagram. Use T flip-flop.
2. Write truth table.
3. Draw timing diagram.
4. If the output frequency is 11 kHz what is the clock input ?

Soln. :

Step 1 : Write the truth table :

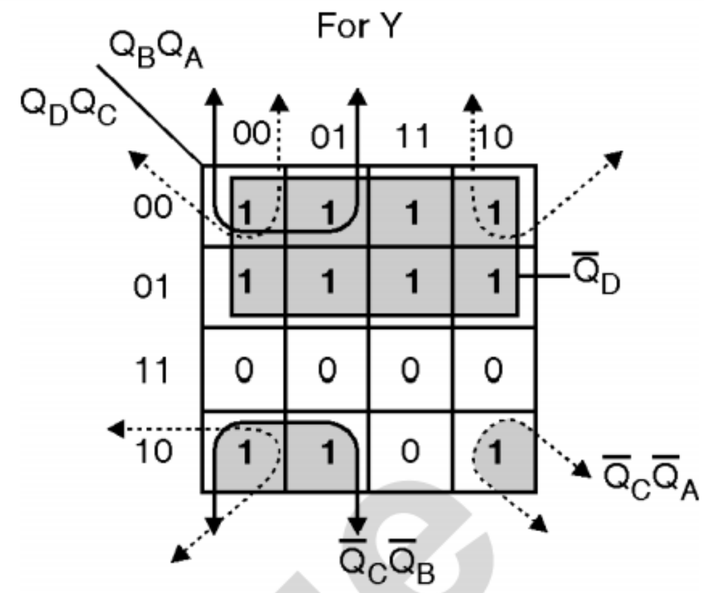
(C-6268) Table P. 5.5.2

Q_D	Q_C	Q_B	Q_A	Y output
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Invalid states

Step 2 : Draw the K map :

– The K map is shown in Fig. P. 5.5.2(a).



$$\therefore\ Y = \overline{Q}_D + \overline{Q}_C\overline{Q}_A + \overline{Q}_C\overline{Q}_B$$

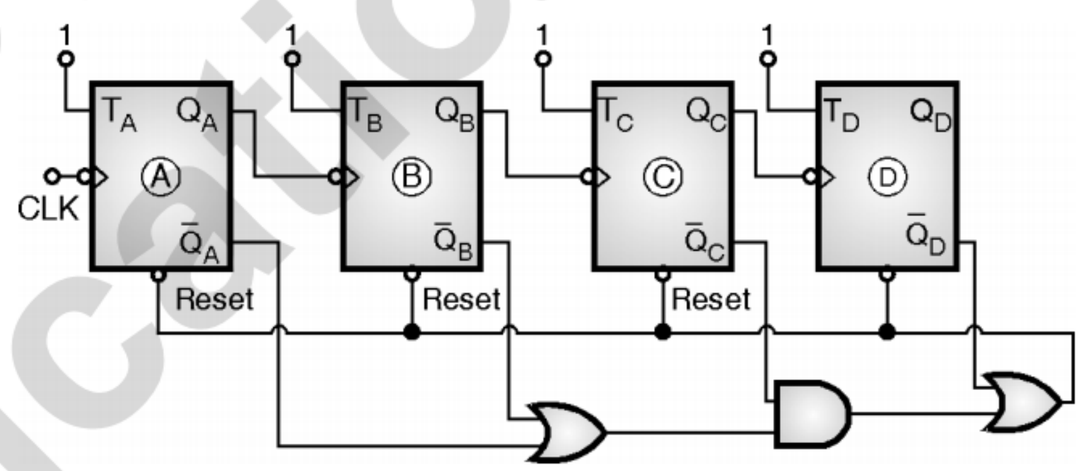
$$\therefore\ Y = \overline{Q}_D + \overline{Q}_C(\overline{Q}_A + \overline{Q}_B)$$

(C-801) Fig. P. 5.5.2(a) : K map

– The simplified equation for Y output of the reset logic is as follows :

$$Y = \overline{Q}_D + \overline{Q}_C\left(\overline{Q}_A + \overline{Q}_B\right)$$

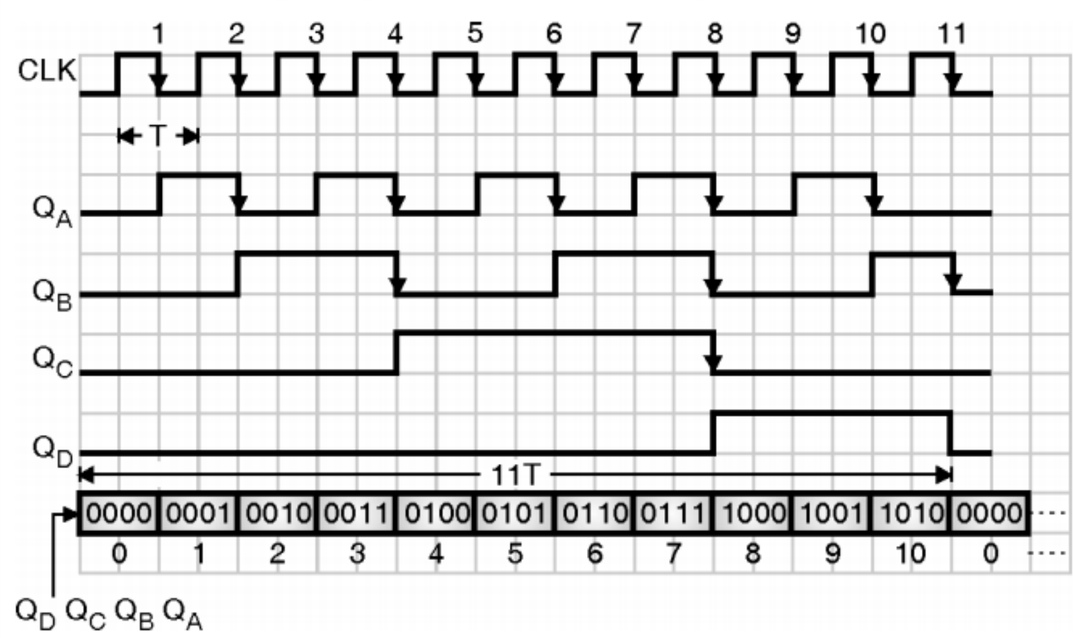
Step 3 : Draw the circuit diagram :



(C-802) Fig. P. 5.5.2(b) : MOD 11 counter

Step 4 : Timing diagram :

– The timing diagram is shown in Fig. P. 5.5.2(c).



(C-803) Fig. P. 5.5.2(c) : Timing diagram

– Output frequency $f_o = \dfrac{f_{CLK}}{11}$

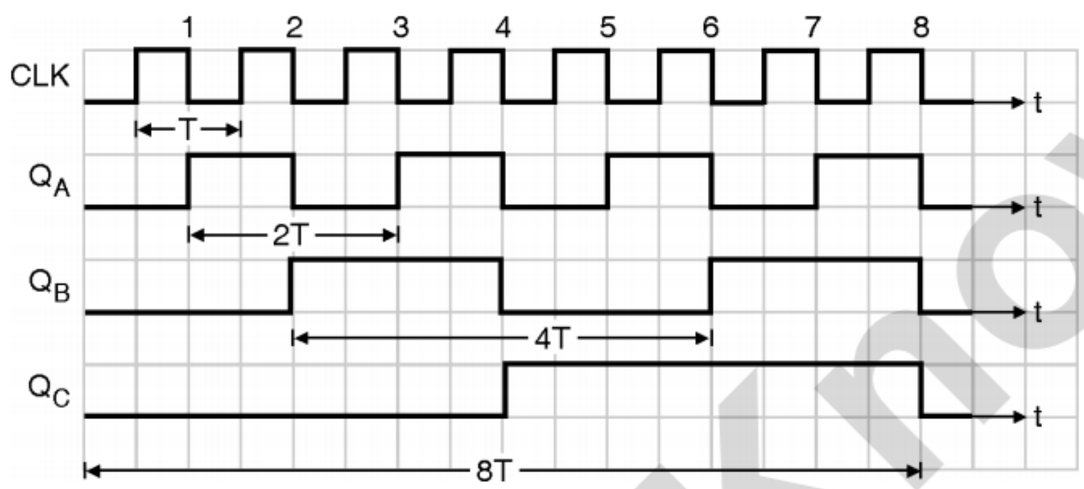
$$\therefore\ f_{CLK} = 11\ f_o = 11 \times 11\ \text{kHz}$$

$$\therefore\ f_{CLK} = 121\ \text{kHz}$$ **...Ans.**

TechKnowledge Publications

5.5.2 Frequency Division Taking Place in Asynchronous Counters :

- In the chapter on flip-flops, we have seen that a flip-flop in toggle mode divides the clock frequency by 2.
- That means the frequency of Q or $\overline{Q}$ output waveform of a toggle flip-flop is exactly half of the clock frequency.
- The concept of frequency division is applicable to the counters as well because we use flip-flops in the toggle mode for counters.
- Refer to the timing waveforms of a 3-bit asynchronous up counter and observe the frequencies of Q_A, Q_B and Q_C waveforms with respect to the clock frequency.



(C-807) Fig. 5.5.1 : Timing waveforms of a 3-bit asynchronous up counter

Conclusions :

- Let one cycle period of the clock signal be T sec. Hence the clock frequency f_{CLK} = (1/T) Hz.
- Output of the least significant flip-flop (i.e. FF-A) has a one cycle period of (2 T) as shown in Fig. 5.5.1. Hence,

$$\text{Frequency of } Q_A \text{ output} = f_A = \frac{1}{2T} = \frac{f_{CLK}}{2}$$

- The one cycle period of Q_B is 4 T. Hence the frequency of Q_B is given by,

$$f_B = \frac{1}{4T} = \frac{f_{CLK}}{4} \qquad \text{..... since } \frac{1}{T} = f_{CLK}$$

- Similarly the frequency of Q_C output is given by

$$f_C = \frac{1}{8T} = \frac{f_{CLK}}{8}$$

Note : In any counter, the signal at the output of last FF (i.e. MSB) will have a frequency equal to the input clock frequency divided by the MOD number of the counter.

Ex. 5.5.3 : How many flip-flops are required to build a binary counter circuit to count from 0 to 2048 ? What is the frequency of the output of last flip-flop for an input clock frequency of 6 MHz ? **Dec. 13, 6 Marks**

Soln. :

To count n clock pulses m flip flops are required, where

$$n = 2^m$$

$$\therefore \quad 2048 = 2^m$$

$$\therefore \quad m = 11$$

11 flip flops are required to count from 0 to 2048.

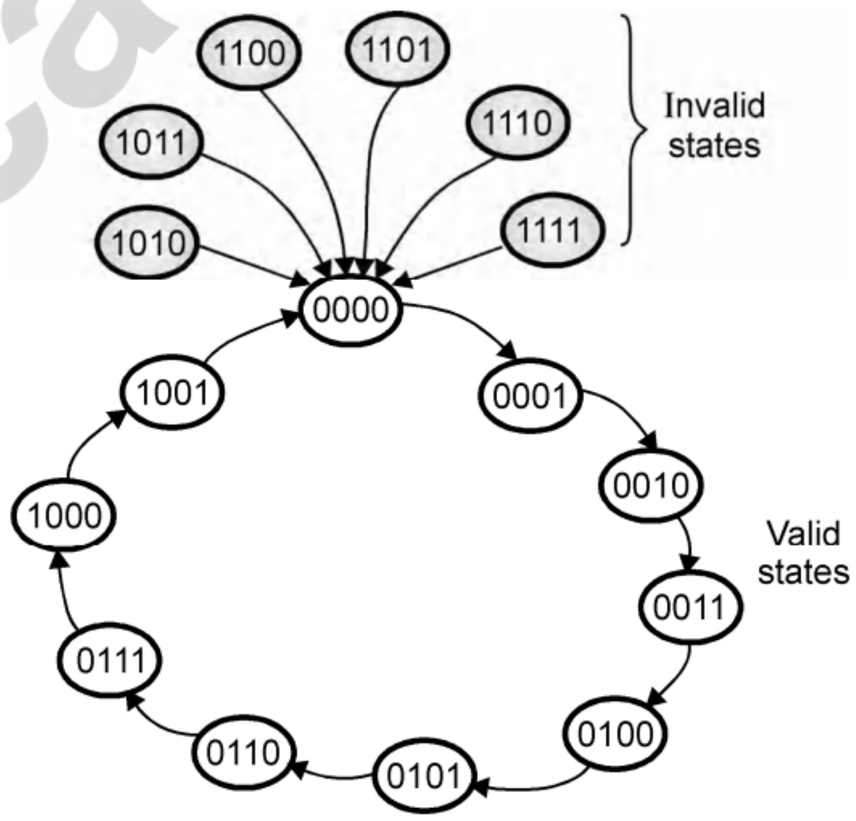
∴ Output frequency,

$$f_o = \frac{f_{CLK}}{2048} = \frac{6 \text{ MHz}}{2048} = 2929.68 \text{ Hz} \qquad \textbf{...Ans.}$$

5.5.3 Decade (BCD) Ripple Counter :

State diagram :

- The state diagram of an BCD ripple counter is shown in Fig. 5.5.2. It shows that a BCD counter counts from 0000 to 1001 i.e. from 0 to 9.



(C-811) Fig. 5.5.2 : State diagram of a BCD counter

- Four JK flip-flops with a clear input are to be used. The output of the **reset logic** is to be connected to the clear input of all the FFs.
- The output of the reset logic Y = 1 for all the valid states from 0 to 9 and Y = 0 for all the invalid outputs of the counter i.e. from 10 to 15.
- If the counter enters into any invalid state then it will be reset automatically.

Technknowledge Publications

Truth table :

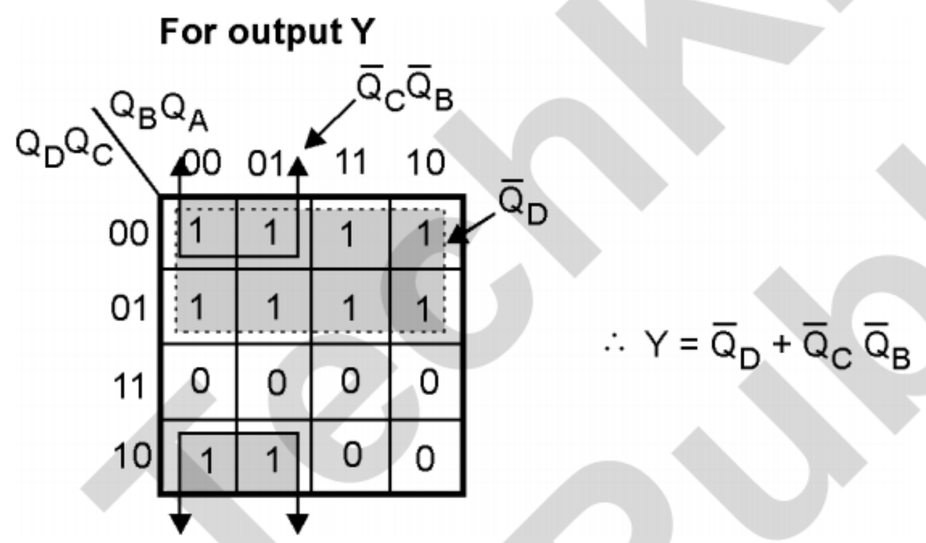
– The truth table of the reset logic is given in Table 5.5.2.

(C-6269) Table 5.5.2 : Truth table of a BCD counter

CLK	Q_D	Q_C	Q_B	Q_A	Output of reset logic	
0	0	0	0	0	1	Valid states
1	0	0	0	1	1	
2	0	0	1	0	1	
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	1	
9	1	0	0	1	1	
–	1	0	1	0	0	Invalid states
–	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	

K-map and simplifications :

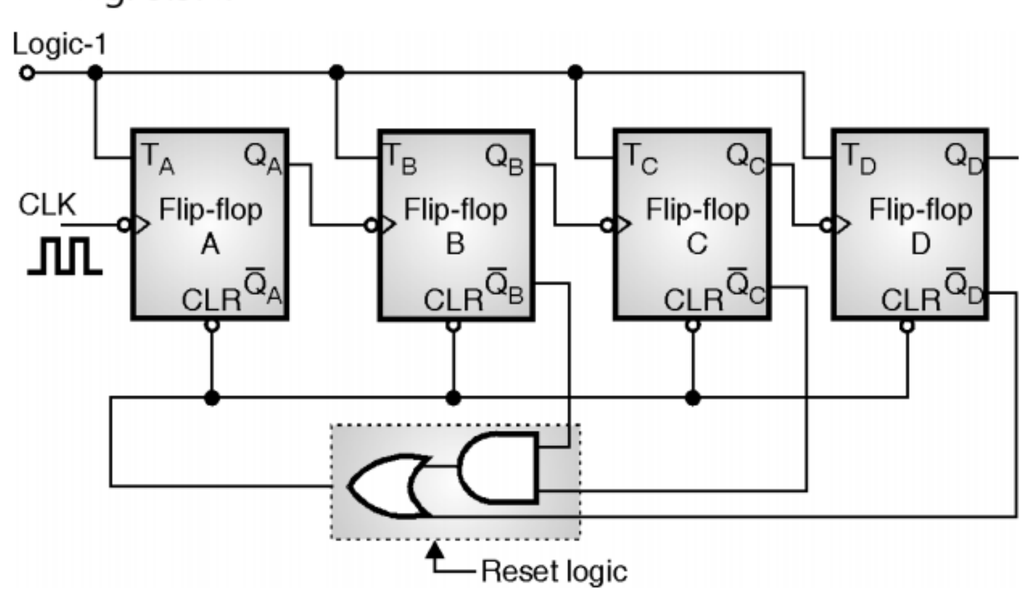
– The K-map for Y output is shown in Fig. 5.5.3.



(C-812) Fig. 5.5.3

Logic diagram :

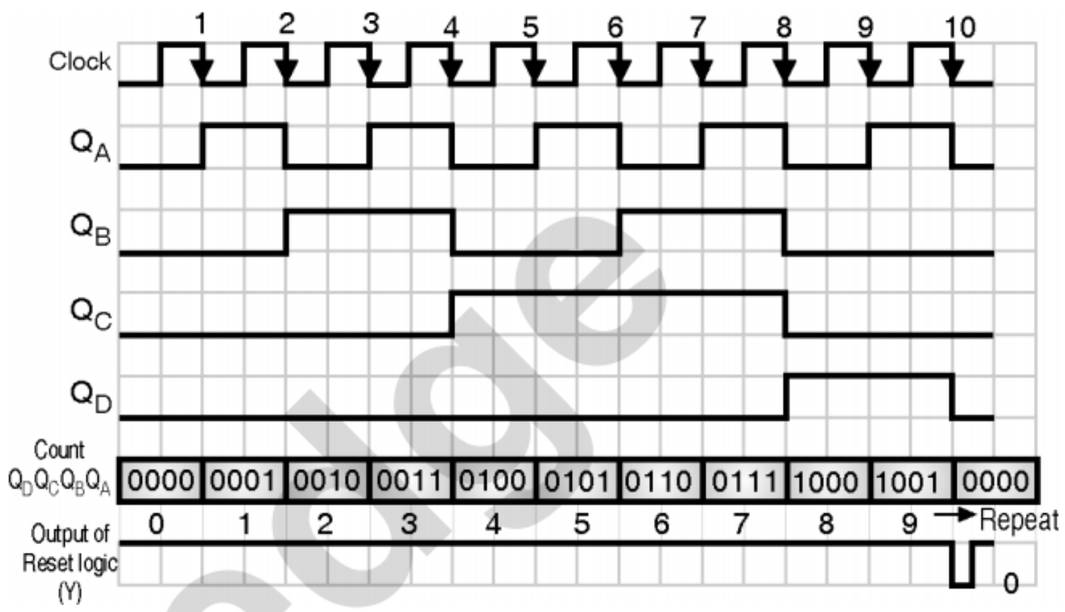
– The logic diagram of a BCD counter are as shown in Fig. 5.5.4.



(C-813) Fig. 5.5.4 : Logic diagram of a BCD counter

Timing diagram :

– The timing diagram of a BCD counter is shown in Fig. 5.5.5.



(C-814) Fig. 5.5.5 : Timing diagram of a BCD ripple counter

5.5.4 Disadvantages of Ripple Counters :

– Every flip-flop has its own propagation delay. In ripple counter the output of the previous FF is used as clock for the next FF.

– Hence the propagation delay goes on accumulating. For a 3-bit ripple counter the propagation delay of the first FF gets added to that of the second FF to decide the transition time for the third stage.

– This accumulated time delay is the main problem with the ripple counters, because the propagation delay goes on increasing with increase in number of flip-flops.

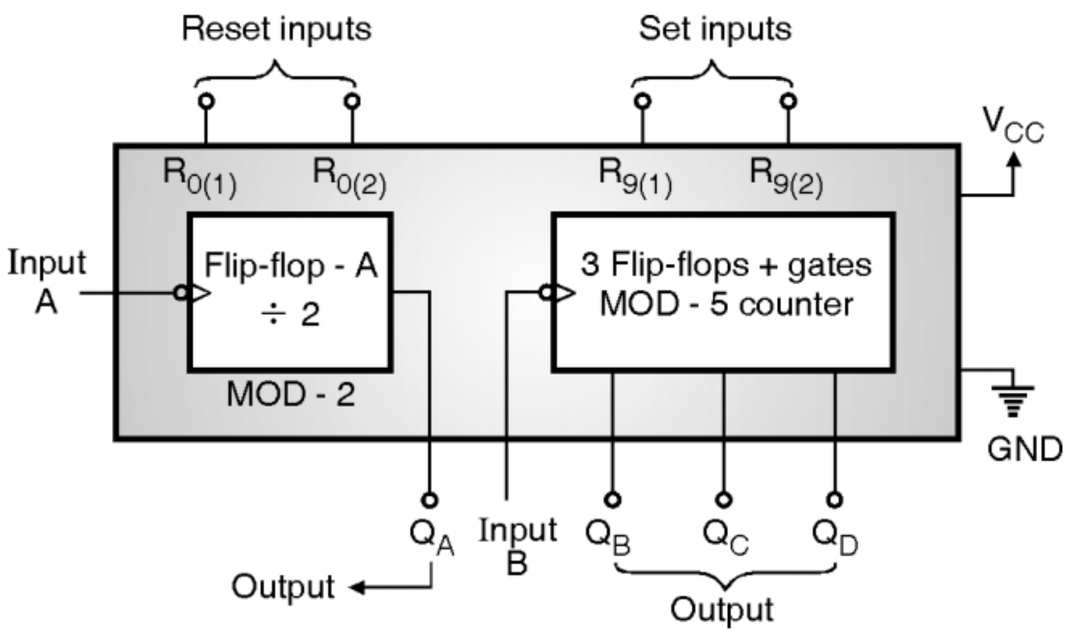
– This will put a limitation on the maximum clock frequency.

5.6 Ripple Counter IC 7490 (Decade Counter) :

Internal structure :

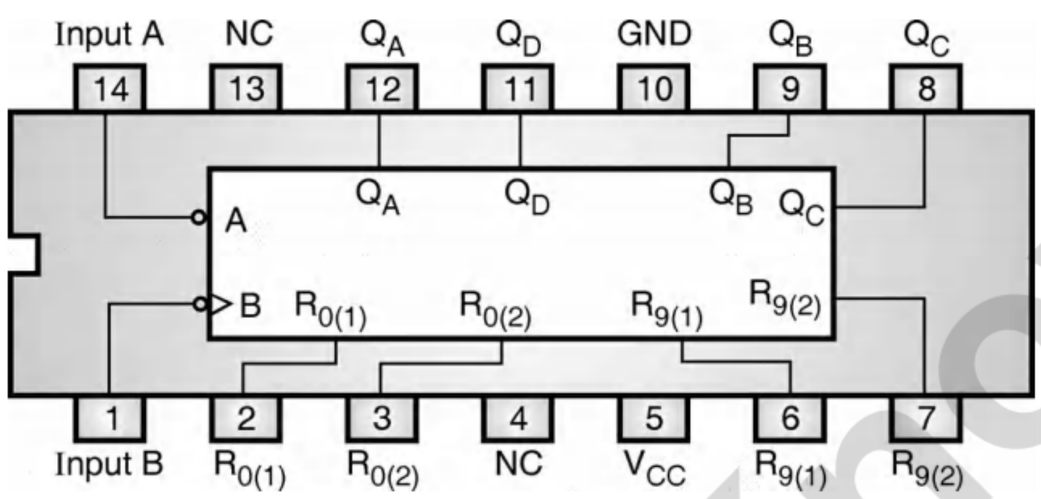
– IC 7490 is a TTL MSI decade counter. It contains four master slave flip-flops and a few logic gates to provide a divide-by-two counter and a three stage binary counter which provides a divide by 5 counter. (MOD-5), as shown in Fig. 5.6.1.

TechKnowledge Publications



(C-1374) **Fig. 5.6.1 : The basic internal structure of IC 7490**

– IC 7490 is a MOD-10 or decade counter. It is a 14 pin IC and its pin configuration as shown in Fig. 5.6.2.



(C-1375) **Fig. 5.6.2 : Pin configuration of IC 7490**

Description :

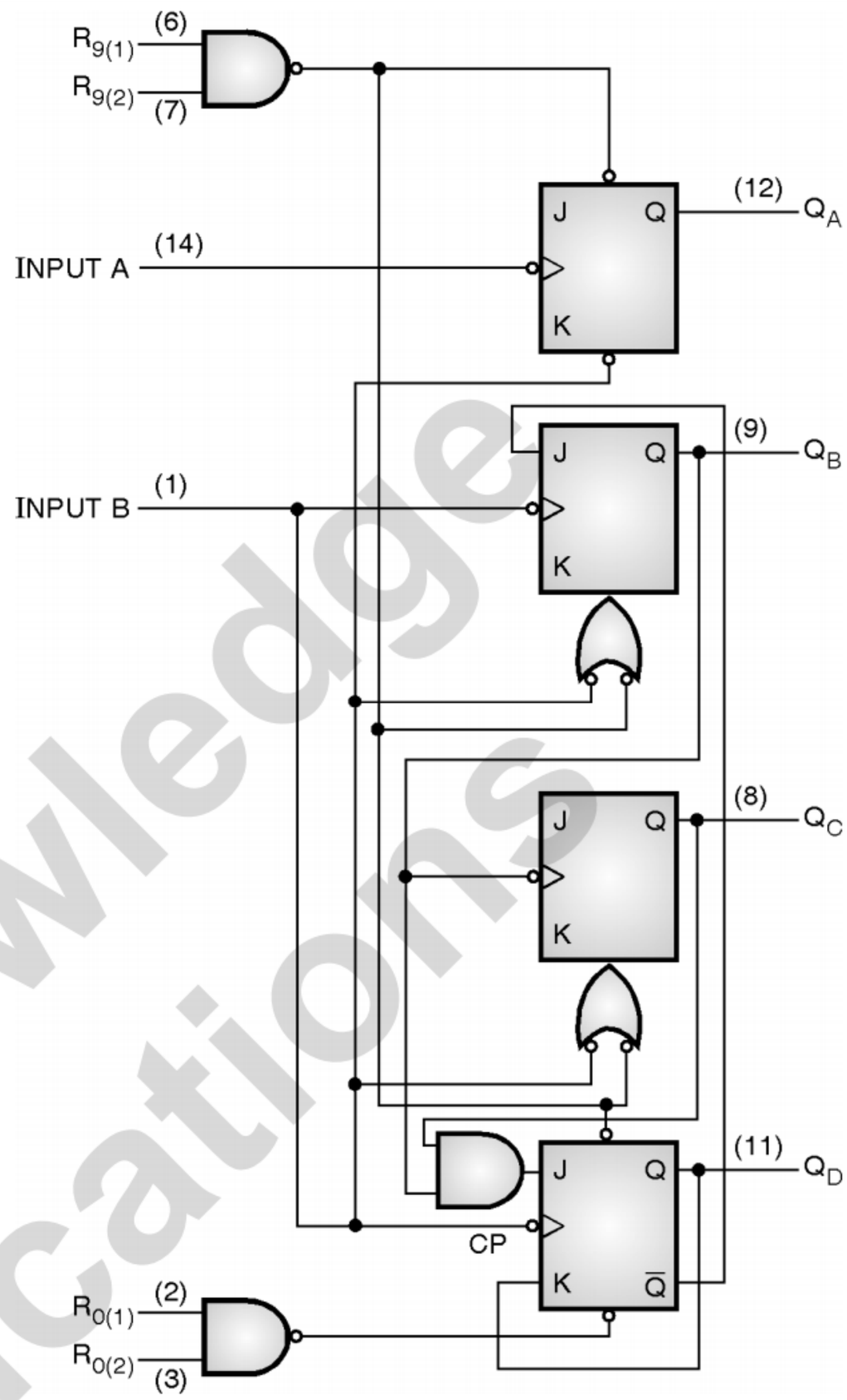
– Table 5.6.1 explains the pin configuration of IC 7490.

Table 5.6.1 : Pin name and description of IC 7490

Pin name	Description
Input B	This is clock input to the internal MOD-5 ripple counter, which is negative edge triggered.
$R_{0(1)}$, $R_{0(2)}$	Gated zero reset inputs.
V_{CC}	+ 5 V DC
$R_{9(1)}$, $R_{9(2)}$	These are gated set to nine inputs.
Q_D, Q_C, Q_B	Outputs of internal MOD-5 counter with Q_D as MSB.
GND	Logic ground. Acts as a reference.
Q_A	Output of internal mod-2 counter or FF-A.
Input A	Clock input to FF-A which is negative edge triggered.

5.6.1 The Internal Diagram of IC 7490 :

– A simplified internal diagram of IC 7490 is shown in Fig. 5.6.3.



(C-1376) **Fig. 5.6.3 : A simplified internal diagram of IC 7490**

Function table :

– The reset/count function table of IC 7490 is shown in Table 5.6.2.

(C-6217) **Table 5.6.2 : 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				If CLK applied it will count or step from previous to next step
0	X	0	X	COUNTER				
0	X	X	0	COUNTER				
X	0	0	X	COUNTER				

Conclusion :

1. If both the reset inputs $R_{0(1)}$ and $R_{0(2)}$ are at logic 1 then all the flip-flops will be reset and the output is given by

$$Q_D\, Q_C\, Q_B\, Q_A = 0\,0\,0\,0$$

TechKnowledge Publications

2. If both the preset inputs $R_{9(1)}$ and $R_{9(2)}$ are at logic 1 then the counter output is set to decimal 9.

   $\therefore \quad Q_D\, Q_C\, Q_B\, Q_A = 1\,0\,0\,1$

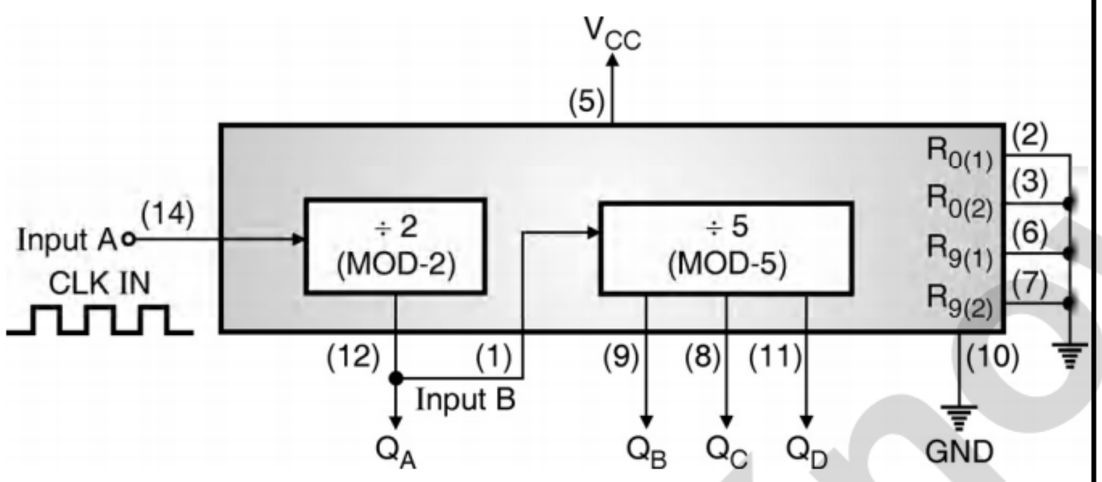
3. If any one pin of $R_{0(1)}$, $R_{0(2)}$ and one of $R_{9(1)}$, $R_{9(2)}$ is low, then the counter will be in the count mode.

Ex. 5.6.1 : Demonstrate the use of IC 7490 as a decade counter and explain its operation.

Dec. 04, 3 Marks, Dec. 17, 2 Marks

Soln. :

– The connection diagram for IC 7490 as decade counter is shown in Fig. P. 5.6.1.



(C-1377) Fig. P. 5.6.1 : 7490 as decade counter

1. Note that the Q_A output of FF-A is externally connected to the input B which is the clock input of the internal mod-5 ripple counter.
2. Hence Q_A will toggle on every falling edge of the clock input whereas the outputs Q_D Q_C Q_B of the MOD-5 counter will increment from 000 to 100 on the low going change of Q_A output.
3. Table P. 5.6.1 summarizes the operation of the 7490 as decade counter. Due to cascading of MOD-2 and MOD-5 counters, the overall configuration becomes a MOD - 10 i.e. decade counter configuration.
4. The reset inputs $R_{0(1)}$, $R_{0(2)}$ and the preset inputs $R_{9(1)}$, $R_{9(2)}$ are connected to ground so as to make them inactive.
5. As shown in Table P. 5.6.1 the counter counts from 0000 to 1001 i.e. from 0 to 9. After 1001 the MOD-5 counter resets to 000 and Q_A changes to 0. Hence the next count after 1001 is 0000 and recycling begins.

(C-6218) Table P. 5.6.1 : Summary of operation of IC 7490 decade counter

Outputs of MOD-5 counter			Output	CLK	Count
Q_D	Q_C	Q_B	Q_A		$Q_DQ_CQ_BQ_A$
0	0	0	0	0	0
0	0	0	1	1 (↓)	1
0	0	1	0	2 (↓)	2
0	0	1	1	3 (↓)	3
0	1	0	0	4 (↓)	4
0	1	0	1	5 (↓)	5
0	1	1	0	6 (↓)	6
0	1	1	1	7 (↓)	7
1	0	0	0	8 (↓)	8
1	0	0	1	9 (↓)	9

FF-A toggles on every falling clock edge. It acts as clock for MOD-5 counter.

MOD-5 counter outputs remain unchanged if Q_A changes from 0 to 1. But MOD-5 counter outputs increment by 1 every time, Q_A changes from 1 to 0

5.6.2 Other Applications of IC 7490 :

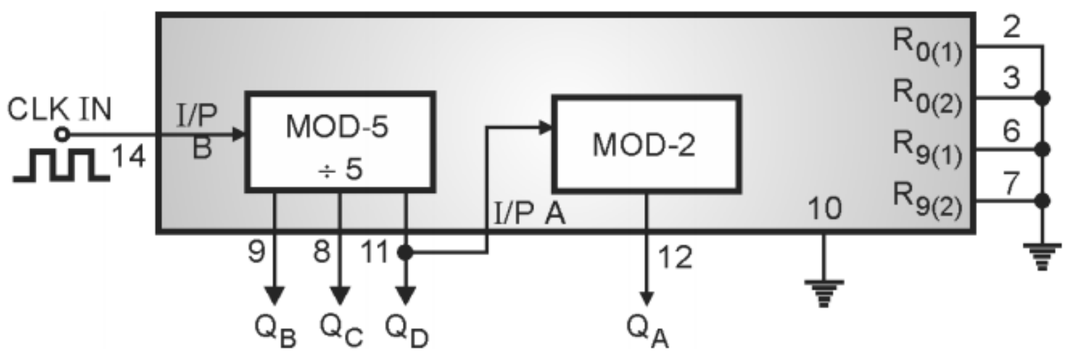
– Some of the other applications of IC 7490 are as follows :

1. Symmetrical Bi-quinary divide by ten counter.
2. Divide by two (MOD-2) and divide by 5 (MOD-5) counter.

5.6.3 Symmetrical Bi-quinary Divide by Ten Counter :

– In this application the clock input is applied to the B input and Q_D output is connected to A input as shown in Fig. 5.6.4.

– The output is obtained at Q_A output. It is a perfect square wave with a 50% duty cycle at a frequency equal to f_{CLK} / 10.



(C-1378) Fig. 5.6.4 : Symmetrical bi-quinary divide by ten counter

Operation :

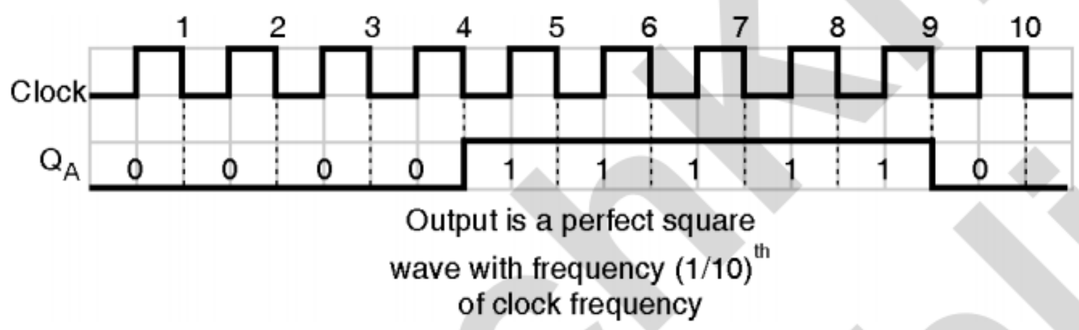
– The operation of this counter has been summarized in Table 5.6.3.

(C-6219) Table 5.6.3 : Summary of operation of symmetrical bi-quinary divide by 10 counter

Clock pulse number	Outputs				Count
	(LSB) Q_B	Q_C	Q_D	(MSB) Q_A	$Q_AQ_DQ_CQ_B$
0	0	0	0	0	0
1 (↓)	1	0	0	0	1
2 (↓)	0	1	0	0	2
3 (↓)	1	1	0 Transition	0	3
4 (↓)	0	0	1	0 Toggle	4
5 (↓)	0	0	0	1	8
6 (↓)	1	0	0	1	9
7 (↓)	0	1	0	1	10
8 (↓)	1	1	0	1	11
9 (↓)	0	0	1	1	12

Output waveform :

– The waveform at Q_A output is shown in Fig. 5.6.5. It shows that Q_A output is a perfect square wave.



(C-1379) Fig. 5.6.5 : Waveforms showing a perfect square wave output

Ex. 5.6.2 : Draw MOD 6 counter using IC 7490. Write truth table. **Dec. 04, 3 Marks**

Soln. :

Step 1 : Truth table :

– Table P. 5.6.2 shows the truth table for a MOD 6 counter.

(C-7764) Table P. 5.6.2

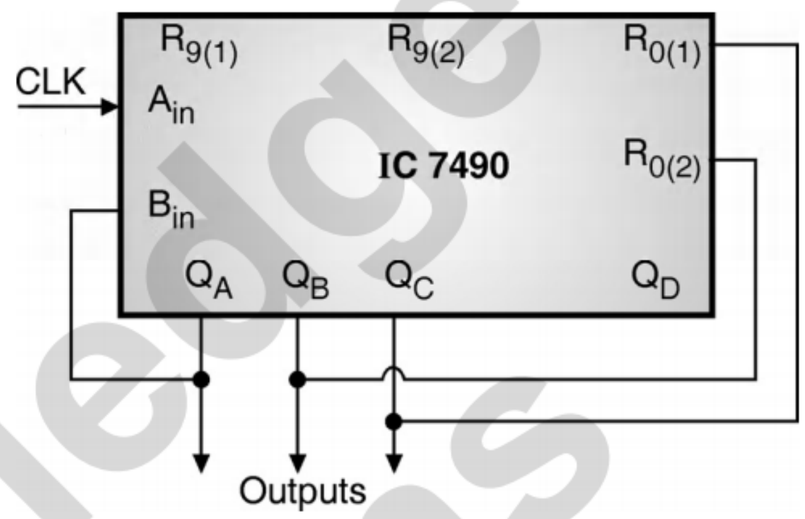
Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
0	0	0

Step 2 : Draw the circuit :

– The total number of states is 6.

– Note that Q_C and Q_B are connected to R_{01} and R_{02} respectively.

– So when the counter output is $(6)_{10}$ i.e. $Q_C\ Q_B\ Q_A$ = 110 then the counter will reset to 000.

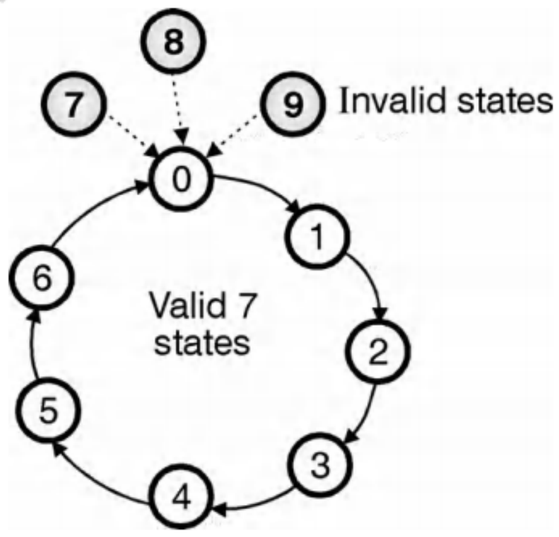


(C-1380) Fig. P. 5.6.2 : MOD 6 counter using IC 7490

Ex. 5.6.3 : Design a MOD-7 counter using 7490.

Soln. :

– MOD - 7 counter counts through the 7 states as shown in Fig. P. 5.6.3(a).



(C-1381) Fig. P. 5.6.3(a) : State diagram for MOD-7 counter

– From all the invalid states (7, 8, 9) and from 6 the counter should reset.

Design of reset logic :

– The output of the reset logic should be 1 corresponding to all the invalid states. The reset logic output is applied to $R_{0(1)}$ and $R_{0(2)}$.

– The truth table for the reset logic is shown in Table P. 5.6.3 and the K-map is shown in Fig. P. 5.6.3(b).

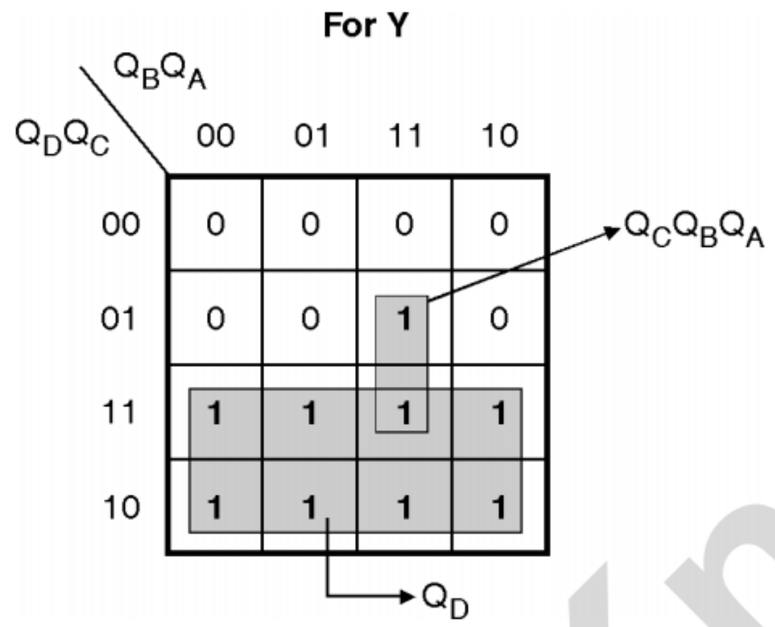
Tech Knowledge Publications

(C-6220) Table P. 5.6.3 : Truth table of the reset logic

Q_D	Q_C	Q_B	Q_A	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1

Invalid states

For Y



(C-1382) Fig. P. 5.6.3(b) : K-map for the output of reset logic

– Note that for all the states beyond 1001, we have entered a logic 1 in the K-map treating all those states as invalid states.

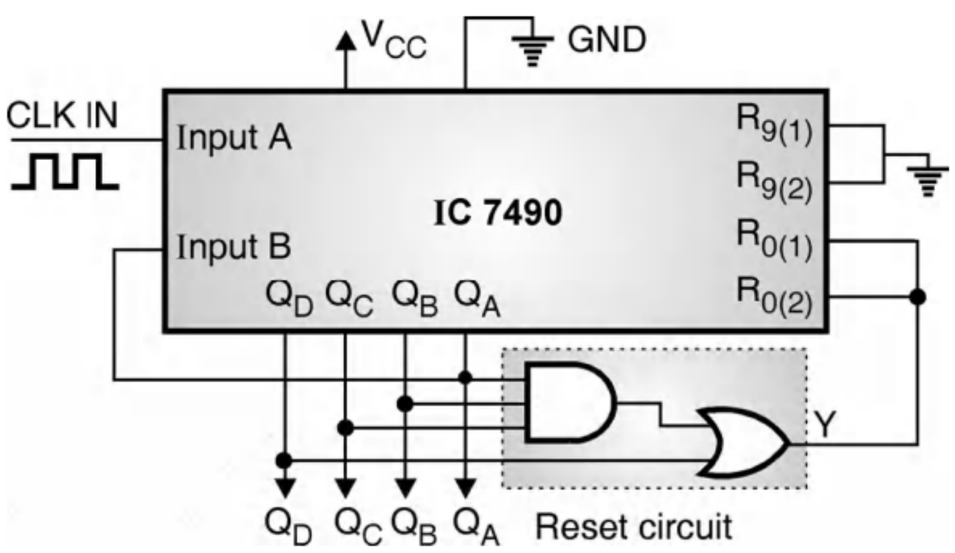
Expression for Y :

– From the K-map, the simplified expression for Y is given by,

$$Y = Q_C Q_B Q_A + Q_D$$

Logic diagram :

– The logic diagram of the MOD-7 counter is shown in Fig. P. 5.6.3(c).



(C-1383) Fig. P. 5.6.3(c) : MOD -7 counter using IC 7490

Ex. 5.6.4 : Draw basic internal architecture of IC 7490. Design a divide by 20 counter using same.

Dec. 11, 8 Marks, May 15, 6 Marks

Soln. :

– For the basic internal structure of IC 7490. Refer section 5.6.1.

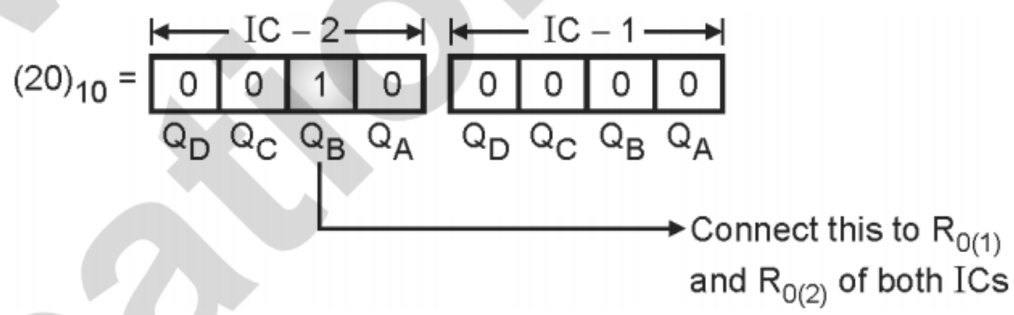
Step 1 : Number of ICs required :

– Upto MOD - 100, two IC 7490s will be sufficient. Hence to implement a divide by 20 counter, we have to use two decade counter ICs.

Step 2 : Design of reset logic :

– To design a divide by 20 (MOD - 20) counter we have to use two IC 7490 counter ICs.

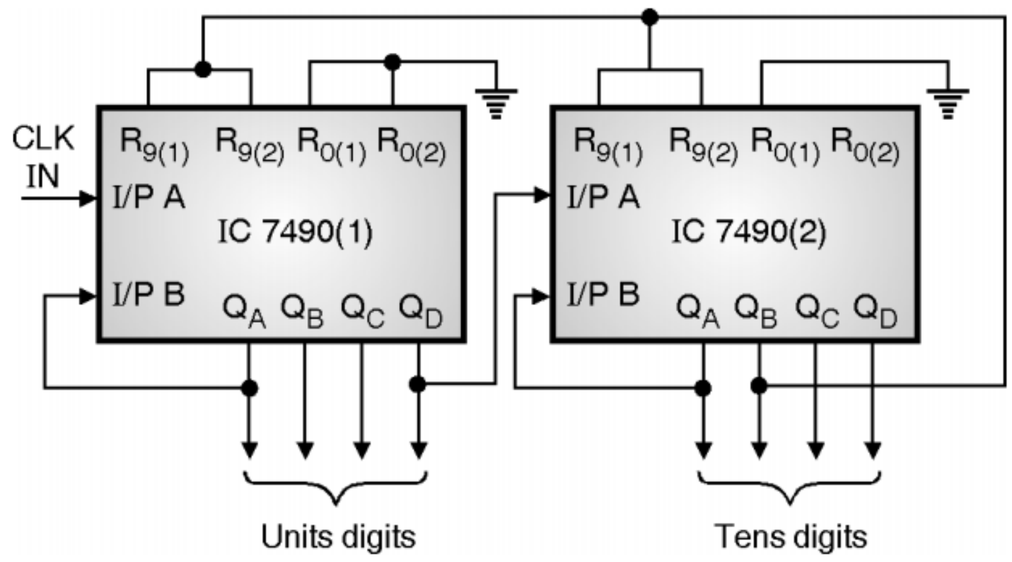
– Both the ICs should reset as soon as the count is equal to 20 decimal.



(C-8116) Fig. P. 5.6.4 : Reset logic

Step 3 : Draw the logic diagram :

– The logic diagram of MOD - 20 counter is as shown in Fig. P. 5.6.4(a).



(C-1985) Fig. P. 5.6.4(a) : Logic diagram of MOD-20 counter using IC 7490

Ex. 5.6.5 : Explain the internal diagram of IC 7490. Design MOD 7 and MOD 98 counter using 7490.

May 12, 8 Marks

Soln. :

– Refer section 5.6.1 for internal diagram of IC 7490. Refer Ex. 5.6.3 for MOD-7 counter.

TechKnowledge Publications

MOD 98 counter :

Step 1 : Number of ICs required :

– Upto MOD - 100, two IC 7490s will be sufficient. Hence to implement a divide by 98 counter, we have to use two decade counter ICs.

Step 2 : Design of reset logic :

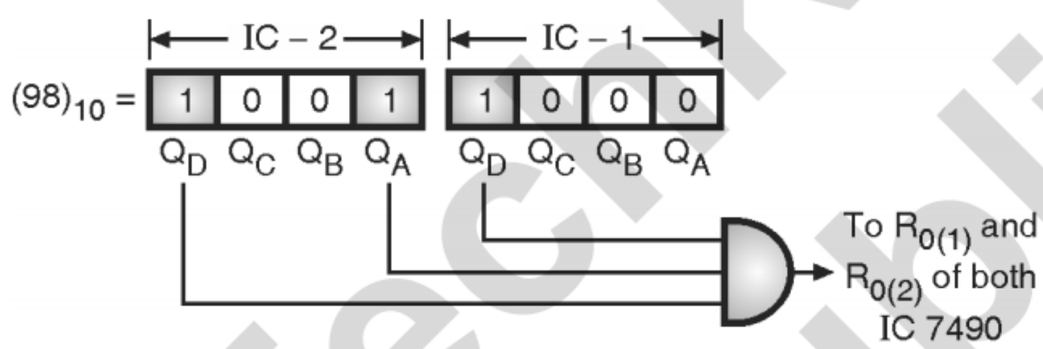
– To design the reset logic, earlier we used to draw the K-maps.

– The 3, 4 or at the most 5 variable K-maps are practically possible to handle. But here, there will be 8 FFs and so there will be 8-variables. So use of K-maps is practically impossible.

– Hence we will simplify the reset logic design as follows :

– A divide by 98 counter counts through 98 states from 0 to 97 and the counter should reset as soon as the count becomes 98.

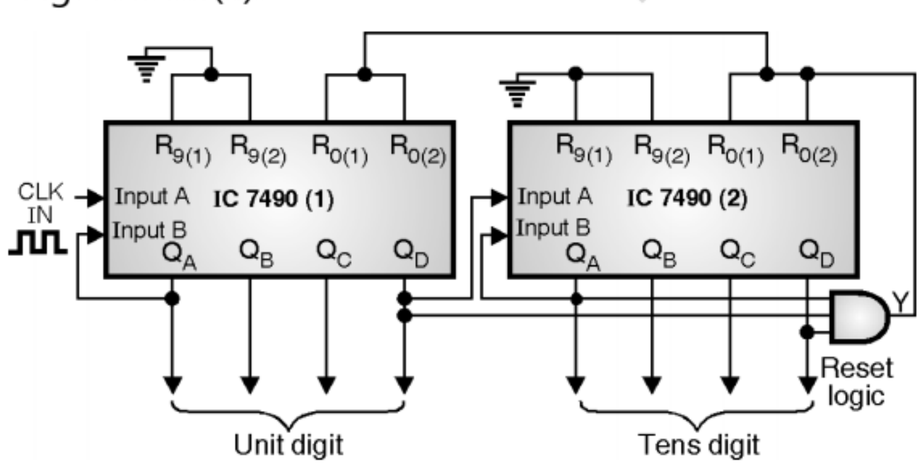
– So in order to reset the counter at 98, connect the Q outputs which are equal to 1 in the count of 98 to an AND gate as shown in Fig. P. 5.6.5 and then connect the AND output to the $R_{0(1)}$ and $R_{0(2)}$ i.e. reset inputs of both the ICs.



(C-6005) Fig. P. 5.6.5

Step 3 : Draw the logic diagram :

– The logic diagram of the MOD-98 counter is shown in Fig. P. 5.6.5(a).



(C-3602) Fig. P. 5.6.5(a)

Ex. 5.6.6 : Design the following using IC7490 :

1. MOD 97 counter
2. MOD 45 counter. **Dec. 12, 8 Marks**

Soln. :

1. MOD 97 counter :

Step 1 : Number of ICs required :

– Upto MOD - 100, two IC 7490s will be sufficient. Hence to implement a divide by 97 counter, we have to use two decade counter ICs.

Step 2 : Design of reset logic :

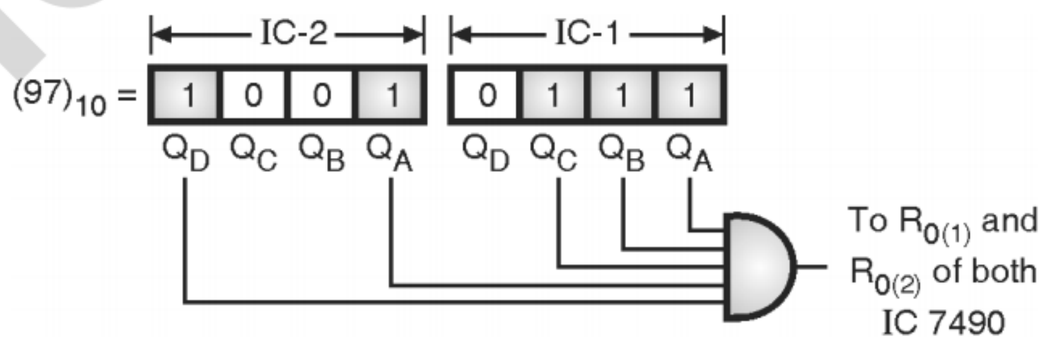
– To design the reset logic, earlier we used to draw the K-maps.

– The 3, 4 or at the most 5 variable K-maps are practically possible to handle. But here, there will be 8 FFs and so there will be 8-variables. So use of K-maps is practically impossible.

– Hence we will simplify the reset logic design as follows :

– A divide by 97 counter counts through 97 states from 0 to 96 and the counter should reset as soon as the count becomes 97.

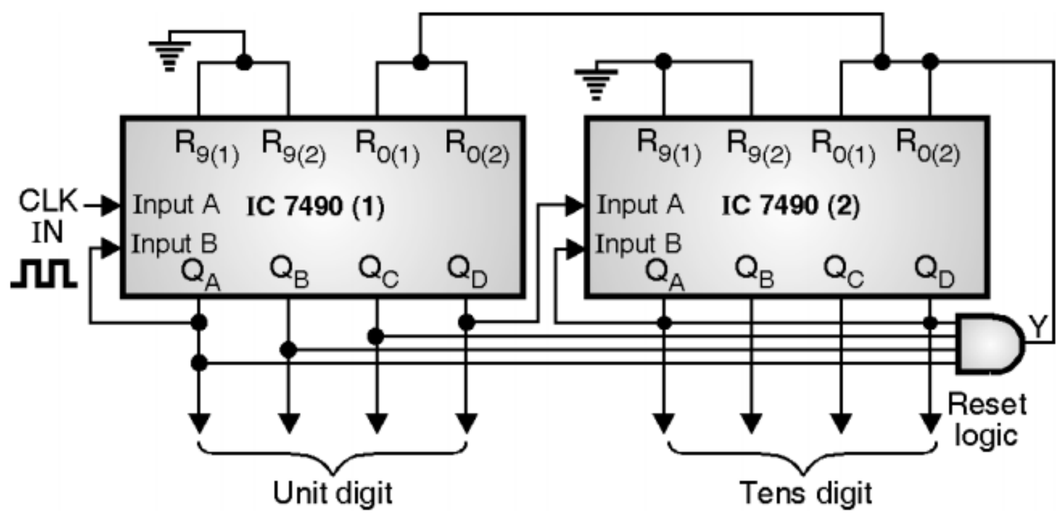
– So in order to reset the counter at 97, connect the Q outputs which are equal to 1 in the count of 97 to an AND gate as shown in Fig. P. 5.6.6(a) and then connect the AND output to the $R_{0(1)}$ and $R_{0(2)}$ i.e. reset inputs of both the ICs.



(C-6006) Fig. P. 5.6.6(a) : Reset logic

Step 3 : Draw the logic diagram :

– The logic diagram of the MOD-97 counter is shown in Fig. P. 5.6.6(b).



(C-3604) Fig. P. 5.6.6(b)

TechKnowledge Publications

2. MOD 45 counter :

Step 1 : Number of ICs required :

– Upto MOD - 100, two IC 7490s will be sufficient. Hence to implement a divide by 45 counter, we have to use two decade counter ICs.

Step 2 : Design of reset logic :

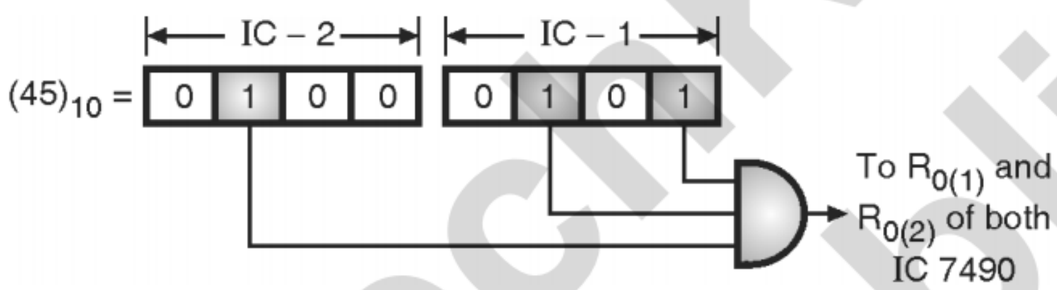
– To design the reset logic, earlier we used to draw the K-maps.

– The 3, 4 or at the most 5 variable K-maps are practically possible to handle. But here, there will be 8 FFs and so there will be 8-variables. So use of K-maps is practically impossible.

– Hence we will simplify the reset logic design as follows :

– A divide by 45 counter counts through 45 states from 0 to 44 and the counter should reset as soon as the count becomes 45.

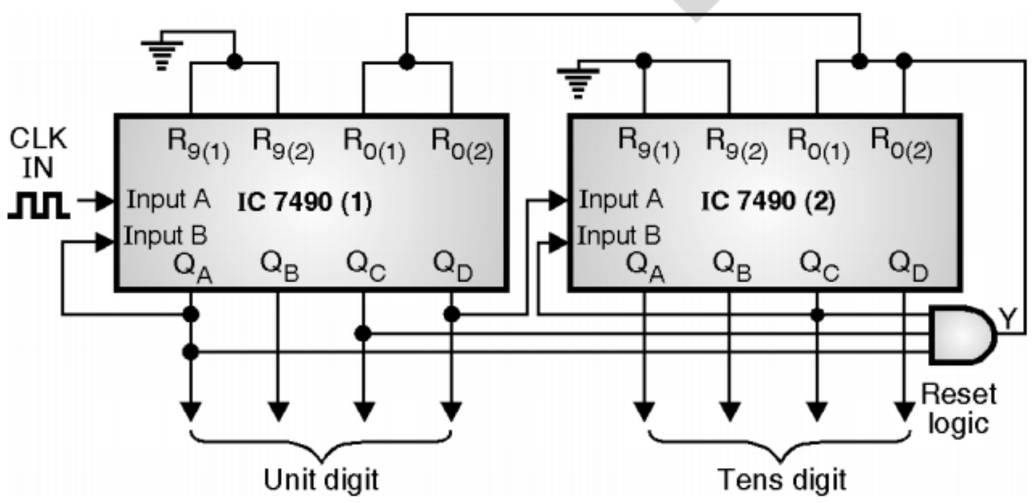
– So in order to reset the counter at 45, connect the Q outputs which are equal to 1 in the count of 45 to an AND gate as shown in Fig. P. 5.6.6(c) and then connect the AND output to the $R_{0(1)}$ and $R_{0(2)}$ i.e. reset inputs of both the ICs.



(C-6007) Fig. P. 5.6.6(c) : Reset logic

Step 3 : Draw the logic diagram :

– The logic diagram of the MOD-45 counter is shown in Fig. P. 5.6.6(d).



(C-3606) Fig. P. 5.6.6(d)

Ex. 5.6.7 : Design MOD-24 counter using 7490.

May 17, 2 Marks

Soln. :

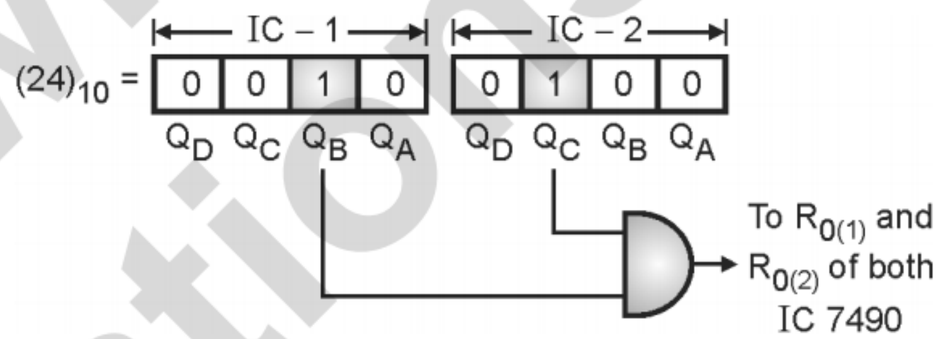
Step 1 : Number of ICs required :

– Upto MOD - 100, two IC 7490s will be sufficient. Hence to implement a divide by 24 counter, we have to use two decade counter ICs.

Step 2 : Design of reset logic :

– A divide by 24 counter counts through 24 states from 0 to 23 and the counter should reset as soon as the count becomes 24.

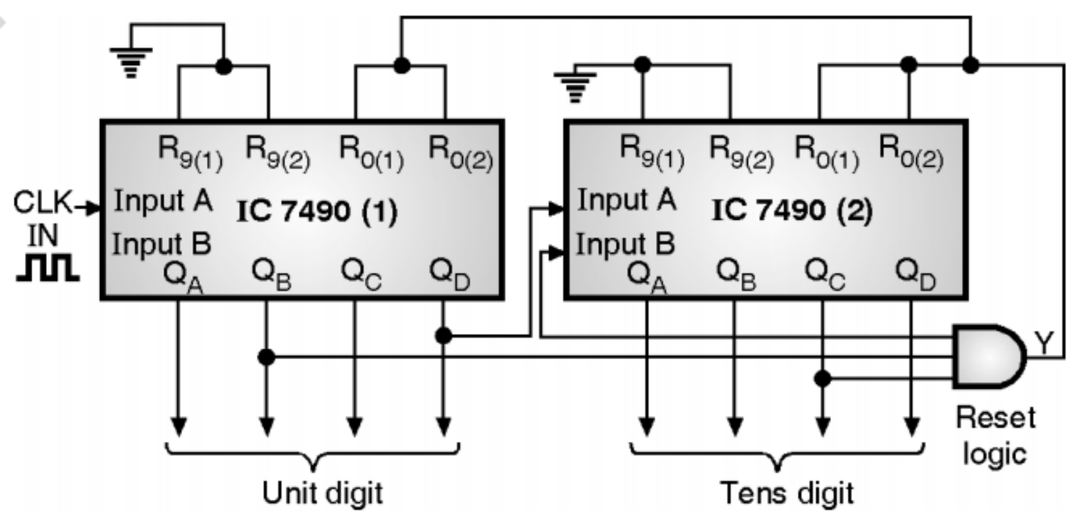
– So in order to reset the counter at 24, connect the Q outputs which are equal to 1 in the count of 24 to an AND gate as shown in Fig. P. 5.6.7(a) and then connect the AND output to the $R_{0(1)}$ and $R_{0(2)}$ i.e. reset inputs of both the ICs.



(C-8117) Fig. P. 5.6.7(a) : Reset logic

Step 3 : Draw the logic diagram :

– The logic diagram of the MOD-24 counter is shown in Fig. P. 5.6.7(b).



(C-5915) Fig. P. 5.6.7(b)

5.7 Problems Faced by Ripple Counters :

– The two major problems associated with the ripple counters are as follows :

1. Generation of unwanted short duration pulses called glitch.
2. Propagation delay.

TechKnowledge Publications

5.8 Synchronous Counters :

Definition :

- If the "clock" pulses are applied to all the flip-flops connected in a counter simultaneously, then such a counter is called as synchronous counter.
- These counters are also known as parallel counters. The state of all the flip flops will change simultaneously in the synchronous counter.

5.8.1 2-Bit Synchronous up Counter :

SPPU : Dec. 19

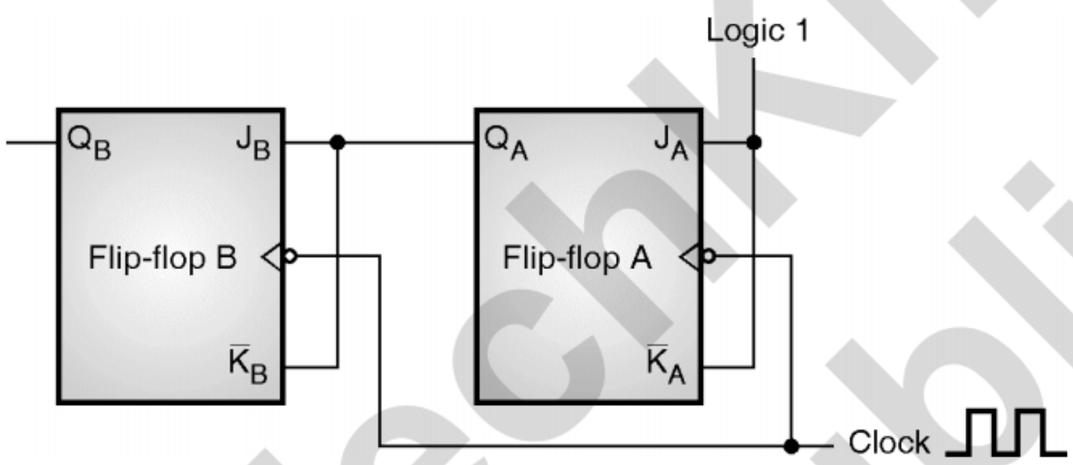
University Questions

Q. 1 Design 2-bit synchronous UP counter using MS-JK flip flop. Draw the circuit diagram.

(Dec. 19, 4 Marks)

Logic diagram :

- A 2-bit or MOD-4 synchronous counter is shown in Fig. 5.8.1.



(C-815) **Fig. 5.8.1 : A 2-bit (MOD-4) synchronous counter**

- The J_A and K_A inputs of FF-A are tied to logic 1. So FF-A will work as a toggle flip-flop. The J_B and K_B inputs are connected to Q_A.
- Hence FF-B will toggle if $Q_A = 1$ and there won't be any state change if $Q_A = 0$, at the instant when the negative clock edge is applied.

Operation :

- Initially let both the FFs be in the reset state. Let FF-A be the LSB flip-flop and FF-B be the MSB flip-flop.

  $\therefore \quad Q_B\,Q_A = 0\,0$ ..... Initially

At the 1st negative clock edge :

- As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will change from 0 to 1.
- But at the instant of application of negative clock edge, $Q_A = 0 \therefore J_B = K_B = 0$. Therefore FF-B will not change its state. So Q_B will remain 0.

  $\therefore \quad Q_B\,Q_A = 0\,1$ ..... After the first clock pulse

At the 2nd negative clock edge :

- At the instant when we apply the second negative clock edge, FF-A toggles again and Q_A changes from 1 to 0.
- But at this instant Q_A was 1. So $J_B = K_B = 1$ and FF-B also will toggle. Hence Q_B changes from 0 to 1.

  $\therefore \quad Q_B\,Q_A = 1\,0$ ..... After the second clock pulse

Next negative clock edges :

- Similarly on application of the third falling clock edge, FF-A will toggle from 0 to 1 but there is no change of state for FF-B.

  $\therefore \quad Q_B\,Q_A = 1\,1$ ..... After the third clock pulse

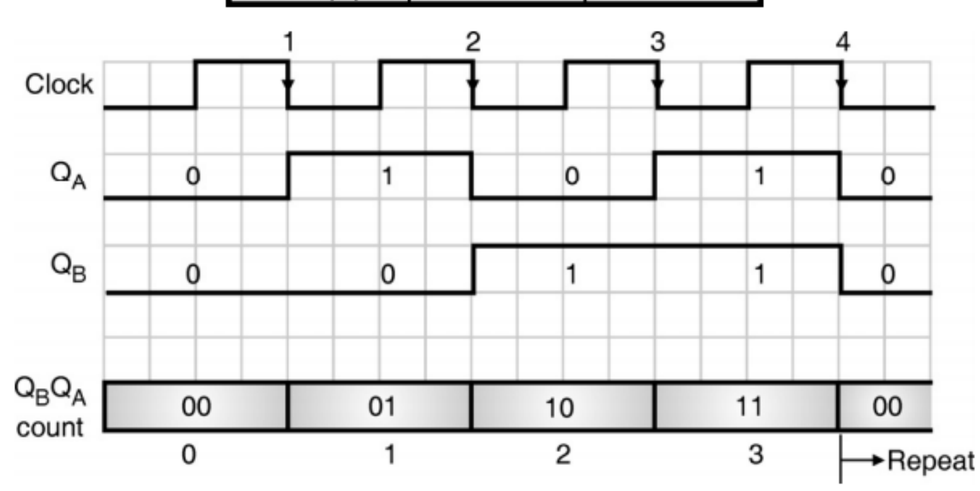
- On application of the next clock pulse, Q_A will change from 1 to 0 as Q_B will also change from 1 to 0. Hence

  $\therefore \quad Q_B\,Q_A = 0\,0$ ..... After the fourth clock pulse

- This is the original state. The operation of counter will repeat after this. The operation is summarised in Table 5.8.1 and the timing diagram is shown in Fig. 5.8.2.
- In this way the 2-bit synchronous counter has four distinct states namely $Q_B\,Q_A$ = 00, 01, 10 and 11.
- The maximum count of a 2-bit counter is $(11)_2$ or $(3)_{10}$.

(C-6224) **Table 5.8.1 : Summary of operation of a 2-bit synchronous counter**

Clock	Counter outputs	
	Q_B (MSB)	Q_A (LSB)
Initially	0	0
1st (↓)	0	1
2nd (↓)	1	0
3rd (↓)	1	1
4th (↓)	0	0



(C-816) **Fig. 5.8.2 : Timing diagram for a 2-bit synchronous counter**

TechKnowledge Publications

5.8.2 3-Bit Synchronous Binary up Counter :

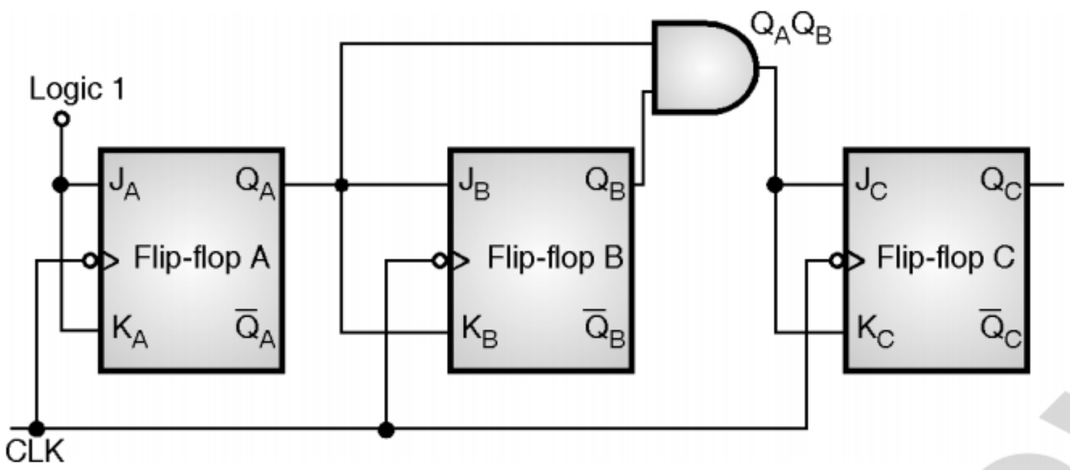
SPPU : May 12

University Questions

Q. 1 Design 3-bit synchronous up-counter using MS JK-flip-flop. **(May 12, 6 Marks)**

Logic diagram :

– We can extend the principle of operation of a 2-bit synchronous counter to a 3-bit counter shown in Fig. 5.8.3.



(C-817) Fig. 5.8.3 : A 3-bit synchronous binary counter

– FF-A acts as a toggle FF since $J_A = K_A = 1$.

– Q_A output of FF-A is applied to J_B as well as K_B. Hence if $Q_A = 1$ at the instant of triggering, then FF-B will toggle but if $Q_A = 0$ then FF-B will not change its state.

– Q_A and Q_B are ANDed and the output of AND gate is applied to J_C and K_C.

– Hence when Q_A and Q_B both are simultaneously high, then $J_C = K_C = 1$ and FF-C will toggle. Otherwise there is no change in the state of FF-C.

Operation :

– Initially all the FFs are in their reset state.

$\therefore \quad Q_C\, Q_B\, Q_A = 0\,0\,0$

1st clock pulse :

– FF-A toggles and Q_A changes to 1 from 0. But since $Q_A = 0$ at the instant of application of 1st falling clock edge, $J_B = K_B = 0$ and Q_B does not change state. $\therefore Q_B$ remains 0.

– Similarly Q_C also does not change state.

$\therefore \quad Q_C = 0.$

$\therefore \quad Q_C\, Q_B\, Q_A = 0\,0\,1$ ..... After 1st clock pulse

2nd clock pulse :

– FF-A toggles and Q_A becomes 0.

– But at the instant of application of 2nd falling clock edge Q_A was equal to 1. Hence $J_B = K_B = 1$. Hence FF-B will toggle and Q_B becomes 1.

– Output of AND gate is 0 at the instant of negative clock edge. So $J_C = K_C = 0$. Hence Q_C remains 0.

$\therefore \quad Q_C\, Q_B\, Q_A = 0\,1\,0$ ...After the 2nd clock pulse

3rd clock pulse :

– After the 3rd clock pulse, the outputs are $Q_C\, Q_B\, Q_A = 0\,1\,1$.

4th clock pulse :

– Note that $Q_B = Q_A = 1$. Hence output of AND gate = 1 and $J_C = K_C = 1$, at the instant of application of 4th negative edge of the clock.

– Hence on application of this clock pulse, FF-C will toggle and Q_C changes from 0 to 1.

– FF-A toggles as usual and Q_A becomes 0.

– Since Q_A was equal to 1 earlier, FF-B will also toggle to make $Q_B = 0$.

$\therefore \quad Q_C\, Q_B\, Q_A = 1\,0\,0$ ...After the 4th clock pulse

– Thus the counting progresses.

– After the 7th clock pulse the output is 111 and after the 8th clock pulse, all the flip-flops toggle and change their outputs to 0. Hence $Q_C\, Q_B\, Q_A = 0\,0\,0$ after the 8th pulse and the operation repeats.

– Table 5.8.2 summarizes operation of the three bit synchronous counter.

(C-6225) Table 5.8.2 : Summary of operation of a 3-bit synchronous counter

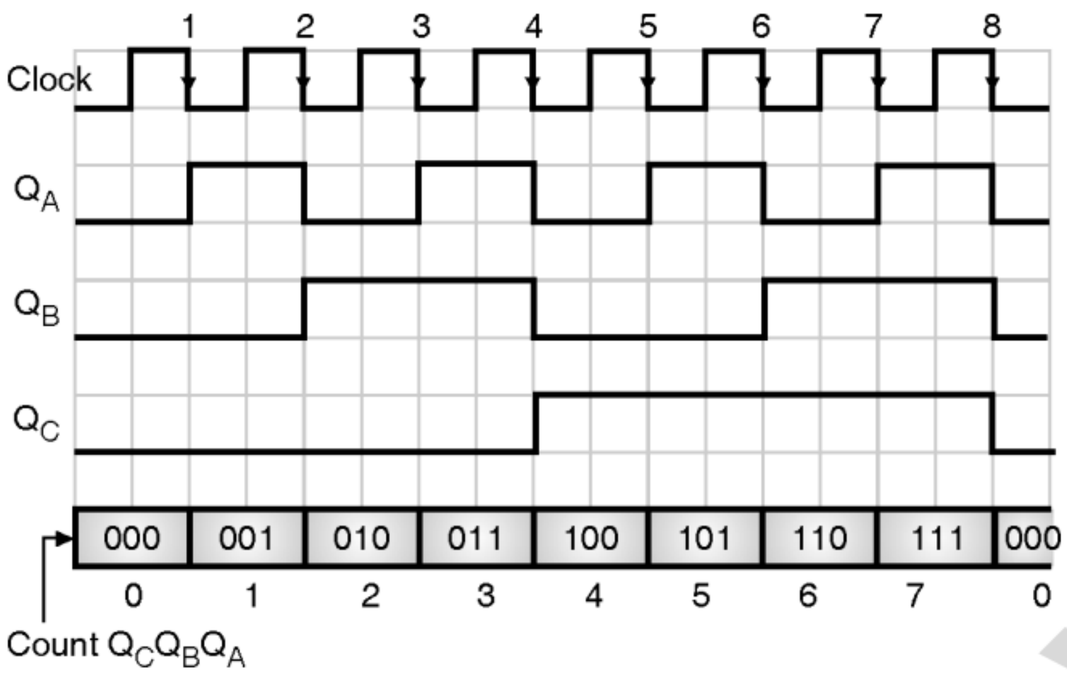
Clock	Q_C	Q_B	Q_A
0	0	0	0
1st (↓)	0	0	1
2nd (↓)	0	1	0
3rd (↓)	0	1	1
4th (↓)	1	0	0
5th (↓)	1	0	1
6th (↓)	1	1	0
7th (↓)	1	1	1

Timing diagram :

– Timing diagram for a 3-bit synchronous counter is shown in Fig. 5.8.4.

TechKnowledge Publications

- The number of states through which this counter progresses is 8 namely Q_C Q_B Q_A = 000, 001, 010, 011, 100, 101, 110, 111.
- The maximum count is $(111)_2$ or $(7)_{10}$.
- Note that the waveforms of synchronous counter are exactly same as those of an asynchronous counter.



(C-818) Fig. 5.8.4 : Timing diagram for a 3-bit synchronous counter

5.8.3 Design of the 3 Bit Synchronous Counter : SPPU : May 18

University Questions

Q. 1 Design 3-bit synchronous counter using T flip-flop. **(May 18, 4 Marks)**

- Let us now design a 3 bit synchronous counter first using the T flip flops and then using JK flip flops.

Design using T flip flops :

Steps to be followed :

Step 1 : Decide the number of flip flops.

Step 2 : Write the excitation table of T flip flop.

Step 3 : Write the excitation table of the counter.

Step 4 : From the circuit excitation table write K-maps and obtain simplified equations.

Step 5 : Draw the logic diagram.

Step 1 : Decide number of FFs :

- A 3 bit counter goes through 8 states. So it needs three flip flops.

Step 2 : Excitation table of T FFs :

- Table 5.8.3(a) shows the excitation table of T FF.

(C-7842) Table 5.8.3(a) : Excitation table of a T FF

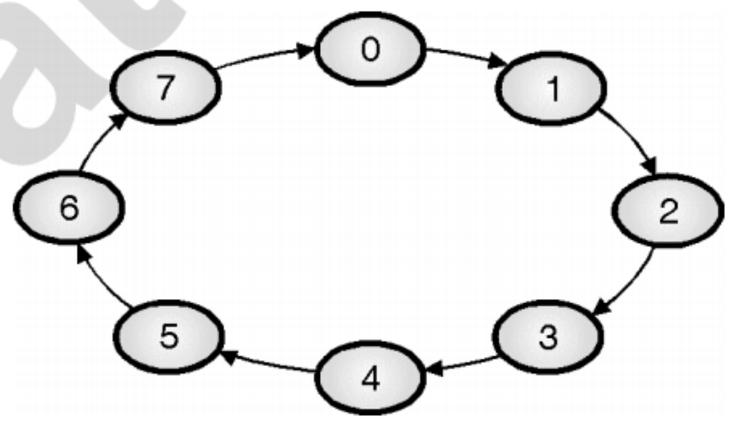
Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Step 3 : State diagram and circuit excitation table :

- Count sequence for a 3 bit up counter is given in Table 5.8.3(b) and Fig. 5.8.5 shows the corresponding state diagram.

(C-6226) Table 5.8.3(b)

Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



(C-819) Fig. 5.8.5 : State diagram

- Table 5.8.3(c) shows the circuit excitation table.

(C-7838) Table 5.8.3(c) : Circuit excitation table

Present state			Next state			Flip flop input		
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A
0	0	0	**0**	0	1	**0**	0	1
0	0	1	**0**	1	0	**0**	1	1
0	1	0	**0**	1	1	**0**	0	1
0	1	1	**1**	0	0	**1**	1	1
1	0	0	**1**	0	1	**0**	0	1
1	0	1	**1**	1	0	**0**	1	1
1	1	0	**1**	1	1	**0**	0	1
1	1	1	**0**	0	0	**1**	1	1

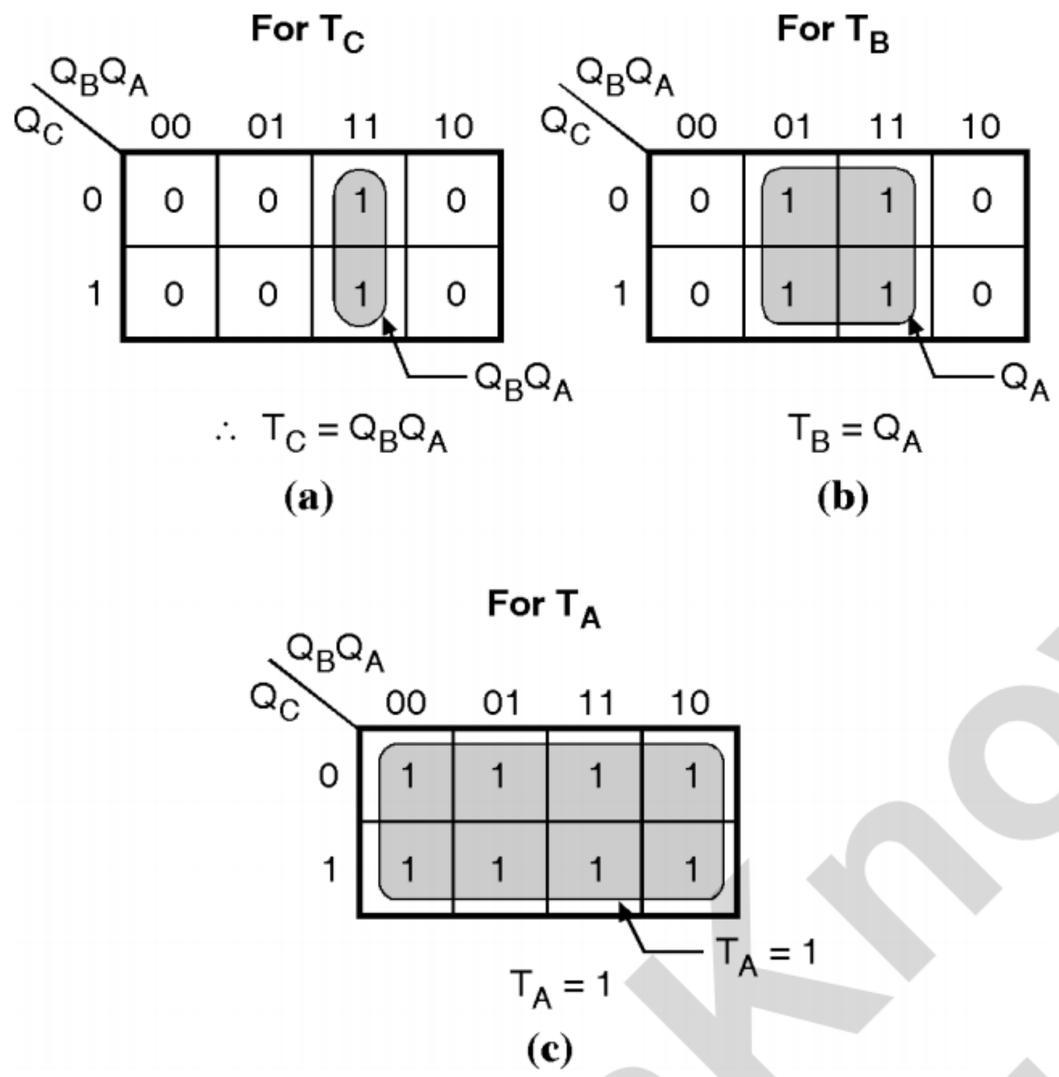
- Table 5.8.3(c) has been written by referring to the present and next state of an output and the required value to input is written as per the excitation table of T FF.

Tech Knowledge Publications

– For example consider the shaded columns of Table 5.8.3(c). i.e. Q_C, Q_{C+1} and T_C

– Consider the first row. $Q_C = 0$, $Q_{C+1} = 0$. So as per the excitation table of a T FF T_C should be 0. Similarly all other entries are made.

Step 4 : Write K maps and obtain simplified equations :

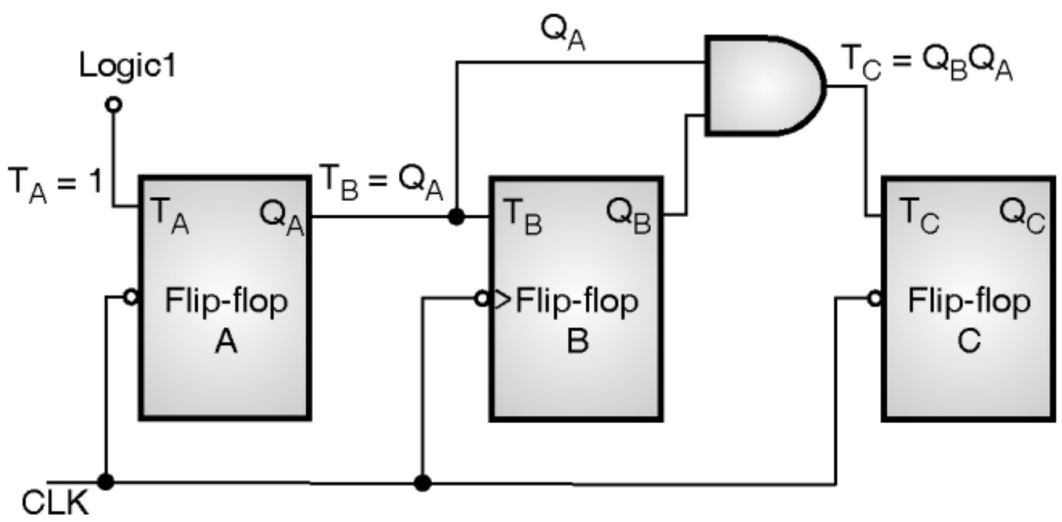


(C-1506) **Fig. 5.8.6 : K-maps for different FF inputs**

– Refer Figs. 5.8.6(a), (b), (c) for the K-maps corresponding to all the FF inputs. The simplified equations for T_A, T_B and T_C also are shown.

Step 5 : Draw the logic diagram :

– By using the simplified equations for T_A, T_B and T_C we can draw the logic diagram for the 3 bit synchronous shown in Fig. 5.8.7.



(C-1507) **Fig. 5.8.7 : Logic diagram of a 3-bit synchronous counter**

Design using JK flip flops :

Step 1 : Number of flip flops :

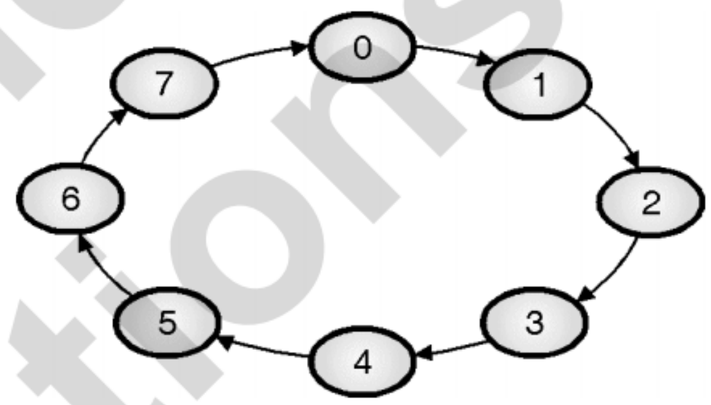
– For a 3 bit counter, we need 3 flip-flops.

Step 2 : Excitation table of JK FF :

(C-7761) **Table 5.8.4 : Excitation table of JK FF**

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3 : State diagram and circuit excitation table :



(C-1508) **Fig. 5.8.8 : State diagram**

(C-7050) **Table 5.8.5 : Circuit excitation table**

Present state			Next state			Flip flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Step 4 : K maps and simplified expressions for all FF inputs :

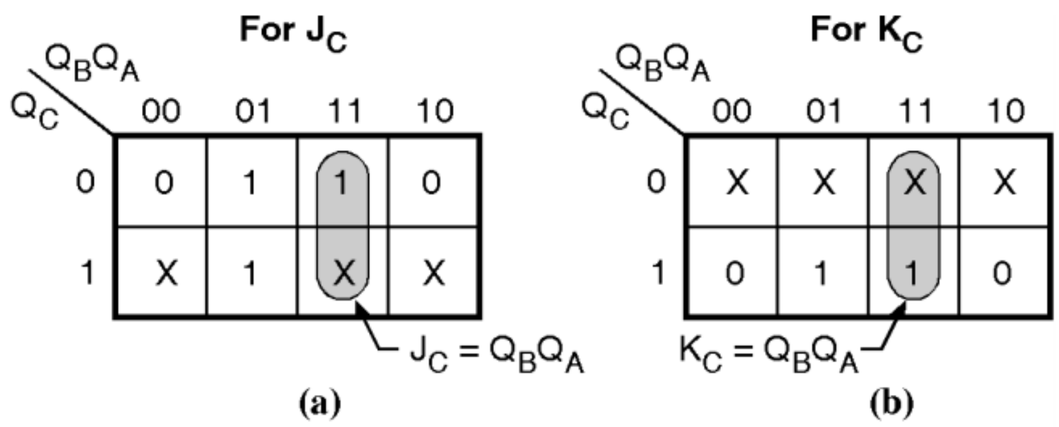
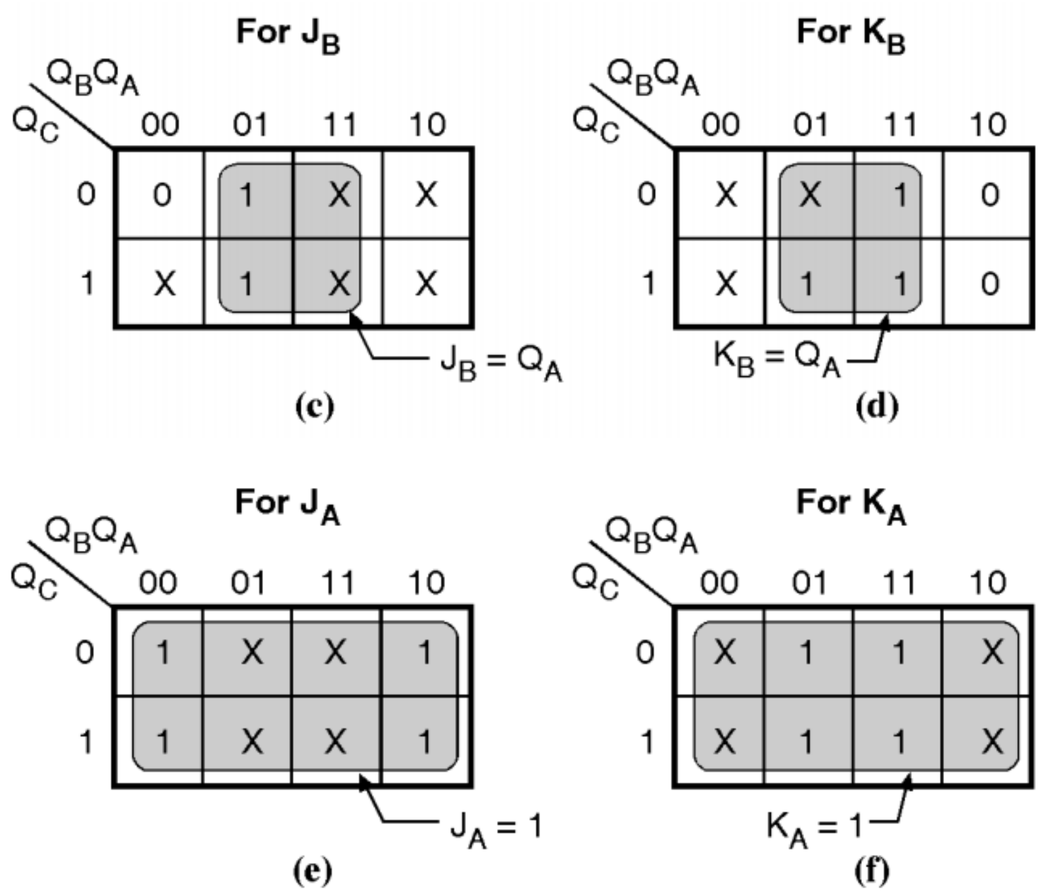


Fig. 5.8.9 (cont...)

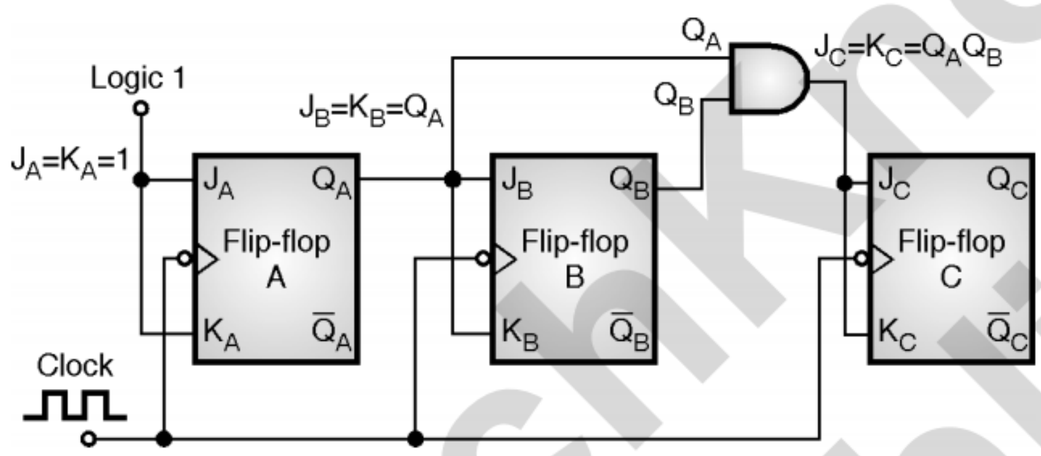
TechKnowledge Publications



(C-1509) Fig. 5.8.9 : K-maps and simplified equations for different FF inputs

Step 5 : Logic diagram :

– Fig. 5.8.10 shows the logic diagram of a 3 bit synchronous counter using JK flip-flops.



(C-1510) Fig. 5.8.10 : 3 bit synchronous counter using JK FFs

5.8.4 Four Bit Synchronous Up Counter :

SPPU : Dec. 10

University Questions

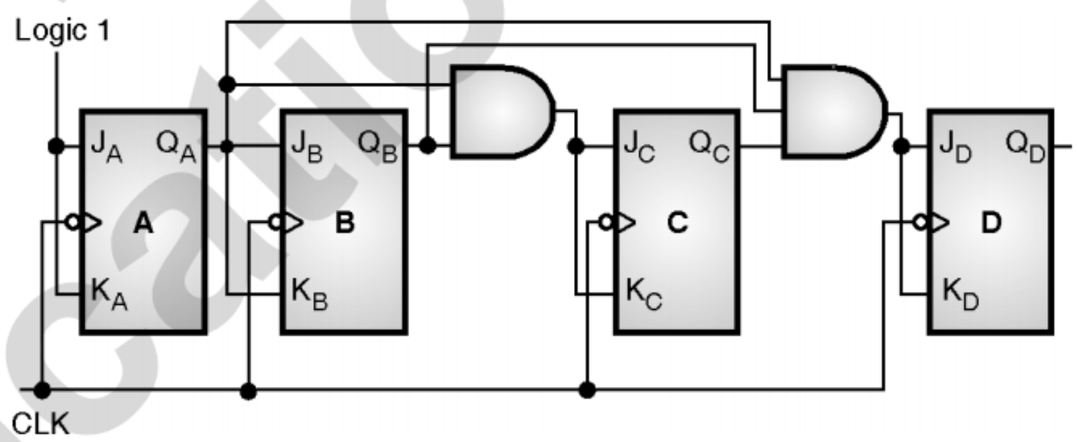
Q. 1 Draw a 4-bit synchronous counter. Also explain timing diagram for the same. **(Dec. 10, 10 Marks)**

Logic diagram :

– All the concepts of a synchronous counter are extended to design a 4-bit synchronous counter shown in Fig. 5.8.11.

– Note that J_D and K_D of the FF-D are connected to the output of an AND gate and the inputs of this AND gate come from the outputs of the three preceding FFs.

– Operating principle of this counter is same as that of the 3-bit counter discussed earlier.

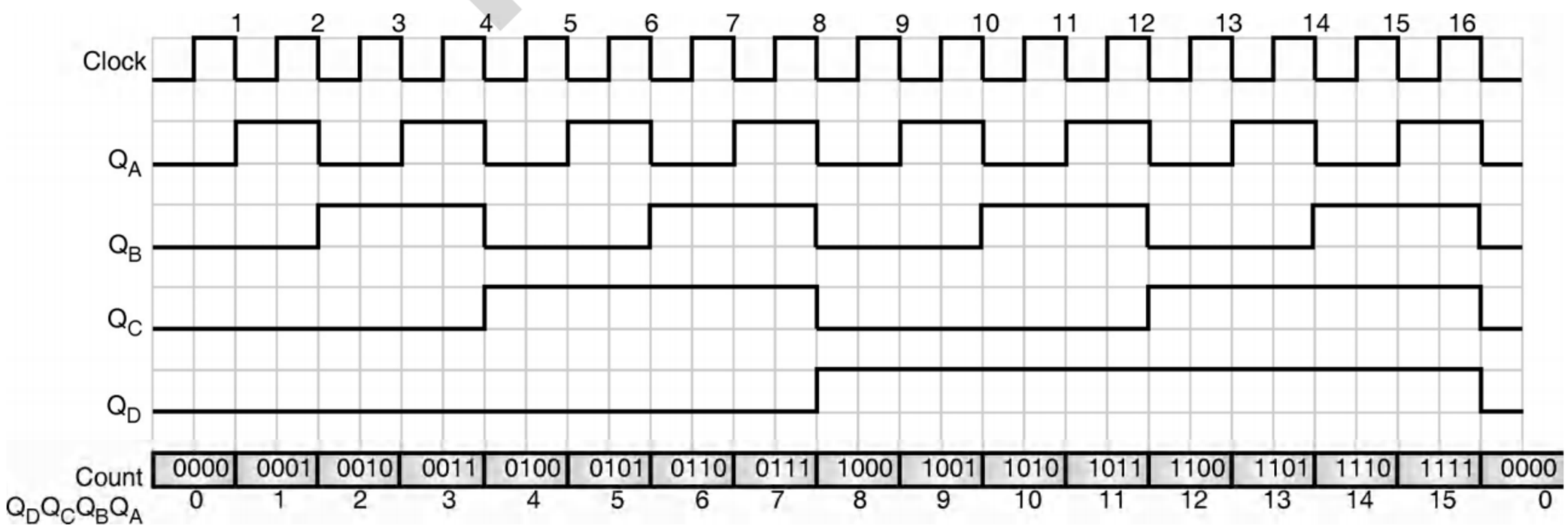


(C-1391) Fig. 5.8.11 : A four bit synchronous counter

Timing diagram :

– The timing diagram of a four bit synchronous counter is as shown in Fig. 5.8.12.

– Note that the principle of frequency division is applicable to the synchronous counters as well.



(C-1392) Fig. 5.8.12 : Timing diagram of a 4-bit synchronous counter

TechKnowledge Publications

5.9 Modulo – N Synchronous Counters :

– We have already discussed the MOD-N asynchronous counters. Now let us discuss the Modulo-N synchronous counters.

– The steps to be followed to design a MOD-N synchronous counter are as follows.

Steps to be followed :

Step 1 :	Decide the number of FFs and type of FF to be used.
Step 2 :	Write the circuit excitation table.
Step 3 :	From the circuit excitation table write down the K-maps and obtain simplified expressions for the outputs.
Step 4 :	Draw the logic diagram and timing diagram.

5.9.1 Synchronous Decade Counter :

– For the design of a synchronous decade counter follow the steps given below :

Step 1 : Write the excitation table for T FF and circuit excitation table :

– Excitation table for T FF is shown in Table 5.9.1.

(C-7842) Table 5.9.1 : Excitation table for T flip-flop

Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

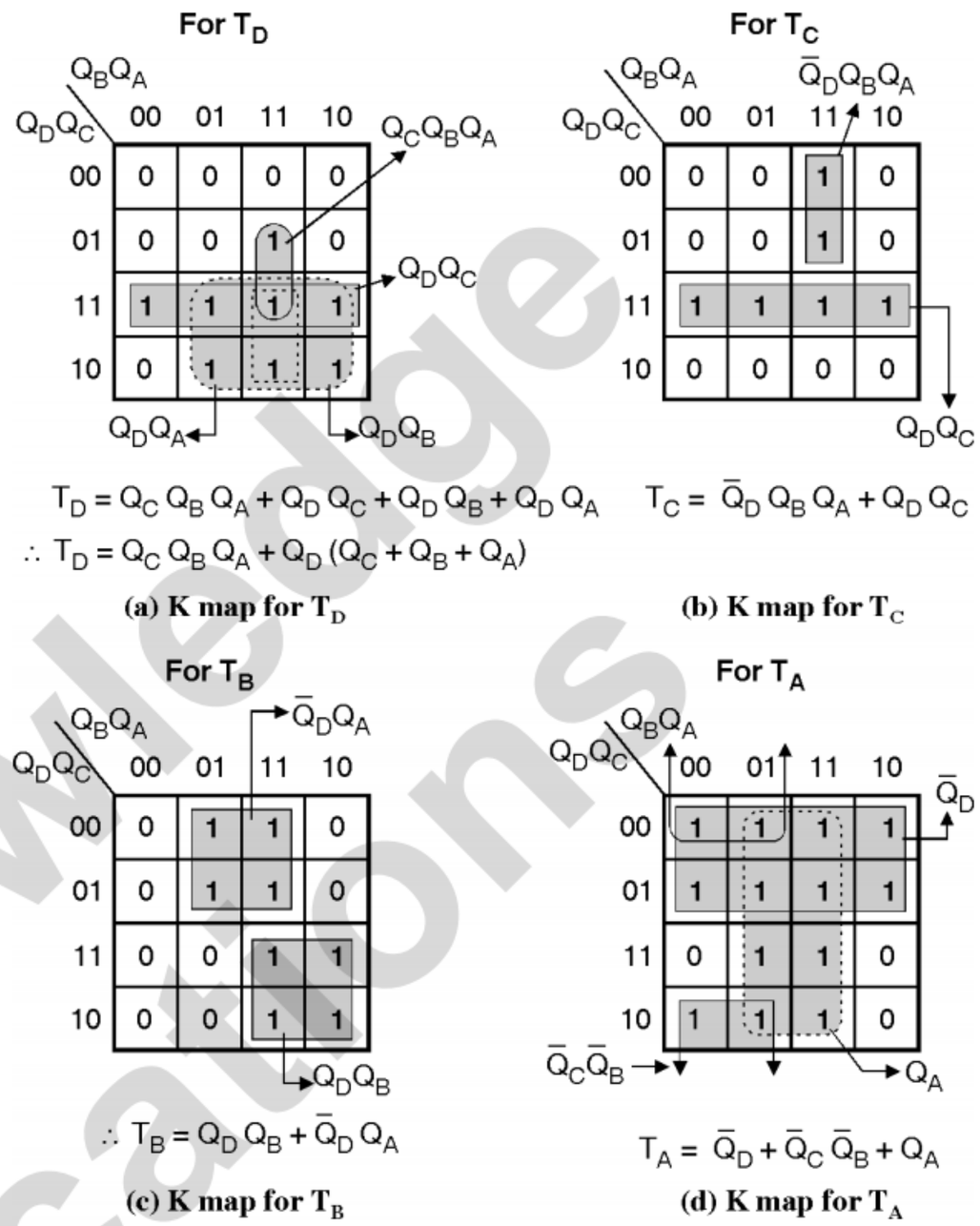
– The circuit excitation table is shown in Table 5.9.2.

(C-7839) Table 5.9.2 : Circuit excitation table

Present state				Next state				Flip flop inputs			
Q_D	Q_C	Q_B	Q_A	Q_{D+1}	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	0	0	0	0	1	0	1	0
1	0	1	1	0	0	0	0	1	0	1	1
1	1	0	0	0	0	0	0	1	1	0	0
1	1	0	1	0	0	0	0	1	1	0	1
1	1	1	0	0	0	0	0	1	1	1	0
1	1	1	1	0	0	0	0	1	1	1	1

Step 2 : K-maps and simplifications :

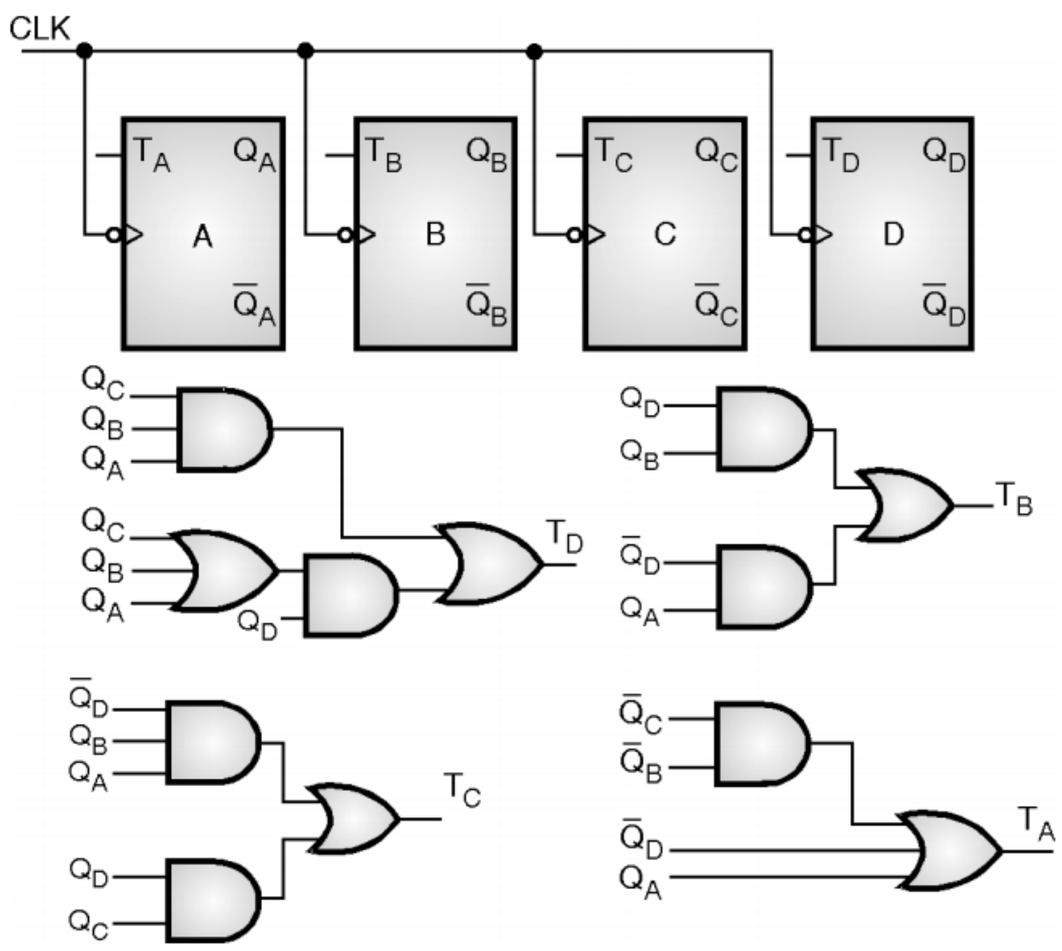
– K-maps for T_D, T_C, T_B, T_A and their simplified expressions are given in Figs. 5.9.1(a), (b), (c) and (d).



(C-847) Fig. 5.9.1

Step 3 : Draw the logic diagram :

– Fig. 5.9.2 shows the logic diagram of a synchronous decade counter.



(C-848) Fig. 5.9.2 : Logic diagram of a synchronous decade counter

TechKnowledge Publications

Ex. 5.9.1 : Design a synchronous counter for the sequence shown in Fig. P. 5.9.1(a).

Dec. 13, 4 Marks, May 14, 6 Marks, May 15, 4 Marks

(C-6227) **Table P. 5.9.1(a) : Desired sequence**

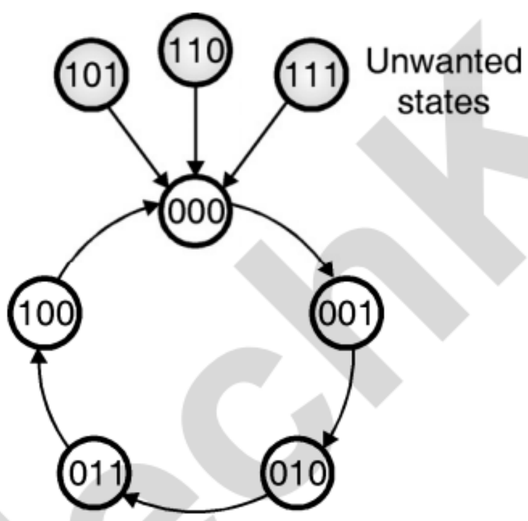
Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

Soln. :

Step 1 : Determine the desired number of FFs :

– From the given sequence the number of FFs is equal to 3. This is a MOD-5 synchronous counter since the number of states is 5.

– The state diagram is shown in Fig. P. 5.9.1(a) which shows that 101, 110, 111 are unwanted states.



(C-849) **Fig. P. 5.9.1(a) : State diagram**

Step 2 : Write the excitation table and state table :

– The type of FF used is JK flip-flop. The excitation table for a JK FF is as shown in Table P. 5.9.1(b).

– We have already seen how to write the excitation table for JK FF.

Excitation table of JK FF :

– Table P. 5.9.1(b) exhibits the excitation table of a JK FF.

(C-7840) **Table P. 5.9.1(b) : Excitation table of JK FF**

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Circuit excitation table :

– The circuit excitation table is as shown in Table P. 5.9.1(c).

(C-7841) **Table P. 5.9.1(c) : Circuit excitation table**

Present state			Next state			Flip flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
1	0	0	**0**	0	1	**0**	×	0	×	1	×
1	0	1	**0**	1	0	**0**	×	1	×	×	1
1	1	0	**0**	1	1	**0**	×	×	0	1	×
1	1	1	**1**	0	0	**1**	×	×	1	×	1
0	0	0	**0**	0	0	×	1	0	×	0	×
0	0	1	**0**	0	0	×	1	0	×	×	1
0	1	0	**0**	0	0	×	1	×	1	0	×
0	1	1	**0**	0	0	×	1	×	1	×	1

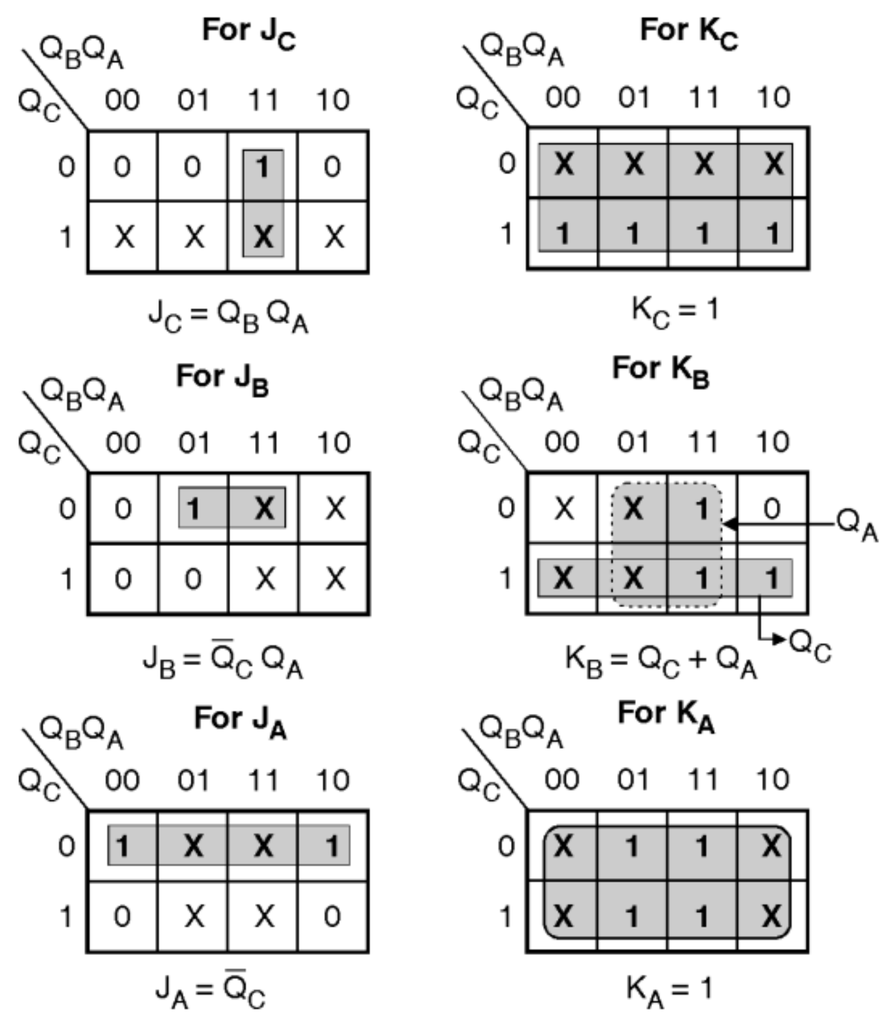
– Refer to the shaded portion of the circuit excitation table. This is nothing but the excitation table of FF-C. The J_C and K_C values have been decided based on Q_C and Q_{C+1}.

– Similarly the entries for J_B and K_B are based on Q_B and Q_{B+1} whereas those for J_A and K_A are based on Q_A and Q_{A+1}.

Step 3 : K-maps and simplifications :

– K-maps for the J and K inputs of all the FFs and the corresponding simplified equations are shown in Fig. P. 5.9.1(b).

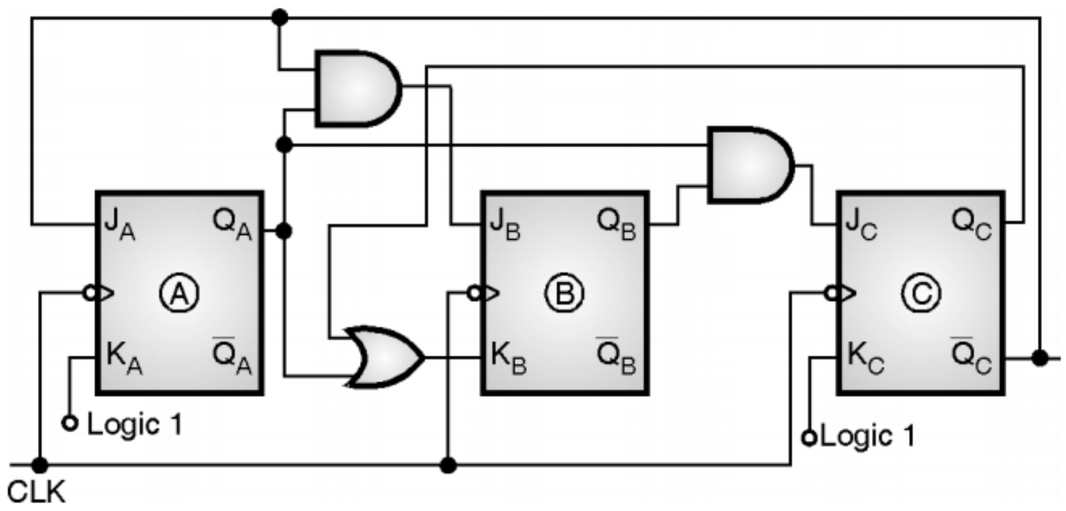
– Note that the inputs to this combinational circuit are Q_A, Q_B, Q_C and outputs are J_A, K_A through J_C, K_C.



(C-850) **Fig. P. 5.9.1(b) : K-maps and simplifications**

Step 4 : Logic diagram :

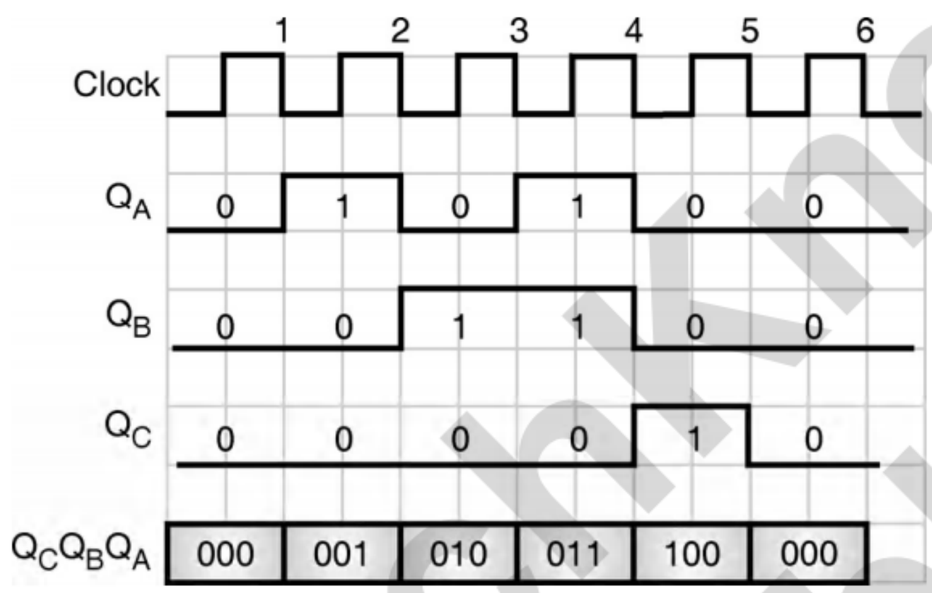
– The logic diagram of MOD-5 synchronous counter is shown in Fig. P. 5.9.1(c).



(C-851) **Fig. P. 5.9.1(c) : Logic diagram of MOD-5 synchronous counter**

Step 5 : Draw the timing diagram :

– The timing diagram of MOD-5 synchronous counter is shown in Fig. P. 5.9.1(d).



(C-852) **Fig. P. 5.9.1(d) : Timing diagram of MOD-5 counter**

Ex. 5.9.2 : Design a MOD-5 synchronous counter using T flip-flops.

Soln. :

Step 1 : Decide number of FFs :

– Since the number of states is 5 we need to use 3 flip-flops.

Step 2 : Excitation tables :

– The excitation table of a T flip-flop is shown in Table P. 5.9.2(a).

(C-7842) **Table P. 5.9.2(a) : Excitation table of a T flip-flop**

Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Circuit excitation table :

– The circuit excitation table is shown in Table P. 5.9.2(b).

– Refer to the shaded portion of the circuit excitation table.

– This is nothing but the excitation table of FF-C. The T_C values are decided based on Q_C and Q_{C+1}.

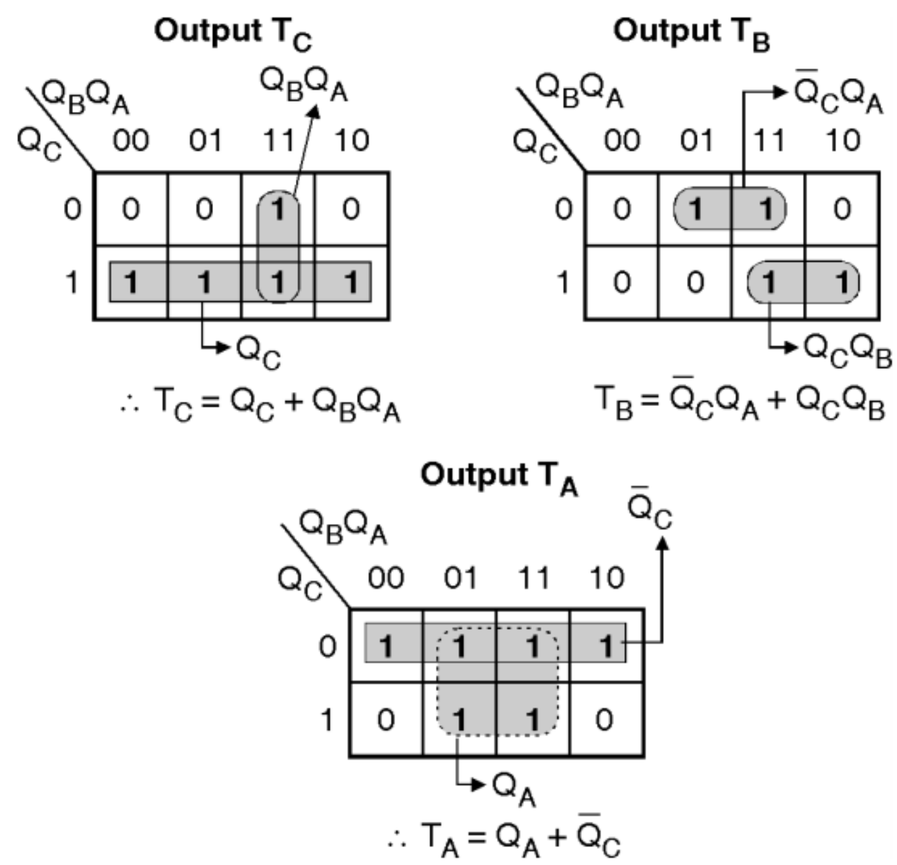
– Similarly the entries for T_B are based on Q_B and Q_{B+1} whereas those for T_A are based on Q_A and Q_{A+1}.

(C-7843) **Table P. 5.9.2(b) : Circuit excitation table**

Present state			Next state			Flip flop inputs		
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A
0	0	0	**0**	0	1	**0**	0	1
0	0	1	**0**	1	0	**0**	1	1
0	1	0	**0**	1	1	**0**	0	1
0	1	1	**1**	0	0	**1**	1	1
1	0	0	**0**	0	0	**1**	0	0
1	0	1	**0**	0	0	**1**	0	1
1	1	0	**0**	0	0	**1**	1	0
1	1	1	**0**	0	0	**1**	1	1

Step 3 : K-maps and simplifications :

– K-maps for the T_A, T_B and T_C inputs of the three FFs and the corresponding simplified equations are shown in Fig. P. 5.9.2(a).

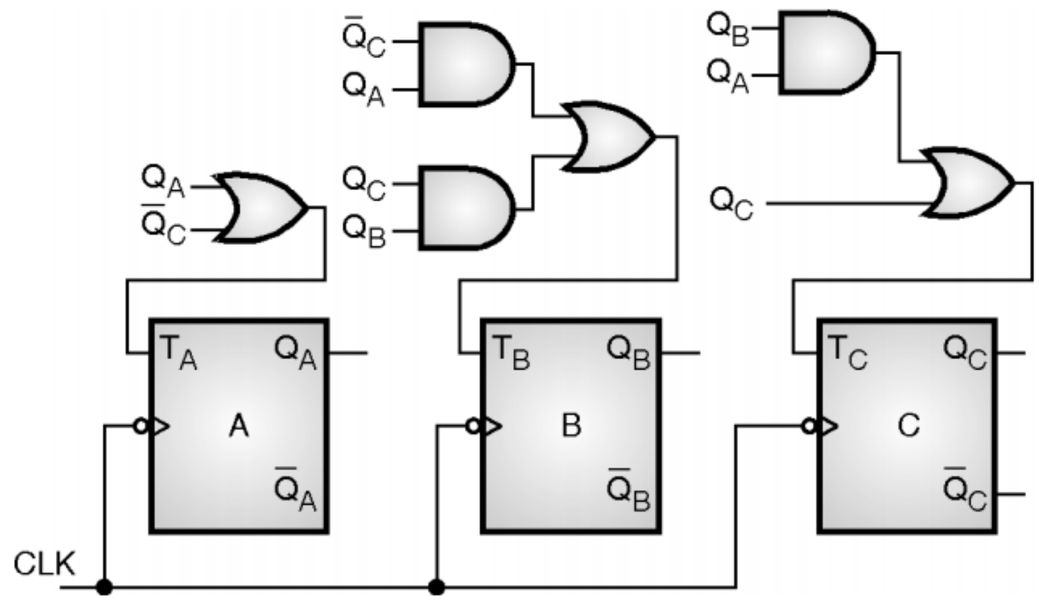


(C-853) **Fig. P. 5.9.2(a) : K-maps and simplifications**

Step 4 : Draw the logic diagram :

– The logic diagram of MOD-5 synchronous counter using T FFs is shown in Fig. P. 5.9.2(b).

TechKnowledge Publications



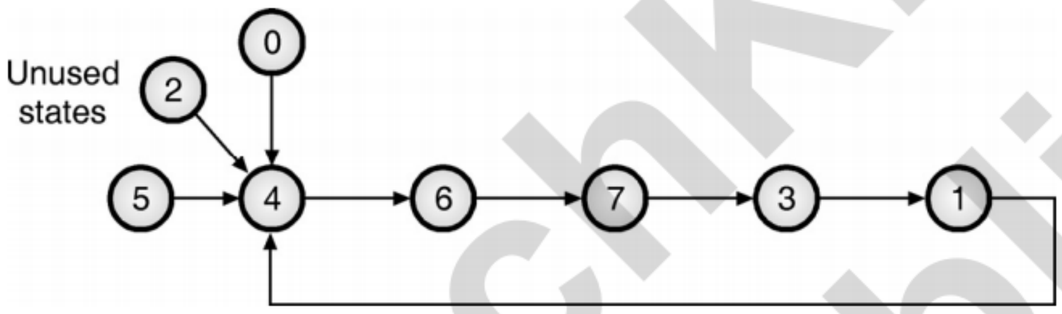
(C-854) Fig. P. 5.9.2(b) : MOD-5 synchronous counter using T flip-flops

Ex. 5.9.3 : Design a synchronous counter for 4 – > 6 – > 7 – > 3 – > 1 – > 4 ... Avoid lockout condition. Use JK flip-flop for design. **Dec. 15, 6 Marks**

Soln. :

Step 1 : Draw the state diagram :

– The state diagram is shown in Fig. P. 5.9.3(a). It shows that the next state of all unused states (i.e. 0, 2 and 5) is forced to be the initial state 4. This is essential in order to avoid the lockout condition.



(C-5166) Fig. P. 5.9.3(a) : State diagram of desired counter

Step 2 : Number of flip flops and type of flip-flop :

– Since the highest state is 7 i.e. 111 we need to use 3 flip flop.

Step 3 : Write the state table :

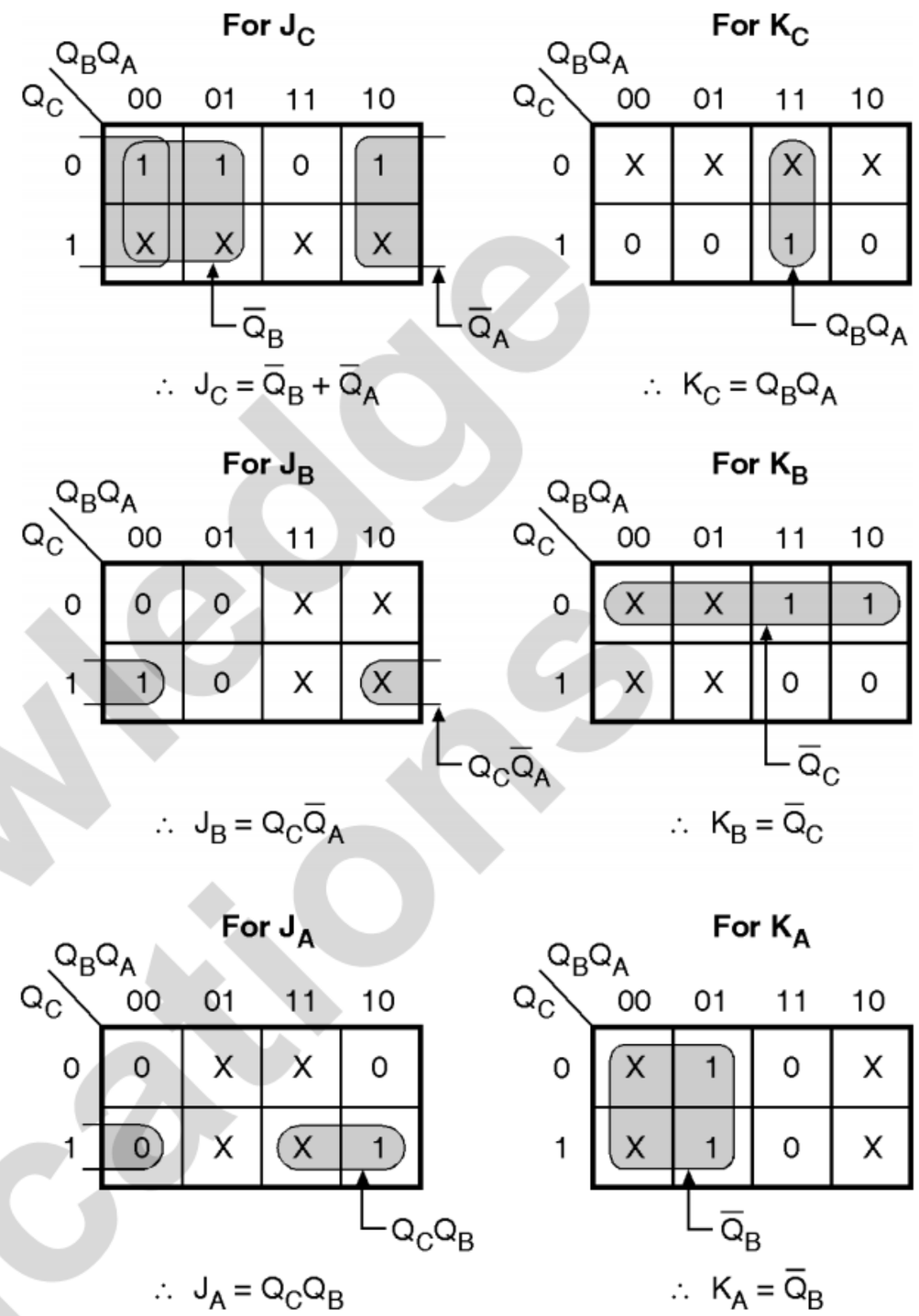
– Table P. 5.9.3 shows the state table for the required counter.

(C-8118) Table P. 5.9.3

Present state			Next state			Flip flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Step 4 : K-maps and simplification :

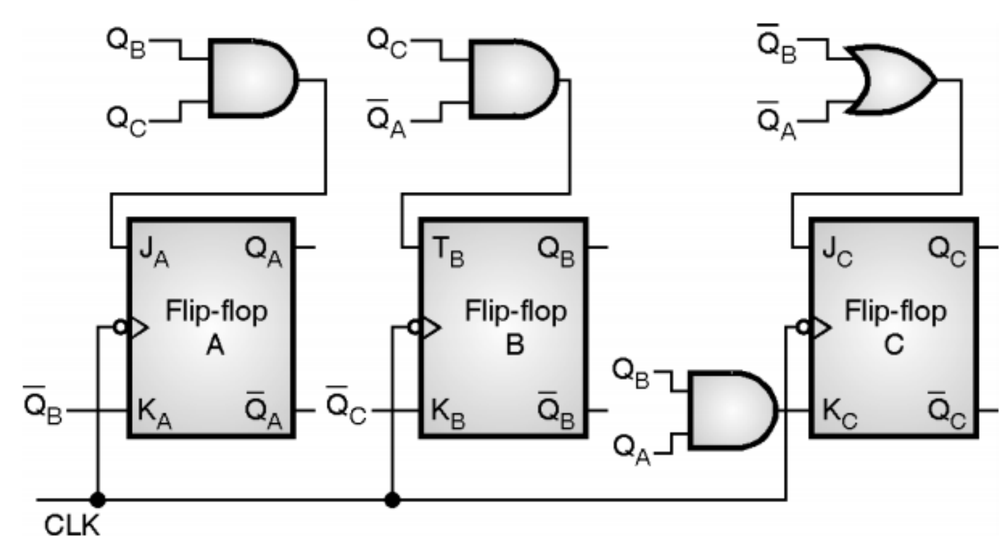
– Fig. P. 5.9.3(b) illustrates the K-maps for all the FF inputs and the corresponding simplified expressions.



(C-5167) Fig. P. 5.9.3(b)

Step 5 : Draw the logic diagram :

– The logic diagram of the required synchronous counter is as shown in Fig. P. 5.9.3(c).



(C-5168) Fig. P. 5.9.3(c) : Logic diagram using JK flip flop

5.10 UP / DOWN Synchronous Counter :

– The operating principle of a synchronous up down counter is same as that of the up/down ripple counters. But the design procedure and realization are different.

TechKnowledge Publications

– The mode control (M) is used for selecting the mode of operation.

– The steps to be followed for the design are as follows :

Steps to be followed :

Step 1 : Write the circuit excitation table.

Step 2 : Write K-maps and obtain the simplified expressions.

Step 3 : Draw the logic diagram.

5.10.1 3-bit Up/Down Synchronous Counter :

SPPU : Dec. 08, May 10

University Questions

Q. 1 Explain with a neat diagram working of 3-bit up-down synchronous counter. Draw necessary timing diagram. **(Dec. 08, May 10, 10 Marks)**

– The number of FFs to be used is three. We will use three toggle flip-flops.

– Let upcounting takes place with M = 0 and down counting takes place for M = 1.

Step 1 : Write the circuit excitation table :

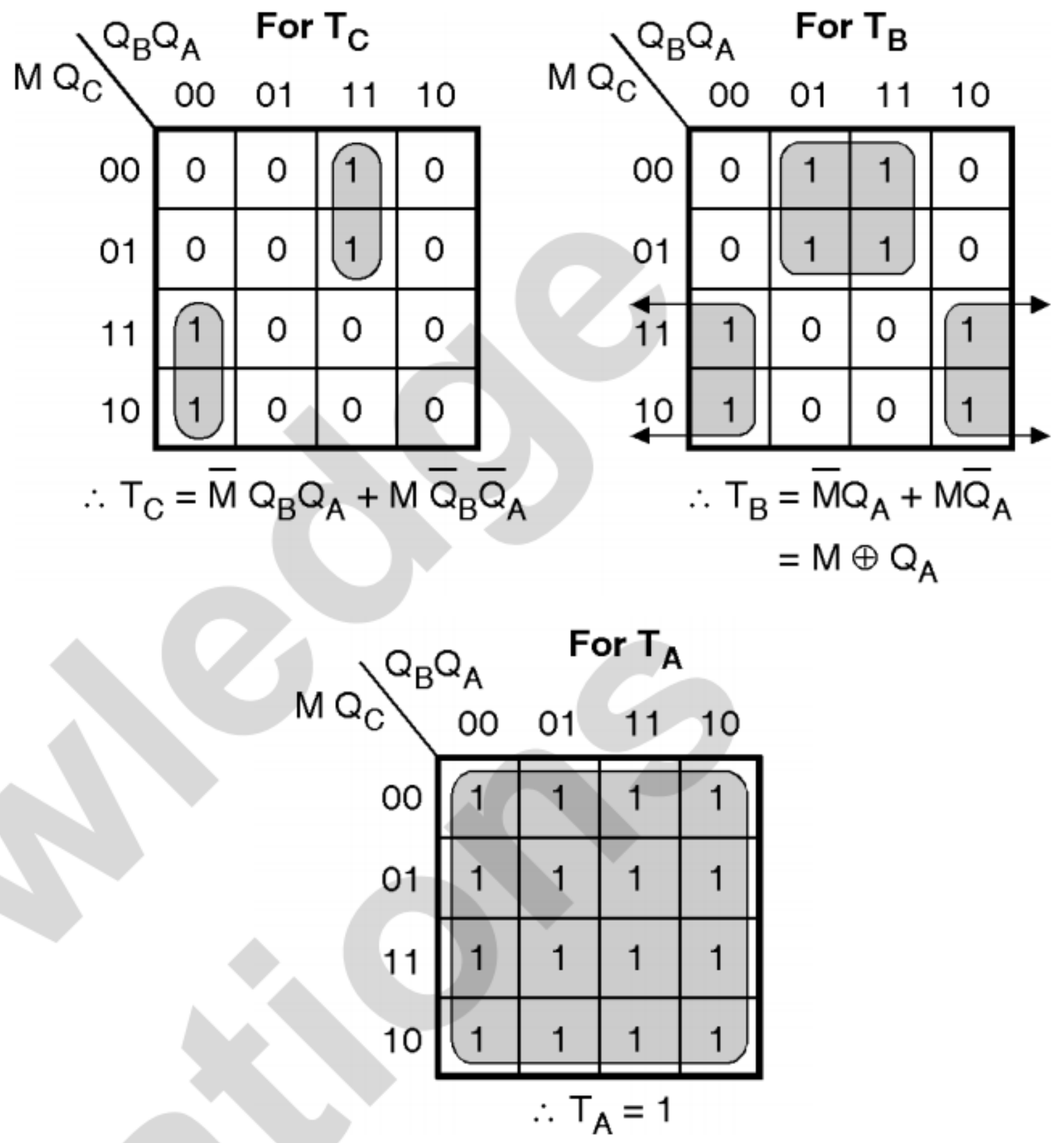
– The circuit excitation table is shown in Table 5.10.1.

(C-6228) Table 5.10.1 : Excitation table for a 3-bit up/down synchronous counter

Mode control M	Present state			Next state			Flip-flop inputs			
	Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A	
0	0	0	0	0	0	1	0	0	1	Up counting
0	0	0	1	0	1	0	0	1	1	
0	0	1	0	0	1	1	0	0	1	
0	0	1	1	1	0	0	1	1	1	
0	1	0	0	1	0	1	0	0	1	
0	1	0	1	1	1	0	0	1	1	
0	1	1	0	1	1	1	0	0	1	
0	1	1	1	0	0	0	1	1	1	
1	0	0	0	1	1	1	1	1	1	Down counting
1	0	0	1	0	0	0	0	0	1	
1	0	1	0	0	0	1	0	1	1	
1	0	1	1	0	1	0	0	0	1	
1	1	0	0	0	1	1	1	1	1	
1	1	0	1	1	0	0	0	0	1	
1	1	1	0	1	0	1	0	1	1	
1	1	1	1	1	1	0	0	0	1	

Step 2 : K-maps and simplified equations :

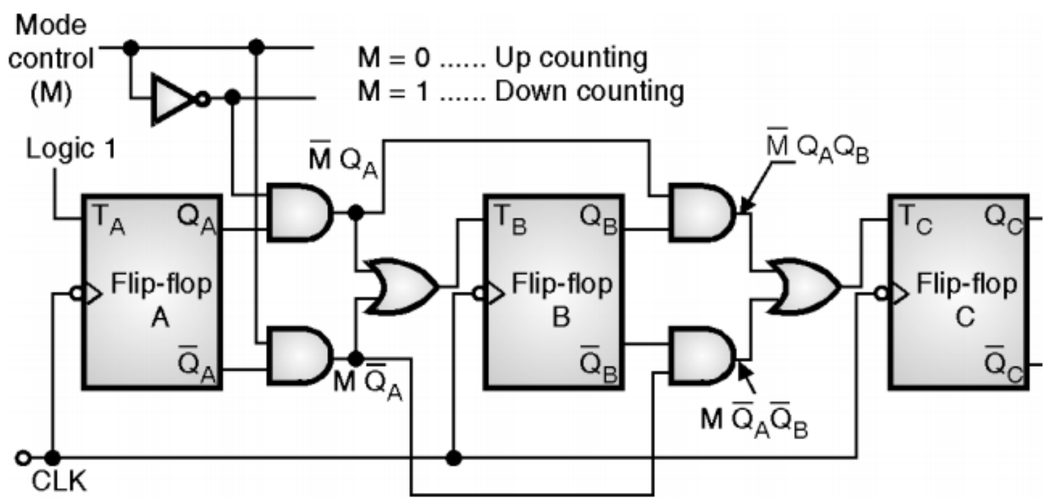
– The K-maps and simplified equations for T_C, T_B and T_A are shown in Fig. 5.10.1.



(C-855) Fig. 5.10.1 : K-maps and simplified expressions

Step 3 : Draw the logic diagram :

– The logic diagram for a 3-bit synchronous up down counter is shown in Fig. 5.10.2.



(C-856) Fig. 5.10.2 : Logic diagram of a 3-bit synchronous up down counter

5.10.2 Advantages of Synchronous Counter :

1. All the FFs receive the clock signal simultaneously i.e. at the same instant, the propagation delay problem is reduced to a great extent. Recall that the propagation delay is one of the problems of the ripple counter.

TechKnowledge Publications

2. The total propagation delay between the instant of application of clock edge and the instant at which the MSB output changes is equal to sum of propagation delay of one FF and that of one AND gate.

∴ Total propagation delay = t_d of one FF + t_d of one gate

3. This is much less than the propagation delay of an asynchronous (ripple) counter.

   Due to reduced propagation delay, the synchronous counters can operate at a much higher clock frequency than the asynchronous counters.

5.10.3 Comparison of Synchronous and Asynchronous Counters :

SPPU : May 07, May 12, Dec. 12, Dec. 18

University Questions

Q. 1 What is the difference between synchronous counters and asynchronous counters ? **(May 07, May 12, Dec. 12, 4 Marks)**

Q. 2 Compare synchronous and asynchronous counter. **(Dec. 18, 2 Marks)**

– Comparison of synchronous and asynchronous counters is given in Table 5.10.2.

Table 5.10.2 : Comparison of synchronous and asynchronous counters

Sr. No.	Parameter	Asynchronous counter	Synchronous counter
1.	Circuit complexity	Logic circuit is simple	With increase in number of states, the logic circuit becomes complicated.
2.	Connection pattern	Output of the preceding FF, is connected to clock of the next FF.	There is no connection between output of preceding FF and CLK of next one.
3.	Clock input	All the FFs are not clocked simultaneously.	All FFs receive clock signal simultaneously.
4.	Propagation delay	P.D. = n × (t_d) where n is number of FFs and t_d is p.d. per FF.	P.D. = $(t_d)_{FF}$ + $(t_d)_{gate}$. It is much shorter than that of asynchronous counter.
5.	Maximum frequency of operation	Low because of the long propagation delay.	High due to shorter propagation delay.

5.11 Lock Out Condition :

SPPU : Dec. 08

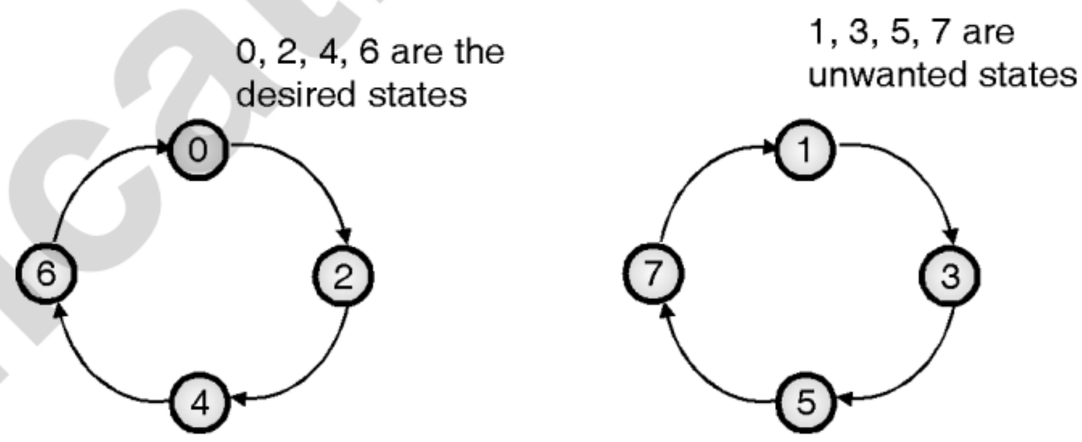
University Questions

Q. 1 What is lock out condition ? **(Dec. 08, 3 Marks)**

– A counter is supposed to follow the sequence of only the desired states as shown in Fig. 5.11.1(a).

– If it enters into an unused or unwanted state, then it is expected to return back to a desired state.

– Instead if the next state of an unwanted state is again an unwanted state as shown in Fig. 5.11.1(b) then the counter is said to have gone into the lockout conditions.



(a) State diagram showing the desired sequence **(b) State diagram showing the lockout condition**

(C-831) **Fig. 5.11.1**

5.11.1 Bushless Circuit :

SPPU : Dec. 08, Dec. 18

University Questions

Q. 1 How lock out condition avoided ? **(Dec. 08, 3 Marks)**

Q. 2 How lockout condition in counter is avoided ? **(Dec. 18, 2 Marks)**

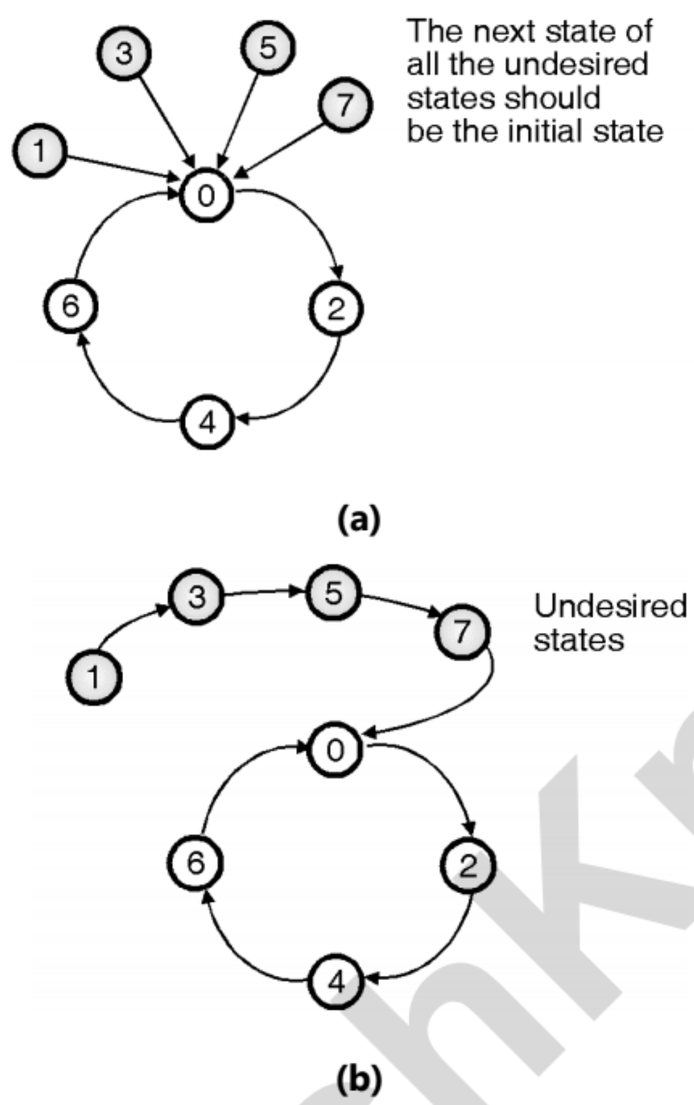
Definition :

– The sequential circuit which enters into the lockout condition is called as the bushless circuit.

Tech Knowledge Publications

How to avoid lockout situation ?

- In order to avoid the lockout condition, we have to use some additional logic circuit to ensure that the next state of the counter is its initial state, if it enters into an undesired state.
- Thus the next state of each unwanted state should be the initial state as shown in Fig. 5.11.2(a).



(C-832) **Fig. 5.11.2 : Ways to avoid lockout condition**

- However it is not necessary to terminate all the undesired states into the initial state as shown in Fig. 5.11.2(a).
- It will be sufficient to terminate only one unused state into the initial state as shown in Fig. 5.11.2(b).
- If the circuit enters into say state "3" then its next states will be "5" and "7". As "7" has next state of "0" which is the desired state, the lockout condition is avoided.

Sequential counter design using JK and T flip flops :

Ex. 5.11.1 : By using suitable flip-flops design a counter to go through the following states.

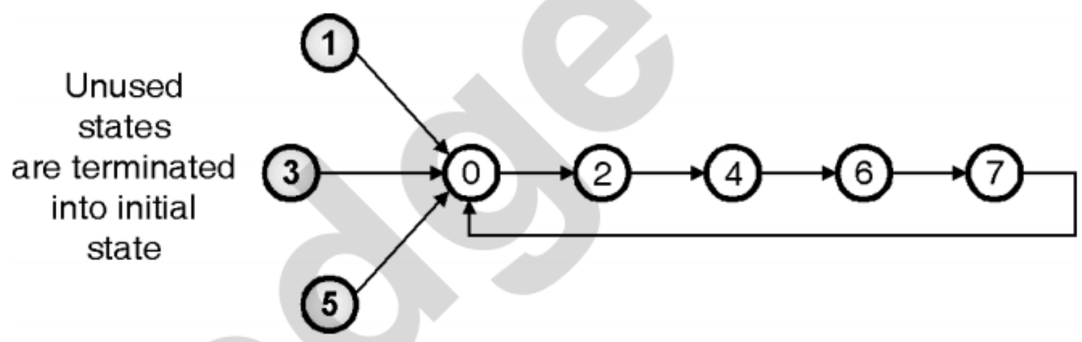
0 – 2 – 4 – 6 – 7 – 0

Avoid the lockout condition.

Soln. :

Step 1 : Draw the state diagram :

- The state diagram is shown in Fig. P. 5.11.1(a). It shows that the next state of all the unused states (1, 3 and 5) is forced to be the initial state (0).
- This is essential in order to avoid the lockout condition.



(C-833) **Fig. P. 5.11.1(a) : State diagram of desired counter**

Step 2 : Number of flip-flops and type of flip-flop :

- Since the highest state is 7 i.e. 111 we need to use three flip-flops. Let us use JK flip-flops.

Step 3 : Write the state table :

- Table P. 5.11.1(a) shows the state table for the required counter.

(C-8053)**Table P. 5.11.1(a)**

Present state			Next state			Flip flop inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	0	0	X	1	X	0	X
0	0	1	0	0	0	0	X	0	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Step 4 : K maps and simplification :

- Fig. P. 5.11.1(b) illustrates the K-maps for all the FF inputs and the corresponding simplified expressions.

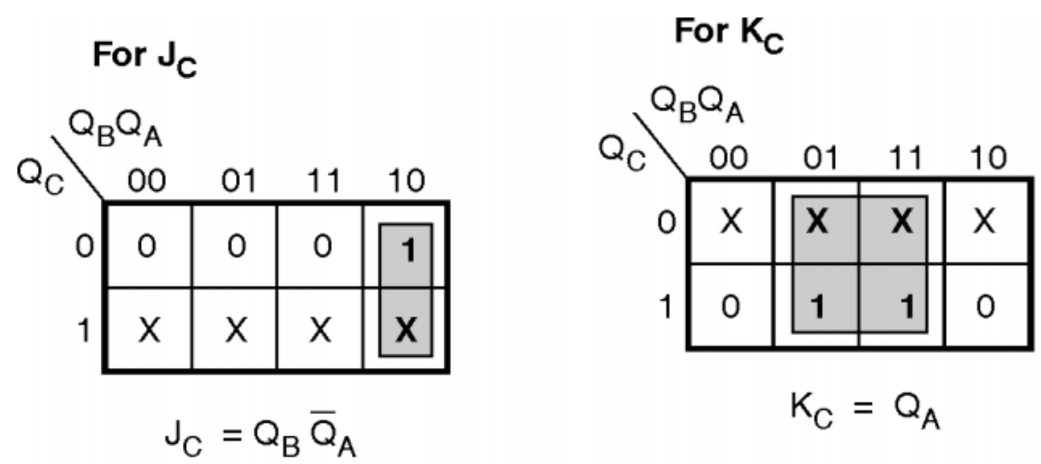
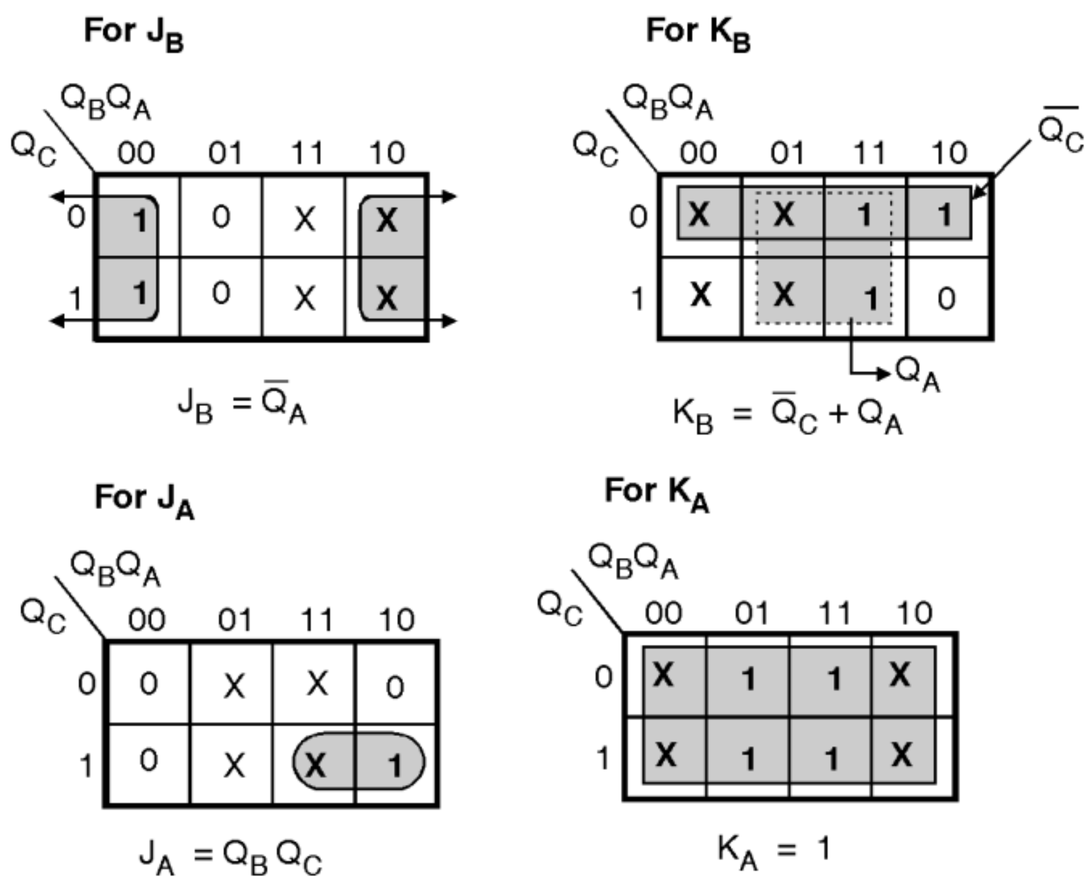


Fig. P. 5.11.1(b) (cont...)

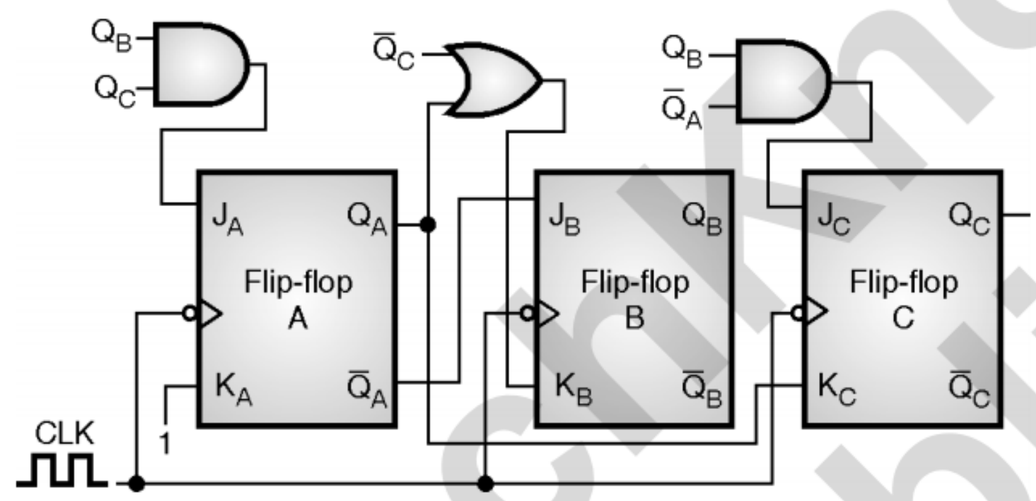
TechKnowledge Publications



(C-834) **Fig. P. 5.11.1(b) : K-maps and simplification**

Step 5 : Draw the logic diagram :

The logic diagram for the required counter is shown in Fig. P. 5.11.1(c).



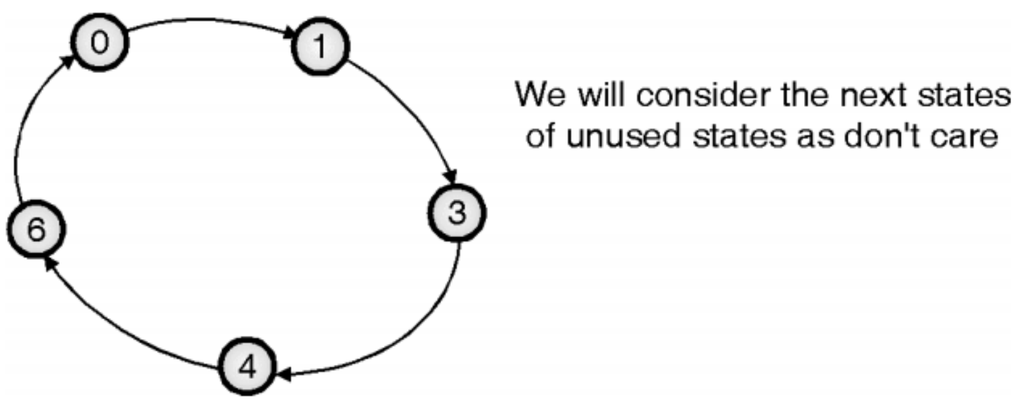
(C-835) **Fig. P. 5.11.1(c) : Logic circuit of the required counter**

Ex. 5.11.2 : By using suitable FFs design a counter to go through the states 0-1-3-4-6-0. Draw the logic diagram. Examine the action of counter for the unused states.

Soln. :

Step 1 : Draw the state diagram :

– The state diagram is as shown in Fig. P. 5.11.2(a). Let us use three T flip-flops.



(C-836) **Fig. P. 5.11.2(a) : State diagram**

Step 2 : Write the state table :

– Assuming the next state of unused states to be don't care conditions the state table is written as shown in Table P. 5.11.2(a).

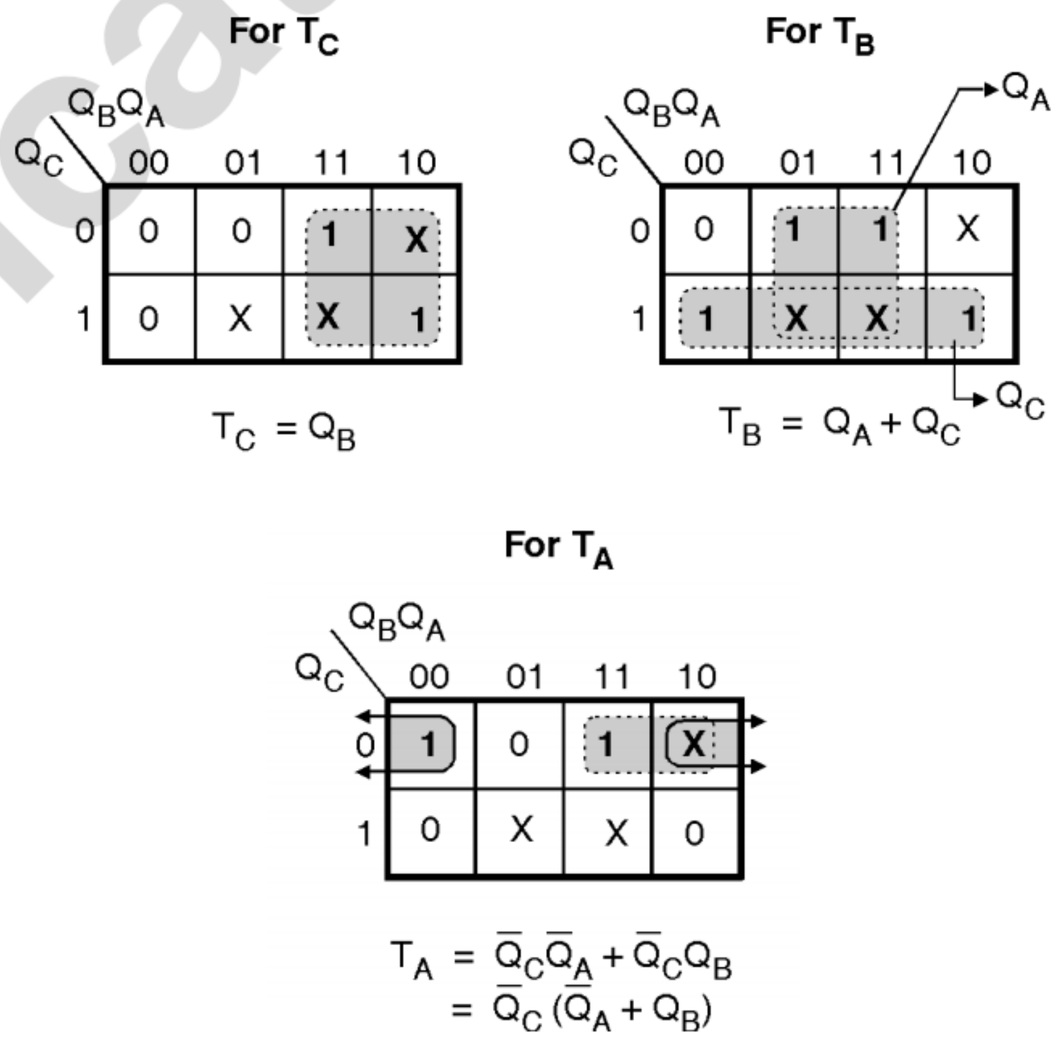
(C-6270) **Table P. 5.11.2(a) : State table**

Present states			Next states			Flip-flop inputs		
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	X	X	X	X	X	X
0	1	1	1	0	0	1	1	1
1	0	0	1	1	0	0	1	0
1	0	1	X	X	X	X	X	X
1	1	0	0	0	0	1	1	0
1	1	1	X	X	X	X	X	X

Next state of the unused state is considered as don't care

Step 3 : K-maps and simplifications :

– The K-maps and the corresponding simplified expressions are as shown in Fig. P. 5.11.2(b).

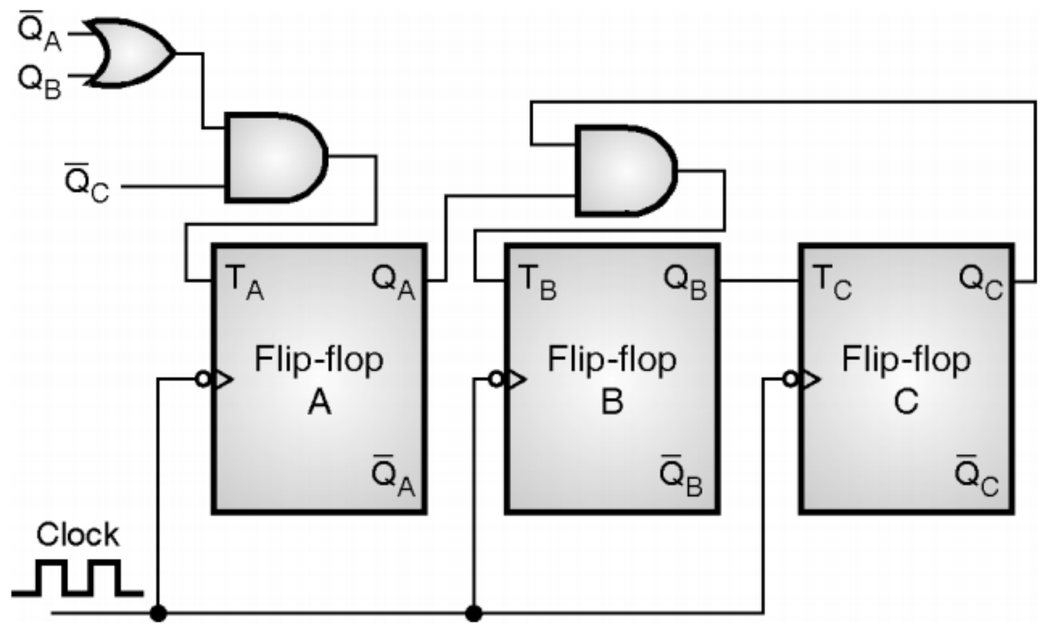


(C-837) **Fig. P. 5.11.2(b) : K-maps and simplifications**

Step 4 : Draw the logic diagram :

– The logic diagram of the required counter is shown in Fig. P. 5.11.2(c).

TechKnowledge Publications



(C-838) **Fig. P. 5.11.2(c) : Logic circuit of the required counter**

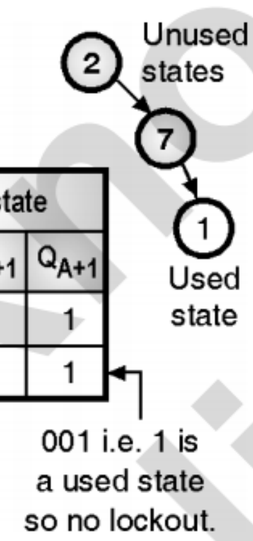
Step 5 : Examine the counter action for the unused states :

For state 010 i.e. 2 :

$Q_C = 0, Q_B = 1$ and $Q_A = 0$.

(C-839) **Table P. 5.11.2(b)**

Present state			FF inputs			Next state		
Q_C	Q_B	Q_A	$T_C = Q_B$	$T_B = Q_A+Q_C$	$T_A = \overline{Q}_C(\overline{Q}_A+Q_B)$	Q_{C+1}	Q_{B+1}	Q_{A+1}
0	1	0	1	0	1	1	1	1
1	1	1	1	1	0	0	0	1

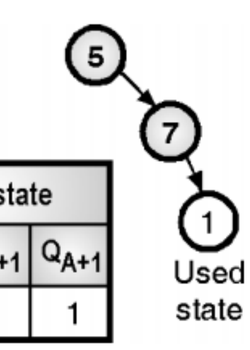


For state 101 i.e. 5 :

– Refer Table P. 5.11.2(c) to know the behaviour of the counter for state 5.

(C-840) **Table P. 5.11.2(c)**

Present state			FF inputs			Next state		
Q_C	Q_B	Q_A	$T_C = Q_B$	$T_B = Q_A+Q_C$	$T_A = \overline{Q}_C(\overline{Q}_A+Q_B)$	Q_{C+1}	Q_{B+1}	Q_{A+1}
1	0	1	0	1	0	1	1	1



– In Table P. 5.11.2(b) we have already proved that the counter returns to state "1" from state "7", which is used state.

For state 111 i.e. 7 :

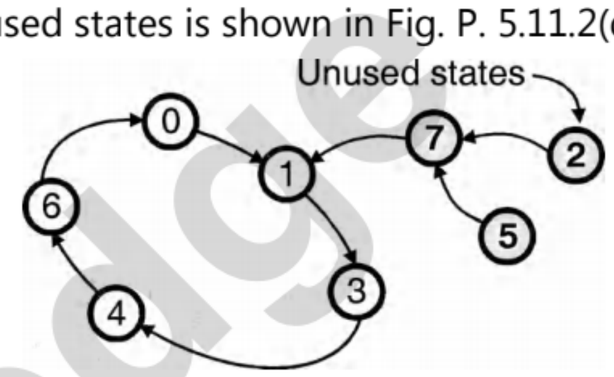
– In Table P. 5.11.2(b) we have already proved that the counter returns to state 1 from "7".

Conclusion :

The counter returns to a used state 1 if by chance it enters into one of the unused states 2, 5 or 7. Hence lockout condition is automatically avoided.

Step 6 : Draw the modified state diagram :

– The modified state diagram showing the used as well as the unused states is shown in Fig. P. 5.11.2(d).



(C-841) **Fig. P. 5.11.2(d) : State diagram showing used and unused states**

5.12 Bush Diagram :

– In the previous section we have discussed the meaning of lock-out condition.

– The lock-out condition can be avoided by taking a precaution called **"bushing".** Bushing means adding branches in the state diagram.

– The invalid states of a counter are branched in the bush diagram in such a way that even if the counter enters into an invalid state, it will return to one of its valid states after one, two or three clock cycles.

– The concept of bushing can be well understood by referring to the following example.

Ex. 5.12.1 : For the given state diagram in Fig. P. 5.12.1(a) of a counter draw the bush diagrams required to bring the counter from invalid to a valid state in one, two or three clock cycles.

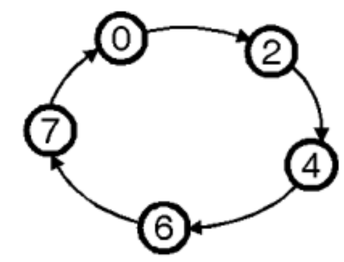


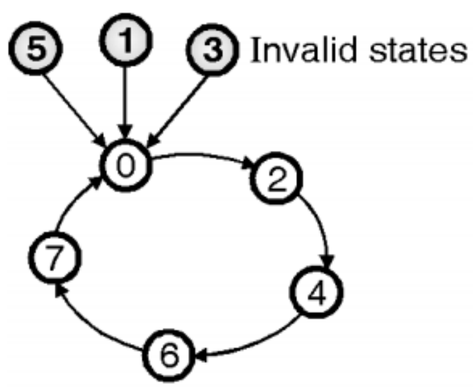
Fig. P. 5.12.1(a) : Given state diagram

Soln. :

– Refer Figs. P. 5.12.1(b) to (d).

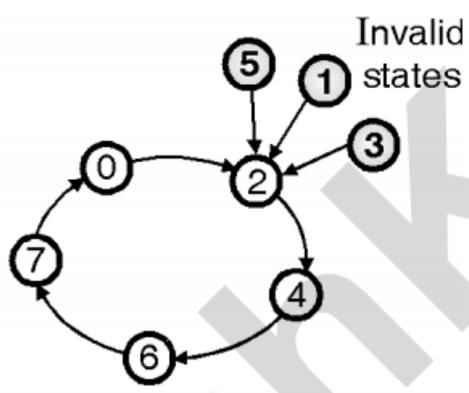
Technical Publications

– Fig. P. 5.12.1(b) shows that the next state of each invalid state 1, 3 or 5 is the initial state 0. Hence even if the counter enters into an unused state, it will return to a valid state after only one clock cycle.



(C-842) **Fig. P. 5.12.1(b) : Entry from invalid to valid state after only one clock pulse**

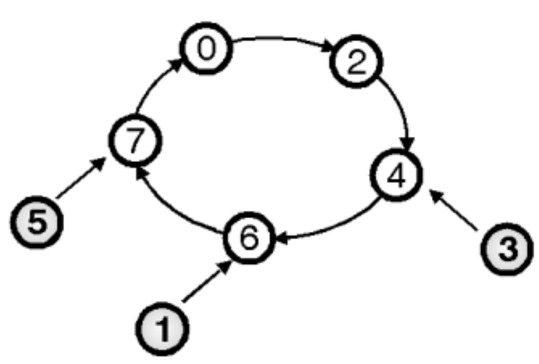
– Similarly for the state diagram shown in Fig. P. 5.12.1(c), it will take only one clock pulse to bring the counter into a used state if it enters into an unused one.



(C-842) **Fig. P. 5.12.1(c) : Entry from invalid to valid state after only one clock pulse**

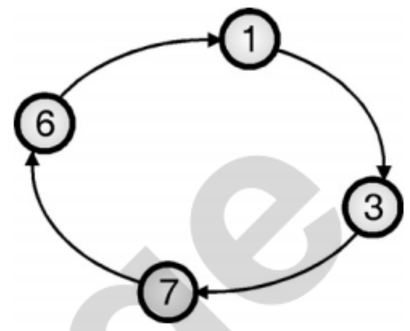
– Refer the state diagram of Fig. P. 5.12.1(d). If the counter enters into state "7" then it will return to "0" after one clock pulse.

– If it is in the state "5" then it will return to 0 state after two clock pulses and if it is in the state "1" then it will return to "0" state after three clock pulses.



(C-842) **Fig. P. 5.12.1(d) : Entry from invalid to valid state after 1, 2 or 3 clock cycles**

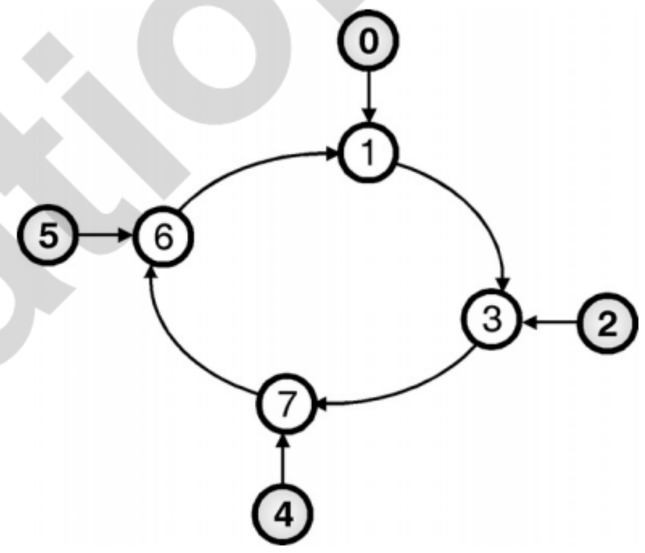
Ex. 5.12.2 : Design a synchronous counter from the state diagram shown in Fig. P. 5.12.2(a). Avoid lockout condition. Draw the bush diagram to avoid the lockout condition.



(C-843) **Fig. P. 5.12.2(a) : Given state diagram**

Soln. :

– So as to avoid the lockout condition, we have to ensure that the counter should return back to a valid state, as soon as it enters into an unused state. The bush diagram is shown in Fig. P. 5.12.2(b).



(C-844) **Fig. P. 5.12.2(b) : Bush diagram**

– 0, 2, 4 and 5 are the unused states. They are forcibly terminated into 1, 3, 7 and 6 respectively.

Step 1 : Number of FFs and the type of FF :

– Let us use T-type FF.

– Since the highest state in the state diagram is 7, we have to use three T flip-flops.

Step 2 : Write the circuit excitation table :

– Let us obtain the circuit excitation table from the bush diagram of Fig. P. 5.12.2(b).

– The excitation table for a T flip-flop is shown in Table P. 5.12.2(a) and the circuit excitation table is shown in Table P. 5.12.2(b).

TechKnowledge Publications

(C-7842) Table P. 5.12.2(a) : Excitation table for a toggle FF

Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

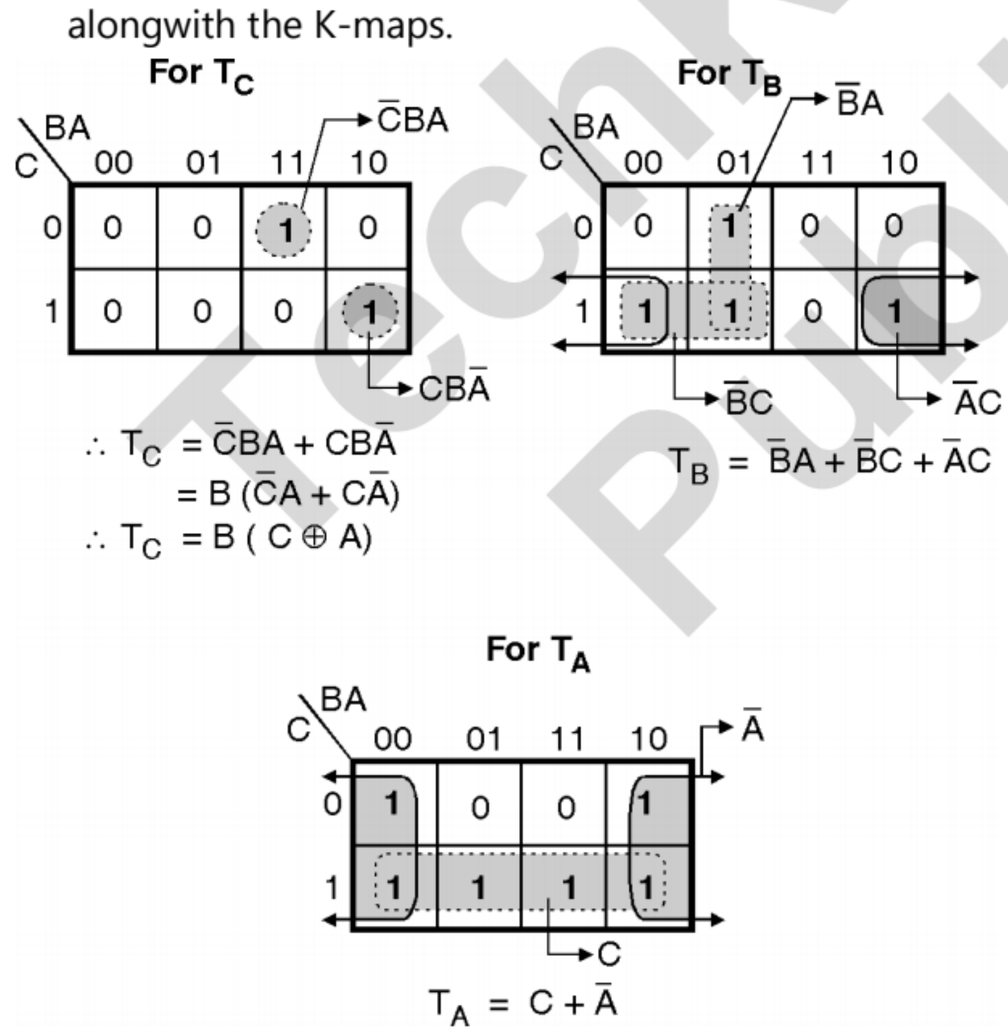
(C-6229) Table P. 5.12.2(b) : Circuit excitation table

Present state			Next state			FF inputs		
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	T_C	T_B	T_A
0	**0**	**0**	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	**1**	**0**	0	1	1	0	0	1
0	1	1	1	1	1	1	0	0
1	**0**	**0**	1	1	1	0	1	1
1	**0**	**1**	1	1	0	0	1	1
1	1	0	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1

Invalid states

Step 3 : Draw the K-maps and obtain simplified expressions :

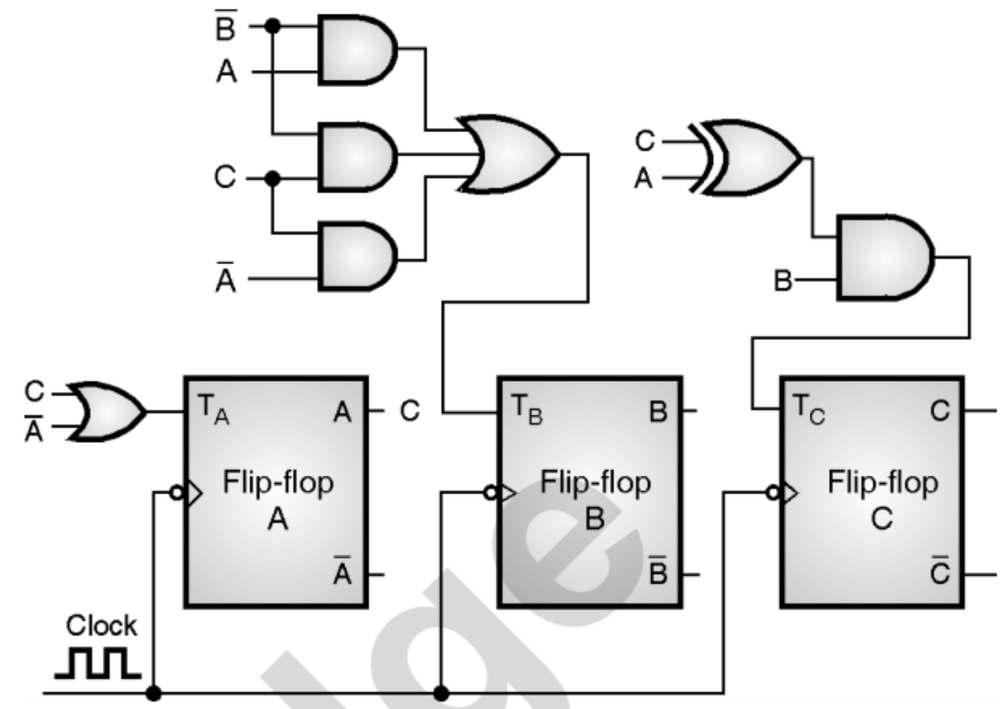
– Fig. P. 5.12.2(c) shows the K-maps for various T inputs. The simplified Boolean expressions also are given alongwith the K-maps.



(C-845) Fig. P. 5.12.2(c) : K-maps and simplifications

Step 4 : Logic diagram :

– The logic diagram is as shown in Fig. P. 5.12.2(d).



(C-846) Fig. P. 5.12.2(d) : Logic diagram of the required counter

5.13 Applications of Counters :

Some of the applications of counters are :

1. In digital clock.
2. In the frequency counters.
3. In time measurement.
4. In digital voltmeters.
5. In the counter type A to D converter.
6. In the digital triangular wave generator.
7. In the frequency divider circuits.

5.14 Ring Counter :

SPPU : Dec. 08, Dec. 18, Dec. 19

University Questions

Q. 1 Write short note on ring counter. **(Dec. 08, 6 Marks)**

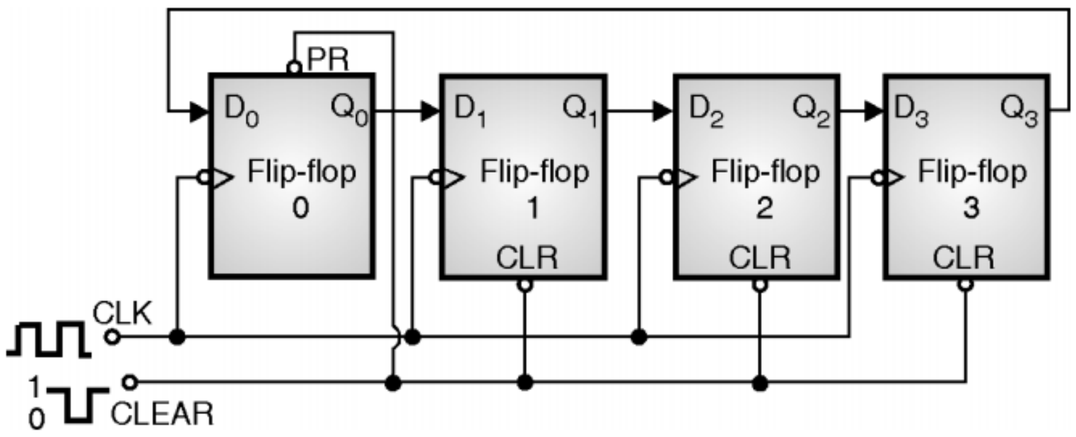
Q. 2 Draw and explain Ring counter using JK flip-flop (Timing diagram is expected.) **(Dec. 18, 4 Marks)**

Q. 3 Draw and explain Ring counter using D flip-flop. Also draw the necessary timing diagram. **(Dec. 19, 6 Marks)**

Logic diagram :

– Fig. 5.14.1 shows a typical application of shift registers called Ring Counter.

– The connections reveal that they are similar to the connections for shift right operation, except for one change.



(C-871) Fig. 5.14.1 : A four bit ring counter

– Output of FF-3 is connected to data input D_0 of FF-0. Ring counter is a special type of shift register.

Operation :

– Initially a low clear (CLR) pulse is applied to all the flip-flops.

– Hence FF-3, FF-2 and FF-1 will reset but FF-0 will be preset. So the outputs of the shift register are :

$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 0\ 1.$

– Now the clear terminal is made inactive by applying a high level to it.

– The clock signal is then applied to all the flip-flops simultaneously. Note that all the flip-flops are negative edge triggered.

On the first negative going CLK edge :

– As soon as the first falling edge of the clock hits, only FF-1 will be set because $Q_0 = D_1 = 1$.

– The FF-0 will reset because $D_0 = Q_3 = 0$ and there is no change in the status of FF-2 and FF-3.

– Hence after the first clock pulse the outputs are

$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 1\ 0.$

On the second falling edge of clock :

– At the second falling edge of the clock, only FF-2 will be set because $D_2 = Q_1 = 1$.

– FF-1 will reset since $D_1 = Q_0 = 0$. There is no change in status of FF-3 and FF-0.

– So after the second clock pulse the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 1\ 0\ 0.$

– Similarly after the third clock pulse the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0 = 1\ 0\ 0\ 0.$

– And after the fourth one the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0 = 0\ 0\ 0\ 1.$

– These are the outputs from where we started. Hence the operation repeats from this point onwards.

Number of output states :

– The number of output states for a ring counter will always be equal to the number of flip-flops. So for a 4-bit ring counter the number of states is equal to 4.

– The operation of a four bit ring counter is summarized in Table 5.14.1.

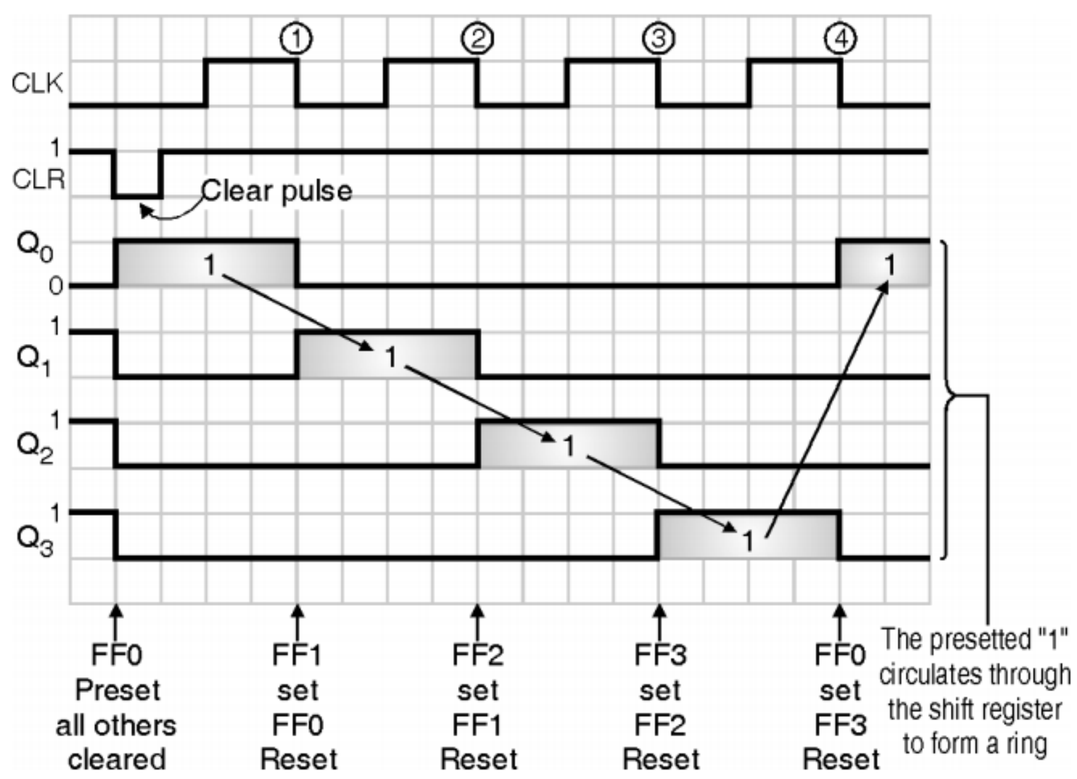
(C-871(a)) Table 5.14.1 : Summary of operation of a ring counter

CLR	CLK	Q_0	Q_1	Q_2	Q_3	
⊓	X	1	0	0	0	← FF-0 Preset, others cleared
1	↓	0	1	0	0	The presetted 1 follows a circular path to form a ring
1	↓	0	0	1	0	
1	↓	0	0	0	1	
1	↓	1	0	0	0	

Waveforms for the ring counter :

– The waveforms for the 4-bit ring counter are as shown in Fig. 5.14.2.

– These waveforms clearly show that the presetted "1" shifts one bit per clock cycle and forms a ring. Hence the name ring counter.



(C-1360) Fig. 5.14.2 : Waveforms of a four bit ring counter

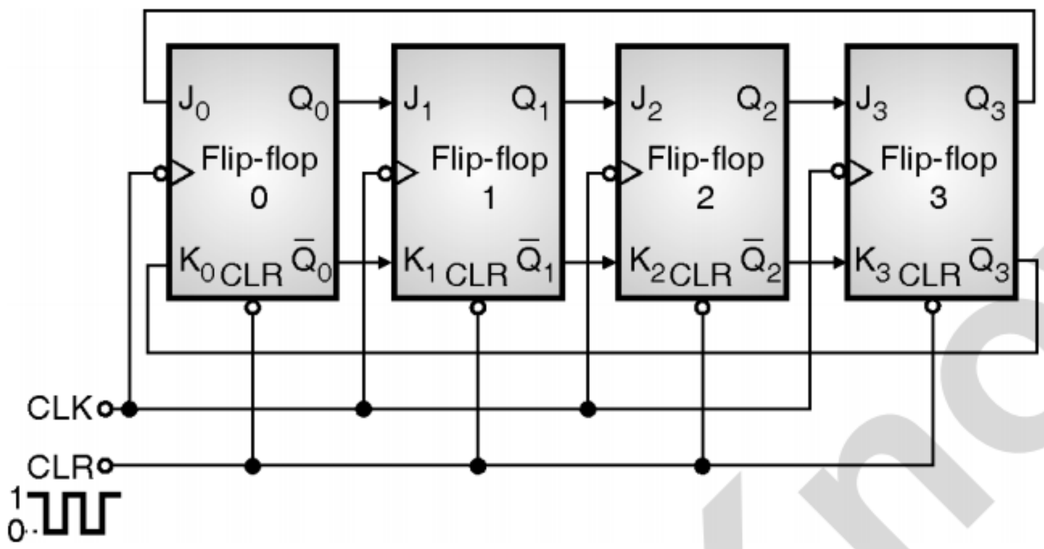
Tech Knowledge Publications

Applications of ring counter :

- Ring counters are used in those applications in which several operations are to be controlled in a sequential manner.
- For example in resistance welding the operations called squeeze, hold, weld and off are to be performed sequentially.
- We can use a ring counter to initiate these operations.

Ring counter using JK Flip Flop :

- A ring counter can also be constructed using the JK flip flops, as shown in Fig. 5.14.3.



(C-1361) Fig. 5.14.3 : Ring counter using JK flip flops

5.15 Johnson's Counter (Twisted / Switch Tail Ring Counter) :

SPPU : Dec. 08, May 10, May 11, Dec. 17

University Questions

Q. 1 Write short notes on Johnson counter.

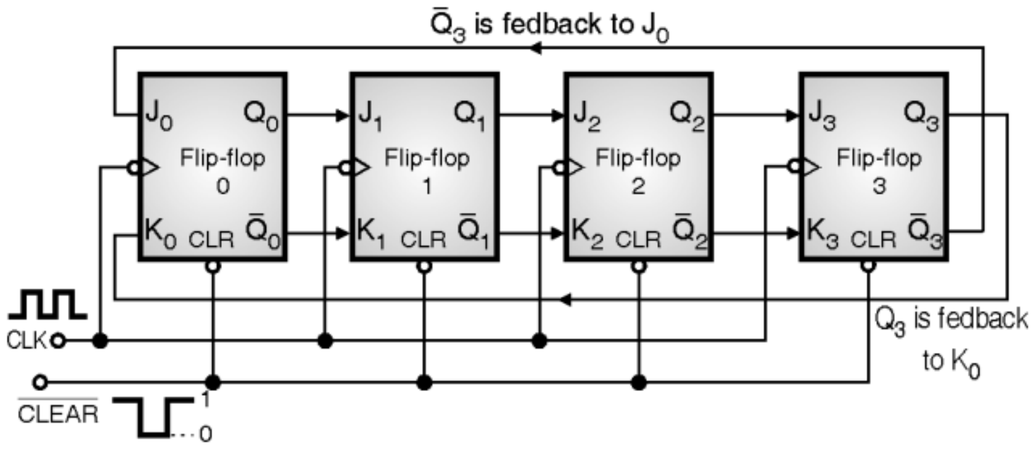
(Dec. 08, May 10, 6 Marks, Dec. 17, 4 Marks)

Q. 2 Explain twisted ring counter in brief.

(May 11, 4 Marks)

Logic diagram :

- In the ring counter the outputs of FF-3 were connected directly to the inputs of FF-0 i.e. Q_3 to J_0 $\overline{Q}_3$ to K_0.
- Instead if the outputs are cross coupled to the inputs i.e. if Q_3 is connected to K_0 and $\overline{Q}_3$ is connected to J_0 then the circuit is called as twisted ring counter or Johnson's counter.
- The Johnson's counter is shown in Fig. 5.15.1.



(C-1362) Fig. 5.15.1 : Twisted ring counter or Johnson counter

- All the flip-flops are negative edge triggered, and clock pulses are applied to all of them simultaneously.
- The clear inputs of all the flip-flops are connected together and connected to an external clear signal. Note that all these clear inputs are active low inputs.

Operation :

- Initially a short negative going pulse is applied to the $\overline{\text{clear}}$ input of all the flip-flops. This will reset all the flip-flops. Hence initially the outputs are, $Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 0.
- But $\overline{Q}_3$ = 1 and since it is coupled to J_0 it is also equal to 1.

  $\therefore$ J_0 = 1 and K_0 = 0 .... Initially

On the first falling edge of clock pulse :

- As soon as the first negative edge of clock arrives, FF-0 will be set. Hence Q_0 will become 1.
- But there is no change in the status of any other flip-flop.
- Hence after the first negative going edge of the clock the flip-flop outputs are,

  $Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 1

On the second negative going clock edge :

- Before the second negative going clock edge, Q_3 = 0 and $\overline{Q}_3$ = 1. Hence J_0 = 1 and K_0 = 1. Also Q_0 = 1. Hence J_1 = 1.
- Hence as soon as the second falling clock edge arrives, FF-0 continues to be in the set mode and FF-1 will now set. Hence Q_1 will become 1 and = 0.
- There is no change in the status of any other flip-flop.

Tech Knowledge Publications

– Hence after the second clock edge the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 1 1.

– Similarly after the third clock pulse, the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0$ = 0 1 1 1.

– And after the fourth clock pulse, the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0$ = 1 1 1 1.

Note that now $\overline{Q}_3$ = 0 i.e. J_0 = 0 and K_0 = 1.

– Hence as soon as the fifth negative going clock pulse strikes, FF-0 will reset.

– But the outputs of the other flip-flops will remain unchanged. So after the fifth clock pulse, the outputs are,

$Q_3\ Q_2\ Q_1\ Q_0$ = 1 1 1 0 ..... after the 5th clock pulse

– This operation will continue till we reach the all zero output state. (i.e. $Q_3\ Q_2\ Q_1\ Q_0$ = 0 0 0 0).

– The operation of Johnson's counter is summarised in Table 5.15.1.

(C-6295) Table 5.15.1 : Summary of operation of Johnson's counter

$\overline{CLEAR}$	CLK	Q_3	Q_2	Q_1	Q_0	State number	Decimal equivalent
⊔	Initially	0	0	0	0	1	0
1	↓	0	0	0	1	2	1
1	↓	0	0	1	1	3	3
1	↓	0	1	1	1	4	7
1	↓	1	1	1	1	5	15
1	↓	1	1	1	0	6	14
1	↓	1	1	0	0	7	12
1	↓	1	0	0	0	8	8
1	↓	0	0	0	0	1	0

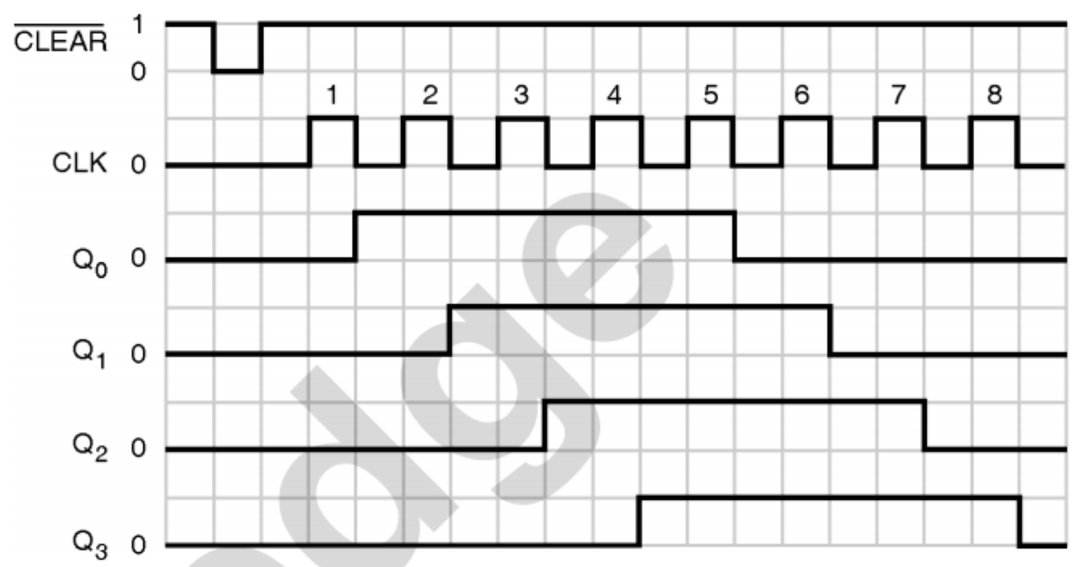
– Note that there are 8 distinct states of output.

– In general we can say that the number of states of a Johnson's counter is twice the number of flip-flops used.

– Therefore for a 4-flip-flop Johnson's counter, there are 8-distinct output states.

Waveforms for Johnson's counter :

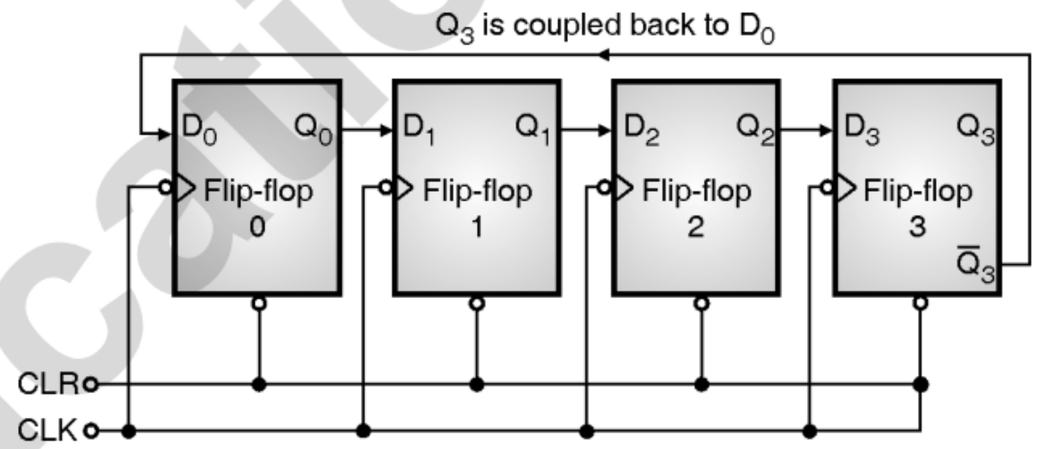
– The waveforms for a 4-bit Johnson's counter are shown in Fig. 5.15.2.



(C-875) Fig. 5.15.2 : Waveforms of Johnson counter

5.15.1 Johnson's Counter using D Flip-flops :

– It is possible to construct the Johnson counter using D flip-flops as shown in Fig. 5.15.3.



(C-1364(b)) Fig. 5.15.3 : Johnson counter using D flip-flops

– The operation of this circuit is summarized in Table 5.15.2.

(C-6296) Table 5.15.2 : Summary of operation of Johnson counter

CLR	CLK	Q_0	Q_1	Q_2	Q_3
⊔		1	0	0	0
1	↓	1	1	0	0
1	↓	1	1	1	0
1	↓	1	1	1	1
1	↓	0	1	1	1
1	↓	0	0	1	1
1	↓	0	0	0	1
1	↓	0	0	0	0
1	↓	1	0	0	0

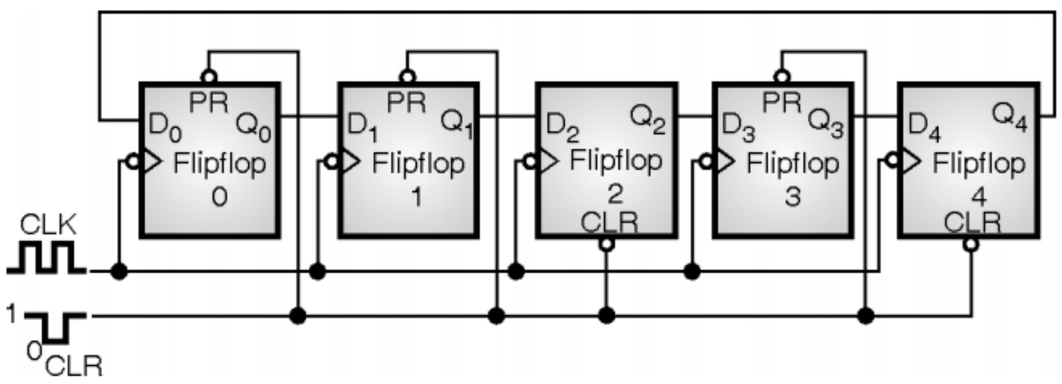
TechKnowledge Publications

Ex. 5.15.1 : Explain ring counter design having initial state 01011 from initial state explain all possible states in the ring.

Dec. 10, 10 Marks

Soln. :

– Since there are 5 bits in the given initial state, we have to use 5 flip flops as shown in Fig. P. 5.15.1.



(C-3562) Fig. P. 5.15.1 : A 5 bit ring counter

– When we apply a low going clear (CLR) pulse, then flip flops 0, 1, B and 3 will be preset to 1 output while flip flops 2 and 4 are reset to 0 output.

$\therefore \quad Q_4\, Q_3\, Q_2\, Q_1\, Q_0 = 01011$ ...Initially

The remaining states are as shown in Table P. 5.15.1.

(C-3563) Table P. 5.15.1 : Ring counter states

CLR	CLK	Q_0	Q_1	Q_2	Q_3	Q_4
⊓	X	1	1	0	1	0
1	↓	0	1	1	0	1
1	↓	1	0	1	1	0
1	↓	0	1	0	1	1
1	↓	1	0	1	0	1
1	↓	1	1	0	1	0

FFs 0, 1 and 3 are preset

Ex. 5.15.2 : Explain the ring counter design for the initial condition 10110.

May 11, 4 Marks

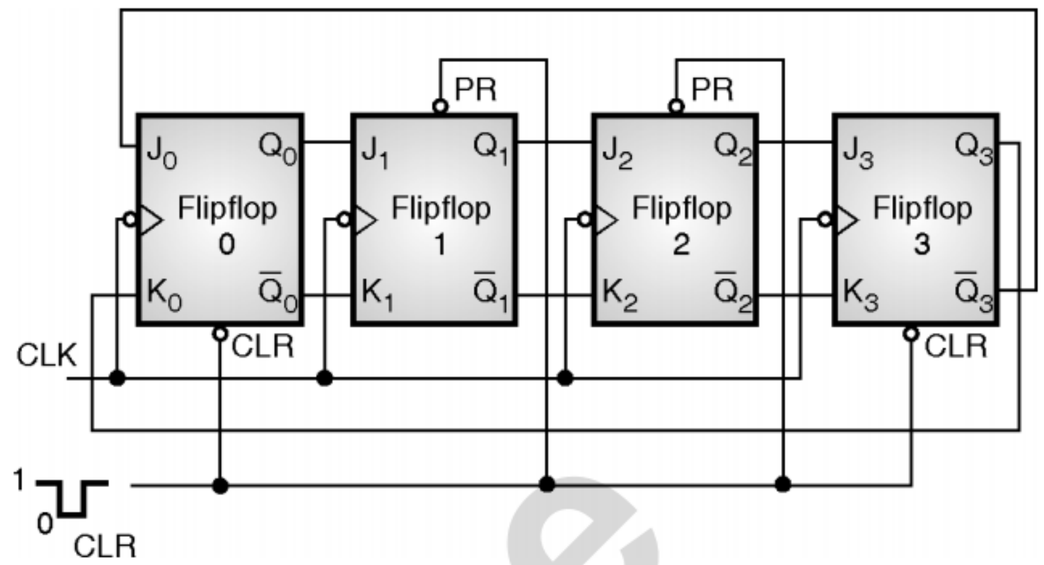
Soln. : Similar to the previous example.

Ex. 5.15.3 : Explain the Johnson's counter design for initial state 0110. From initial state explain and draw all possible states.

Dec. 09, 8 Marks

Soln. :

– The required Johnson's counter is shown in Fig. P. 5.15.3.



(C-3564) Fig. P. 5.15.3 : Required Johnson's counter

– Counters 0 and 3 are reset to 0 while counters 1 and 2 and preset to 1 initially.

$\therefore \quad Q_3\, Q_2\, Q_1\, Q_0 = 0\,1\,1\,0$ ...initially

– The other possible states of this Johnson's counter are listed in Table P. 5.15.3.

(C-3565) Table P. 5.15.3 : All possible states of a Johnson's counter

Clear	CLK	Q_3	Q_2	Q_1	Q_0
⊓	X	0	1	1	0
1	↓	1	1	0	1
1	↓	1	0	1	0
1	↓	0	1	0	0
1	↓	1	0	0	1

Review Questions

Q. 1 What is counter ? What are its types ?

Q. 2 Explain the terms "synchronous" and "asynchronous".

Q. 3 What is the difference between synchronous and asynchronous counter ?

Q. 4 What are the merits and demerits of the synchronous counter over the asynchronous counter ?

Q. 5 State the procedure for designing mod counters from N-bit ripple counter.

Q. 6 What is the similarity between a decade counter and a binary counter ? Explain in brief.

Q. 7 What are the applications of counter ?

Q. 8 How is a counter different from a register ?

Q. 9 Write the count sequence of three bit binary down counter.

Tech Knowledge Publications

Q. 10 How does a counter works as frequency divider ?

Q. 11 What is synchronous counter ?

Q. 12 Define modulus of counter. State the states of MOD 5 counter.

Q. 13 Explain the working of 4-bit ripple counter (asynchronous counter) using T flip-flop with suitable circuit diagram and timing diagram.

Q. 14 Write a note on Mod-8 synchronous up-counter using T flip-flop.

Q. 15 Write a note on Mod-8 synchronous down counter using T flip-flop.

Q. 16 Draw the divide by 7 asynchronous up counter using T flip flop. Write truth table. Draw timing diagram.

Q. 17 Draw the circuit for mod-12 counter. Explain the same with neat waveforms.

Q. 18 Compare synchronous and ripple counters.

Q. 19 Compare counters and registers.

Q. 20 Design a 3 bit synchronous counter using JK fillip flops.

Q. 21 What is lock out ? How is it avoided ?

Q. 22 Explain decade counter.

❑❑❑

TechKnowledge Publications

Unit III

Chapter 6

Synchronous Sequential Circuit Design

Technical Knowledge Publications

Syllabus

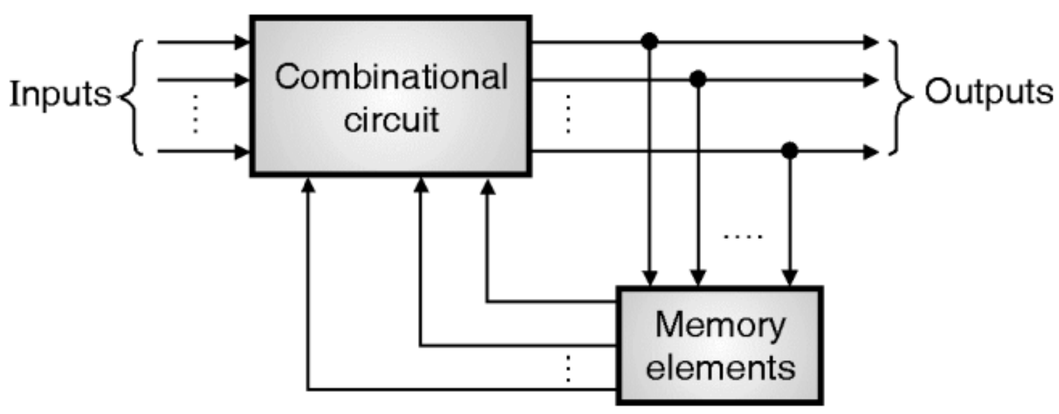
Models – Moore and Mealy, State diagram and state table, Design procedure, Sequence generator and detector.
Exemplar / Case study : Electronic Voting Machine (EVM).

Chapter Contents

6.1	General Model of Sequential Circuits	6.7	State Reduction and Assignments
6.2	Models for Synchronous Sequential Circuits	6.8	State Assignment
6.3	Moore Circuit	6.9	Design of Clocked Synchronous State Machine using State Diagram
6.4	Mealy Circuit	6.10	Design using Unused States
6.5	Finite State Machine (FSM)	6.11	Sequence Generator
6.6	Analysis of Clocked Sequential Circuits	6.12	Sequence Detector

6.1 General Model of Sequential Circuits :

- The general model of a sequential circuit is shown in Fig. 6.1.1.
- It shows that a sequential circuit is made of a combinational circuit alongwith the memory elements.



(C-652) **Fig. 6.1.1 : General model of a sequential circuit**

- The memory elements will be always connected between output and input of the combinational circuit. Therefore it is said to be appearing in the feedback path.
- The output of a sequential circuit depends on the present state as well as the past state.
- The present state is stored in the memory elements. These present states are applied to the combinational circuit along with the external inputs.
- Thus the outputs of the combinational block i.e. the outputs and the next state of the sequential circuit are decided by the external inputs and the present state.

6.1.1 Classification of Sequential Circuits :

- The sequential circuits are classified into two categories as follows :
  1. Synchronous sequential circuits.
  2. Asynchronous sequential circuits.

Synchronous sequential circuits :

- In the synchronous sequential circuits, the contents of memory elements can be changed only at the rising or falling edges of clock signal.

Asynchronous sequential circuits :

- In the asynchronous sequential circuit, the contents of memory elements can be changed at any instant of time.

Table 6.1.1 : Comparison of synchronous and asynchronous sequential circuits

Sr. No.	Synchronous sequential circuits	Asynchronous sequential circuits
1.	These circuits are easy to design.	Difficult to design.
2.	A clocked FF acts as a memory element.	An unclocked FF or time delay element is used as memory element.
3.	Slower, because the delays correspond to those of the memory element.	Faster as the clock is not present.
4.	The status of memory element is affected only at the active edge of clock if the input is changed.	The status of memory element will change any time as soon as the input is changed.

6.2 Models for Synchronous Sequential Circuits :

- There are two models developed for representing the synchronous sequential circuits namely :
  1. Moore circuit 2. Mealy circuit
- In the synchronous sequential circuits, the combinational circuit block consists of a combination of logic gates where as the flip flops are used as memory elements.
- A synchronous sequential circuit is called as a **Moore circuit** if its output depends only on the present state of flip flops. The output does not depend on the external inputs.
- A synchronous sequential circuit is called as a **Mealy circuit** if its output is dependent on the present state of flip flops as well as the external inputs.

6.3 Moore Circuit :

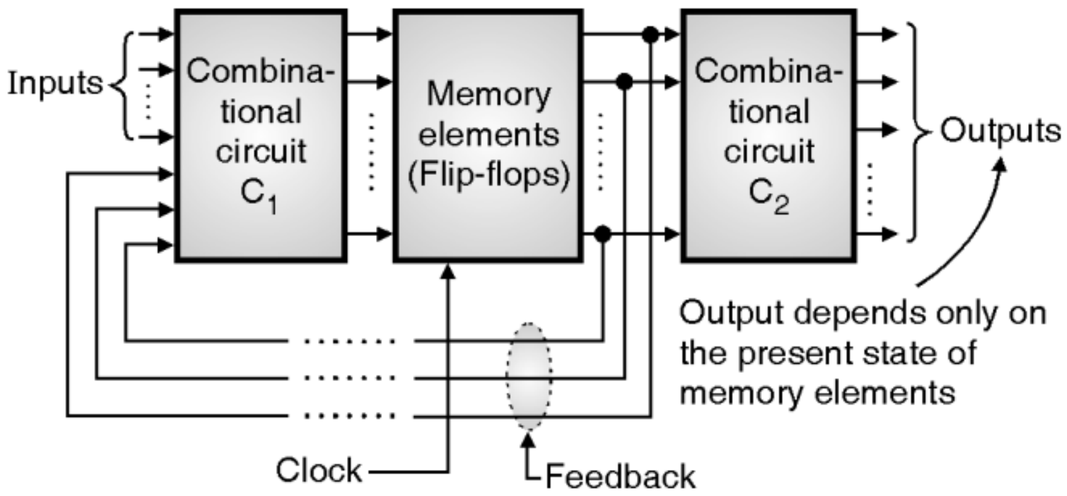
Definition :

- A Moore machine is a finite-state machine whose output values are determined only by its current state.

TechKnowledge Publications

Block diagram :

– The general block diagram of a Moore synchronous sequential circuit is as shown in Fig. 6.3.1.



(C-670) Fig. 6.3.1 : A general form of Moore circuit

Explanation :

– From this block diagram it is illustrated that for a Moore circuit the output depends only on the present state of the memory elements i.e. flip flops.

– The external inputs do not influence the outputs.

– The combinational circuit C_1 on the input side is sometimes called as the next state decoder.

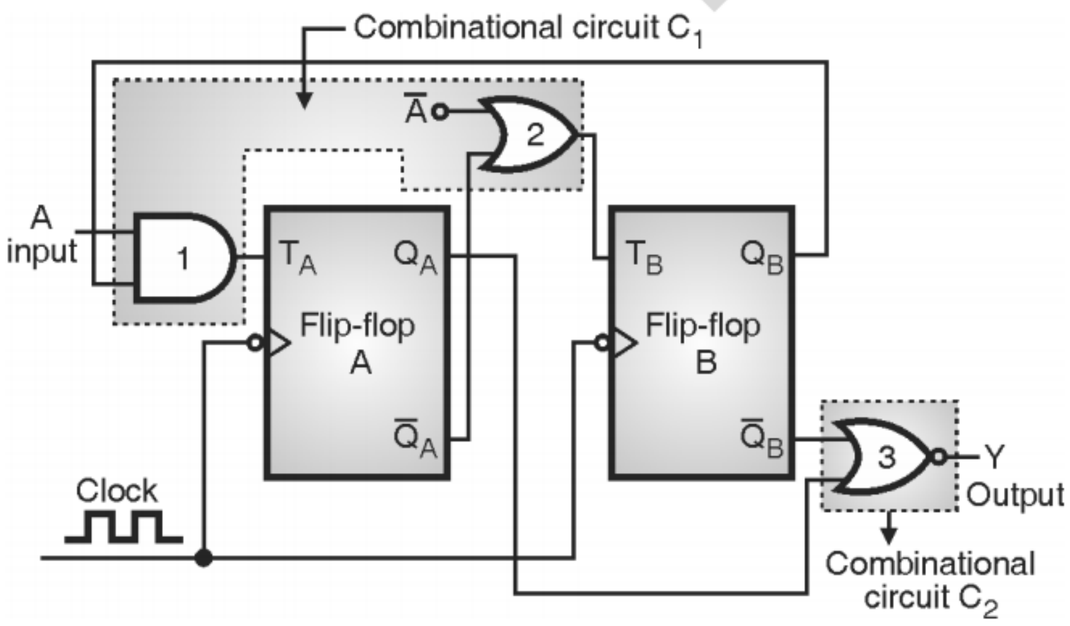
– The external inputs and the feedback signals from the memory elements are applied to this combinational circuit.

– The outputs of the memory elements (present state) are applied to another combinational circuit C_2 which is also called as output decoder. The outputs are produced at the output of this block.

– Note that the clock signal is applied only to the memory element block which consists of the flip flops.

6.3.1 Example of Moore Circuit :

– The example of a Moore circuit is shown in Fig. 6.3.2.



(C-671) Fig. 6.3.2 : Example of Moore circuit

– Flip flops A and B act as the memory elements for the circuit.

– Gates 1 and 2 form the input combinational circuit C_1, because the external inputs A and $\bar{A}$ are applied to one of their inputs, where as the other inputs are coming from the Q_B and $\bar{Q}_A$ outputs of the memory elements.

– Gate-3 acts as the output decoder or combinational circuit C_2. The inputs to this gate are from the outputs of the two memory elements FF-A and FF-B. Note that the external inputs are not connected to the output decoder block.

$$\therefore Y = \overline{Q_A + \bar{Q}_B}$$

– Thus the output Y depends only on the present state of flip flops.

– Both the flip flops are negative edge triggered and the clock input is applied to both the flip flops simultaneously.

– Therefore the output Y will change only after the application of clock pulse.

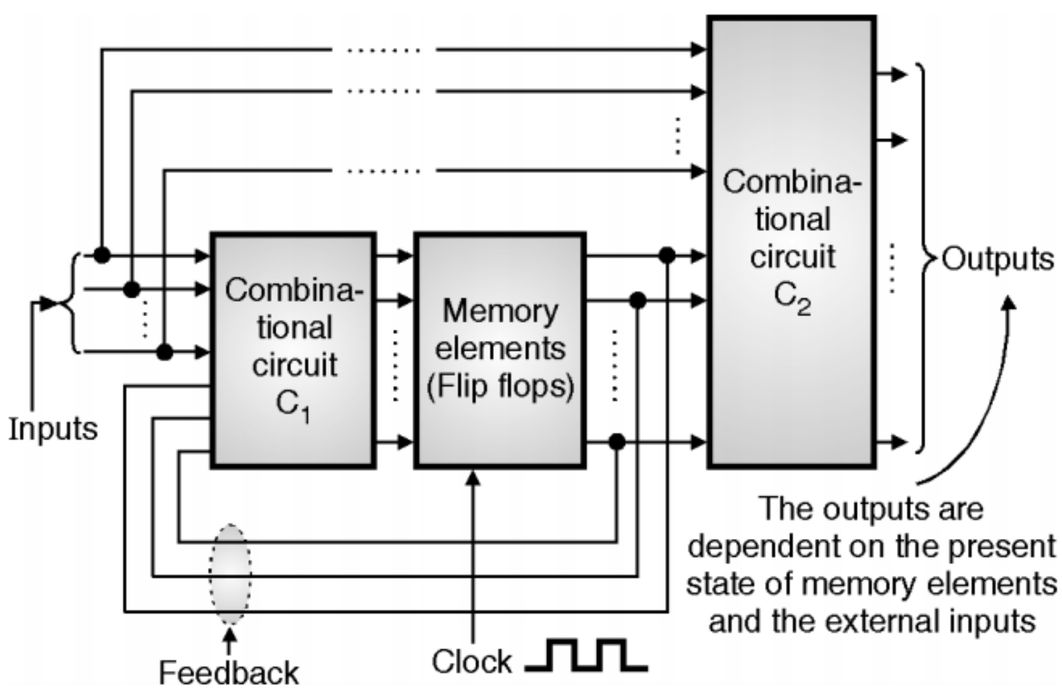
6.4 Mealy Circuit :

Definition :

– A Mealy machine, is a finite-state machine whose output values are determined both by its current state and by the values of its inputs.

Block diagram :

– The generalised block diagram of a Mealy circuit is shown in Fig. 6.4.1.



(C-673) Fig. 6.4.1 : A general form of Mealy circuit

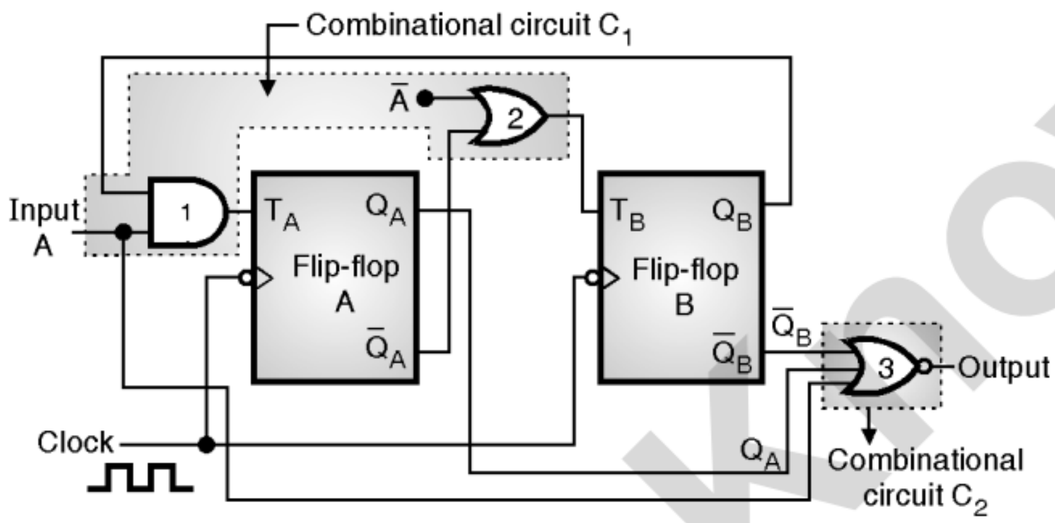
TechKnowledge Publications

Explanation :

- The input combinational circuit (C_1) which is also called as the next state decoder receives inputs from the outputs of memory elements (present state) and from the external inputs.
- But the most important point to be noted is that the output combinational circuit C_2 (also called as output decoder) receives signal from the output of memory elements as well as the external inputs.
- This shows that the outputs of Mealy circuit is dependent on the present state of memory elements as well as the external inputs.

6.4.1 Example of Mealy Circuit :

- The logic circuit shown in Fig. 6.4.2 is an example of Mealy circuit.



(C-674) **Fig. 6.4.2 : Example of Mealy circuit**

- As usual, the flip flops A and B form the memory element. Both the FFs are negative edge triggered and the clock signal has been applied to both the flip flops simultaneously.
- Gates 1 and 2 form the input combinational circuit C_1 and gate-3 forms the output combinational circuit i.e. C_2.
- Note that the inputs to gate-3 are Q_A, $\bar{Q}_B$ and A. That means the output is now dependent on the outputs of memory elements (present state) as well as the external input A.

$$Y = Q_A + \bar{Q}_B + A$$

- However any changes in "A" in between the successive negative clock edges will not affect the outputs of the flip flops.
- But it will definitely affect the final output, because it will affect the combinational circuit C_2.
- Thus a false output may be produced.
- To eliminate this problem, the external inputs should be allowed to change their state only in synchronization with the active edge of clock and not in between the active clock edges of the successive clock pulses.

6.4.2 Comparison of Moore and Mealy Models : SPPU : May 08, Dec. 16

University Questions

Q. 1 Explain the difference between Moore and Mealy concepts. **(May 08, 4 Marks)**

Q. 2 Compare Moore and Mealy model. **(Dec. 16, 2 Marks)**

Table 6.4.1 : Comparison of Moore and Mealy models

Sr. No.	Moore model	Mealy model
1.	The final output depends only on the present state of memory elements.	The final output depends on the present state of memory elements and the external inputs.
2.	The output changes only after the active clock edge.	Output can change in between the clock edges if the external inputs change.
3.	The implementation of a logic function needs more number of states than Mealy circuit.	Implementation of the same logic function requires less number of states than Moore circuit.

6.5 Finite State Machine (FSM) :

- When describing the two models, some books or technical sources refer to a sequential circuit as finite state machine of FSM.
- Therefore the two models are referred to as Moore FSM and Mealy FSM.
- The word finite is used for indicating that there are limited number states, which can be attained by the Finite State Machine (FSM).

6.5.1 Advantages of FSM :

1. FSMs are simple to implement.
2. After development and realization, the testing of FSM is also easy.
3. The time required to design, implement and test a FSM is short.
4. FSM is a well proven modeling technique for the synchronous sequential circuits.
5. FSMs are flexible. That means we can implement them in many different ways.

Tech Knowledge Publications

6.5.2 Disadvantages of FSM :

1. FSM technique is not suitable for solving all types of problems. It can be used only if we can decompose the system behavior into distinct states. The transitions among these states also should be well defined and known to the designer.
2. FSMs are predictable which makes them useless in some application areas such as computer gaming.
3. FSMs become too complicated for larger systems and become difficult to manage.

6.5.3 State Machine Notations :

– Various important names used in the state machine analysis are as follows :

1. Input variable :

– All the variable which get generated outside the sequential circuit are called as the input variables.

2. Output variable :

– All the variable that leave the sequential circuit are called as the output variables.

3. State variable :

– As the output of flipflops (memory) defines the state of a sequential machine, we can define the state variables as the outputs of flipflops.

∴ State variables = Outputs of flip flops

4. Excitation variable :

– Excitation variables are defined as the inputs to the memory (flipflops).

– This name is given because the variable excites the memory to change its state.

– State variables are produced by the excitation variables.

∴ Excitation variables = Inputs to flip flops

5. State :

– The state of a sequential machine is dependent on the content of memory.

– If the memory is realized using flipflops, then the machine state is defined by the Q output of flipflops.

∴ State = Q output of flip flops

– The relation between state variables and number of states is :

$$y = 2^x$$

where y = Maximum number of possible states

And x = Number of state variables

6.6 Analysis of Clocked Sequential Circuits :

– The sequential circuit to be analysed is given to us.

– The meaning of analysis is that we have to identify the type of circuit, write its state table and draw the state diagram.

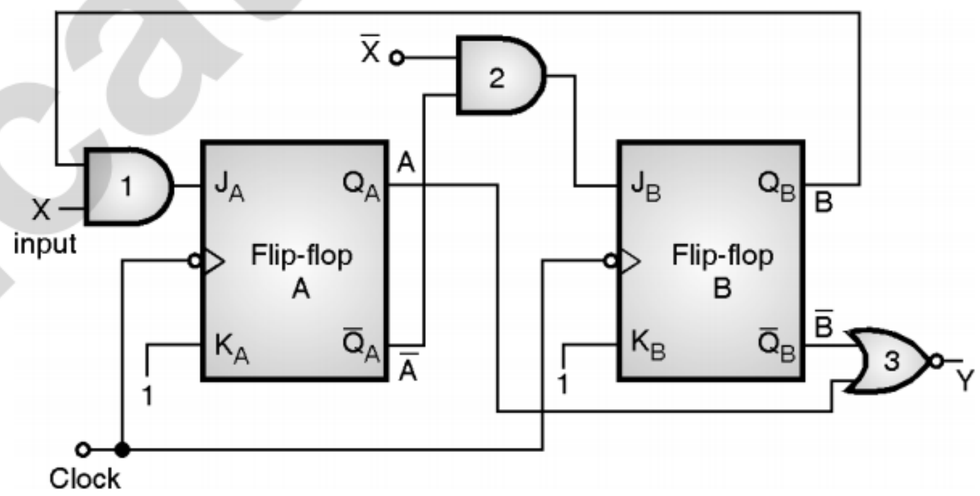
– This procedure will be clear after solving the following examples.

Analysis procedure :

1. Identify the type of synchronous sequential circuit (Moore or Mealy).
2. Derive the state table.
3. Draw the state diagram.

– This procedure will be clear after solving the following example.

Ex. 6.6.1 : Identify the type of circuit given in Fig. P. 6.6.1(a), write the state table and draw the state diagram for the same.



(C-675) **Fig. P. 6.6.1(a) : Given circuit**

Soln. :

Step 1 : Type of circuit :

– Since output $Y = \overline{A + \bar{B}}$, it is dependent only on the present state of the memory elements. Hence this is a Moore circuit.

Step 2 : Write the state table :

– The state table is as shown in Table P. 6.6.1(a).

(C-8101) **Table P. 6.6.1(a) : State table**

Present state	Next state		Output Y
A B	X = 0	X = 1	

Description :

- The present state column in the state table represents the FF outputs before application of clock pulse.
- The next state column represents the outputs of FF outputs after the application of clock pulse.
- It is dependent on the present state as well as the external input X.
- Output Y column values are obtained from the present state column of the state table.

Derivation of state table :

- The derivation of state table is as follows. Let us consider each present state and obtain the corresponding next state and Y output.

Present state AB = 00 : **(C-6392)**

Next state with X = 0	Next state with X = 1
• $J_A = X \cdot B = 0 \cdot 0 = 0$ and $K_A = 1$	• $J_A = X \cdot B = 1 \cdot 0 = 0$ and $K_A = 1$
• Hence $A_{n+1} = 0$	• Hence $A_{n+1} = 0$
• $J_B = \overline{X} \cdot \overline{A} = 1 \cdot 1 = 1$ and $K_B = 1$	• $J_B = \overline{X} \cdot \overline{A} = 0 \cdot 1 = 0$ and $K_B = 1$
• So FF-B will toggle	
• Hence $B_{n+1} = 1$	• Hence $B_{n+1} = 0$
• Hence next state = 01	• Hence next state = 00
Output $Y = \overline{A + \overline{B}} = \overline{0 + 1}$ $\therefore\ Y = 0$	

Present state AB = 01 : **(C-6393)**

Next state with X = 0	Next state with X = 1
• $J_A = X \cdot B = 0 \cdot 1 = 0$ and $K_A = 1$	• $J_A = X \cdot B = 1 \cdot 1 = 1$ and $K_A = 1$
• Hence $A_{n+1} = 0$	• Hence FF-A toggles
• $J_B = \overline{X} \cdot \overline{A} = 1 \cdot 1 = 1$ and $K_B = 1$	• $\therefore\ A_{n+1} = 1$
• Hence $B_{n+1} = 0$ as FF-B toggles	• $J_B = \overline{X} \cdot \overline{A} = 0 \cdot 1 = 0$ and $K_B = 1$
• Hence next state = 00	• Hence $B_{n+1} = 0$
	• Hence next state = 10
Output $Y = \overline{A + \overline{B}} = \overline{0 + 0} = 1$	

Present state AB = 10 : **(C-6394)**

Next state with X = 0	Next state with X = 1
• $J_A = X \cdot B = 0 \cdot 0 = 0$ and $K_A = 1$	• $J_A = X \cdot B = 1 \cdot 0 = 0$ and $K_A = 1$
• Hence $A_{n+1} = 0$	• Hence $A_{n+1} = 0$
• $J_B = \overline{X} \cdot \overline{A} = 1 \cdot 0 = 0$ and $K_B = 1$	• $J_B = \overline{X} \cdot \overline{A} = 0 \cdot 0 = 0$ and $K_B = 1$
• Hence $B_{n+1} = 0$	• Hence $B_{n+1} = 0$
• Therefore next state = 00	• Therefore next state = 00
Output $Y = \overline{A + \overline{B}} = \overline{1 + 1} = 0$	

- Similarly we can obtain the next state and output for the present state of 1 1 as follows :

  Present state : AB = 11

  Next state (X = 0) : 00 next state (X = 1) = 00

  Output Y = 0

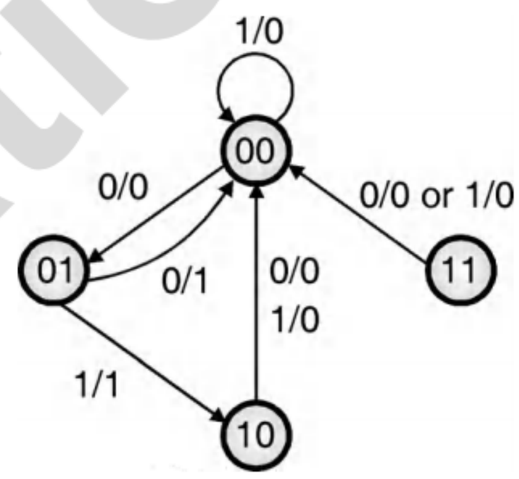
- The complete state table is given in Table P. 6.6.1(b).

(C-8079) Table P. 6.6.1(b) : State table

Present state	Next state A_{n+1} B_{n+1}		Output Y
A B	X = 0	X = 1	
0 0	0 1	0 0	0
0 1	0 0	1 0	1
1 0	0 0	0 0	0
1 1	0 0	0 0	0

Step 3 : Draw the state diagram :

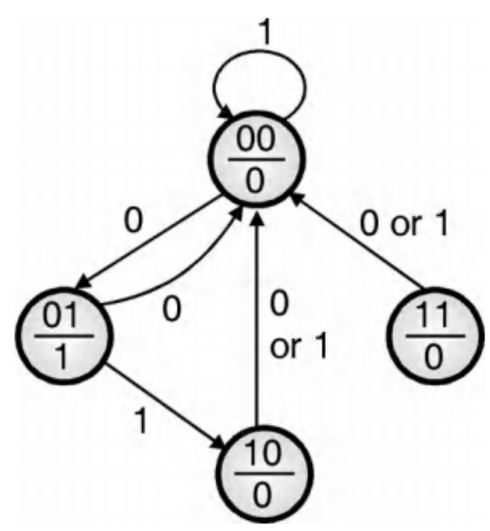
- The state diagram is drawn with the help of state table. The state diagram is shown in Fig. P. 6.6.1(b).



(C-676) Fig. P. 6.6.1(b) : State diagram

Step 4 : Modified state diagram :

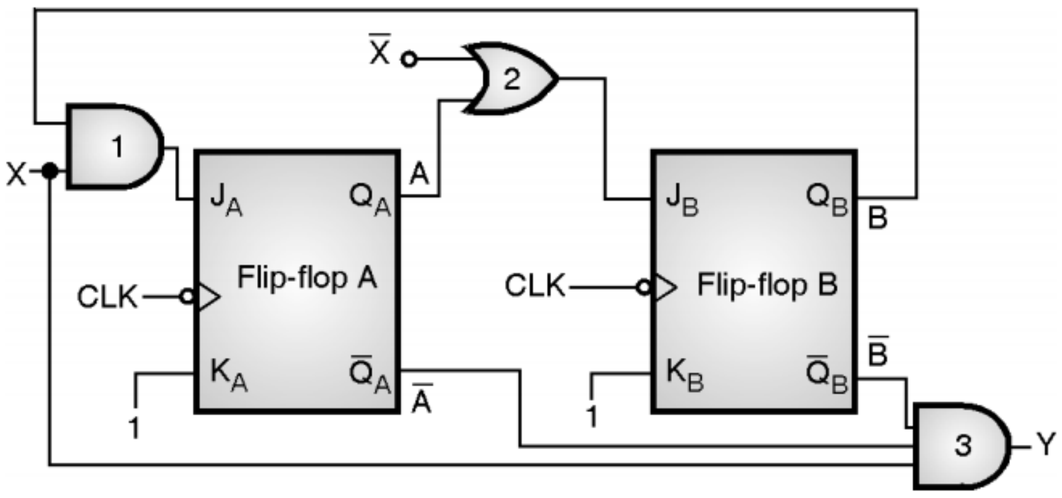
- The modified state diagram for Moore circuit is shown in Fig. P. 6.6.1(c).
- Note that the binary number corresponding to Y output has been included inside the circle itself.



(C-677) Fig. P. 6.6.1(c) : Modified state diagram for the given Moore circuit

TechKnowledge Publications

Ex. 6.6.2 : Identify the circuit shown in Fig. P. 6.6.2(a). Write the state table and draw state diagram for the same.



(C-678) Fig. P. 6.6.2(a) : Given circuit

Soln. :

Step 1 : Identify the type of circuit :

– The given circuit is a Mealy circuit since the Y output depends on the present state as well as the X input.

Step 2 : Write the state table :

– The state table for given circuit is given in Table P. 6.6.2.

– Note that the output Y contains two columns one each for X = 0 and X = 1. This is because in Melay circuit the output depends on external input as well as the present state.

$$Y = \bar{A} \cdot \bar{B} \cdot X \quad ...(1)$$

where $\bar{A}\,\bar{B}$ represents the present state of flip-flops.

(C-8080) Table P. 6.6.2 : State table

Present state	Next state $A_{n+1}\ B_{n+1}$		Output $Y = \bar{A} \cdot \bar{B} \cdot X$	
A B	X = 0	X = 1	X = 0	X = 1
0 0	0 1	0 0	0	1
0 1	0 0	1 0	0	0
1 0	0 1	0 1	0	0
1 1	0 0	0 0	0	0

Here, $J_A = X \cdot B$ and $J_B = \bar{X} + A$

Derivation of state table :

– Let me demonstrate the procedure to obtain the next state and output corresponding to first two present states AB = 00 and 01.

– You can follow the same procedure for the remaining states.

Present state AB = 00 :

Next state with X = 0	Next state with X = 1
• $J_A = X \cdot B = 0 \cdot 0 = 0$ and $K_A = 1$	• $J_A = X \cdot B = 1 \cdot 0 = 0$ and $K_A = 1$
• Hence $A_{n+1} = 0$	• Hence $A_{n+1} = 0$
• $J_B = \bar{X} + A = 1 + 0 = 1$ and $K_A = 1$ Hence FF-B will toggle	• $J_B = \bar{X} + A = 0 + 0 = 0$ and $K_A = 1$
• Therefore $B_{n+1} = 1$	• Hence $B_{n+1} = 0$
• Hence next state = 01	• Therefore the next state = 00
Output Y	
• For X = 0, $Y = \bar{A}\bar{B}X = 1 \cdot 1 \cdot 0 = 0$	
• For X = 1, $Y = \bar{A}\bar{B}X = 1 \cdot 1 \cdot 1 = 1$	

(C-6396)

Present state AB = 01 :

Next state with X = 0	Next state with X = 1
• $J_A = X \cdot B = 0 \cdot 1 = 0$ and $K_A = 1$	• $J_A = X \cdot B = 1 \cdot 1 = 1$ and $K_A = 1$
• Hence $A_{n+1} = 0$	So FF-A will toggle
• $J_B = \bar{X} + A = 1 + 0 = 1$ and $K_B = 1$	• Hence $A_{n+1} = 1$
So FF-B will toggle	• $J_B = \bar{X} + A = 0 + 0 = 0$ and $K_B = 1$
• Hence $B_{n+1} = 0$	• Hence $B_{n+1} = 0$
• Therefore next state = 00	• Therefore the next state = 10
Output Y	
• For X = 0, $Y = \bar{A}\bar{B}X = 1 \cdot 0 \cdot 0 = 0$	
• For X = 1, $Y = \bar{A}\bar{B}X = 1 \cdot 0 \cdot 1 = 0$	

(C-6397)

– Similarly we can obtain the next state and output Y for the remaining two present states to complete the state table.

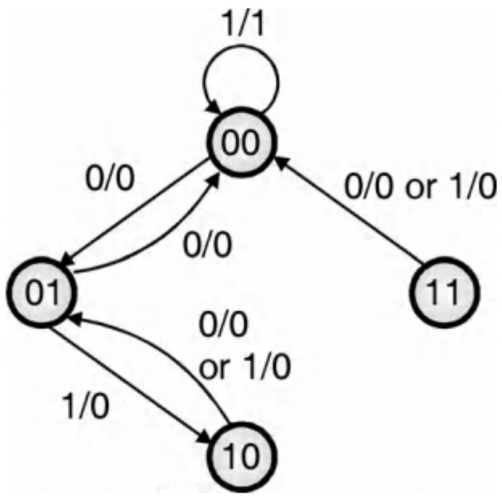
Step 3 : Draw the state diagram :

– The state diagram for the given Melay circuit is shown in Fig. P. 6.6.2(c). The state table has been repeated in Fig. P. 6.6.2(b) for reference.

Present state	Next state $A_{n+1}\ B_{n+1}$		Output Y	
A B	X = 0	X = 1	X = 0	X = 1
0 0	0 1	0 0	0	1
0 1	0 0	1 0	0	0
1 0	0 1	0 1	0	0
1 1	0 0	0 0	0	0

(C-8081) (b) State table repeated
Fig. P. 6.6.2(Contd...)

TechKnowledge Publications



(c) State diagram

(C-679) **Fig. P. 6.6.2**

6.7 State Reduction and Assignments :

- In the analysis of a sequential circuit we always starts from a circuit diagram which has been given to us and derive the state table and then draw the state diagram as we did in the solved examples of previous section.
- In order to design a sequential circuit a set of specifications is given to us. We design the required circuit and finally draw the logic diagram of the designed logic circuit.
- In this section we are going to discuss some properties of synchronous sequential circuits which are useful in reducing the number of gates and flip-flops while implementing the designed circuits.

State reduction techniques :

- In order to minimize the number of components required to implement the designed circuit, we need to reduce the number of states the circuit is going through.
- There are two state reduction techniques :
  1. Row elimination method.
  2. Implication table method.

6.7.1 Row Elimination Method :

- In the design of synchronous sequential circuits the first step is to draw the state diagram and then write the state table.
- Some of the states in the state diagram/state table could be redundant states.
- These redundant states can be avoided by the technique called state reduction technique.
- Thus the state reduction helps us to eliminate the redundant states which ultimately results in reducing the number of flip-flops and gates to be used in a sequential circuit.

Identification of a redundant state :

- If two states in a state diagram are equivalent, then one of them is treated as redundant and can be eliminated.
- Those two states are said to be equivalent if every same output and same next state is produced for every possible set of inputs.
- Reducing the number of states will not alter the input output relation of the circuit being designed.

6.7.2 Example on Row Elimination Method :

- The specifications of a sequential circuit are given in the form of a state diagram.
- The steps to followed for state reduction are as follows.

Steps to be followed :

Step 1 : Derive the state table from the given state diagram.

Step 2 : Find out the equivalent states.

Step 3 : Eliminate one of the every pair of equivalent states and write a new state table.

Step 4 : In the new state table, check for equivalent states and continue till all the redundant states are eliminated.

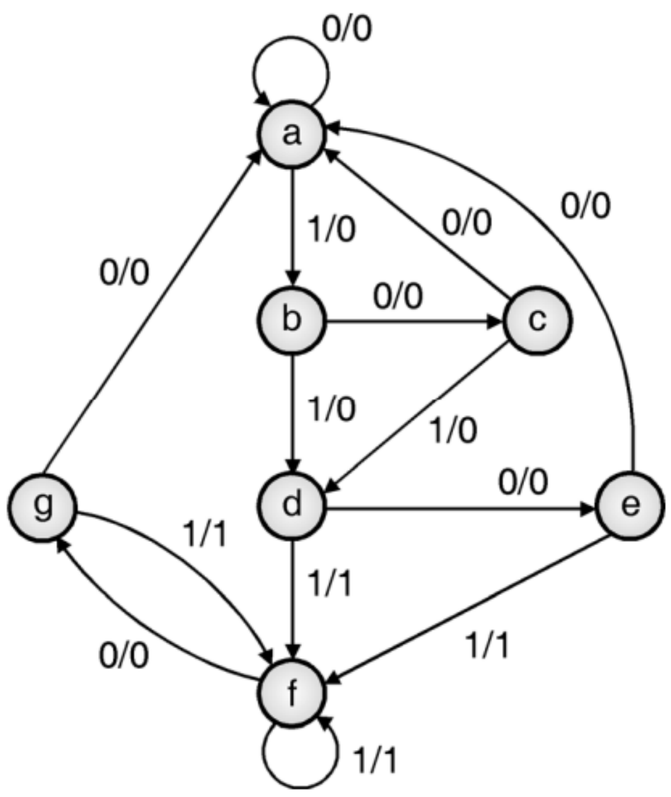
Step 5 : Draw the state diagram with reduced number of states.

- The concept of state reduction using the row elimination method has been illustrated in the following example.

Ex. 6.7.1 : How will you perform state reductions in Sequential Logic Design ? Reduce the number of states from the following State Diagram. Starting with initial state table, clearly justify the reduced state and reduced state table. Also draw the reduced state diagram.

Dec. 07, 10 Marks, May 14, 6 Marks

TechKnowledge Publications



(C-680) Fig. P. 6.7.1(a) : Given circuit diagram

Soln. :

Step 1 : Write the state table :

– Table P. 6.7.1(a) represents the state table derived from the given state diagram.

(C-6236) Table P. 6.7.1(a) : State table

Present state	Next state		Output Y	
	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**

States "e" and "g" are equivalent states

Step 2 : Identify the equivalent states :

– Two states are said to be equivalent if, for all the possible input combinations, they give exactly the same output and send the circuit either to the same state or to an equivalent state.

– Apply this principle to the state table of Table P. 6.7.1(a). Look at the state table for two present states which go the same next states and have same output (Y) for both the input conditions (X = 0 and X = 1).

– The states "e" and "g" in Table P. 6.7.1(a) are found to be equivalent states, because both go to the next states "a" and "f" for X = 0 and 1 respectively with Y = 0 and 1 for X = 0 and 1 respectively.

Step 3 : Eliminate one of the equivalent states and write a new state table :

– As the states "e" and "g" are equivalent, we can remove one of them.

– Let us remove state "g" i.e. the last row containing the present state "g", and then replace "g" by "e" each time it appears in the next state column shown in the new state table of Table P. 6.7.1(b).

(C-6237) Table P. 6.7.1(b) : Modified state table with state reduction

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~~ e	f	0	1

After replacing "g" by "e" the states "d" and "f" have become equivalent

State g has been eliminated

"g" is replaced by "e"

– Note that in the last row, we have replaced "g" by "e".

– After doing so, the states "d" and "f" become equivalent states. So eliminate "f".

Step 4 : Replace "f" by "d" and prepare a new state table :

– The reduced state table is as follows (Table P. 6.7.1(c)). Note that each "f" in the next state column is replaced by "d" and the last row for present state "f" has been removed completely.

(C-6238) Table P. 6.7.1(c) : Reduced state table

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	~~f~~ d	0	1
e	a	~~f~~ d	0	1

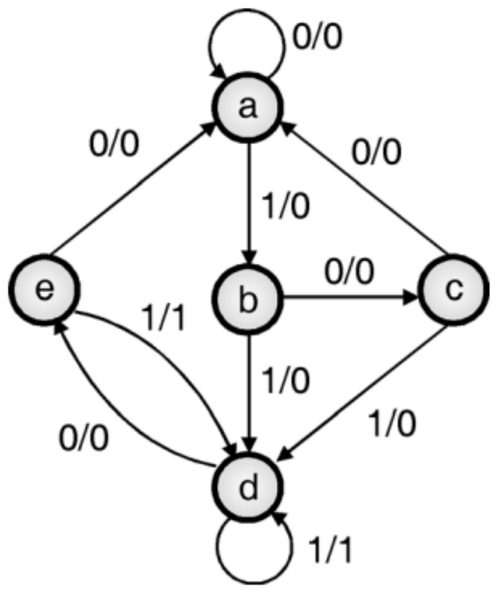
"f" is replaced by "d"

State "f" has been eliminated

– In Table P. 6.7.1(c) all the 5 states are different from each other. Therefore the process of state reduction is complete and this table represents the reduced state table, after state reduction.

Step 5 : Draw the state diagram :

– The state diagram for reduced state table is shown in Fig. P. 6.7.1(b).



(C-681) Fig. P. 6.7.1(b) : Reduced state diagram

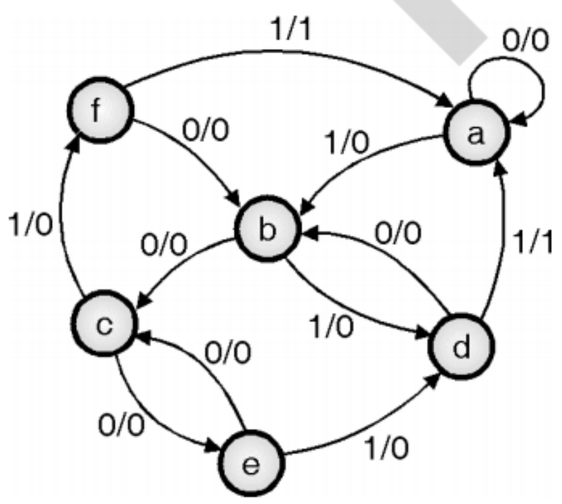
6.7.3 Reduction in Number of FFs due to State Reduction :

- By using the state reduction technique, we reduce the number of states and this will reduce the number of flip flops required to be used to realize the circuit.
- If 2 flip flops are used then 2^2 = 4 states are produced or with 3 flip flops 2^3 = 8 states will be produced.
- Thus in general "m" flip-flops produce 2^m states and a reduction in number of states may (or may not) result in a reduction in number of flip-flops.

6.7.4 Implication Table Method :

- This is another method used to solve the **state reduction** problem. This method is more systematic than the row elimination method discussed earlier.
- If the number of original states in the given example is n then there will be (n–1) rows and (n–1) columns for the implication table.
- The procedure of state reduction using implication table method will be clear after going through the following example.

Ex. 6.7.2 : For the state diagram of Fig. P. 6.7.2(a) obtain the reduced state table and reduced state diagram using : 1. Row elimination method and 2. Implication table method.



(C-3663) **Fig. P. 6.7.2(a) : Given stage diagram**

Soln. :

1. Row elimination method :

- Write the original state table from state diagram and reduce it.

(C-3664)**Table P. 6.7.2(a) : Original state table**

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	c	d	0	0
c	e	f	0	0
d	b	a	0	1
e	c	d	0	0
f	b	a	0	1

- States b and e are equivalent as there next state and output are same. So we will retain state b and discard state e.

(C-3664) **Table P. 6.7.2(b)**

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	c	d	0	0
c	~~e~~ b	f	0	0
d	b	a	0	1
f	b	a	0	1

- States d and f are equivalent hence we retain d and discard f.
- In Table P. 6.7.2(c) then next state for X = 1 and the output are same for present states b and c.
- The next state of b for X = 0 is c and next state of c for X = 0 is b.
- Thus bc are equivalent if their next states cb are equivalent which is always true. Hence states b and c are equivalent. We retain b and discard c.

(C-3665)**Table P. 6.7.2(c)**

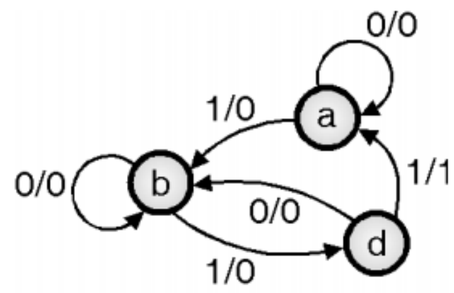
Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	c	d	0	0
c	~~e~~ b	~~f~~ d	0	0
d	b	a	0	1

- The final simplified state table is as shown in Table P. 6.7.2(d). It contains only three states a, b and d.

Table P. 6.7.2(d) : Final simplified table

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	~~c~~ b	d	0	0
d	b	a	0	1

- The simplified state diagram is as shown in Fig. P. 6.7.2(b).



(C-3666) Fig. P. 6.7.2(b) : Reduced state diagram

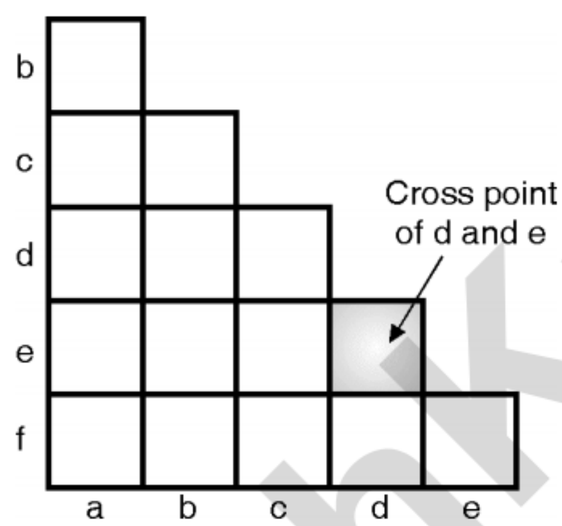
2. Implication table method :

Step 1 : Draw the implication table :

– Fig. P. 6.7.2(c) shows the implication table. As there are 6 states in the original state diagram, there will be 5 rows and 5 columns for the implication table.

– The cross point in an implication table is the location where a ROW meets a COLUMN.

– At the cross points we test the conditions for equivalence between the states crossing each other.



(C-3667) Fig. P. 6.7.2(c) : Implication table

– The implication table is filled up by using the original state table. (Table P. 6.7.2(a)) which was derived from the original state diagram.

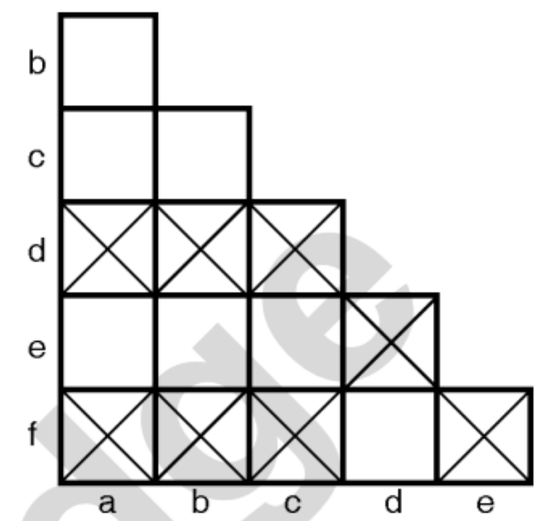
Step 2 : Identify the states which are not equivalent :

– In Table P. 6.7.2(a), we identify the non-equivalent states i.e. the states having unequal outputs. These states are denoted by putting a double cross (X) in the respective cross points.

– In this problem the states d and f only have output = 1 for X = 1. This is different than the other states.

– Hence d and f states cannot be equivalent to other states (a, b, c and e).

– So we put a double cross at all the cross points corresponding to states a, b, c, e with d and f as shown in Fig. P. 6.7.2(d).



(C-3668) Fig. P. 6.7.2(d) : Implication table for step 2

Step 3 : Write the necessary conditions for equivalence of intersecting states. For example start from the first column (state a) and top row (state b). Refer to the original state table of Table P. 6.7.2(a) i.e. rows a and b.

– If we want states a and b to be equivalent then a should be equivalent to c and b should be equivalent to d. This is written at the cross point of a and b in the implication table of Table P. 6.7.2(e).

– Similarly fill up all the remaining points as shown in Fig. P. 6.7.2(e).

– Note that states b and e are equivalent so we put a double tick (✓✓) at their cross point.

Step 4 : Now we will use the relationship obtained in steps 2 and 3 specially the double cross and double ticks to see if other cross points also can be crossed or ticked.

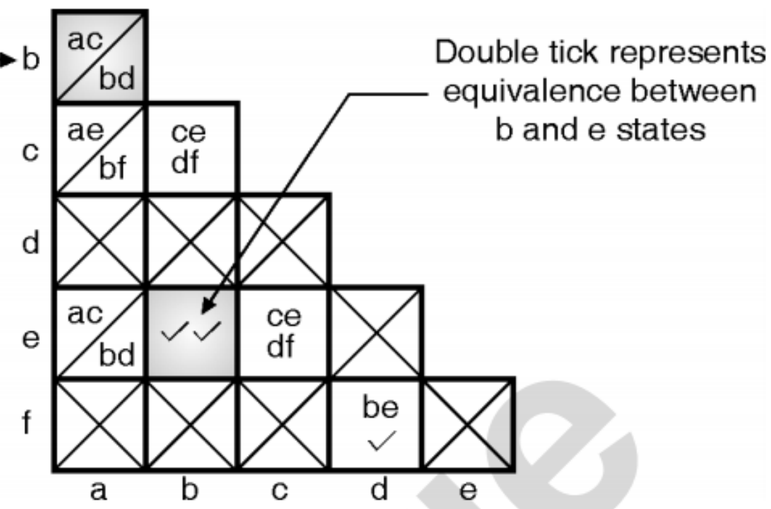
Step 5 : Keep repeating step 4 and cross or tick as many cross points as possible. For example ab and ae cross points should be crossed because they need ac to be equivalent and ac has been already crossed in the previous step.

– Now there is no further ticking or crossing, we go to the final step.

TechKnowledge Publications

Table P. 6.7.2(e) : Original state table

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	c	d	0	0
c	e	f	0	0
d	b	a	0	1
e	c	d	0	0
f	b	a	0	1

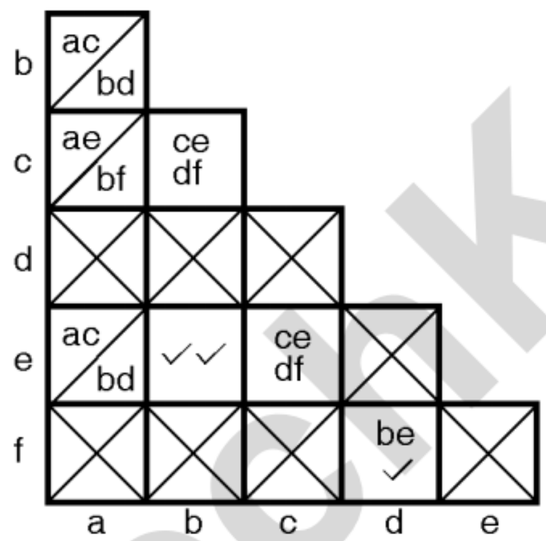


(C-3669) **Fig. P. 6.7.2(e)**

Step 6 : Check pairwise equivalence :

Column e :

– Start from the rightmost column i.e. e of the implication table of Fig. P. 6.7.2(f). The only cross point is ef which is double crossed, there is no other cross point in this column. Therefore there is no equivalence possible in this column so we write e : e in Fig. P. 6.7.2(g).



(f)

e : e

d : e (df)

c : e (df) (ce) ≡ (df) (ce)

b : (df) (ce) (bc) ≡ (df) (bce)

a : a (df) (bce)

∴ P = (a) (df) (bce)

(g)

(C-3670) **Fig. P. 6.7.2**

Column d :

– In column d of Fig. P. 6.7.2(f) it is observed that d and f are equivalent. The crosspoint of d and e is double crossed. Hence in Fig. P. 6.7.2(g) we write

d : e (df)

Column c :

– In column c we find that ce are equivalent as df are equivalent. Hence in Fig. P. 6.7.2(g) we write

c : e (df) (ce) ≡ (df) (ce)

Column b :

– In column b the state be are equivalent and the states bc also become equivalent as ce and df are already considered equivalent. Hence in Fig. P. 6.7.2(g) we write

b : (df) (ce) (bc) ≡ (df) (bce)

– This is because when bc are equivalent and be are equivalent bce are equivalent.

Column a :

– This column does not give any equivalence. So we write

a : (df) (bce)

Final result :

– The final partition table is represented as :

P = (a) (df) (bce)

– These three partitions represent that three states are sufficient to solve the given problem.

∴ P = (a) (b) (d)

– Where b represents equivalent states (bce) and d represents equivalent states (df).

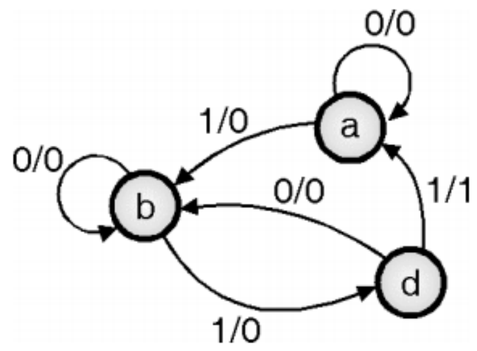
– The reduced state table corresponding to these three states is as shown in Table P. 6.7.2(f) and the reduced state diagram is as shown in Fig. P. 6.7.2(h).

(C-8083) **Table P. 6.7.2(f) : Reduced state table**

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	b	d	0	0
d	b	a	0	1

TechKnowledge Publications

– The simplified state diagram is as shown in Fig. P. 6.7.2(h).



(C-3671) Fig. P. 6.7.2(h) : Reduced state diagram

Note : We get the same results using both methods.

Ex. 6.7.3 : For the state table given in Table P. 6.7.3(a) obtain the reduced state diagram (Moore FSM) using the row elimination and implication table methods.

(C-5951) Table P. 6.7.3(a) : Given state table

PS	NS X = 0	NS X = 1	Output Z
a	a	b	0
b	c	d	0
c	d	e	1
d	c	b	0
e	b	c	1

Soln. :

1. Row elimination method :

– In Table P. 6.7.3(a) rows b and d are equivalent. So let us retain state "b" and discard state "d" to obtain Table P. 6.7.3(b).

(C-5952) Table P. 6.7.3(b) : Table after elimination of one row

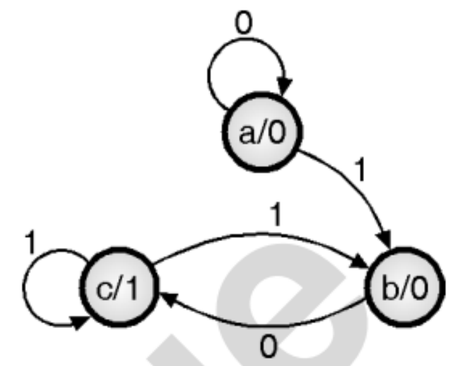
PS	NS X = 0	NS X = 1	Output Z
a	a	b	0
b	c	~~d~~ b	0
c	~~d~~ b	~~e~~ c	1
e	b	c	1

– In Table P. 6.7.3(b) rows "c" and "e" are equivalent. So let us retain state "c" and discard state "e" to obtain Table P. 6.7.3(c).

(C-8084) Table P. 6.7.3(c) : Final table after reducing two rows

Present state	Next state X = 0	Next state X = 1	Output Z
a	a	b	0
b	c	b	0
c	b	c	1

– No further simplification is possible. Hence Table P. 6.7.3(c) is the final reduced table. We can draw the reduced state diagram as shown in Fig. P. 6.7.3(a).

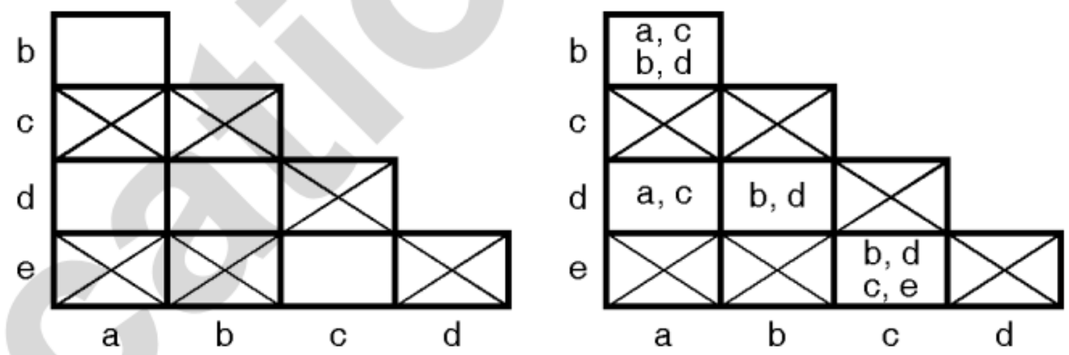


(C-5953) Fig. P. 6.7.3(a) : Reduced state diagram with row elimination method

2. Implication table method :

Step 1 : Draw implication table and identify the non-equivalent states.

– The states having unequal outputs are non-equivalent states. Put a X (double cross) in the respective cross points as shown in Fig. P. 6.7.3(b).



(b) Implication table for step 1 (c) Implication table for step 2

(C-5954) Fig. P. 6.7.3

Step 2 : Write the necessary conditions for the equivalence of intersecting states as shown in Fig. P. 6.7.3(c).

Step 3 : Further simplification :

– From Fig. P. 6.7.3(c) it is evident that because states "a" and "c" can never be equivalent (X at the crosspoint ac) the states "ab" and "ad" can never be equivalent.

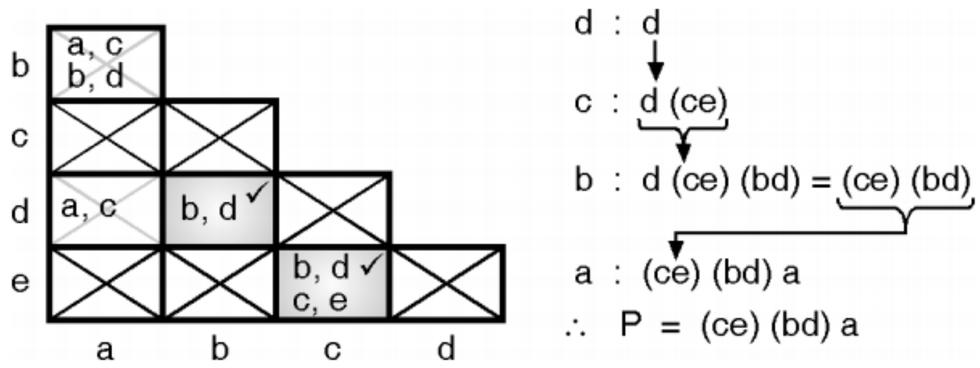
– As "bd" pair is dependent on itself for equivalency. Hence it is (b and d) an equivalent pair of states.

– As "bd" are equivalent and "ce" pair is dependent on itself for equivalency, the states "c" and "e" are equivalent.

Step 4 : Check the pairwise equivalence :

– We will move columnwise starting from "d" to get the final partition P as shown in Fig. P. 6.7.3(d).

TechKnowledge Publications



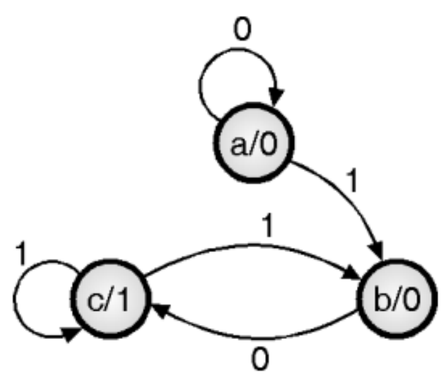
(C-5955) Fig. P. 6.7.3(d)

– Thus the final partition table is represented as

$$P = (ce)\ (bd)\ (a)$$

– Thus three states "abc" or "ade" are sufficient to solve the given example.

Step 5 : Draw state diagram :



(C-5956) Fig. P. 6.7.3(e) : Reduced state diagram with implication table method

Note :	Thus we get the same results using both the methods.

Ex. 6.7.4 : Reduce the following state table using implication chart method.

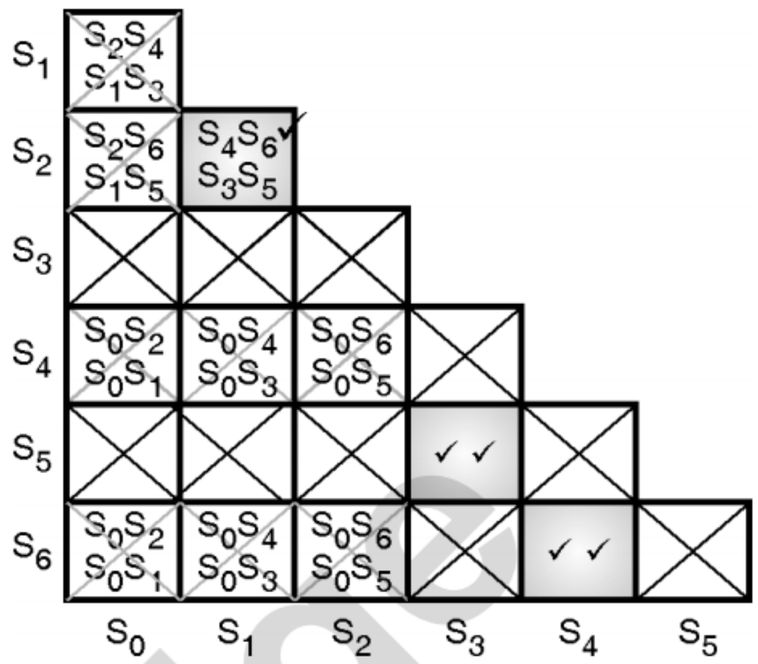
(C-8085) Table P. 6.7.4(a) : Given state table

Present state	Next state		Output Z	
	X = 0	X = 1	X = 0	X = 1
S_0	S_2	S_1	0	0
S_1	S_4	S_3	0	0
S_2	S_6	S_5	0	0
S_3	S_0	S_0	1	1
S_4	S_0	S_0	0	0
S_5	S_0	S_0	1	1
S_6	S_0	S_0	0	0

Soln. :

Step 1 : Identify nonequivalent states on the basis of unequal outputs and place a X (double cross) in the respective squares (Fig. P. 6.7.4(a)).

Step 2 : Enter the necessary conditions for the equivalency of intersecting states (Fig. P. 6.7.4(a)).



(C-5964) Fig. P. 6.7.4(a) : Implication table for steps 1 and 2

Step 3 : Further simplification :

– Note that pair $\{S_3, S_5\}$ and $\{S_4, S_6\}$ are equivalent. Hence pair $\{S_1, S_2\}$ becomes equivalent.

– Since $\{S_0, S_3\}$ pair is not equivalent (double cross) the pairs $\{S_1, S_6\}$ and $\{S_1, S_4\}$ cannot be equivalent. Put a cross in the squares corresponding to them.

– Since $\{S_1, S_5\}$ and $\{S_1, S_3\}$ are non-equivalent pairs (double cross) hence pairs $\{S_0, S_1\}$ and $\{S_0, S_2\}$ cannot be equivalent. Put a cross.

– Since $\{S_0, S_1\}$ and $\{S_0, S_2\}$ are non-equivalent pairs, the pair $\{S_0, S_4\}$ and $\{S_0, S_6\}$ also cannot be equivalent.

– Finally as $\{S_0, S_5\}$ are non-equivalent states (double cross X), the states $\{S_2, S_4\}$ and $\{S_2, S_6\}$ can't be equivalent. Put cross.

Step 4 : Check pairwise equivalence :

– Move columnwise from S_5 to get the final partition P as shown in Fig. P. 6.7.4(b).

$S_5 : S_5$

$S_4 : S_5\ (S_4, S_6)$

$S_3 : S_5\ (S_4, S_6)\ (S_3\ S_5) = (S_4\ S_6)\ (S_3\ S_5)$

$S_2 : (S_4\ S_6)\ (S_3\ S_5)\ S_2$

$S_1 : (S_4\ S_6)\ (S_3\ S_5)\ S_2\ (S_1\ S_2)$

$S_0 : (S_4\ S_6)\ (S_3\ S_5)\ (S_1\ S_2)\ S_0$

$\therefore\ P = (S_4)\ (S_3)\ (S_2)\ (S_0)$

Fig. P. 6.7.4(b)

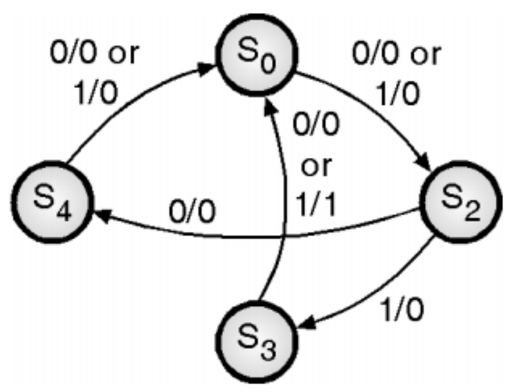
TechKnowledge Publications

– Thus only four states S_0, S_2, S_3 and S_4 are sufficient to solve this example.

Step 5 : Reduced state table and state diagram :

(C-8086) Table P. 6.7.4(b) : Reduced state table

Present state	Next state		Output Z	
	X = 0	X = 1	X = 0	X = 1
S_0	S_2	S_2	0	0
S_2	S_4	S_3	0	0
S_3	S_0	S_0	1	1
S_4	S_0	S_0	0	0



(C-5965) Fig. P. 6.7.4(c) : Reduced state diagram

6.8 State Assignment :

– In the previous section we have learnt the state reduction technique to minimize the number of flip-flops. Now we will learn another technique which is used to reduce the cost of the combinational circuit.

– Such a technique is called as state assignment technique.

– It plays an important role in minimizing the number of gates used in the combinational circuit which drives the flip flops.

– Thus far we were using alphabets a, b, ... for representing states of a synchronous sequential circuits.

– But using the state assignment technique we can assign binary values to these states, in such a way that it will reduce the cost of the combinational circuit which drives the flip flops.

– The concept of state assignment will be clear after solving the following example.

Ex. 6.8.1 : For the given state table obtain the reduced state table using state assignment procedure.

(C-8087) Table P. 6.8.1(a) : Given state table

Present state	Next state		Output (Y)	
	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

Soln. :

Step 1 : Assign various possible binary values to different states :

– The assignment of various possible binary values to different states is demonstrated in Table P. 6.8.1(b).

– Note that the binary values of the state are not important as long as their sequence maintains the proper input output relationship.

– Hence assignment of any binary number to any state is satisfactory if we assign a unique number to each state.

– Table P. 6.8.1(b) shows three examples of possible binary assignment.

(C-8088) Table P. 6.8.1(b) : Three possible binary state assignments

State	Assignment 1	Assignment 2	Assignment 3
a	001	000	000
b	010	010	100
c	011	011	010
d	100	101	101
e	101	111	011

– Assignment-1 in Table P. 6.8.1(b) is a straightforward binary assignment, whereas the remaining two are done arbitrarily.

Step 2 : Obtain the reduced state table :

– Table P. 6.8.1(c) represents the reduced state table with binary assignment 1 substituted in place of letter symbols of Table P. 6.8.1(a).

– Such a state table with binary assignment is sometimes called as a **transition table.**

– Note that if we substitute binary assignment 2 or 3, then we would get a completely new transition table, with the same input output relationship.

TechKnowledge Publications

(C-8089) Table P. 6.8.1(c) : Reduced state table or transition table

Present state	Next state		Output (Y)	
	X = 0	X = 1	X = 0	X = 1
001(a)	001	010	0	0
010(b)	011	100	0	0
011(c)	001	100	0	0
100(d)	101	100	0	1
101(e)	001	100	0	1

- This type of state table is used to derive the combinational part of the sequential circuit.
- The complexity of the combinational circuit obtained depends on the binary state assignment chosen.
- Therefore various procedures have been tried to select a particular binary assignment from so many possible ones.
- The most important criterion is that the selected binary assignment should result in a simpler combinational circuit.
- But it is surprising to know that there is no state assignment procedure that will guarantee a minimal-cost combinational circuit.

Ex. 6.8.2 : Determine the minimum state table equivalent shown in Table P. 6.8.2(a).

(C-8090) Table P. 6.8.2(a) : Given table

Present state	X = 0		X = 1	
	Next state	Output	Next state	Output
1	1	0	1	0
2	1	1	6	1
3	4	0	5	0
4	1	1	7	0
5	2	0	3	0
6	4	0	5	0
7	2	0	3	0

Soln. :

Step 1 : Rearrange the table as shown in Table P. 6.8.2(b) :

(C-6239) Table P. 6.8.2(b)

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
1	1	1	0	0
2	1	6	1	1
3	4	5	0	0
4	1	7	1	0
5	2	3	0	0
6	4	5	0	0
7	2	3	0	0

States 3 and 6 are equivalent

States 5 and 7 are equivalent

Step 2 : Identify the equivalent states :

In Table P. 6.8.2(b) the states 5 and 7 are equivalent. And states 3 and 6 also are equivalent.

Step 3 : Eliminate one of the equivalent states and write a new state table :

- Eliminate state 7 and then replace 7 by 5. Similarly eliminate state 6 and replace 6 by 3 to obtain the modified state table as shown in Table P. 6.8.2(c).

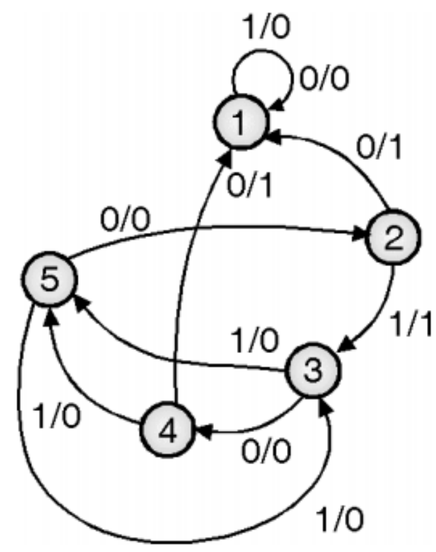
(C-6240) Table P. 6.8.2(c) : Modified state table with state reduction

Present state	Next state		Output		
	x = 0	x = 1	x = 0	x = 1	
1	1	1	0	0	
2	1	~~6~~ 3	1	1	← 6 replaced by 3
3	4	5	0	0	
4	1	~~7~~ 5	1	0	← 7 replaced by 5
5	2	3	0	0	

Step 4 : Write the reduced state table and draw the reduced state diagram :

(C-8091) Table P. 6.8.2(d) : Reduced state table

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
1	1	1	0	0
2	1	3	1	1
3	4	5	0	0
4	1	5	1	0
5	2	3	0	0



(C-682) Fig. P. 6.8.2

6.9 Design of Clocked Synchronous State Machine using State Diagram :

- Design of a clocked sequential circuits will start from the set of given specifications and it will end with drawing a logic diagram.

TechKnowledge Publications

– The stepwise design procedure is as follows :

Step 1 : A state diagram or timing diagram or some other information is given, which describes the behaviour of the circuit that is to be designed.

Step 2 : Draw the state table.

Step 3 : The number of states can be reduced by state reduction methods.

Step 4 : Assign binary values to each state for the states in steps 2 and 3 using state assignment technique.

Step 5 : Determine the number of flip-flops required and assign a letter symbol to each one.

Step 6 : Decide the type of FF to be used.

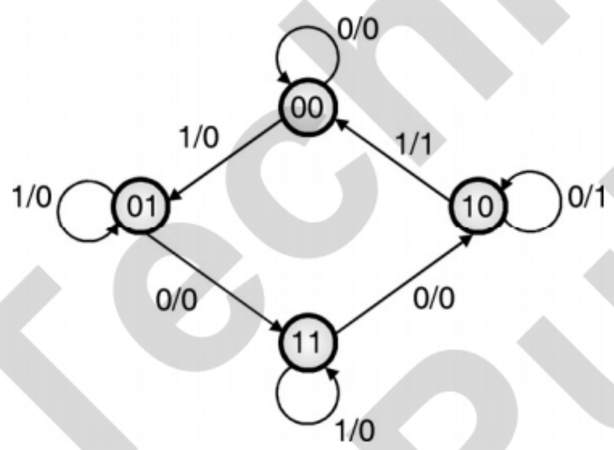
Step 7 : Derive the circuit excitation table and output table from the state table.

Step 8 : Obtain the expressions for the circuit output and flip-flop inputs.

Step 9 : Draw the logic diagram.

– The design procedure of clocked sequential circuits will be clear by solving the following example.

Ex. 6.9.1 : For the state diagram given in Fig. P. 6.9.1(a) draw the clocked sequential circuit using T flip-flops.



(C-685) **Fig. P. 6.9.1(a) : Given state diagram**

Soln. :

Step 1 : Write the state table :

– The state table is written from the given state diagram as shown in Table P. 6.9.1(a).

(C-7837) **Table P. 6.9.1(a) : State table**

Present state		Next state				Output	
		X = 0		X = 1			
A	B	A_{n+1}	B_{n+1}	A_{n+1}	B_{n+1}	X = 0	X = 1
0	0	0	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	1	0	0	0	1	1
1	1	1	0	1	1	0	0

Step 2 : Number of flip-flops :

– As seen from the state table, there are no equivalent states. So there won't be any reduction in the state diagram.

– The circuit goes through four states. Hence we need to use 2 flip-flops.

Step 3 : Write the circuit excitation table :

– The circuit excitation table is shown in Table P. 6.9.1(b). The type of FF used is T type FF.

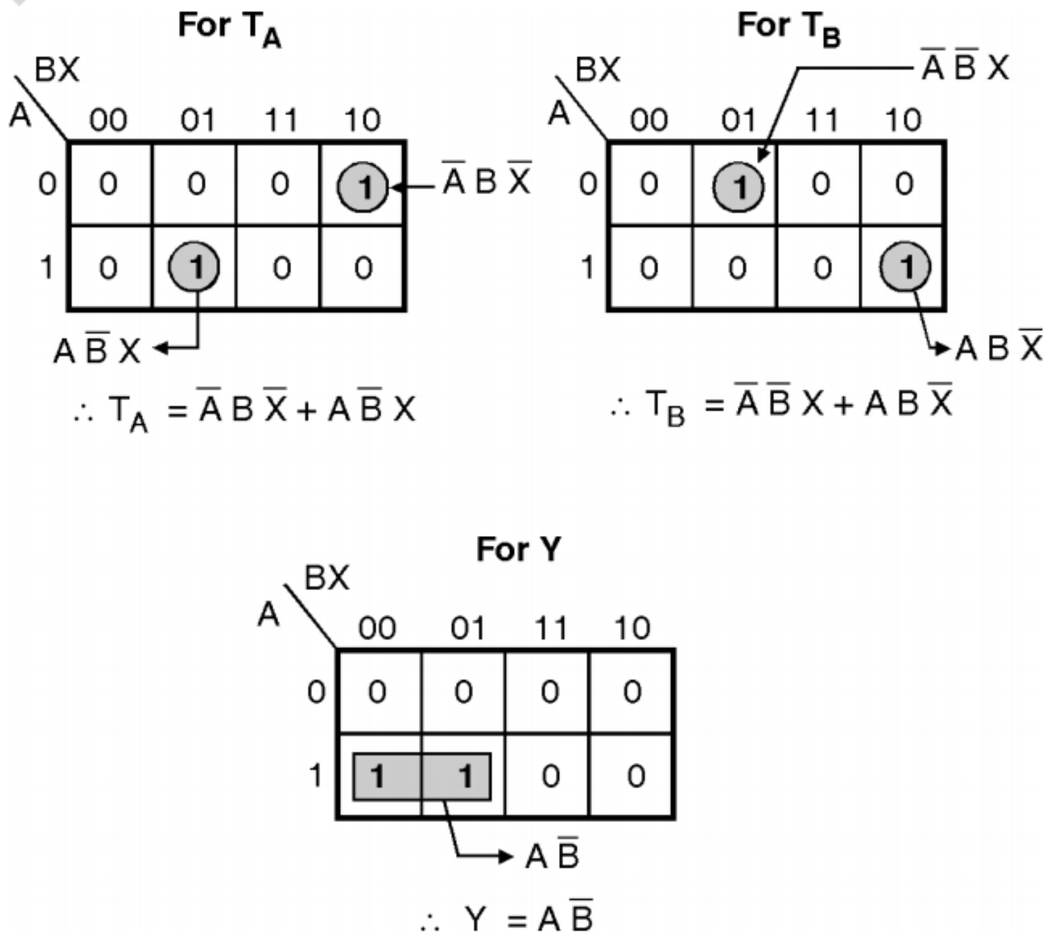
(C-7760) **Table P. 6.9.1(b) : Circuit excitation table**

Present state		Input	Next state		FF Inputs		Output
A	B	X	A_{n+1}	B_{n+1}	T_A	T_B	Y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	0	0	0
1	0	0	1	0	0	0	1
1	0	1	0	0	1	0	1
1	1	0	1	0	0	1	0
1	1	1	1	1	0	0	0

– The shaded portion of the circuit excitation table corresponds to the excitation table of a T FF.

Step 4 : K maps and simplifications :

– Fig. P. 6.9.1(b) shows the K-maps and corresponding simplified expressions for T_A T_B i.e. inputs of the two T flip-flops and the Y output.

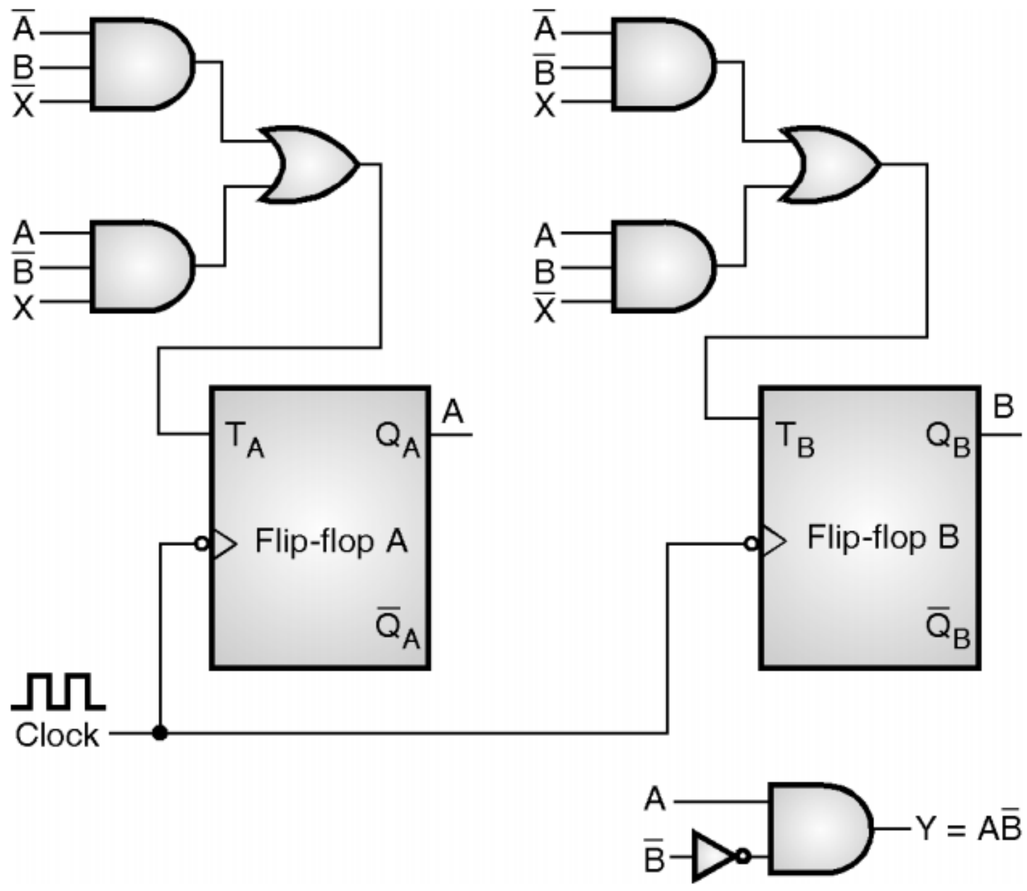


(C-686) **Fig. P. 6.9.1(b) : K-maps and simplifications**

Step 5 : Draw the logic diagram :

– The logic diagram for required clocked sequential circuit is shown in Fig. P. 6.9.1(c).

TechKnowledge Publications



(C-687) **Fig. P. 6.9.1(c)**

Note : We can use the design procedure in order to design the synchronous counters.

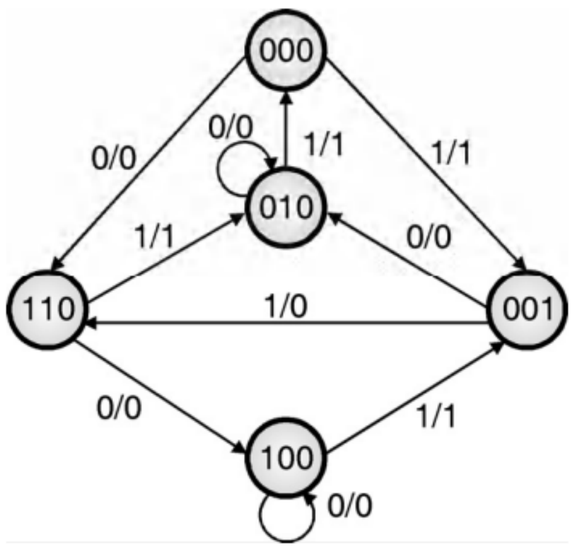
6.10 Design using Unused States :

- Sometimes a sequential circuit may use less number of states than the available maximum number of states.
- While designing such a sequential circuit, we may treat the unused states as don't care conditions.
- This is called as design with unused state and it leads to a more reliable design as well as reduction in the number of components.
- Solve the following example to get the idea about designing with unused states.

6.10.1 Synthesis Table :

- The circuit excitation table is also known as the synthesis table.

Ex. 6.10.1 : Implement the following state diagram using D flipflops.



(C-688) **Fig. P. 6.10.1(a) : Given state diagram**

Soln. :

- From the given state table it is evident that 3 flip flops are required to be used.
- The unused states are 011, 101 and 111.

Step 1 : Write the circuit excitation table :

- The required circuit excitation table is shown in Table P. 6.10.1(a).
- This state table has been derived from the given state diagram.

(C-8092) **Table P. 6.10.1(a) : Circuit excitation table**

Present state			Input X	Next state			Output Y	Flip flop inputs		
A	B	C		A_{n+1}	B_{n+1}	C_{n+1}		D_A	D_B	D_C
0	0	0	0	1	1	0	0	1	1	0
0	0	0	1	0	0	1	1	0	0	1
0	0	1	0	0	1	0	0	0	1	0
0	0	1	1	1	1	0	0	1	1	0
0	1	0	0	0	1	0	0	0	1	0
0	1	0	1	0	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	0	0
1	0	0	1	0	0	1	1	0	0	1
1	1	0	0	1	0	0	0	1	0	0
1	1	0	1	0	1	0	1	0	1	0

- Note that the entries in the column D_A have been made by taking into consideration the entries in the columns A and A_{n+1}.
- Similarly the entries in the columns of D_B and D_C have been made by considering B, B_{n+1} and C, C_{n+1} respectively.
- The excitation table for a general D FF is shown in Table P. 6.10.1(b).

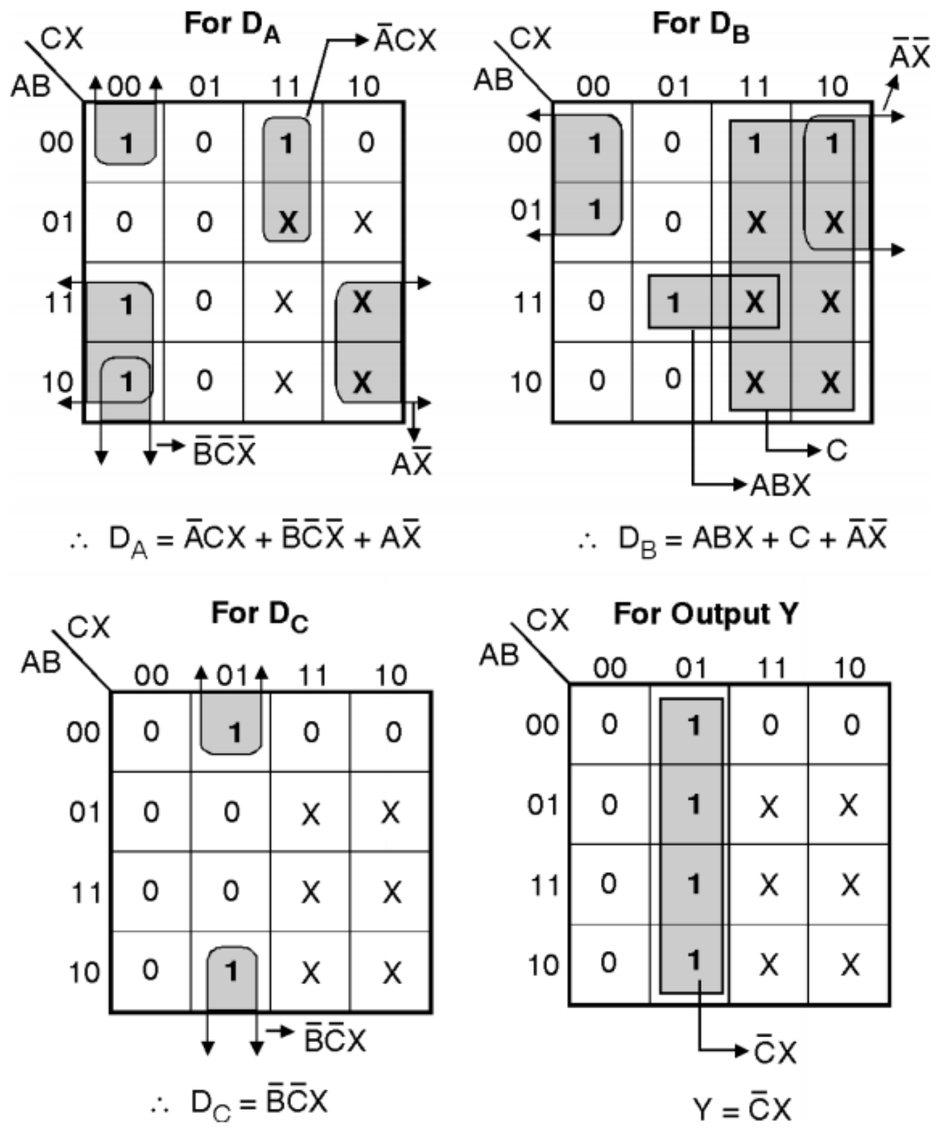
(C-7828) **Table P. 6.10.1(b) : Excitation table for a D FF**

Q output		Input
Present state	Next state	D_n
0	0	0
0	1	1
1	0	0
1	1	1

Step 2 : K maps and simplifications :

- The K maps for D_A, D_B and D_C and output Y are shown in Fig. P. 6.10.1(b).
- The simplified expression are also derived.

TechKnowledge Publications

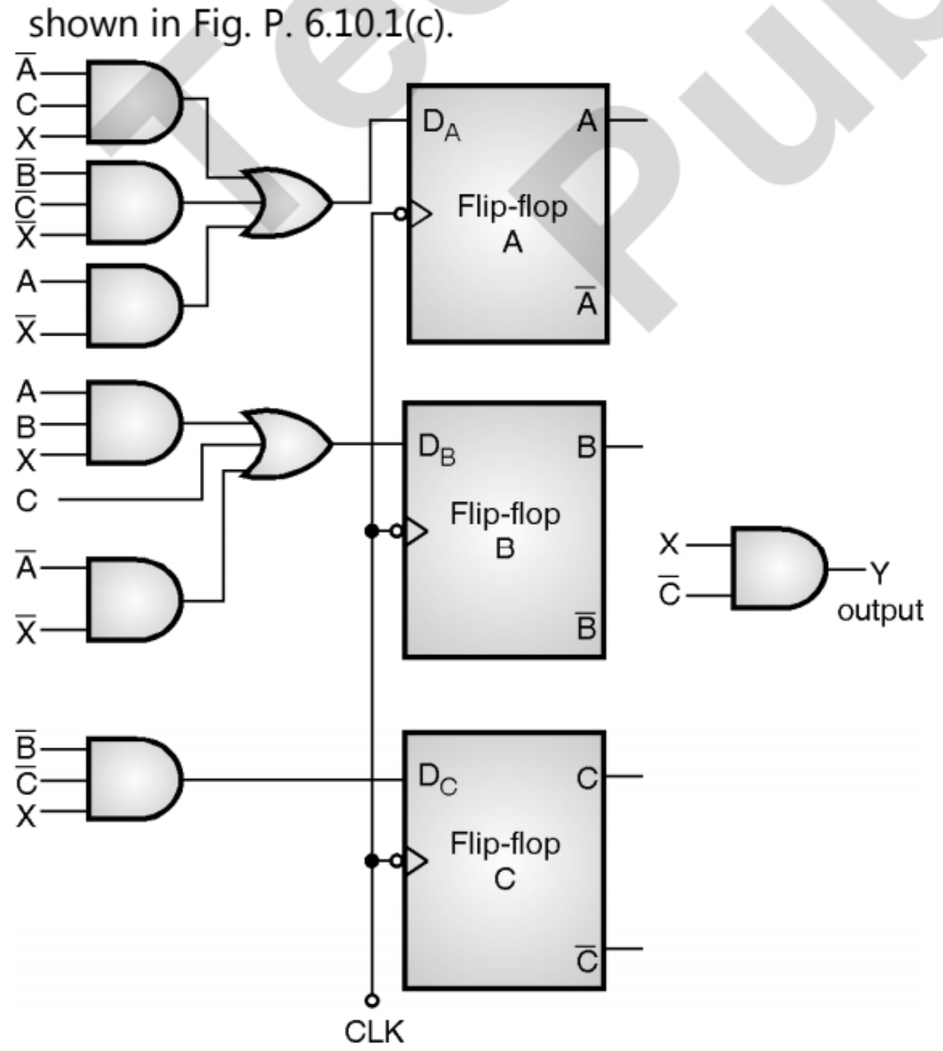


(C-689) Fig. P. 6.10.1(b) : K-maps and simplifications

- Note that these K-maps have been written from the circuit excitation table shown in Table P. 6.10.1(a). The inputs are A, B, C and X.
- But the excitation Table P. 6.10.1(a) does not contain all the possible combinations of A, B, C and X. So the missing combinations are treated as don't care conditions and denoted by X in the K maps. They correspond to 011, 101 and 111.

Step 3 : Logic diagram :

- The logic diagram of the required sequential circuit is shown in Fig. P. 6.10.1(c).



(C-690) Fig. P. 6.10.1(c) : Logic diagram of the required clocked sequential circuit

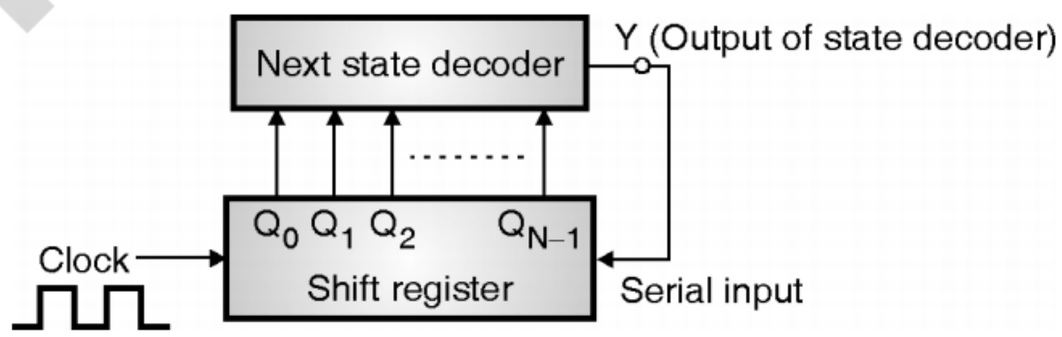
6.11 Sequence Generator :

Definition :

- A sequence generator is a sequential circuit which generates the prescribed sequence at its output. The output sequence is produced in synchronization with the clock input.
- It is possible to design a sequence generator using either counters or using the shift registers.

6.11.1 Sequence Generator using Shift Register :

- The sequence generator is a circuit which generates a desired sequence of bits at its output in synchronization with the clock.
- Some of the applications of the sequence generator are as follows :
  1. Random bit generator.
  2. Counters.
  3. Code generators.
  4. Period and sequence generator.
- Fig. 6.11.1 shows the basic structure of a sequence generator using a shift register.
- The outputs of an N bit shift register (Q_0 through Q_{N-1}) are applied as inputs to a combinational circuit called "Next state decoder".



(C-1511) Fig. 6.11.1 : Basic structure of a sequence generator

- And the output (Y) of the next state decoder is applied as the serial input to the shift register.
- The "Next state decoder" is designed by keeping in mind the required sequence.
- Design of a sequence generator using shift registers has been demonstrated in the following examples.

Ex. 6.11.1 : Design a sequence generator to generate the following sequence 1 0 1 1 0 ….

OR

Design **pulse train** generator using shift register to generate the following pulse : …..10110….

TechKnowledge Publications

Soln. :

(C-6242) Table P. 6.11.1 : State stable for the sequence generator

Number of clock pulse	Flip flop outputs			Decimal equivalent	
	Q_2	Q_1	Q_0		
1	1	0	1	5	same
2	0	1	0	2	
3	1	0	1	5	
4	1	1	0	6	
5	0	1	1	3	

Prescribed sequence delayed by 2 bits

Prescribed sequence delayed by 1 bit

Prescribed sequence

The length of given sequence = L = 5 bits.

1. The minimum number of flip flops (N) required to generate a sequence of length L is given by,
2. Length of the sequence $L = 2^N - 1$ ...(1)
3. Here L = 5. Therefore number of flip flops N = 3. Thus three flip-flops will have to be used.
4. Write the states of the circuit as shown in Table P. 6.11.1. Note that the column for Q_2 output consists of the prescribed sequence, whereas the column for Q_1 consists of 1 bit delayed version of the prescribed sequence and that for Q_0 consists of 2 bit delayed version.
5. The use of N = 3 i.e. three flip-flops will lead us to the correct generation of prescribed sequence, if and only if all the states in the state table are distinct. Otherwise we will have to increase the number of flip-flops.
6. In Table P. 6.11.1, all the states are not distinct. States 1 and 3 are same. Hence only three flip-flops are not sufficient. So assume N = 4 and prepare a new state table as Table P. 6.11.1(A) in a similar manner as the previous state table.

(C-6243) Table P. 6.11.1(A) : Modified state table using N = 4

Clock pulse number (state)	Flip flop outputs				Decimal equivalent
	Q_3	Q_2	Q_1	Q_0	
1	1	0	1	1	11
2	0	1	0	1	5
3	1	0	1	0	10
4	1	1	0	1	13
5	0	1	1	0	6

– Note that with 4 flip-flop, all the states are distinct states.

7. Design of next state decoder :

– We have to find the output (Y) of the next state decoder which is connected to the serial input of the shift register so that the state changes from present to next as per our requirements.

– For example if $Q_3\ Q_2\ Q_1\ Q_0$ = 1 0 1 1 i.e. (11 decimal). Then what should be the serial input so as to get $Q_3\ Q_2\ Q_1\ Q_0$ = 0 1 0 1 (i.e. 5 decimal) ? The answer is, serial input should be "0" (right shifting is assumed).

– Applying same logic to Y which is output of the next state decoder and serial input to the shift register we find Y for each state as shown in Table P. 6.11.1(B).

(C-6244) Table P. 6.11.1(B) : Output Y of the next state decoder

Number of clock pulse (state)	Flip flop outputs				Output Y	State (Decimal equivalent)
	Q_3	Q_2	Q_1	Q_0		
1	1	0	1	1	0	11
2	0	1	0	1	1	5
3	1	0	1	0	1	10
4	1	1	0	1	0	13
5	0	1	1	0	1	6

Valid states

– Note that the Y output of the next state decoder is applied to D_3 (input of FF – 3) and the shift register is operated in the right shift mode.

8. K-map and simplifications :

– The states 11, 5, 10, 13 and 6 shown in Table P. 6.11.1(B) are called as the stable states.

– So in the K-map corresponding to these states we will enter the corresponding value of Y (0 or 1 as per Table P. 6.11.1(B)).

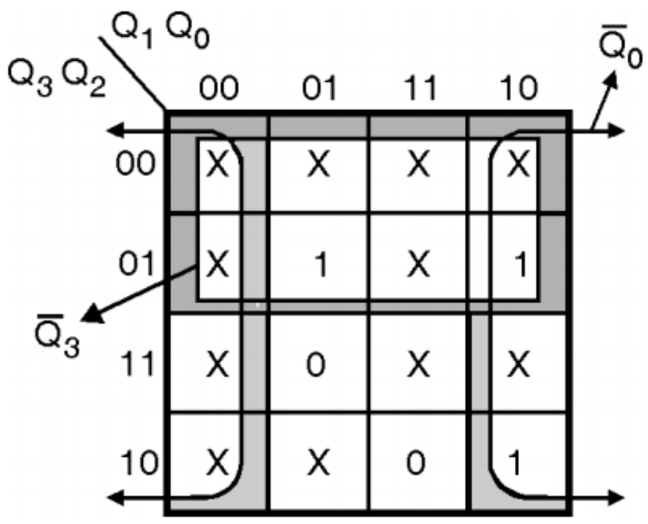
– In the remaining boxes of K-maps we should enter the don't care (X) conditions.

– The K-map is shown in Fig. P. 6.11.1(a).

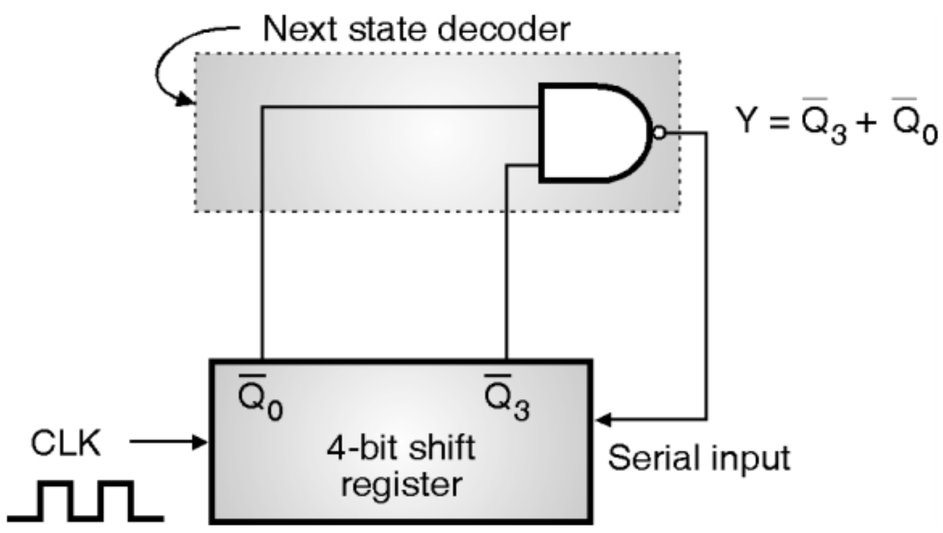
$$\therefore\ Y = \overline{Q_0} + \overline{Q_3}$$

9. Draw the logic diagram :

– The logic diagram is shown in Fig. P. 6.11.1(b).



(C-1513) Fig. P. 6.11.1(a) : K-map and simplifications



(C-1514) Fig. P. 6.11.1(b) : Logic diagram of the sequence generator

Ex. 6.11.2 : Design a sequence generator to generate the following pulse train using shift register

1 1 0 0 1 1 1 0.

Soln. :

Step 1 : Decide the number of flip-flops :

– The length of given sequence L = 8

$$L = 2^N - 1 \quad \text{where N = Number of flip-flops}$$

$$\therefore 8 = 2^N - 1$$

$$\therefore 2^N = 9 \qquad \therefore \mathbf{N = 4}$$

Step 2 : Write the state table for the sequence generator :

– The state table for the sequence generator is shown in Table P. 6.11.2(a).

(C-6245) Table P. 6.11.2(a) : State table for the generator

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

– Note that we have delayed the sequence in the opposite direction. Hence the directions of all the arrows have reversed.

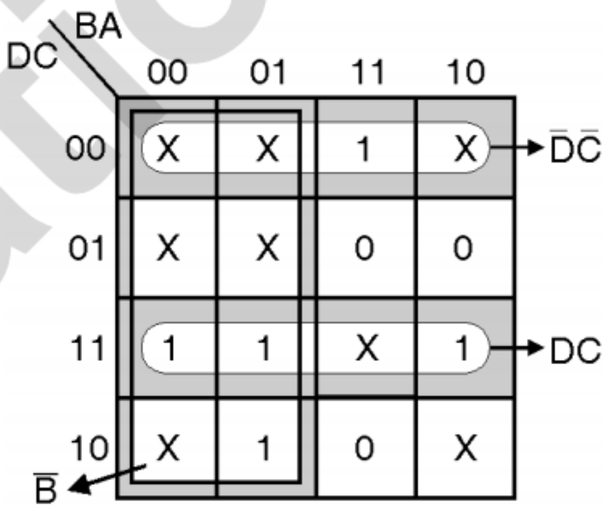
– All the states in Table P. 6.11.2(a) are distinct states. Hence 4 flip-flops will be sufficient.

Step 3 : Find Y :

– Refer Table P. 6.11.2(b) which shows the values of Y i.e. the output of the next state generator.

(C-6246) Table P. 6.11.2(b) : State table to obtain Y

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



(C-1515) Fig. P. 6.11.2(a) : K-map and simplification

Step 4 : K-map and simplifications :

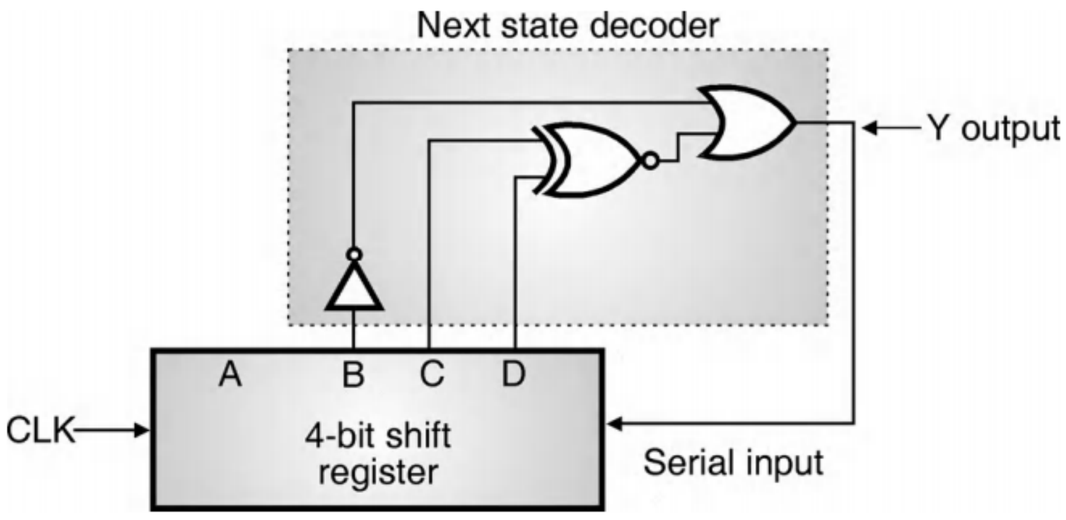
– The K-map for output Y is as shown in Fig. P. 6.11.2(a).

$$\therefore Y = \overline{B} + DC + \overline{D}\,\overline{C}$$

$$\therefore Y = \overline{B} + (\overline{D \oplus C})$$

Step 5 : Draw the logic diagram :

– The logic diagram of the sequence generator is shown in Fig. P. 6.11.2(b).



(C-1516) Fig. P. 6.11.2(b) : Logic diagram of the required sequence generator

TechKnowledge Publications

Ex. 6.11.3 : Design a sequence generator for the sequence 1010 using shift register.

May 18, 6 Marks

Soln. :

Step 1 : Decide the number of flip flops :

The length of given sequence is L = 4

$$L = 2^N - 1 \text{ where N = Number of flip flops.}$$

$$\therefore 4 = 2^N - 1$$

$$2^N = 5$$

$$\therefore N = 2$$

$\therefore$ We use three flip flops.

Step 2 : Writ the state table for the sequence generator :

– The state table for the sequence generator is shown in Table P. 6.11.3(a).

(C-7186) Table P. 6.11.3(a) : State table for the generator

State	C	B	A	Decimal equivalent
1	1	0	1	5
2	0	1	0	2
3	1	0	1	5
4	0	1	0	2

Step 3 : Find Y :

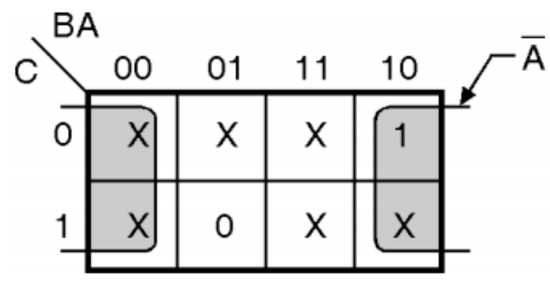
– Refer Table P. 6.11.3(b) which shows the values of Y i.e. the output of the next state generator.

(C-7187) Table P. 6.11.3(b) : State table to obtain Y

State	C	B	A	Y
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0
4	0	1	0	1

Step 4 : k-map and simplification :

– The k-map for output Y is as shown in Fig. P. 6.11.3(a).

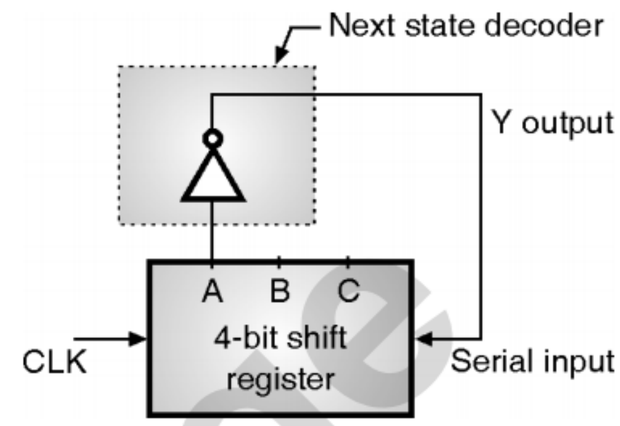


(C-7188) Fig. P. 6.11.3(a) : k-map and simplification

$$\therefore Y = \bar{A}$$

Step 5 : Draw the logic diagram :

– The logic diagram of the sequence generator is shown in Fig. P. 6.11.3(b).



(C-7189) Fig. P. 6.11.3(b) : Logic diagram of the sequence generator

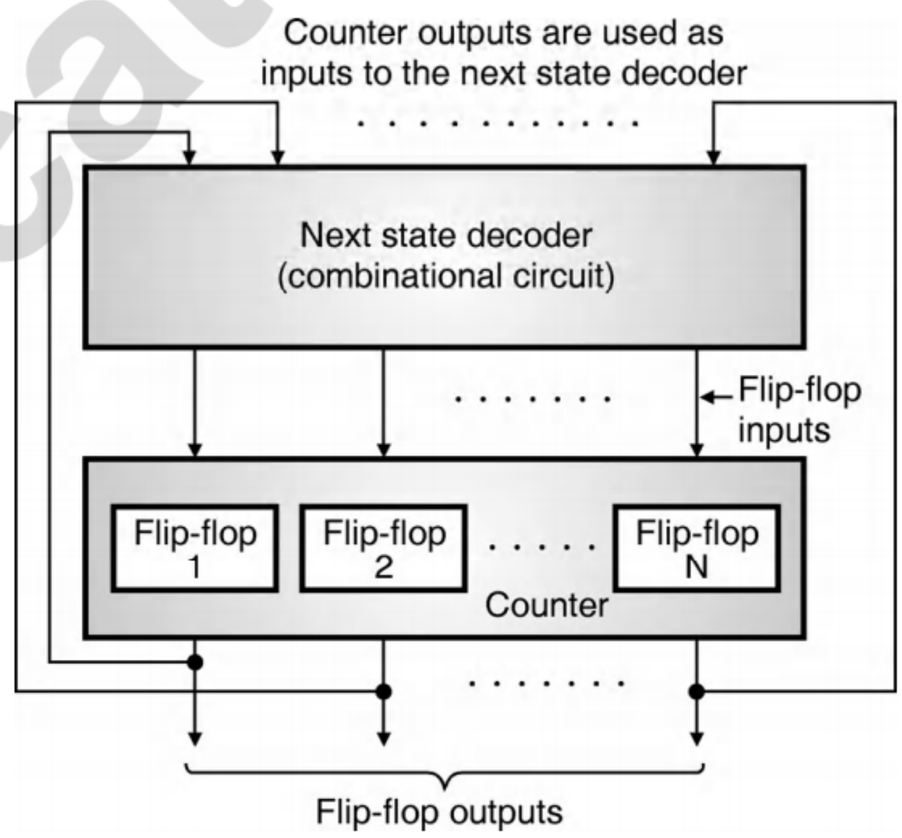
6.11.2 Sequence Generator using Counters :

Block diagram :

– The general block diagram of a sequence generator constructed using counter is shown in Fig. 6.11.2.

– The next state decoder is a combinational circuit.

– The inputs to it are obtained from the flip-flop outputs and its outputs are applied to the inputs of the flip-flops as shown in Fig. 6.11.2.



(C-857) Fig. 6.11.2 : A general block diagram of a sequence generator

Deciding the number of flip-flops :

– We will be given the required sequence of 1's and 0's that is to be generated such as 1011011 ....

– Based on the given sequence we have to decide the number of flip-flops as follows :

1. Count the number of 1's and 0's in the given sequence.
2. Choose the higher number of the two. Let this number be N.

TechKnowledge Publications

3. The number of flip flops (n) is then calculated as, $N \le 2^{n-1}$

– Let us apply this principle to the given sequence, i.e. 1011011 ...

– Number of 1's = 5, number of 0's = 2. So select higher one of them i.e. 5. So N = 5.

– But $N = 2^{n-1}$ Hence $5 \le 2^{n-1}$

$\therefore$ n = 4 ... so four FFs will be required.

Ex. 6.11.4 : For the following sequence, design a sequence generator using JK flip-flops.
1 0 0 1 0 0 1.

Soln. :

Step 1 : Decide the number of flip-flops :

– Number of 1's = 3 and number of 0's = 4.

– So select higher one of them i.e. 4. So N = 4.

– Hence $N = 2^{n-1}$, so $4 \le 2^{n-1}$

– Therefore number of flip-flops n = 3.

Step 2 : Write the state table :

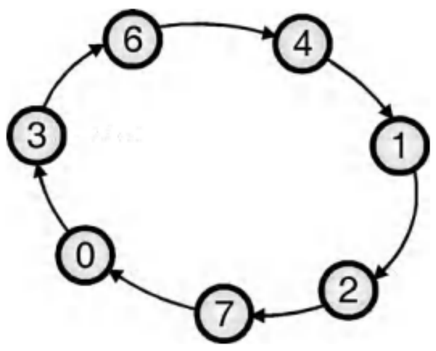
– Assume that the required sequence is obtained at the output of flip-flop C. The state assignment for the other outputs is shown in Table P. 6.11.4(a).

(C-6247) Table P. 6.11.4(a)

Flip flop outputs			State
C	B	A	
1	0	0	4
0	0	1	1
0	1	0	2
1	1	1	7
0	0	0	0
0	1	1	3
1	1	0	6

These entries have been made in an arbitrary manner

This is the given sequence



(C-858) Fig. P. 6.11.4(a) : State diagram

– There is only one unused state i.e. 5. We assume that the next state of 5 is don't care XXX.

Step 3 : Draw the state diagram :

From the state table, draw the state diagram as shown in Fig. P. 6.11.4(a).

Step 4 : Write the excitation table :

The excitation table of a JK FF is given in Table P. 6.11.4(b) and the circuit excitation table is shown in Table P. 6.11.4(c).

(C-7761) Table P. 6.11.4(b) : Excitation table of JK FF

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

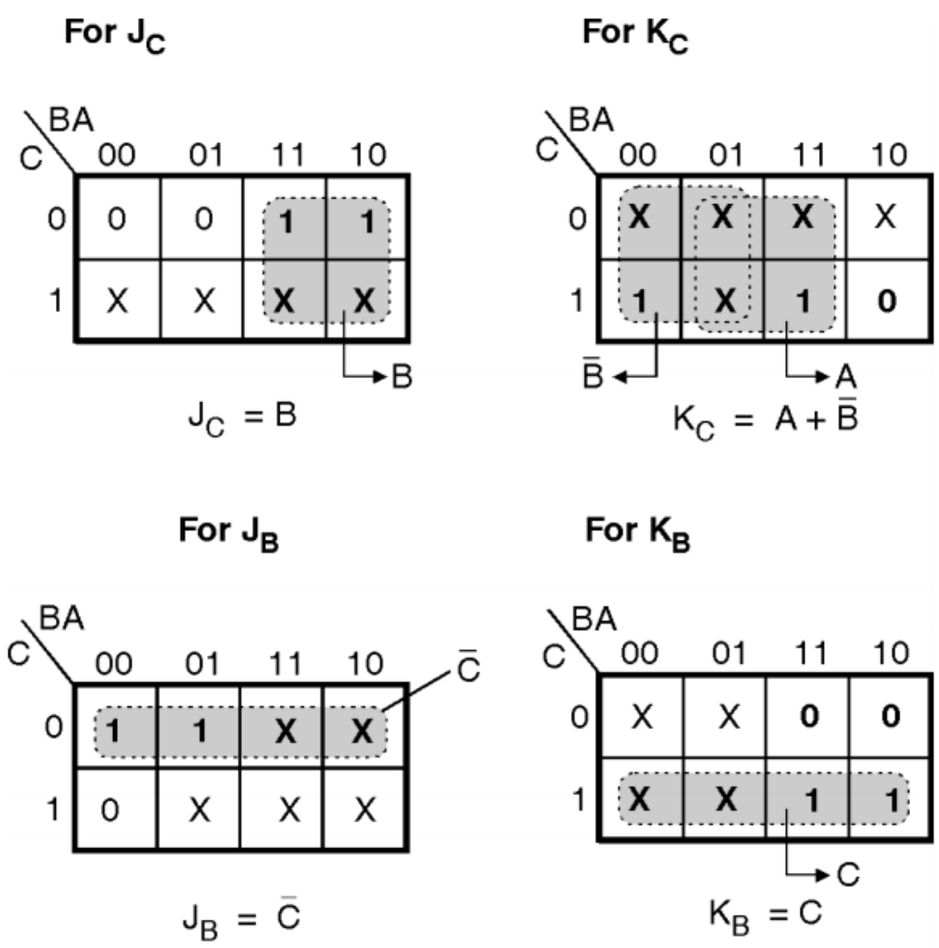
(C-6398) Table P. 6.11.4(c) : Circuit excitation table

Present state			Next state			Flip-flop inputs					
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	1	0	X	1	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	1	1	1	X	X	0	1	X
0	1	1	1	1	0	1	X	X	0	X	1
1	0	0	0	0	1	X	1	0	X	1	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	1	0	0	X	0	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

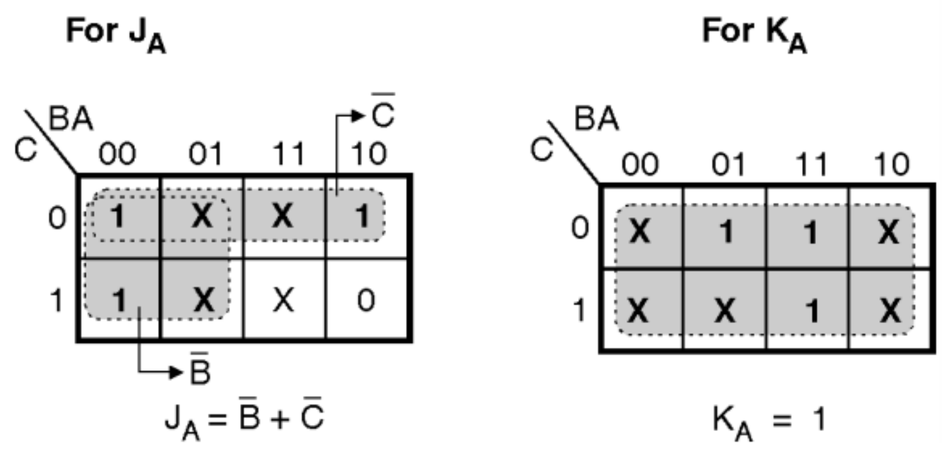
← The next state and FF inputs are don't care for the unused state.

Step 5 : K-maps and simplified Boolean expressions :

– Fig. P. 6.11.4(b) shows the K-maps and corresponding simplified boolean expressions for all the FF inputs.



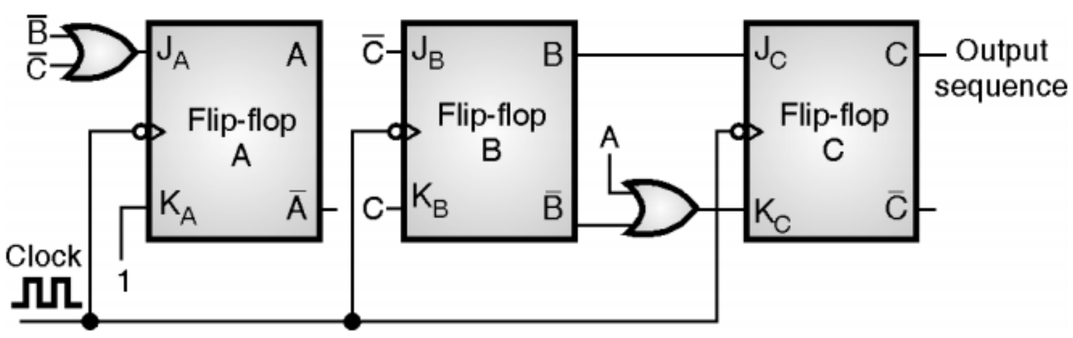
(C-859) Fig. P. 6.11.4(b) (Contd...)



(C-859) **Fig. P. 6.11.4(b) : K-map and simplification**

Step 6 : Draw the logic diagram :

– The logic diagram is shown in Fig. P. 6.11.4(c).



(C-860) **Fig. P. 6.11.4(c) : Logic diagram of the required sequence generator**

– Note that the required sequence is obtained at the C output.

– All the flip-flops are clocked simultaneously.

Ex. 6.11.5 : Design a sequence generator using T FFs.

(C-861) **Fig. P. 6.11.5(a) : Given state diagram**

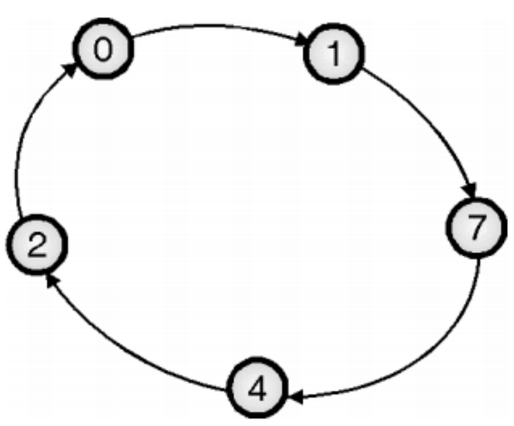
Draw the bush diagram to check for the lockout conditions.

Soln. :

– We shall assume that the next state of each unused state is a don't care condition, while writing the circuit excitation table.

Step 1 : Draw the state diagram :

– The state diagram is as shown in Fig. P. 6.11.5(b).



(C-862) **Fig. P. 6.11.5(b) : State diagram**

Step 2 : Number of FFs :

– Since the highest state is "7" i.e. 111, we have to use three T flip-flops.

Step 3 : Write the excitation table :

– Table P. 6.11.5(a) shows the excitation table for a T FF.

– Table P. 6.11.5(b) shows the circuit excitation table.

(C-7842) **Table P. 6.11.5(a) : Excitation table for a T FF**

Present state Q_n	Next state Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(C-6248) **Table P. 6.11.5(b) : Circuit excitation table**

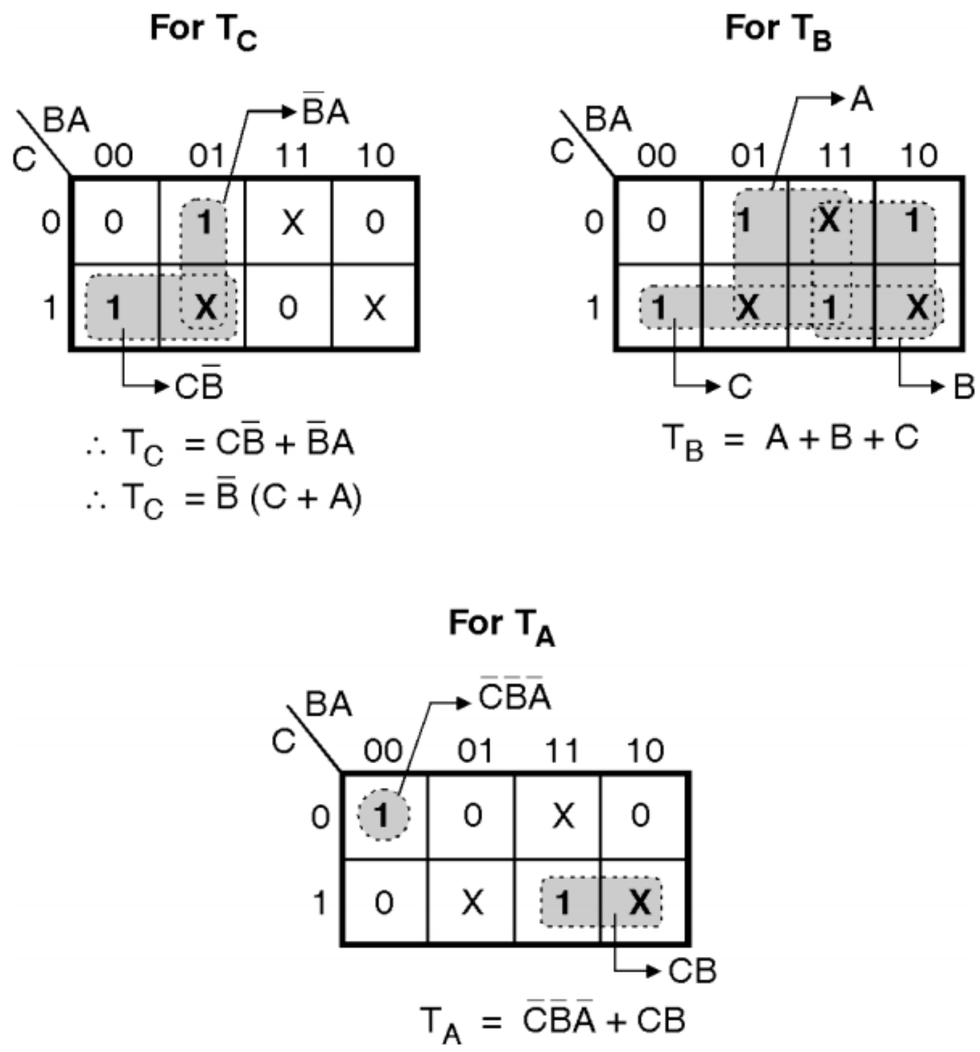
Present state			Next state			Flip-flop inputs		
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	1	1	1	1	1	0
0	1	0	0	0	0	0	1	0
0	1	1	X	X	X	X	X	X
1	0	0	0	1	0	1	1	0
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	1	0	0	0	1	1

Next state of each unused state is don't care. We can assume don't care conditions to be logic 1's

FF inputs are don't care for the unused inputs

Step 4 : K-maps and simplifications :

– Refer Fig. P. 6.11.5(c) for the K-maps and simplified boolean expressions.

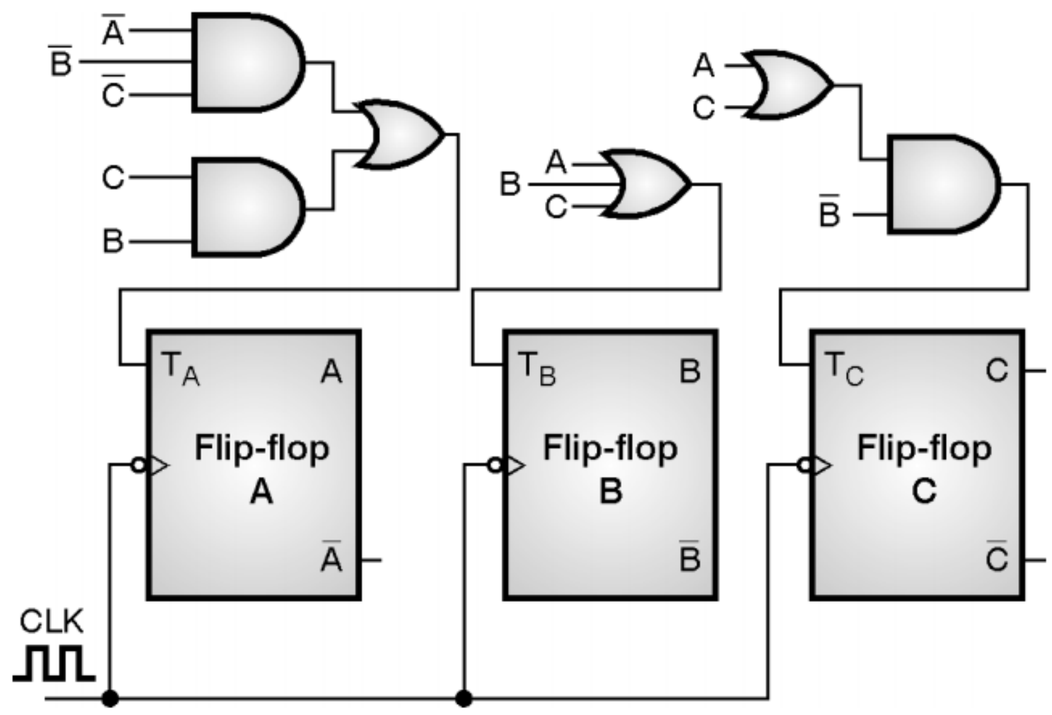


(C-863) **Fig. P. 6.11.5(c) : K-maps and simplifications**

TechKnowledge Publications

Step 5 : Logic diagram :

– Fig. P. 6.11.5(d) shows the required logic diagram.



(C-864) Fig. P. 6.11.5(d) : Logic diagram of the required counter

Step 6 : Examine the counter action for the unused states :

– The unused states are 3, 5 and 6.

For state 3 :

C = 0, B = 1, A = 1.

(C-6402) Table P. 6.11.5(c)

Present state			FF inputs			Next state			
C	B	A	$T_C = \bar{B}(C+A)$	$T_B = A+B+C$	$T_A = \bar{C}\bar{B}\bar{A} + CB$	C_{n+1}	B_{n+1}	A_{n+1}	
0	1	1	0	1	0	0	0	1	→ Valid state
1	0	1	1	1	0	0	1	1	
1	1	0	0	1	1	1	0	1	

– Thus if counter goes to state 3 it returns to state 1 which is a valid state.

For state 5 :

– Refer Table P. 6.11.5(c). The next state of 5 is 3 which ultimately terminates into 1.

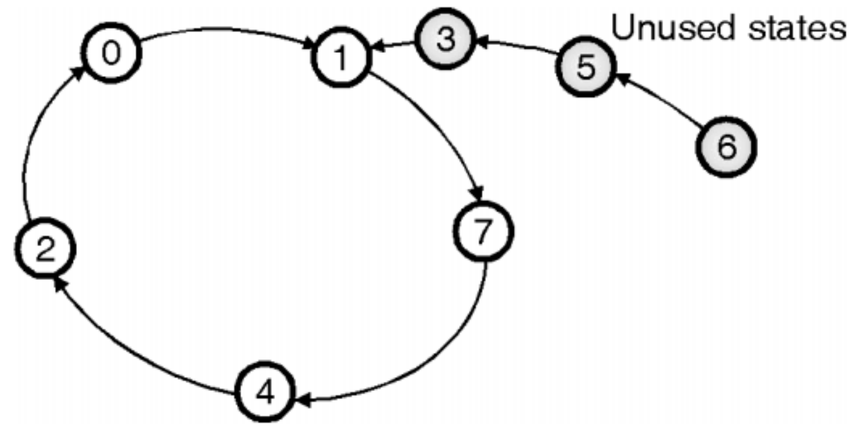
For state 6 :

– Refer Table P. 6.11.5(c). The next state of 6 is 5 and that of 5 is 3 which ultimately gets terminated into 1 which is a valid state.

Step 7 : Draw the bush diagram :

– The bush diagram is shown in Fig. P. 6.11.5(e).

– From the bush diagram it is evident that lockout condition will be avoided.



(C-865) Fig. P. 6.11.5(e) : Bush diagram

Ex. 6.11.6 : Design binary sequence generator to generate binary sequence '11010' using JK flip-flops. How to avoid lockout condition in designed sequence generator ?

Dec. 11, 10 Marks

Soln. :

Step 1 : Decide the number of flip-flops :

Number of 1's = 3 and number of 0's = 2

$\therefore$ N = 3, Hence $N \le 2^{n-1}$

$\therefore$ $3 \le 2^{3-1}$

$\therefore$ Number of flip-flops n = 3.

Step 2 : Write the state table :

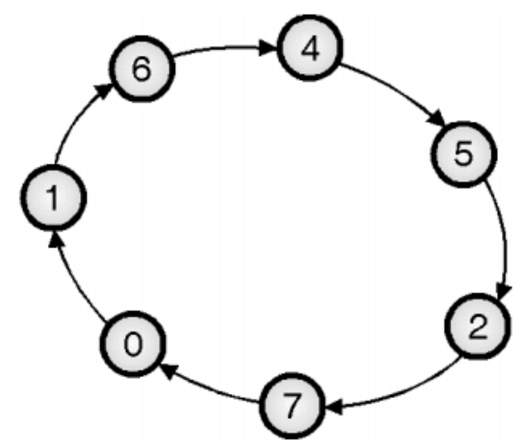
– Assume that the required sequence is obtained at the output of flip-flop C. The state assignment for the other outputs is shown in Table P. 6.11.6.

(C-6249) Table P. 6.11.6

Flip flop outputs			State
C	B	A	
1	0	0	4
1	0	1	5
0	1	0	2
1	1	1	7
0	0	0	0
1	0	1	1
1	1	0	6

These entries have been made in an arbitrary manner

This is the given sequence



(C-1970) Fig. P. 6.11.6 : State diagram

TechKnowledge Publications

– There is only one unused state i.e. 3. We assume that the next state of 3 is don't care XXX.

Step 3 : Draw the state diagram :

– From the state table, draw the state diagram as shown in Fig. P. 6.11.6.

Step 4 : Write the excitation table :

– The excitation table of a JK FF is given in Table P. 6.11.6(a) and the circuit excitation table is shown in Table P. 6.11.6(b).

(C-7761)Table P. 6.11.6(a) : Excitation table of JK FF

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

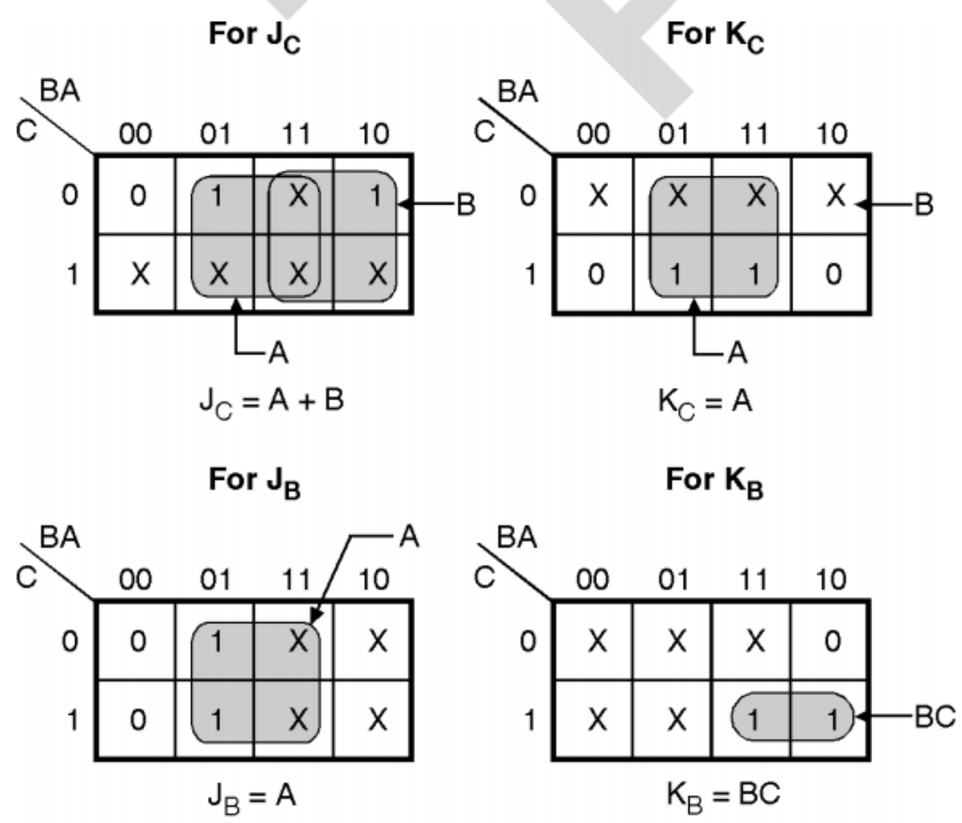
(C-6403)Table P. 6.11.6(b) : Circuit excitation table

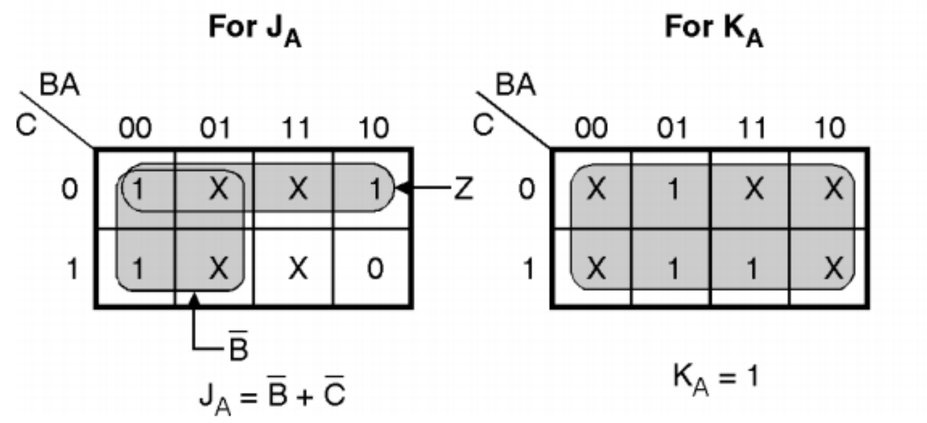
Present state			Next state			Flip-flop inputs					
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	1	0	1	X	1	X	X	1
0	1	0	1	1	1	1	X	X	0	1	X
0	1	1	X	X	X	X	X	X	X	X	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	1	0	X	1	1	X	X	1
1	1	0	1	0	0	X	0	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

← The next state and FF inputs are don't care for the unused state.

Step 5 : K-maps and simplified Boolean expressions :

– Fig. P. 6.11.6(a) shows the K-maps and corresponding simplified boolean expressions for all the FF inputs.

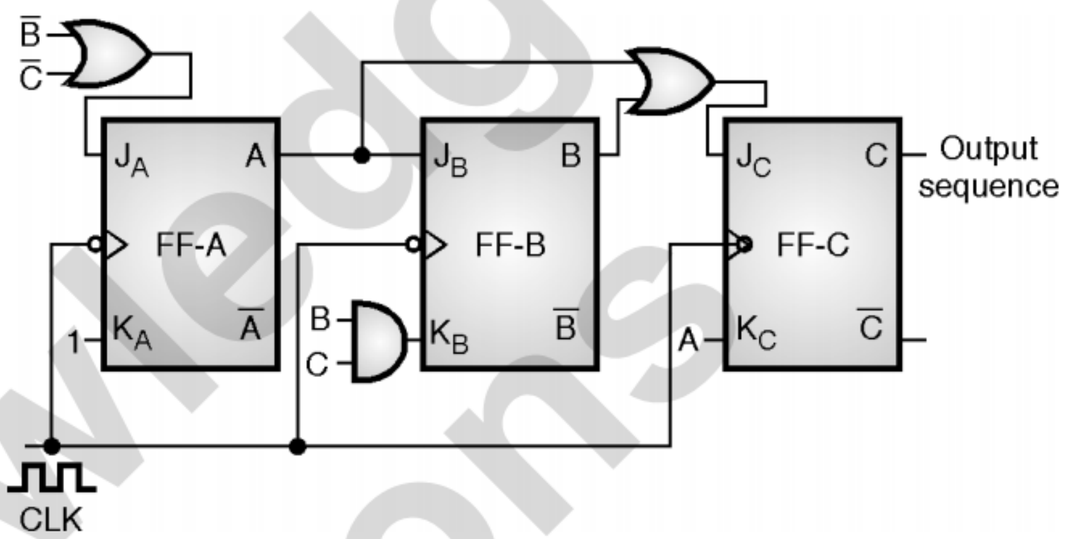




(C-1971) Fig. P. 6.11.6(a)

Step 6 : Draw the logic diagram :

– The logic diagram is as shown in Fig. P. 6.11.6(b).



(C-1972) Fig. P. 6.11.6(b) : Logic diagram of the required sequence generator

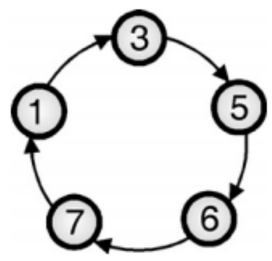
Ex. 6.11.7 : Design given sequence generator using J-K FF. Sequence is 1 → 3 → 5 → 6 → 7 → 1

Dec. 14, 6 Marks

Soln. :

Step 1 : Draw the state diagram :

– The state diagram is as shown in Fig. P. 6.11.7(a).



(C-4918) Fig. P. 6.11.7(a)

Step 2 : Number of FFs :

– Since the highest state is "7". i.e. 111 we have to use three JK flip flops.

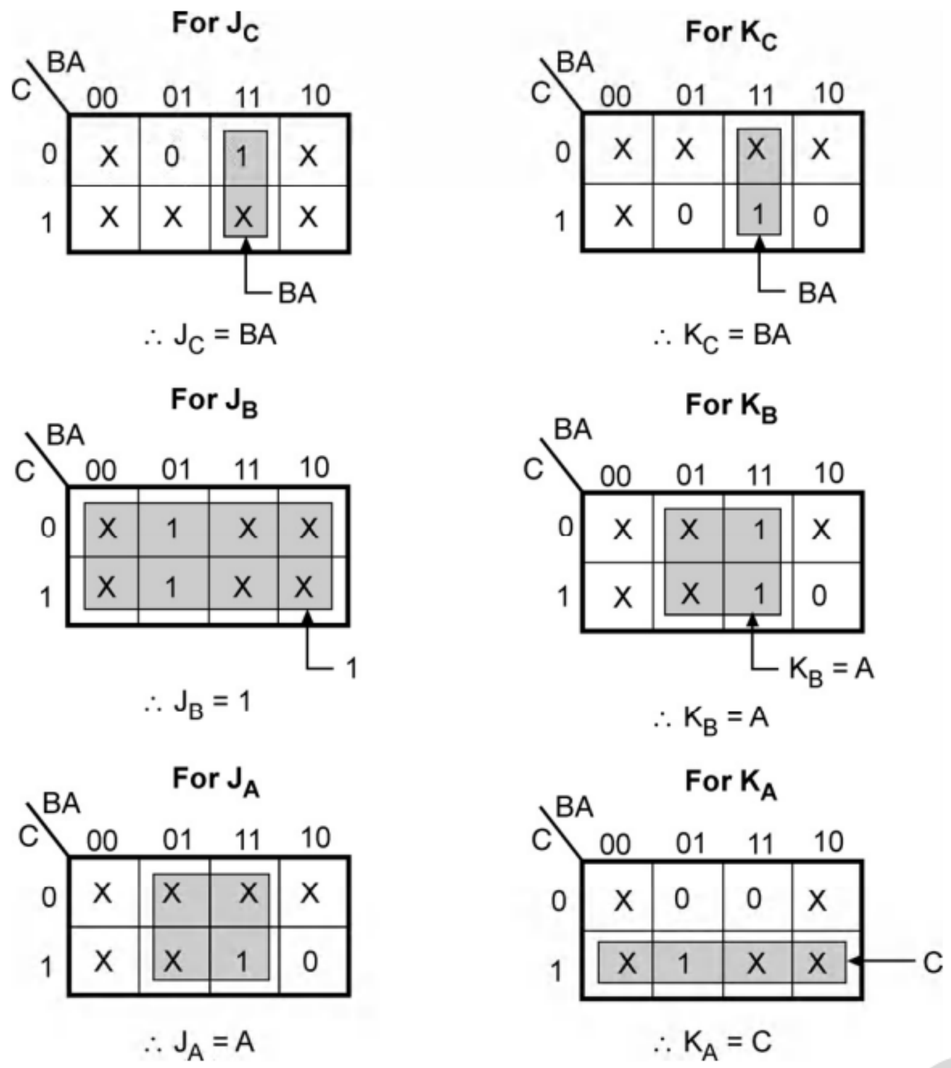
Step 3 : Write the excitation table :

(C-8123) Table P. 6.11.7 : Circuit excitation table

Present state			Next state			Flip flop inputs					
C	B	A	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	X	X	X	X	X	X	X	X	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	X	X	X	X	X	X	X	X	X
0	1	1	1	0	1	1	X	X	1	X	0
1	0	0	X	X	X	X	X	X	X	X	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	1	X	1	X	1	0	X

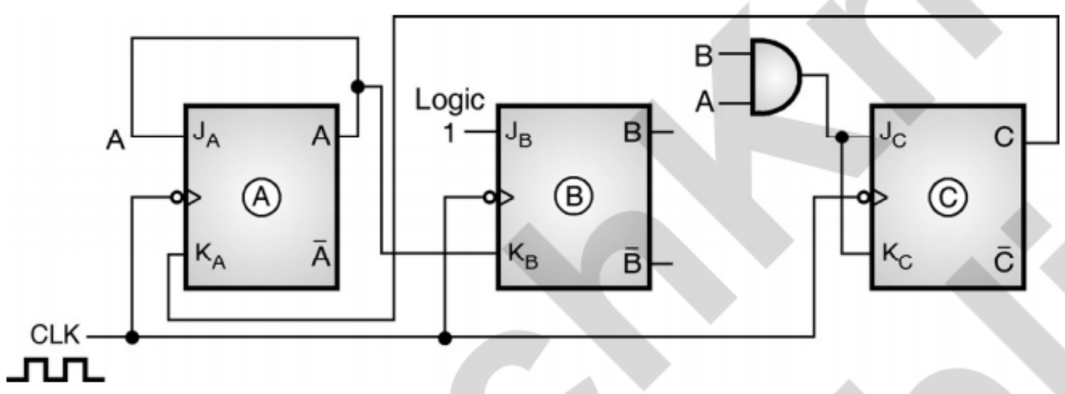
TechKnowledge Publications

Step 4 : K-maps and simplification : **(C-4919)**



Step 5 : Draw the logic diagram :

The logic diagram is as shown in Fig. P. 6.11.7(b).



(C-4920) **Fig. P. 6.11.7(b) : Logic diagram**

6.12 Sequence Detector :

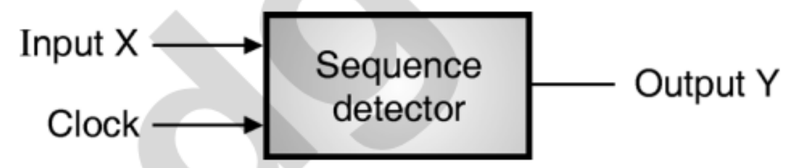
Definition :

– **A sequence detector** is a synchronous FSM which is designed to detect a specific sequence of bits arriving at its input.

– The detector will produce a logic "1" at its output (Y) as soon as it detects the specified sequence of bits. This concept is illustrated in Fig. 6.12.1(b).

– As shown in Fig. 6.12.1(b), in the long data string at the input (X), the desired input sequence may sometimes get overlapped.

– Fig. 6.12.1(a) shows the block diagram of a sequence detector.



(C-691) **Fig. 6.12.1(a) : Block schematic of a sequence detector**

– Now let us see how to design a sequence detector which is a logic circuit used to detect a desired sequence.

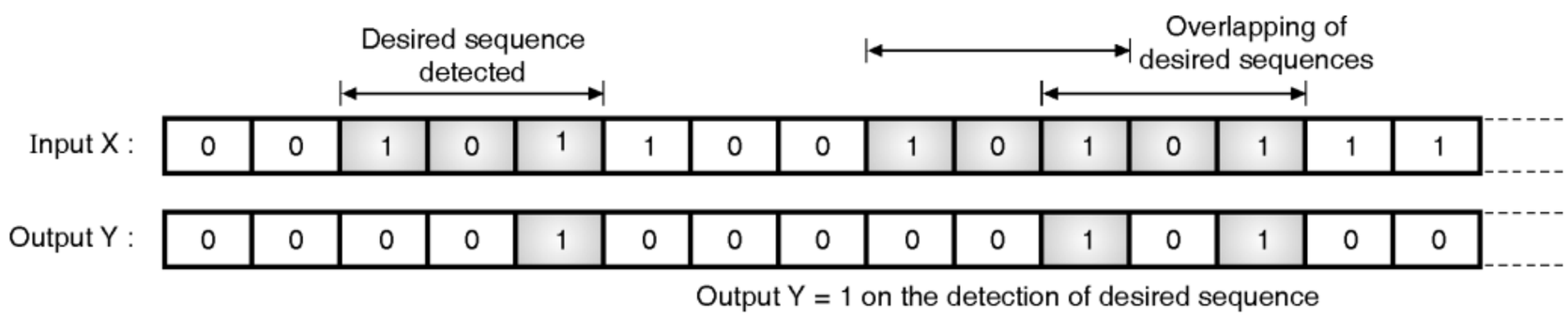
– The available or received sequence is applied at the input of the detector.

– It checks the sequence bit by bit. If required bit is at its input then the detector moves to the next state.

– Detector output will be equal to zero as long as the complete desired sequence is not detected.

– The output will be equal to 1 if the complete desired sequence has been detected.

– The sequence to be detected is given to us. Follow the steps given below to design the sequence detector.



(C-5986) **Fig. 6.12.1(b) : Principle of operation of a sequence detector**

Steps to design a sequence detector :

Step 1 :	A sequence to be detected is given to us.
Step 2 :	Develop the state diagram.
Step 3 :	Write the state table and circuit excitation table.

Step 4 : From the circuit excitation table write K-maps and obtain simplified equations.
Step 5 : Draw the logic diagram.

– The design of a sequence detector is demonstrated in the following example.

Ex. 6.12.1 : Design a sequence detector to detect three or more consecutive 1's in a string of bits coming through an input line.

May 06, 12 Marks, May 08, 6 Marks

Soln. :

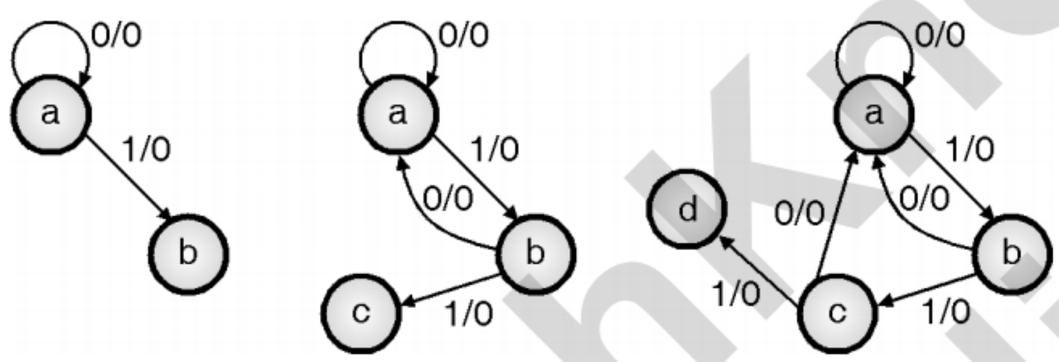
Step 1 : Develop the states diagram :

State "a" :

– Let the initial state be "a". If the input is 0 then the detector will stay in the same state.

– But if the input is equal to 1 then it will move to the next state "b" as shown in Fig. P. 6.12.1(a).

– For both the values of input the output remains to be equal to "0" since the complete sequence has not yet been detected.



(a) State "a" **(b) State "b"** **(c) State "c"**

(C-692) **Fig. P. 6.12.1 : Developed of state diagram**

State "b" :

– Let the detector be in state "b".

– If the next input is 1, then the circuit will move on to next state "c". This indicates the arrival of two consecutive 1's as shown in Fig. P. 6.12.1(b).

– But if the input is "0" then we go back to state "a". The output will be equal to "0" for both the values of input.

State "c" :

– If the next input is "1" then the circuit will move to next state "d" indicating the arrival of three consecutive 1's as shown in Fig. P. 6.12.1(c). Output will be zero.

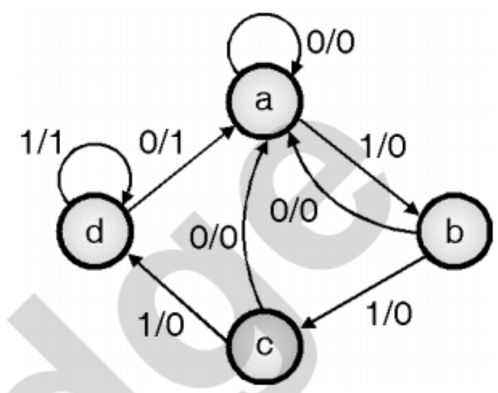
– But if the input is 0 then the circuit goes back to state "a" output will be equal to 0.

State "d" :

– If more 1's as detected, then the circuit stays in state "d", whereas if the next input is 0 then we should return to state "a".

– The complete state diagram is shown in Fig. P. 6.12.1(d).

– In this way the circuit stays at "d" as long as there are three or more consecutive 1's received.



(C-693) **Fig. P. 6.12.1(d) : Complete state diagram**

Step 2 : Assign binary codes to the various states :

– Let the binary code assignment for the states a to d be as shown in Table P. 6.12.1(a).

(C-8119) **Table P. 6.12.1(a) : State assignment table**

State name	Assigned binary state
a	00
b	01
c	10
d	11

Step 3 : Write down the state table :

– Table P. 6.12.1(b) represents the state table for the sequence detector to be designed.

(C-6250) **Table P. 6.12.1(b) : State table for sequence detector**

Present state		Input	Next state		Output Y	
A	B	X	A_{n+1}	B_{n+1}		
0	0	0	0	0	0	Output is 0 because the desired sequence is yet to be detected
0	0	1	0	1	0	
0	1	0	0	0	0	
0	1	1	1	0	0	
1	0	0	0	0	0	
1	0	1	1	1	0	
1	1	0	0	0	1	Output is 1 because the desired sequence has been detected
1	1	1	1	1	1	

Step 4 : Decide the type of flip-flop and number of flip-flops :

– Since the number of states is 4, we have to use 2 flip-flops A and B.

– Let us choose the D flip-flops for implementing the detector.

Step 5 : Write the excitation table :

– The circuit excitation table is shown in Table P. 6.12.1(d) and the excitation table for a D flip-flop is shown in Table P. 6.12.1(c).

TechKnowledge Publications

(C-7828) Table P. 6.12.1(c) : Excitation table of a D FF

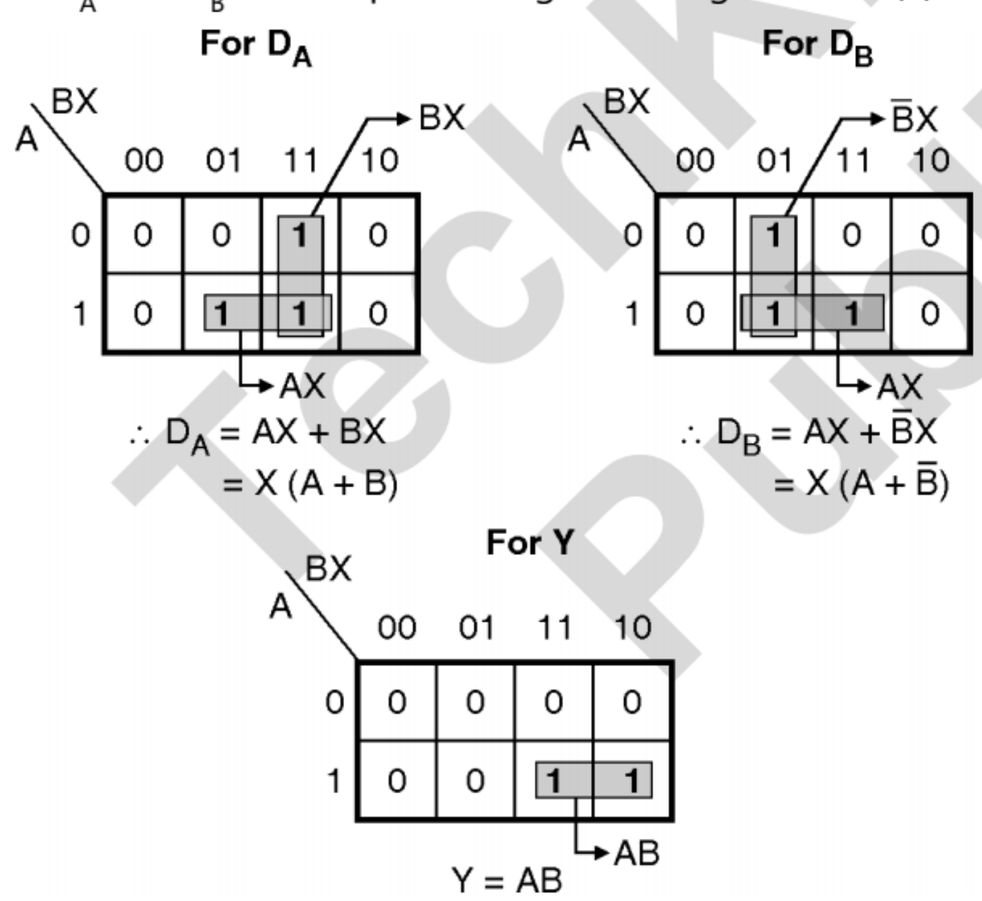
Q output		Input
Present state	Next state	D_n
0	0	0
0	1	1
1	0	0
1	1	1

(C-8120) Table P. 6.12.1(d) : Circuit excitation table

Present state		Input X	Next state		Output Y	FF inputs	
A	B		A_{n+1}	B_{n+1}		D_A	D_B
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	1

Step 6 : Write K-maps and simplify :

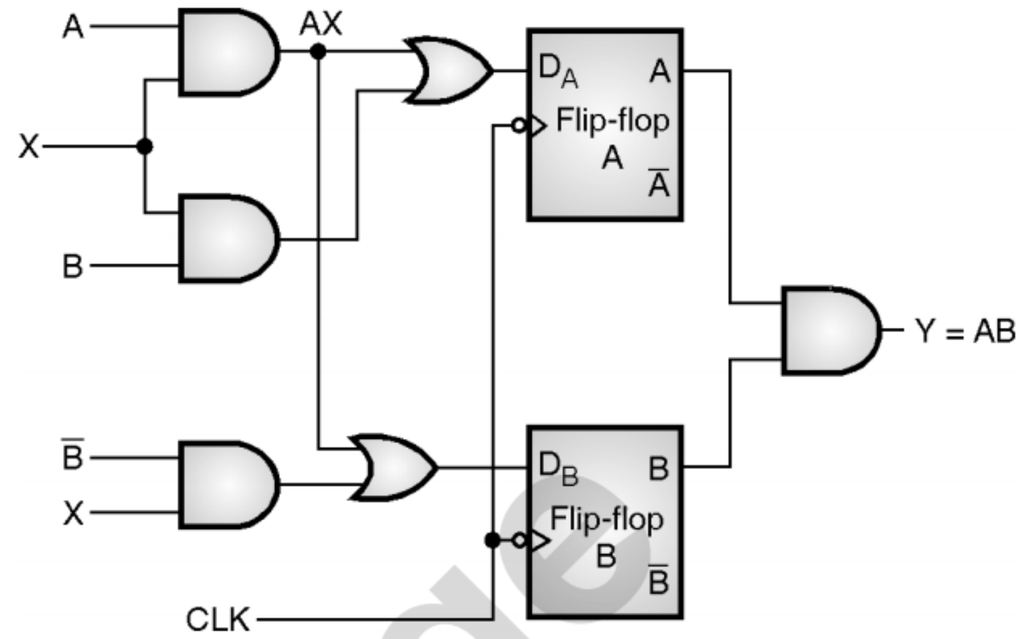
– The K-maps and simplified equations for the FF inputs D_A and D_B and output Y are given in Fig. P. 6.12.1(e).



(C-694) Fig. P. 6.12.1(e) : K-maps and simplifications

Step 7 : Draw the logic diagram :

– The logic diagram of the required sequence detector is shown in Fig. P. 6.12.1(f).



(C-695) Fig. P. 6.12.1(f) : Logic diagram of the required sequence detector

Ex. 6.12.2 : Design a sequence detector to detect a sequence …..101….. using D-flip-flop. (Use Moore Machine)

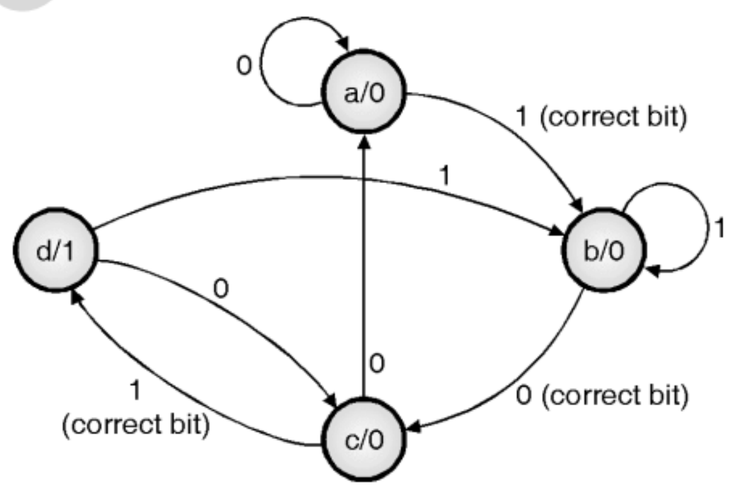
Soln. :

Step 1 : Draw the state diagram :

– In the Moore machine, the output depends only on the flip-flop outputs and not dependent on the input (x).

– Hence in the state diagram the output is written with the state as shown in Fig. P. 6.12.2.

– Let the initial state be "a".



(C-1565) Fig. P. 6.12.2 : State diagram

Step 2 : Write the state table :

(C-8095) Table P. 6.12.2 : State table

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	1
d	c	b	0	0

Step 3 : Assign states :

– The binary assignment for the states is as follows :

a = 00, b = 01, c = 10, d = 11

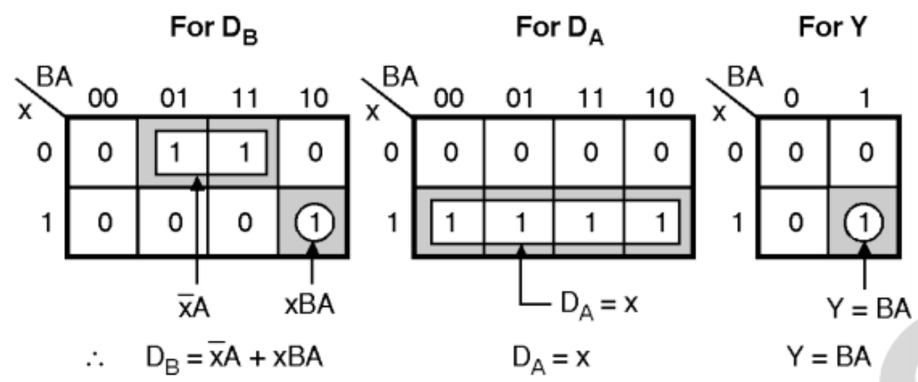
TechKnowledge Publications

Step 4 : Write the excitation table :

(C-8096) Table P. 6.12.2(a) : Excitation table

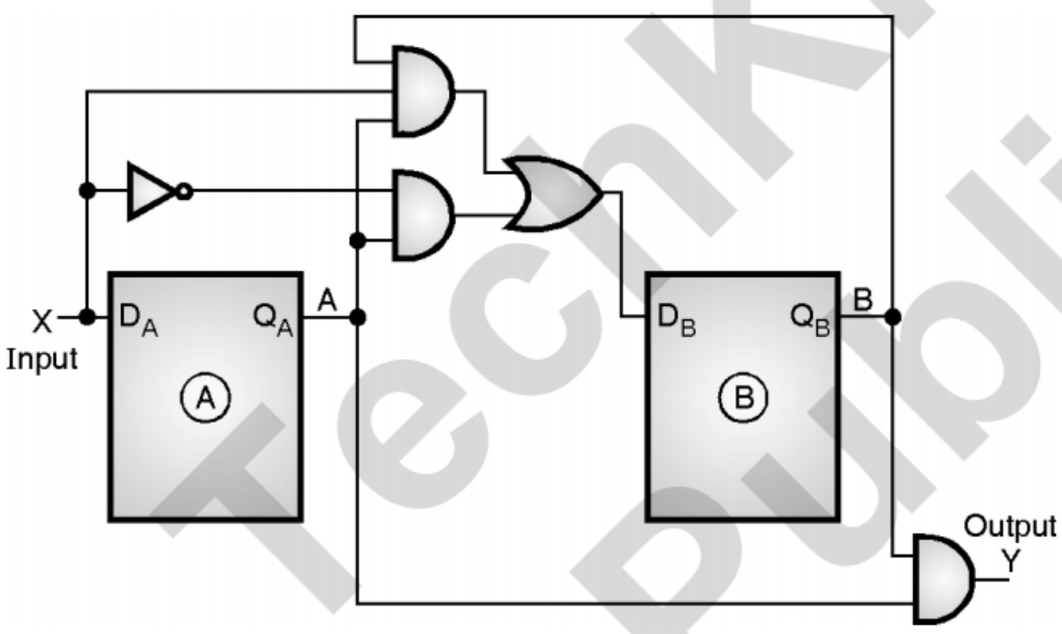
Input X	Present state B	Present state A	Next state B + 1	Next state A + 1	F/F inputs D_B	F/F inputs D_A	Output Y
0	0	0 (a)	0	0(a)	0	0	0
0	0	1 (b)	1	0(c)	1	0	0
0	1	0 (c)	0	0(a)	0	0	0
0	1	1 (d)	1	0(c)	1	0	0
1	0	0 (a)	0	1(b)	0	1	0
1	0	1 (b)	0	1(b)	0	1	0
1	1	0 (c)	1	1(d)	1	1	0
1	1	1 (d)	0	1(b)	0	1	1

Step 5 : K maps :



(C-1566) Fig. P. 6.12.2(a) : K-maps

Step 6 : Draw the logic diagram :



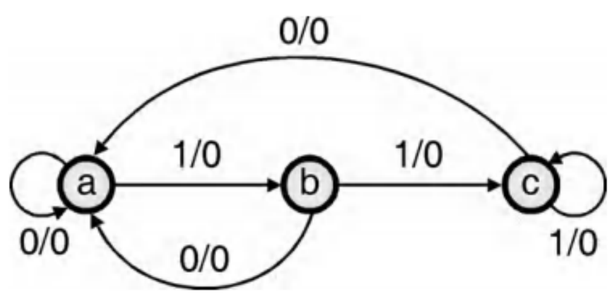
(C-1567) Fig. P. 6.12.2(b) : Logic diagram of sequence detector (Moore)

Ex. 6.12.3 : Design a sequence detector to detect the following sequence using D flip-flops. … 110 …

Soln. :

Step 1 : Draw the state diagram :

The state diagram is as shown in Fig. P. 6.12.3(a).



(C-696) Fig. P. 6.12.3(a) : State diagram

Step 2 : Write the state table :

(C-8093) Table P. 6.12.3(a) : State table

Present state	Next state X = 0	Next state X = 1	Output X = 0	Output X = 1
a	a	b	0	0
b	a	c	0	0
c	a	c	1	0

Step 3 : Assign states :

– The state assignment is carried out in the normal binary sequence as follows :

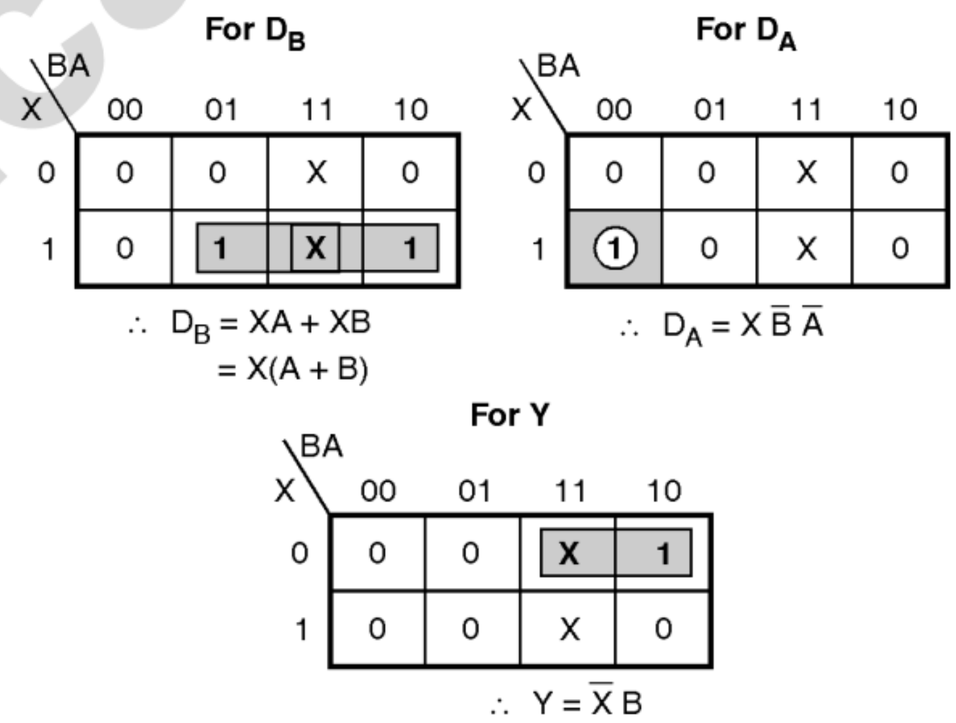
a = 00, b = 01, c = 10.

Step 4 : Write the circuit excitation table :

(C-8094) Table P. 6.12.3(b) : Circuit excitation table

Input X	Present state B	Present state A	Next state B + 1	Next state A + 1	F/F inputs D_B	F/F inputs D_A	Output Y
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	1
0	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0
1	0	1	1	0	1	0	0
1	1	0	1	0	1	0	0
1	1	1	X	X	X	X	X

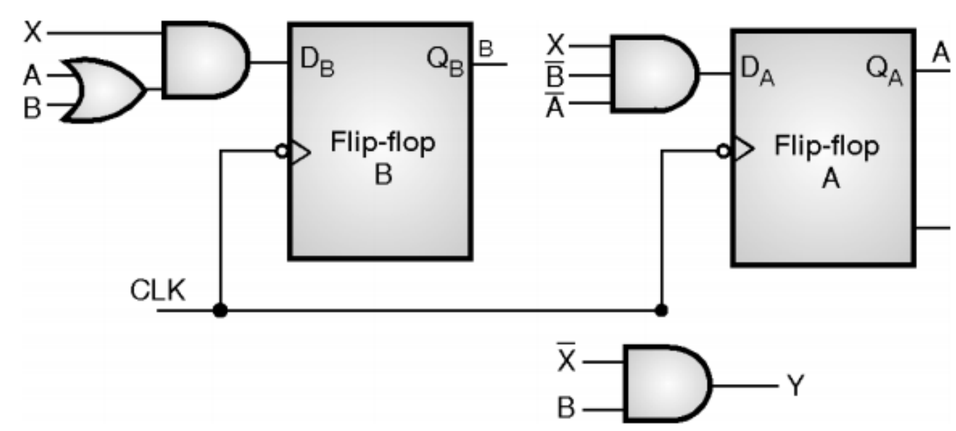
Step 5 : K-maps and simplifications :



(C-697) Fig. P. 6.12.3(b) : K-map and simplifications

Step 6 : Logic circuit :

– The logic circuit is shown in Fig. P. 6.12.3(c).



(C-698) Fig. P. 6.12.3(c) : Logic circuit of the detector

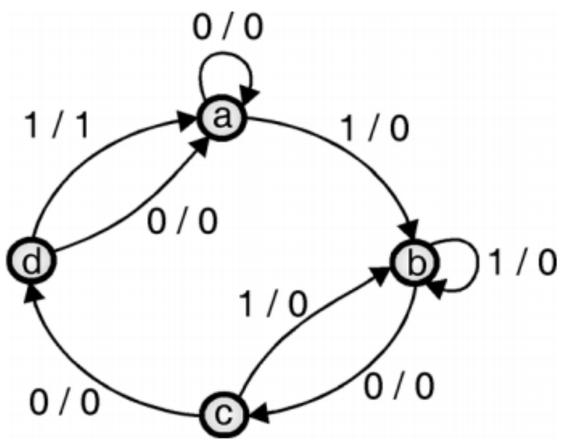
TechKnowledge Publications

Ex. 6.12.4 : Design a sequence detector using JK-flip-flops to detect the following sequence : …1 0 0 1 … Use state diagram, state transition tables and K-map as design tools. Remove all redundant states and draw the final circuit diagram.

Dec. 05, 12 Marks, Dec. 15, 6 Marks

Soln. :

Step 1 : To draw the state diagram :



(C-3615) **Fig. P. 6.12.4(a)**

Step 2 : To write the state transition table :

(C-8121) **Table P. 6.12.4(a) : State table**

Present state	Next state		Output Y	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	0
c	d	b	0	0
d	a	a	0	1

Step 3 : Assign states :

Let a = 00, b = 01, c = 10, d = 11.

Step 4 : To write the circuit excitation table :

– The excitation table for JK flip flop is as shown in Table P. 6.12.4(b).

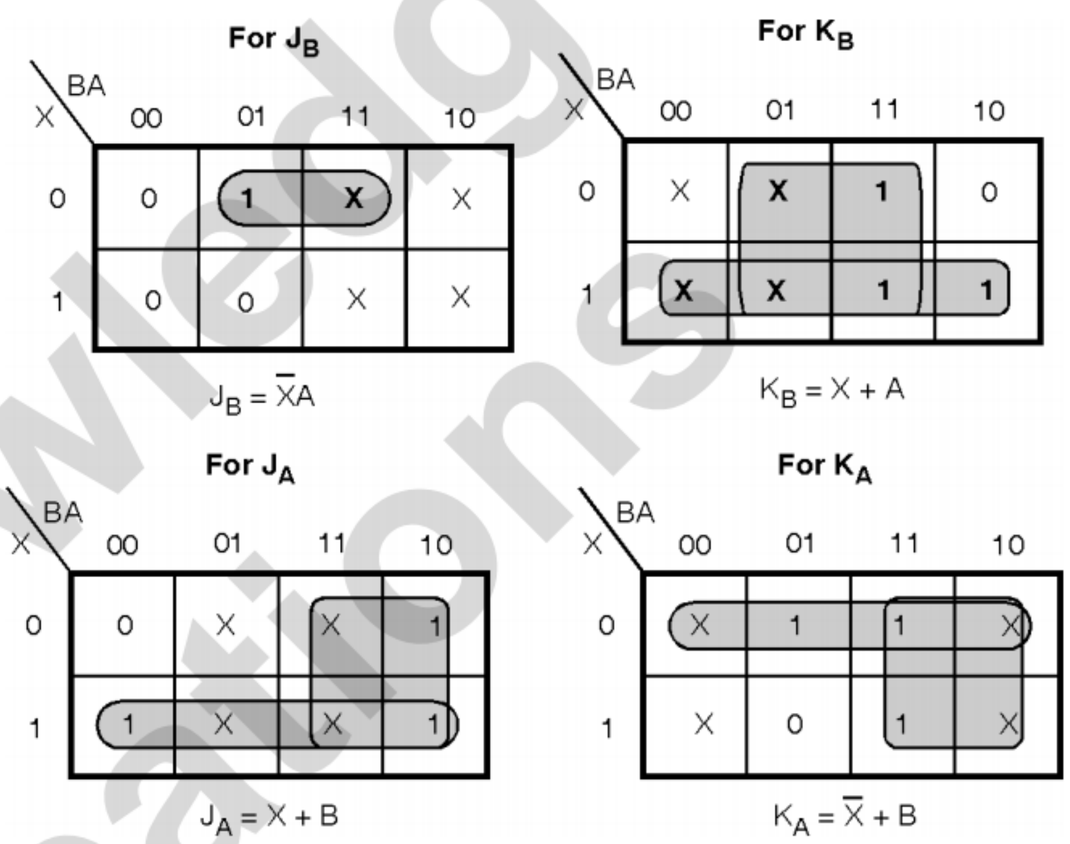
(C-7761) **Table P. 6.12.4(b) : Excitation table of JK flip-flop**

Present state Q_n	Next state Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(C-8122) **Table P. 6.12.4(c) : Circuit excitation table**

Input X	Present state B	Present state A	Next state B_{+1}	Next state A_{+1}	Flip flop inputs J_B	K_B	J_A	K_A	Output Y
0	0	0 (a)	0	0 (a)	0	X	0	X	0
0	0	1 (b)	1	0 (c)	0	X	X	1	0
0	1	0 (c)	1	1 (d)	X	0	1	X	0
0	1	1 (d)	0	0 (a)	X	1	X	1	0
1	0	0 (a)	0	1 (b)	0	X	1	X	0
1	0	1 (b)	0	1 (b)	0	X	X	0	0
1	1	0 (c)	0	1 (b)	X	1	1	X	0
1	1	1 (d)	0	0 (a)	X	1	X	1	1

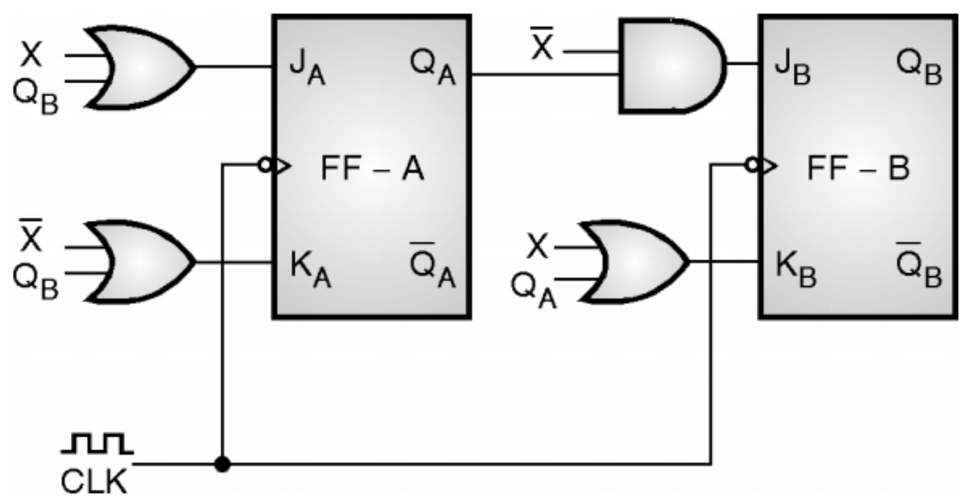
Step 5 : K-maps and simplification :



(C-3616) **Fig. P. 6.12.4(b)**

Step 6 : Logic diagram :

– The circuit diagram is as shown in Fig. P. 6.12.4(c).



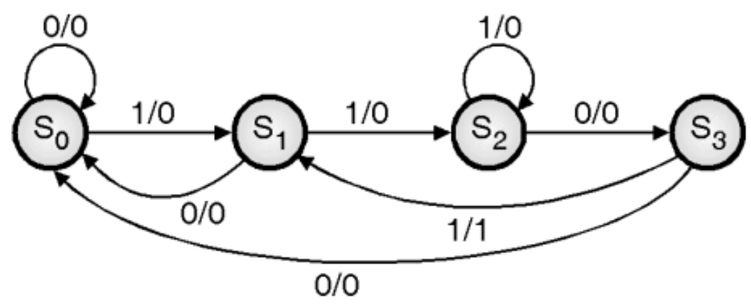
(C-3617) **Fig. P. 6.12.4(c) : Logic diagram**

Ex. 6.12.5 : Design sequence detector to detect the following sequence ……..1101……

May 12, May 17, 4 Marks

Soln. :

Step 1 : Draw the state diagram :



(C-3612) **Fig. P. 6.12.5(a) : State diagram**

Step 2 : Obtain state table :

– Assign binary value to states S_0 to S_3.

$$\therefore\ S_0 = 00,\ S_1 = 01,\ S_2 = 11,\ S_3 = 10$$

(C-8098) Table P. 6.12.5(a)

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

∴ State table will be as shown in Table P. 6.12.5(b).

(C-8099) Table P. 6.12.5(b)

Present state	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

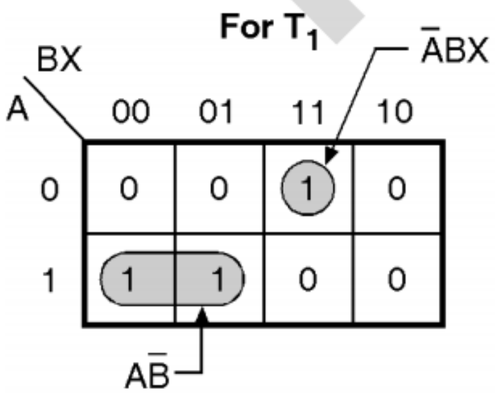
Step 3 : Obtain circuit excitation table :

(C-8100) Table P. 6.12.5(c)

Present state		Input X	Next state		Flip Flop inputs		Output Z
A	B		A_{n+1}	B_{n+1}	T_1	T_2	
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	0	1	0
1	1	1	1	1	0	0	0

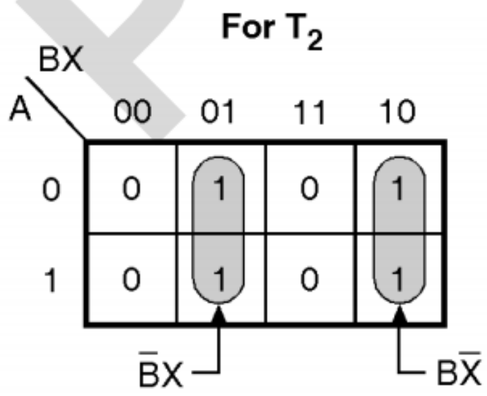
Step 4 : K-maps and simplification :

For T_1



$\therefore\ T_1 = A\bar{B} + \bar{A}BX$

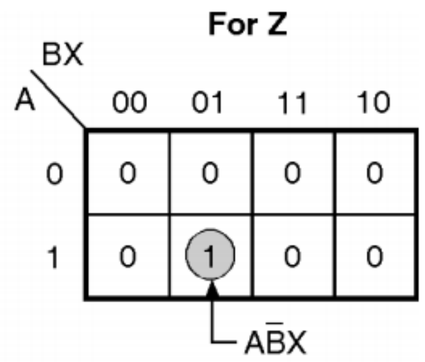
For T_2



$\therefore\ T_2 = \bar{B}X + B\bar{X}$

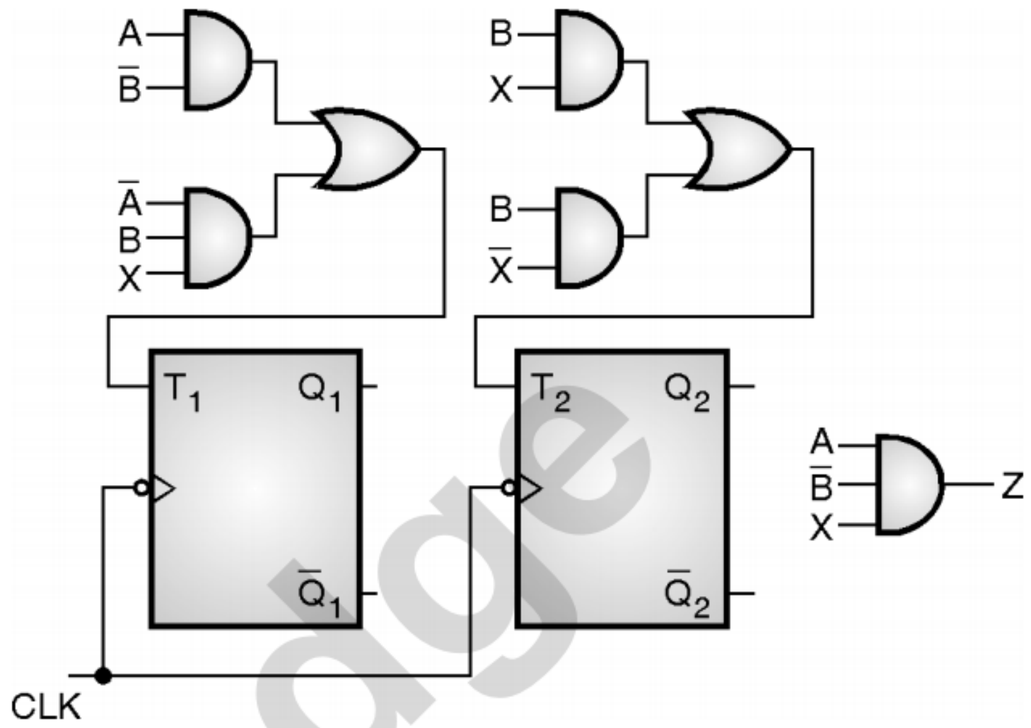
(C-5500)

For Z



$\therefore\ Z = A\bar{B}X$

Step 5 : Draw the logic diagram :



(C-5501) Fig. P. 6.12.5(b) : Logic diagram

Review Questions

Q. 1 What is FSM ?

Q. 2 Compare synchronous and asynchronous sequential circuits.

Q. 3 Write a note on : Moore and Mealy circuits.

Q. 4 Compare Moore and Mealy circuits.

Q. 5 Define the following :

1. Input and output variables
2. State variable 3. State.

Q. 6 Define present state and next state.

Q. 7 Explain the design procedure for a sequence generator.

Q. 8 What is the difference between asynchronous and synchronous sequential circuits ?

Q. 9 What are the advantages of FSM ?

Q. 10 State the disadvantages of FSM.

Q. 11 What is the advantage of state reduction technique ?

Q. 12 Write short note : Ssequence generators.

Q. 13 Write a short note : Sequence detectors.

❑❑❑

TechKnowledge Publications

Unit IV

Chapter 7

Algorithmic State Machines

Technical Publications

Syllabus

Finite State Machines (FSM) and ASM, ASM charts, Notations, Construction of ASM chart and realization for sequential circuits.
Exemplar / Case study : Waveform generator using MUX controller method.

Chapter Contents

7.1	Introduction to ASM (FSM and ASM)
7.2	ASM Chart Notations
7.3	How to use the ASM Chart for Sequential Circuit Design
7.4	Case Study : Waveform Generator using MUX Controller Method

7.1 Introduction to ASM (FSM and ASM) :

SPPU : Dec. 08, Dec. 09, May 11, May 12, Dec. 12, Dec. 15, Dec. 16, May 19

University Questions

Q. 1 What is ASM chart ? **(Dec. 08, Dec. 09, May 11, Dec. 12, 2 Marks)**

Q. 2 Give the features of ASM. **(May 12, 2 Marks)**

Q. 3 What is ASM chart ? Give its application and explain the MUX controller method with the suitable example. **(Dec. 15, May 19, 6 Marks)**

Q. 4 What is an ASM chart ? **(Dec. 16, 2 Marks)**

- An algorithm is a group of statements or steps which tell us how to obtain the solution of a problem.
- After writing the algorithm we draw a flowchart to specify the stepwise procedure and decision paths for an algorithm.
- An Arithmetic State Machine (ASM) chart is similar to the conventional flowchart but we interpret it in a different manner.
- A conventional flowchart does not take into account the time factor, but the ASM chart does take into account the time factor.

Definition :

- An ASM chart is basically a flowchart which represents a hardware algorithm. It is useful to design the hardware of a sequential circuit as per specifications.
- It is used for designing a sequential circuit having a large number of external inputs because with a large number of external inputs it becomes very difficult to use state tables for designing the circuit.
- Thus we conclude that the ASM chart is a special type of flowchart which can be used for describing the sequential operation of a digital system.

7.2 ASM Chart Notations :

SPPU : Dec. 11, Dec. 13, May 15

University Questions

Q. 1 State and explain basic components of ASM chart. Also explain the salient features of ASM chart. **(Dec. 11, Dec. 13, May 15, 6 Marks)**

- An ASM chart is composed of three basic elements :

1. The state box
2. The decision box
3. The conditional box.

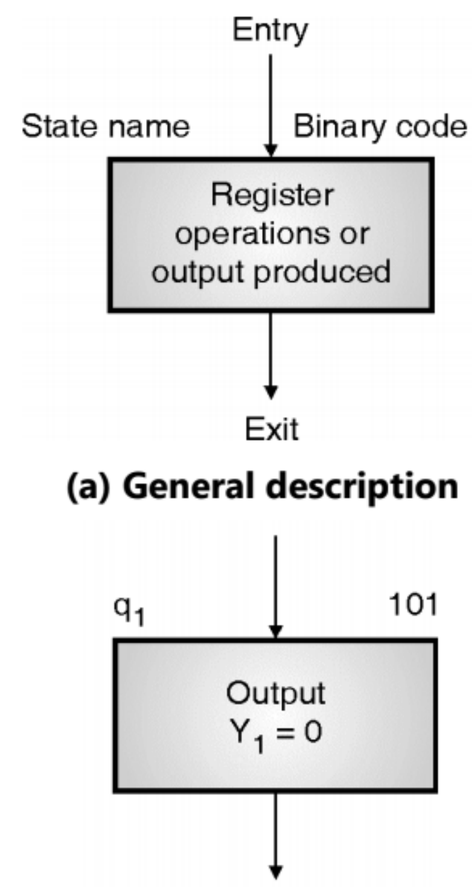
7.2.1 The State Box :

SPPU : Dec. 11, Dec. 13, May 15

University Questions

Q. 1 State and explain basic components of ASM chart. Also explain the salient features of ASM chart. **(Dec. 11, Dec. 13, May 15, 6 Marks)**

- A state box is used for indicating the state of the controller in the control sequence as shown in Fig. 7.2.1.
- The state box is rectangular in shape. Inside this box we write information such as register operations or output signal names which are generated by the controller when it is in this state.
- A symbolic name is allotted to each state and it is written on the upper left corner of the box as shown in Fig. 7.2.1(a).
- For example in Fig. 7.2.1(b) the state name is "q_1".



(a) General description

(b) Typical example

(C-1577) **Fig. 7.2.1 : The state box**

- In the state assignment we assign a binary code to each state. This code is written at the right hand top corner of the state box, as shown in Fig. 7.2.1.
- For example the binary code for state "q_1" is 101 as shown in Fig. 7.2.1(b).
- Fig. 7.2.1(b) shows a typical example of state box.

TechKnowledge Publications

– The input to a state box is indicated by "Entry" and the output is denoted by "Exit".

7.2.2 Decision Box :

SPPU : Dec. 11, Dec. 13, May 15

University Questions

Q. 1 State and explain basic components of ASM chart. Also explain the salient features of ASM chart.

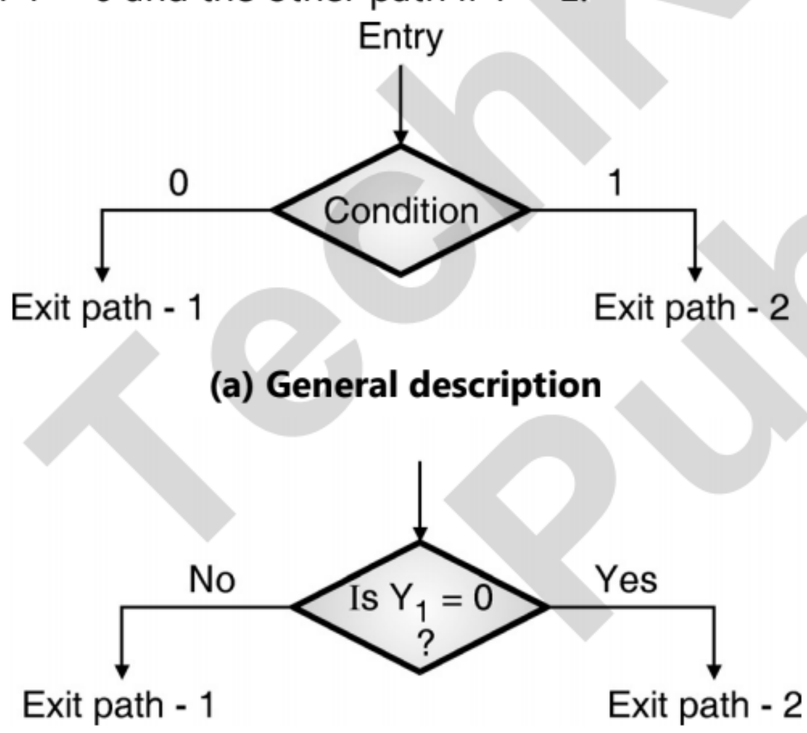
(Dec. 11, Dec. 13, May 15, 6 Marks)

– The decision box is a diamond shaped box as shown in Fig. 7.2.2. It has two or more exit paths. There is only one exit path for the state box.

– The decision box makes a decision based on certain condition and that condition is written inside the decision box. For example in Fig. 7.2.2(b) the condition reads : Is Y = 0?

– Fig. 7.2.2 shows a decision box with two exit paths. One exit path is followed if the condition specified in the decision box is true (1) and the other path will be followed if the condition is false (0).

– For example, in Fig. 7.2.2(b) we will take one exit path for Y = 0 and the other path if Y = 1.



(C-1578) Fig. 7.2.2 : The decision box

– Note that the state and decision boxes are very similar to those used in a conventional flowchart.

7.2.3 Conditional Box :

SPPU : Dec. 11, Dec. 13, May 15

University Questions

Q. 1 State and explain basic components of ASM chart. Also explain the salient features of ASM chart.

(Dec. 11, Dec. 13, May 15, 6 Marks)

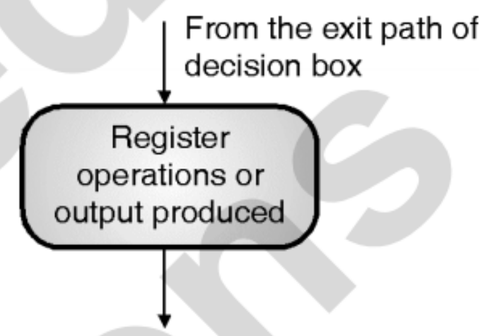
– The third element in the ASM chart is the conditional box. It is unique one in the ASM chart.

– Fig. 7.2.3 shows that the conditional box is oval shaped.

– The rounded corners makes it different from the state box.

– The input path to the conditional box always comes from one of the exit paths of the decision box.

– The register operations or outputs produced listed inside the conditional box are generated during a given state.



(C-1578(a)) Fig. 7.2.3 : The conditional box

7.2.4 ASM Block :

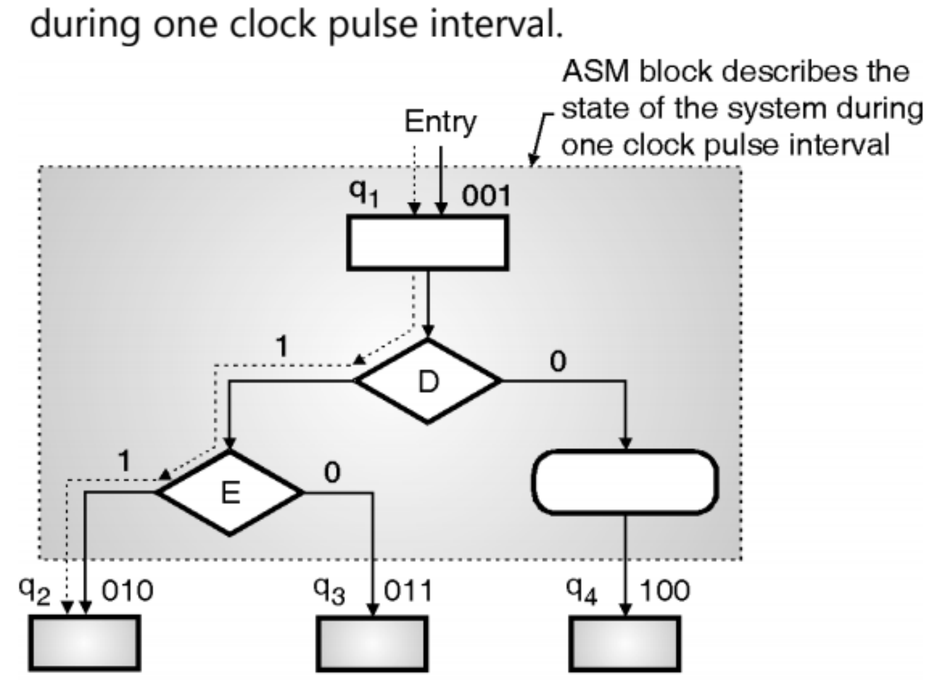
– An ASM block is a structure which consists of one state box and all the conditional and decision boxes as shown in Fig. 7.2.4(a).

– An ASM block has one entrance and multiple exit paths.

– All the decision and conditional boxes are connected to the exit path of the state box q_1 as shown in Fig. 7.2.4(a).

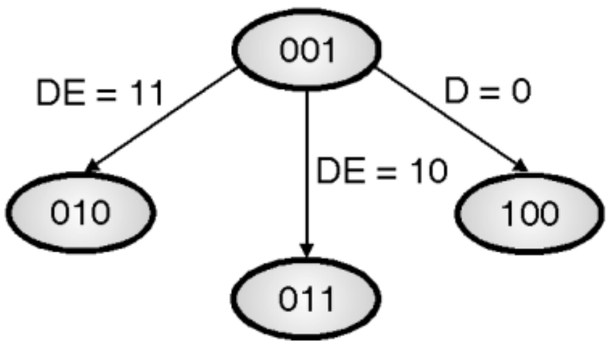
– The state diagram equivalent to this ASM block is shown in Fig. 7.2.4(b).

– Each ASM block tells us about the state of the system during one clock pulse interval.



(a) An ASM block

(C-1579) Fig. 7.2.4 (Contd...)



(b) Equivalent state diagram

(C-1579) Fig. 7.2.4

Comparison of ASM chart and state diagram :

- The operations that are mentioned inside the state and conditional boxes in Fig. 7.2.4(a) are executed when a common clock pulse is applied while the system is in state q_1.
- The same clock pulse will take the system controller from the present state q_1 to one of the next states i.e. q_2, q_3 or q_4 depending on the values of D and E.
- Compare the ASM chart of Fig. 7.2.4(a) with the state diagram of Fig. 7.2.4(b).
- The comparison reveals that the ASM chart is very similar to a state diagram.
- Each state box in an ASM chart is equivalent to one state in the state diagram.
- For example q_1 = 001, q_2 = 010, q_3 = 011 and q_4 = 100.
- The decision box in ASM chart is equivalent to the binary information written along the directed lines which connect the two states in the state diagram.
- For example D = 0 in Fig. 7.2.4(b) is equivalent to the decision box D in Fig. 7.2.4(a).
- Or consider DE = 11 will transfer the controller from the state 001 to 010 in the state diagram of Fig. 7.2.4(b).
- We obtain the same information from the ASM chart as indicated by the dotted line in Fig. 7.2.4(a).

Ex. 7.2.1 : Draw an ASM chart and state table of a 2 bit up-down counter having a mode control input. **Dec. 12, May 14, Dec. 15, 7 Marks, Dec. 16, 6 Marks**

Soln. :

- Let the mode control input be denoted by M and let the counter mode be related to M in the following manner :

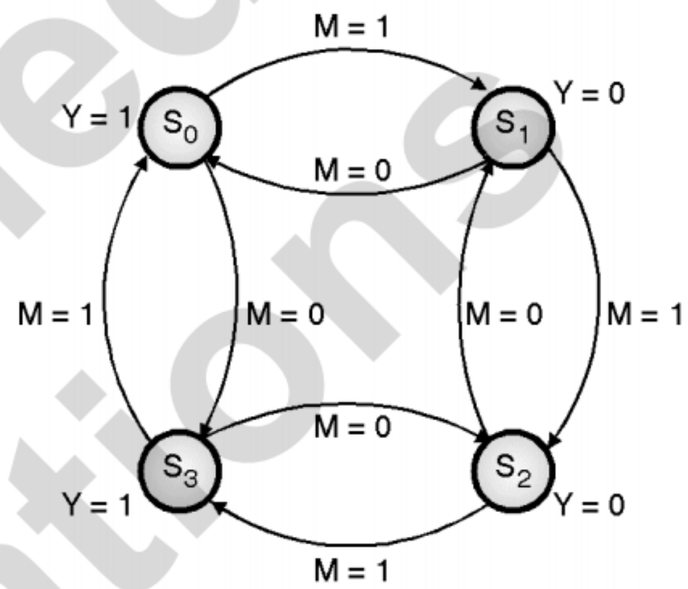
  M = 1 ... UP counting,

  M = 0 ... DOWN counting

- The circuit should produce output = 1 whenever the count becomes minimum (00) or maximum (11).
- Since number of bits = 2 there will be four states namely S_0, S_1, S_2 and S_3 corresponding to 00, 01, 10, 11 and let Y = output.

State diagram :

- The state diagram is shown in Fig. P. 7.2.1(a). Up counting takes place for M = 1 and the counter progresses as S_0, S_1, S_2, S_3, S_0 .....
- Down counting takes place with M = 0 and the counter progresses as $S_3 - S_2 - S_1 - S_0 - S_3$ ...

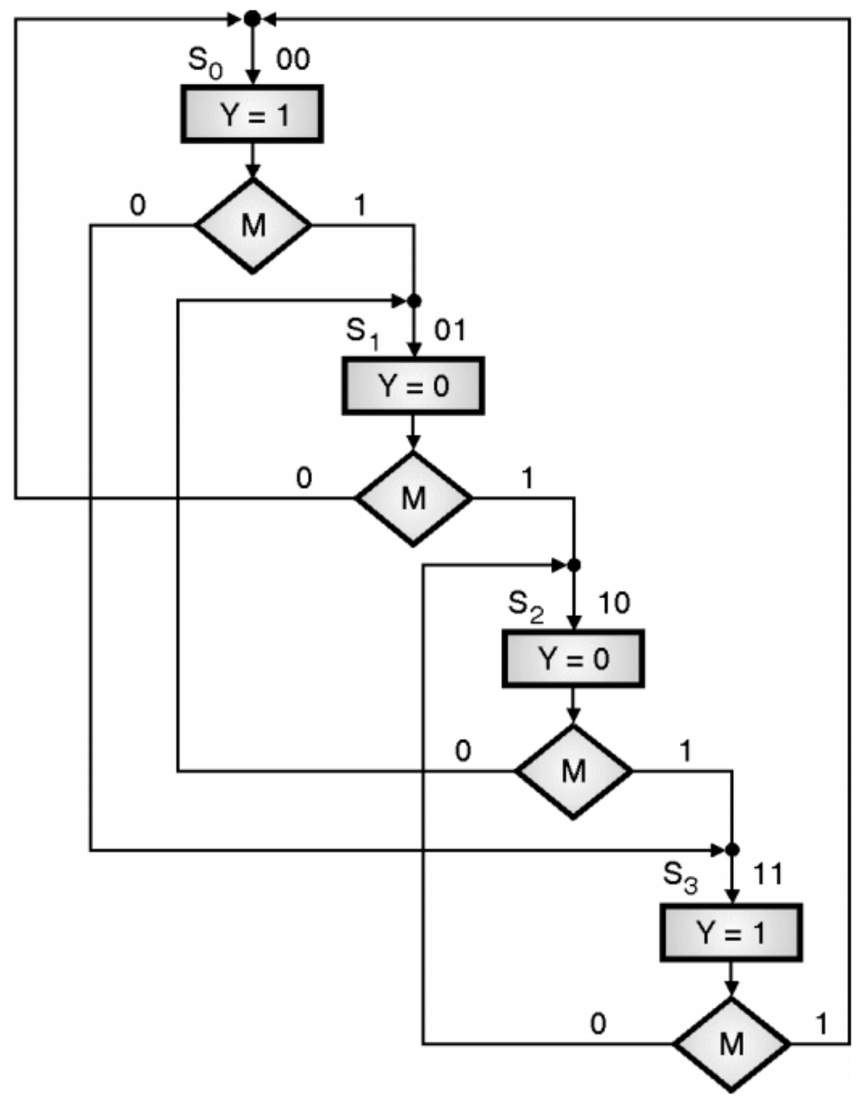


(a) State diagram of a 2 bit up down counter

(C-1580) Fig. P. 7.2.1

ASM chart :

- The ASM chart is shown in Fig. P. 7.2.1(b).

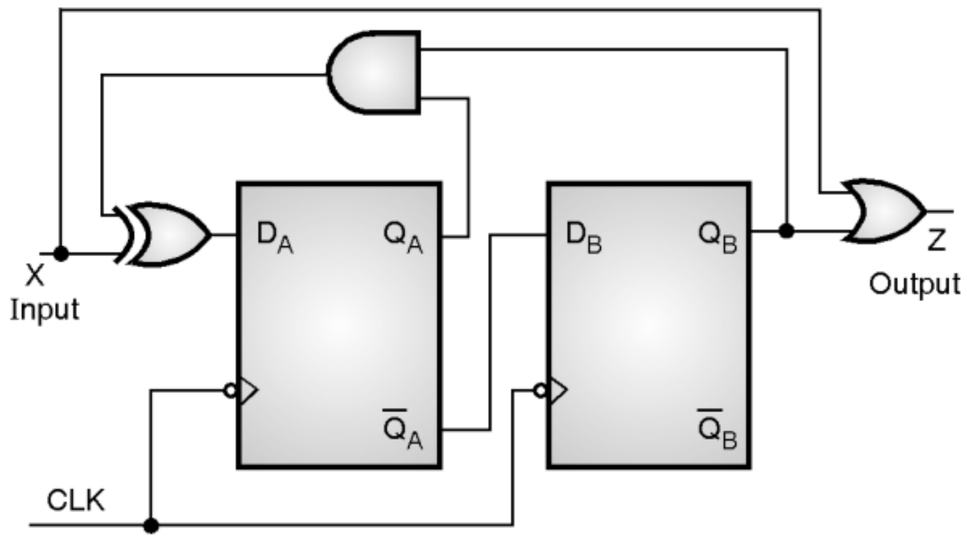


(b) ASM chart of a 2 bit up down counter

(C-1580) Fig. P. 7.2.1

TechKnowledge Publications

Ex. 7.2.2 : Draw the ASM chart and state diagram for the circuit shown in Fig. P. 7.2.2(a).



(C-1581) **Fig. P. 7.2.2(a) : Given circuit**

Soln. :

Step 1 : Draw the state table :

– The state table is shown in Table P. 7.2.2. Note that

$$D_A = X \oplus (Q_A \cdot Q_B)$$

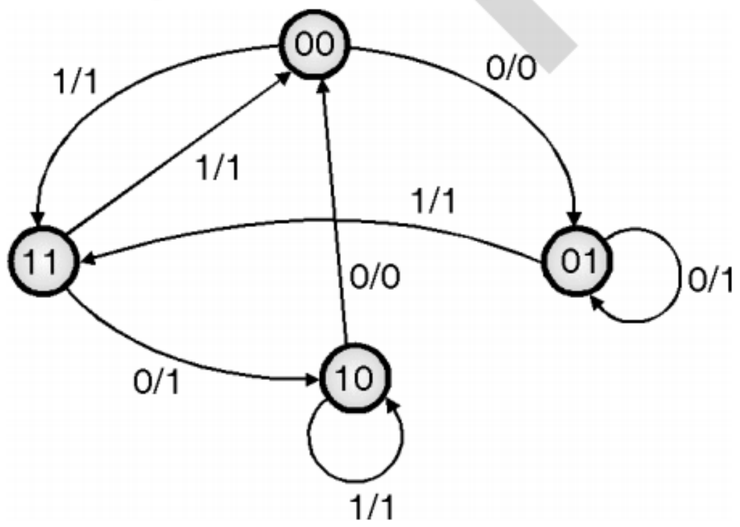
$$D_B = \overline{Q}_A \text{ and } Z = X + Q_B.$$

(C-8124)**Table P. 7.2.2 : State table**

Input X	Present state		FF inputs		Next state		Output Z
	Q_A	Q_B	D_A	D_B	Q_{A+1}	Q_{B+1}	
0	0	0	0	1	0	1	0
0	0	1	0	1	0	1	1
0	1	0	0	0	0	0	0
0	1	1	1	0	1	0	1
1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	0	1	0	1
1	1	1	0	0	0	0	1

Step 2 : Draw the state diagram :

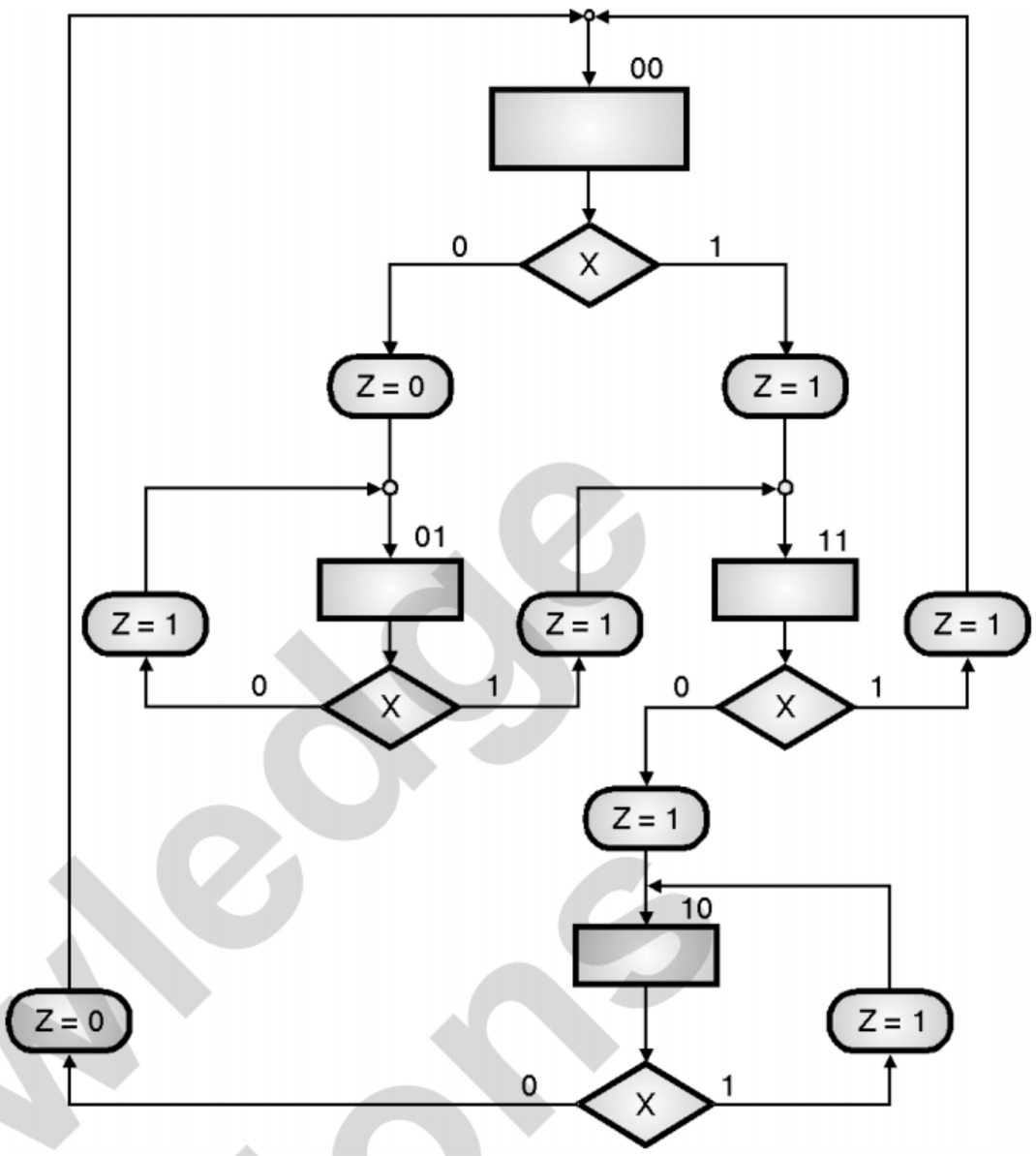
– The state diagram is shown in Fig. P. 7.2.2(b).



(C-1582) **Fig. P. 7.2.2(b) : State diagram**

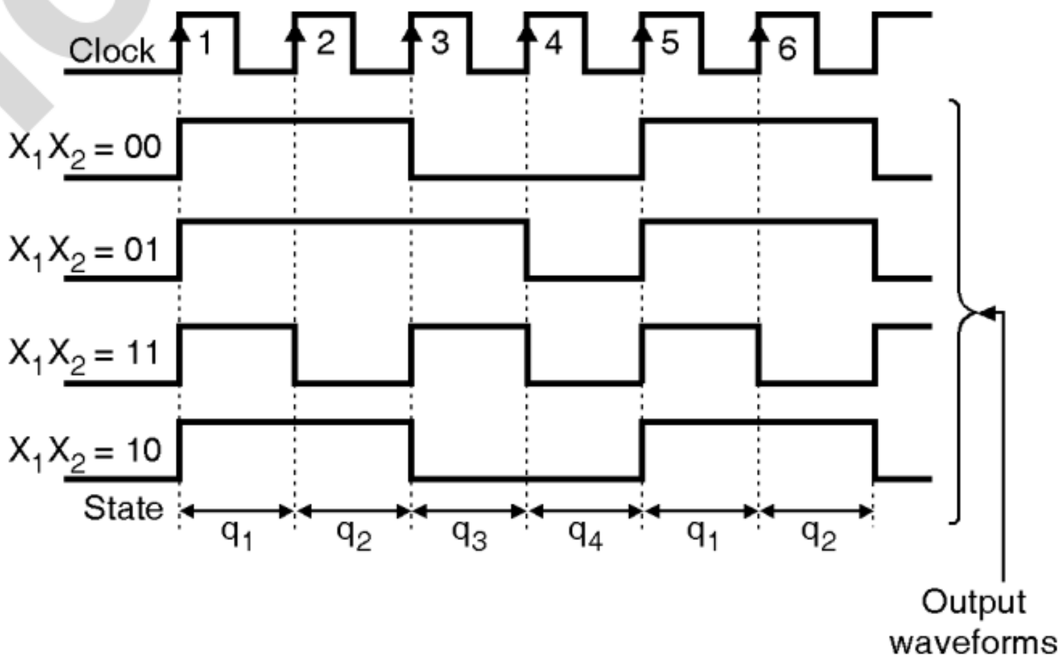
Step 3 : ASM chart :

– ASM chart drawn from the state diagram is shown in Fig. P. 7.2.2(c).



(C-1583) **Fig. P. 7.2.2(c) : ASM chart for the given circuit**

Ex. 7.2.3 : Draw the ASM chart for a signal generator which generates one waveform out of four shown in Fig. P. 7.2.3(a) depending on the combination of two control inputs X_1 and X_2.



(C-1584) **Fig. P. 7.2.3(a) : Output waveforms of a signal generator**

Soln. :

– The ASM chart can be drawn by defining four states q_1, q_2, q_3 and q_4 as shown in Fig. P. 7.2.3(a).

– Each state corresponds to a time period of one clock pulse.

– The output Z in any state is dependent conditionally on the inputs X_1 and X_2.

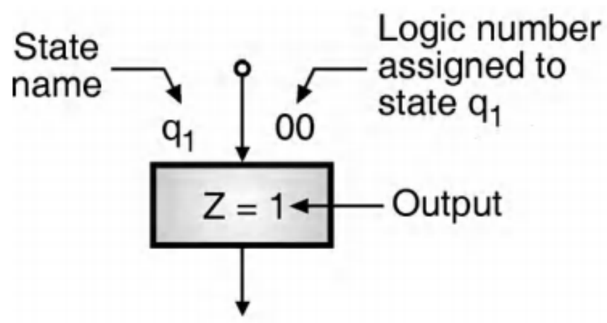
– The ASM chart is as shown in Fig. P. 7.2.3(b) and the description for drawing it is as follows :

Description :

– In state q_1 corresponding to the first clock pulse, all the waveforms are high.

– Hence we can say that the output of the waveform generator (Z) should be 1 irrespective of the values of X_1 and X_2.

– The ASM block for the first clock cycle is therefore shown in Fig. P. 7.2.3(b).

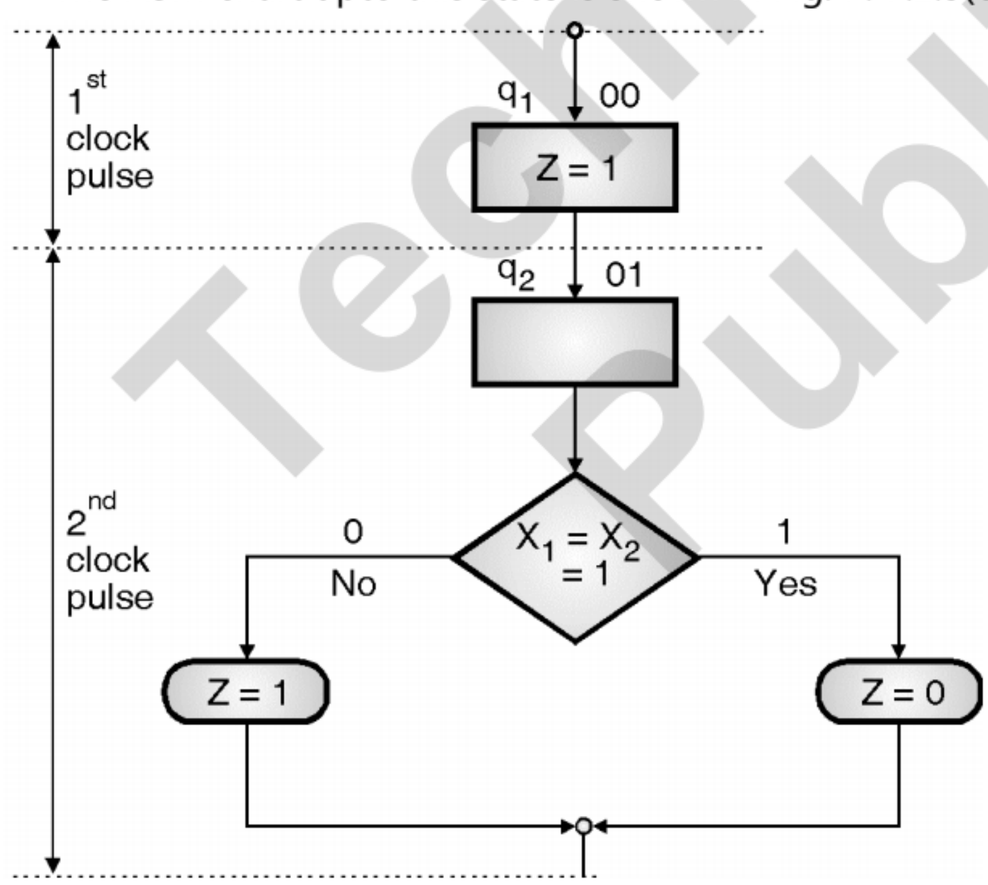


(C-1585) **Fig. P. 7.2.3(b) : ASM block for the first clock cycle**

State q_2 :

– This state corresponds to the second clock pulse. Refer Fig. P. 7.2.3(a) which shows that the output Z = 1 for all the other combinations of inputs except for $X_1 X_2$ = 11.

– The ASM chart upto this state is shown in Fig. P. 7.2.3(c).

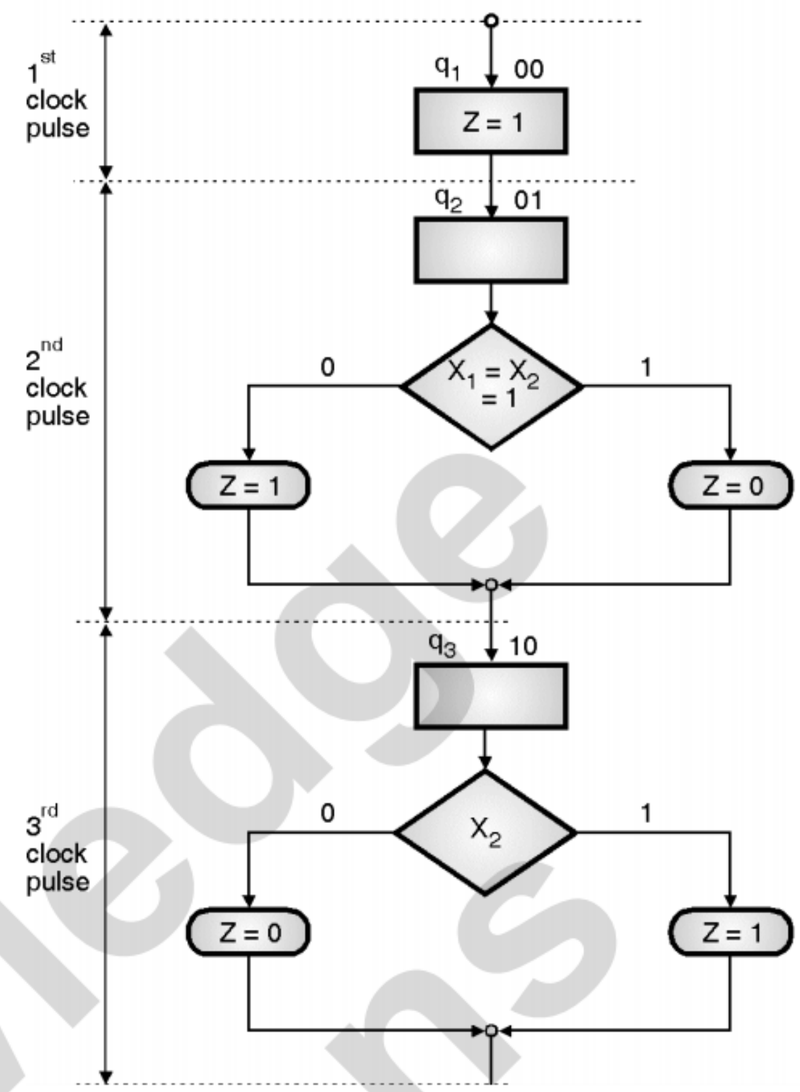


(C-1585) **Fig. P. 7.2.3 (c) : ASM chart upto second clock pulse**

State q_3 :

– This state corresponds to the third clock pulse. Refer Fig. P. 7.2.3(a) to conclude that the generator output is one (Z = 1) if X_2 = 1.

– The ASM chart upto this state is shown in Fig. P. 7.2.3(d).

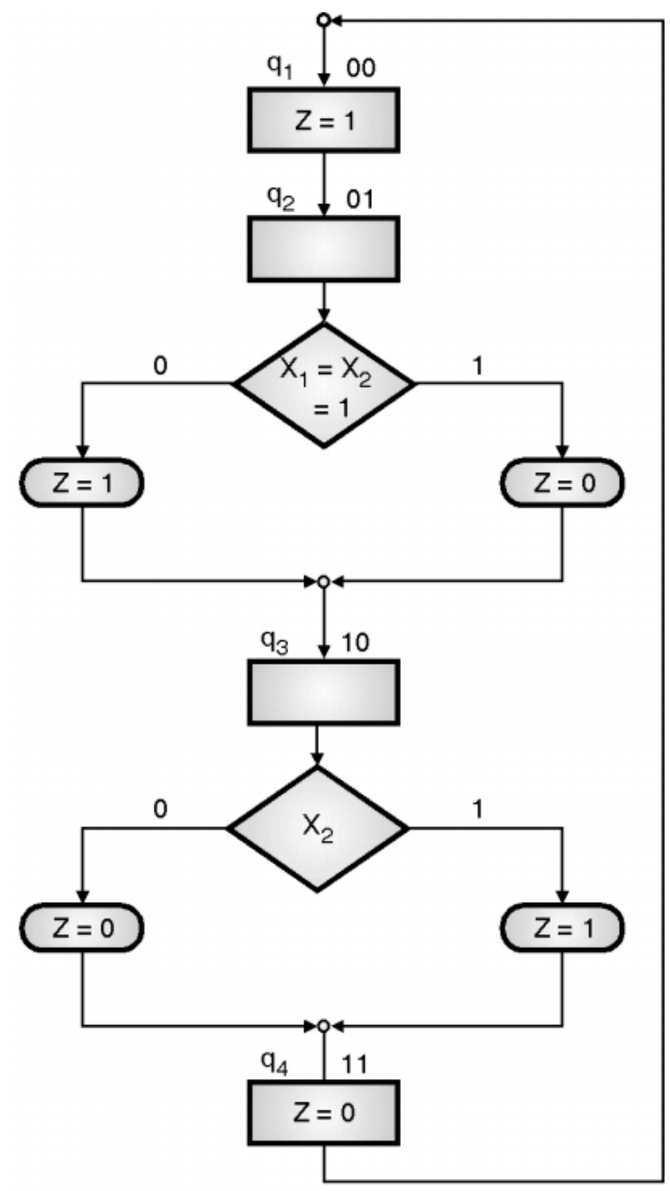


(C-1586) **Fig. P. 7.2.3(d)**

State q_4 :

– This state corresponds to the fourth clock pulse. Output Z of the waveform generator is "0" irrespective of the inputs X_1 and X_2.

– In the next clock cycle the operation starts repeating itself. The complete ASM chart is shown in Fig. P. 7.2.3(e).



(C-1586) **Fig. P. 7.2.3(e) : Complete ASM chart**

TechKnowledge Publications

Ex. 7.2.4 : Draw an ASM chart for the 3-bit down counter having one enable line such that :

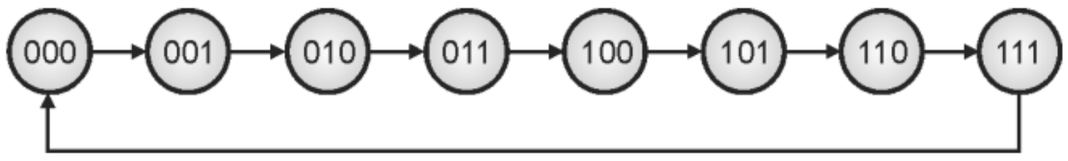
E = 1 Counting enabled),

E = 0 (Counting disabled). Also draw the state diagram. **Dec. 19, 6 Marks**

Soln. :

Step 1 : State Diagram :

– With E = 1, the counting will take place in the following manner.



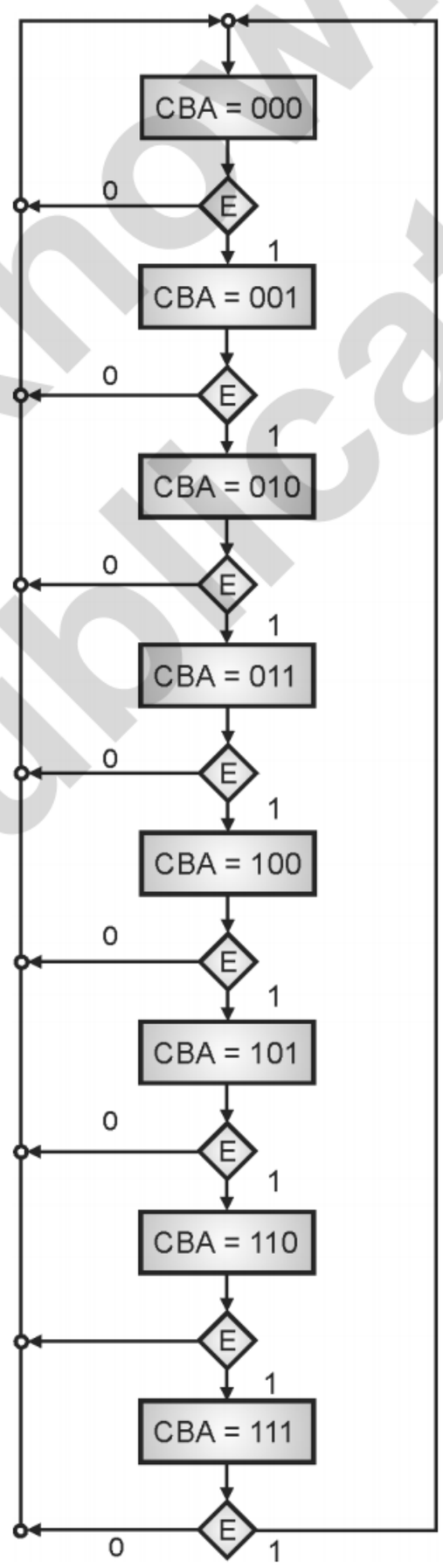
(CT-6) Fig. P. 7.2.4(a) : MOD-8 counter

– But if E = 0, the counting gets disabled and the counter is reset to 000.

– The state diagram is shown in Fig. P. 7.2.4(b) and the ASM chart is shown in Fig. P. 7.2.4(c).

(CT-7) Fig. P. 7.2.4(b) : State diagram

Step 2 : Draw the ASM chart :



(CT-8) Fig. P. 7.2.4(c) : ASM chart

TechKnowledge Publications

7.3 How to use the ASM Chart for Sequential Circuit Design ?

SPPU : May 12, Dec. 13

University Questions

Q. 1 How does ASM technique differ from conventional flow chart ? **(May 12, 4 Marks)**

Q. 2 State and explain basic component of ASM chart. What is the difference between ASM chart and conventional flow chart ? **(Dec. 13, 7 Marks)**

– We can use the ASM chart as a design tool for designing a sequential circuit by following the steps given below :

Steps to be followed :

Step 1 : The assignment or problem is given to us in the form of statements or waveforms.

Step 2 : We have to draw a suitable ASM chart from the given statement or waveform.

Step 3 : Convert the ASM chart to a state table.

Step 4 : Convert the state table into a transition table.

Step 5 : Obtain the Boolean expressions for the FF inputs and circuit outputs.

7.3.1 Realization of the Logic Circuit :

– The logic circuit can be realized after following the procedure mentioned above with the help of one of the following techniques :

1. By using the K-map simplification and flipflops.
2. By using multiplexers and flipflops.

– This procedure will be clear after solving the following examples.

Ex. 7.3.1 : For the waveform generator of Ex. 7.2.3 write the state table and transition table.

Soln. :

Step 1 : Write the state table :

– The state table is normally written in the following way.

(C-8125)**Table P. 7.3.1 : Conventional format of state table**

Present state		Inputs		Next state		Output Z
A_n	B_n	X_1	X_2	A_{n+1}	B_{n+1}	

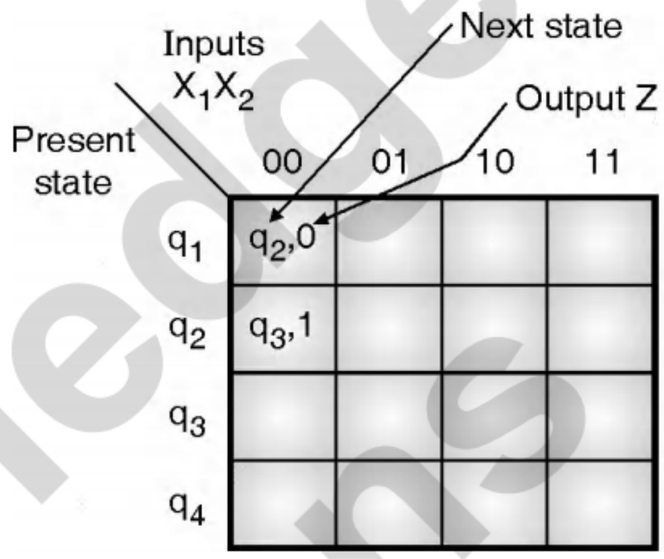
– But this state table format is lengthy so a simpler format as shown in Fig. P. 7.3.1(a) is developed.

– We will use this format of the state table hence forth.

– Substitute the next states and output Z in this format to obtain the state table of Fig. P. 7.3.1(b).

– For example refer to the first row of Fig. P. 7.3.1(b). The present state is q_1 and for all the combinations of X_1 X_2 the next state is q_2 and output is Z = 1.

– Similarly the entries in remaining boxes are made to obtain the state table of Fig. P. 7.3.1(b).



(a) Simplified format of the state table

Present state \ X_1X_2	00	01	10	11
q_1	q_2, 1	q_2, 1	q_2, 1	q_2, 1
q_2	q_3, 1	q_3, 1	q_3, 1	q_3, 0
q_3	q_4, 0	q_4, 1	q_4, 0	q_4, 1
q_4	q_1, 0	q_1, 0	q_1, 0	q_1, 0

(b) State table for the waveform generator of Ex. 7.2.3

(C-1587) **Fig. P. 7.3.1**

Step 2 : To write the transition table :

– By substituting the binary codes for the states q_1 to q_4 we can convert the state table to a transition table.

– Let q_1 = 00, q_2 = 01, q_3 = 10 and q_4 = 11.

– The transition table is shown in Fig. P. 7.3.1(c).

– The circuit realization can be done from the transition table.

Present state \ X_1X_2	00	01	10	11
q_1	01, 1	01, 1	01, 1	01, 1
q_2	10, 1	10, 1	10, 1	10, 0
q_3	11, 0	11, 1	11, 0	11, 1
q_4	00, 0	00, 0	00, 0	00, 0

Note that we have not drawn the K-map so the sequence is not 00, 01, 11, 10 instead it is 00, 01, 10, 11

(C-1588) **Fig. P. 7.3.1(c) : Transition table for the waveform generator of Ex. 7.2.3**

TechKnowledge Publications

Circuit realization using K-map simplification :

– We can realize the logic circuit from the transition table by using the K-map simplification technique.

– The procedure to be followed for this technique is as follows :

Procedure :

Step 1 :	Decide the type and number of flipflops to be used.
Step 2 :	Draw the K-maps for the inputs of all flipflops as well as circuit output, using the transition table.
Step 3 :	Obtain the simplified Boolean expressions for FF inputs and circuit output.
Step 4 :	Draw the logic diagram.

– This procedure is implemented in the following example.

Ex. 7.3.2 : Realize the waveform generator of Ex. 7.2.3 using the K-map simplification technique. Use T-flipflops.

Soln. :

Step 1 : Type and number of flipflops to be used :

– Type of FF : T – FF.

– Number of flipflops : We have four states q_1 to q_4. Therefore two flipflops are to be used.

Step 2 : Circuit excitation table :

The transition table has been repeated in Fig. P. 7.3.2(a) and the circuit excitation table from it has been written in Fig. P. 7.3.2(b).

Present state \ X_1X_2	00	01	10	11
q_1	01, 1	01, 1	01, 1	01, 1
q_2	10, 1	10, 1	10, 1	10, 0
q_3	11, 0	11, 1	11, 0	11, 1
q_4	00, 0	00, 0	00, 0	00, 0

(a) Transition table

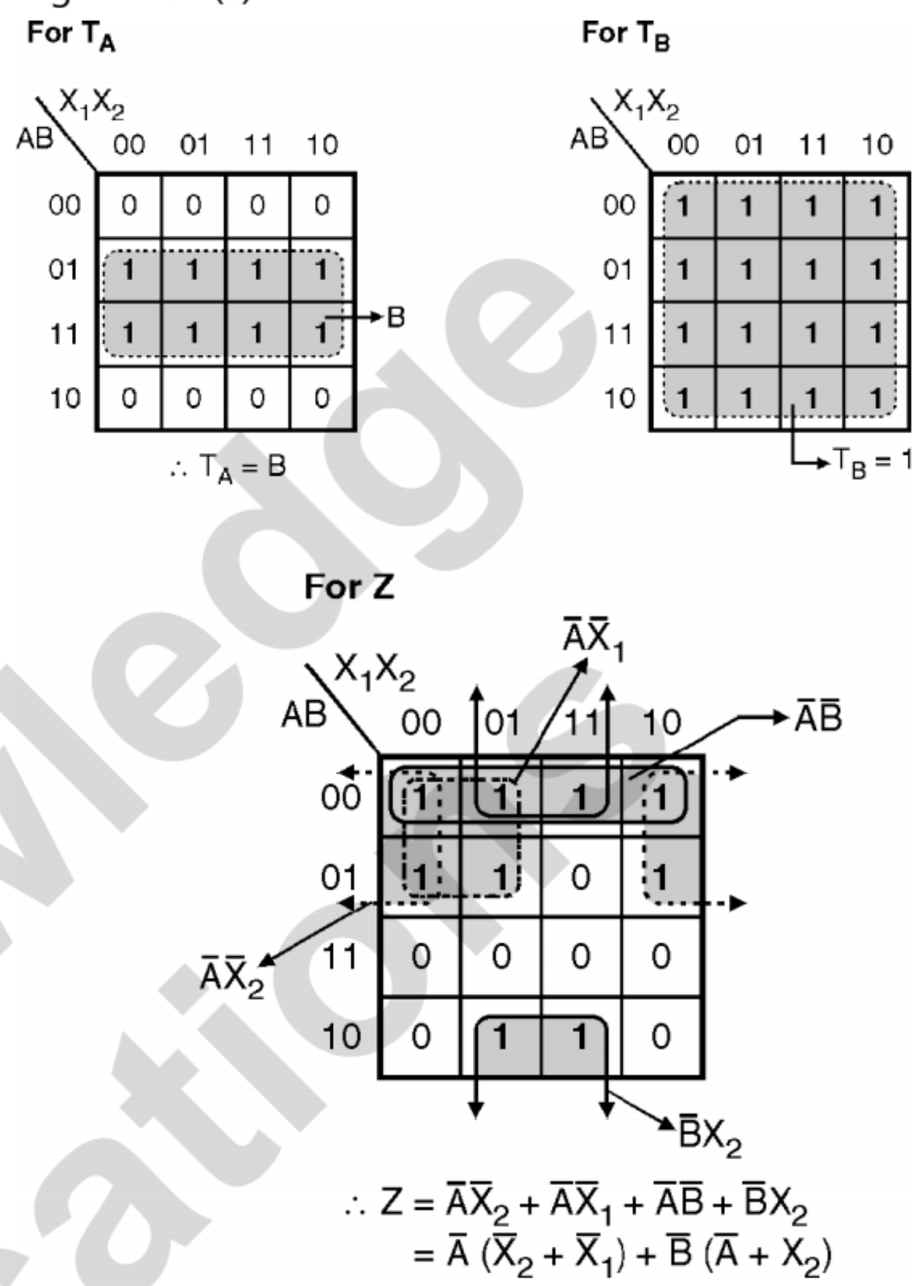
Present state \ X_1X_2	00 (T_A, T_B)	01	10	11
q_1	01, 1, 01	01, 1, 01	01, 1, 01	01, 1, 01
q_2	10, 1, 11	10, 1, 11	10, 1, 11	10, 0, 11
q_3	11, 0, 01	11, 1, 01	11, 0, 01	11, 1, 01
q_4	00, 0, 11	00, 0, 11	00, 0, 11	00, 0, 11

(b) Circuit excitation table

(C-1590) Fig. P. 7.3.2

Step 3 : K-maps and simplified expressions :

– The K-maps and simplified expressions are shown in Fig. P. 7.3.2(c).

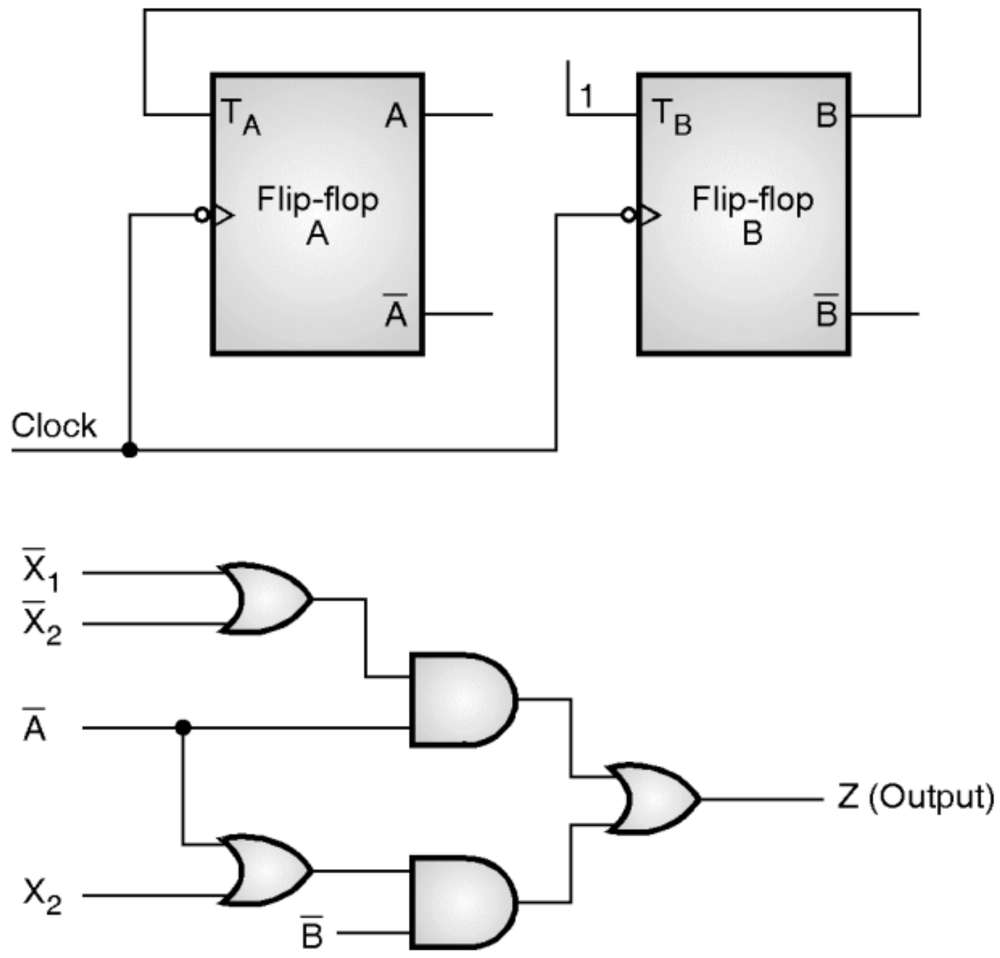


$$\therefore Z = \bar{A}\bar{X}_2 + \bar{A}\bar{X}_1 + \bar{A}\bar{B} + \bar{B}X_2$$
$$= \bar{A}(\bar{X}_2 + \bar{X}_1) + \bar{B}(\bar{A} + X_2)$$

(C-1591) Fig. P. 7.3.2(c) : K-maps and simplification

Step 4 : Draw the logic diagram :

– The waveform generator is as shown in Fig. P. 7.3.2(d).



(C-1592) Fig. P. 7.3.2(d) : Logic diagram of a waveform generator

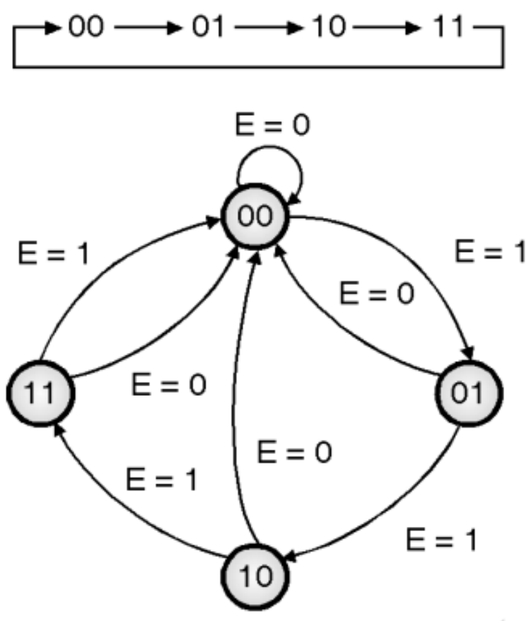
TechKnowledge Publications

Ex. 7.3.3 : Draw the ASM chart for a 2-bit binary counter having one enable line E such that : E = 1 (counting enabled), E = 0 (counting disabled).

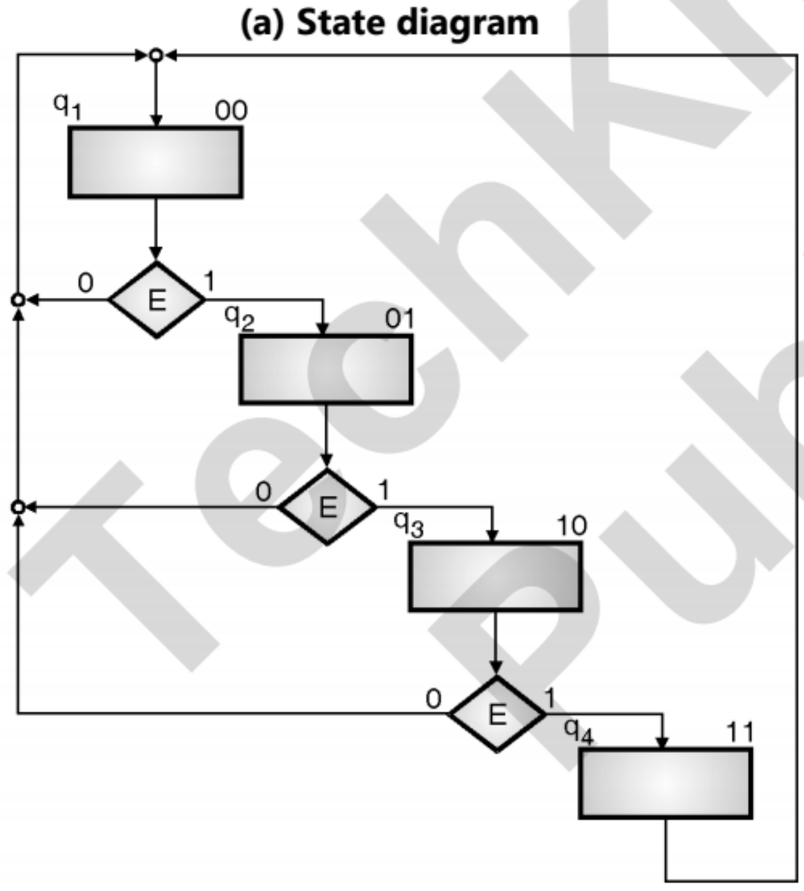
May 12, Dec. 17, Dec. 18, 6 Marks

Soln. :

- With E = 1 the counting will take place in the following manner.
- But if E = 0 let us assume that the counting gets disabled and the counter is reset to 00.
- The state diagram is shown in Fig. P. 7.3.3(a) and the ASM chart is shown in Fig. P. 7.3.3(b).



(a) State diagram



(b) ASM chart

(C-1593) Fig. P. 7.3.3

Ex. 7.3.4 : Draw the ASM chart and state diagram for the synchronous circuit having the following description :

The circuit has control input C, clock and outputs x, y, z.

1. If C = 1, on every rising edge of the clock code on output x, y and z changes from 000 → 010 → 100 → 110 → 000 and repeats.
2. If C = 0 then the circuit holds the present state.

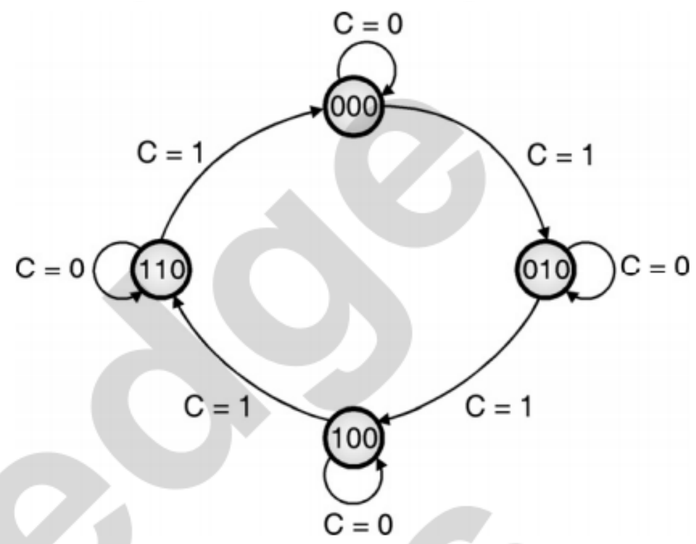
Dec. 14, 6 Marks

Soln. :

Step 1 : State diagram :

- Let the four states be defined as follows :

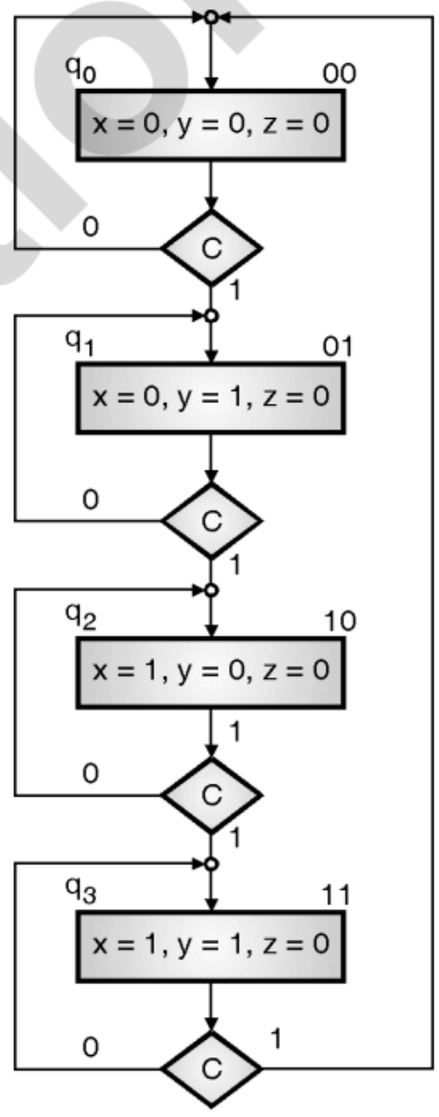
  q_0 = 00, q_1 = 01, q_2 = 10, q_3 = 11.
- The state diagram is shown in Fig. P. 7.3.4(a).



(C-1594) Fig. P. 7.3.4(a) : State diagram

Step 2 : Draw the ASM chart :

- The ASM chart is shown in Fig. P. 7.3.4(b).



(C-1595) Fig. P. 7.3.4(b) : ASM chart

Ex. 7.3.5 : Draw an ASM chart and state table for the synchronous circuit having the following description :

"The circuit has a control input X, clock and outputs A and B. If X = 1, on every clock rising edge the code on BA changes from 00 → 01 → 10 → 11 → 00 and repeats. If X = 0 the circuit holds the present state.

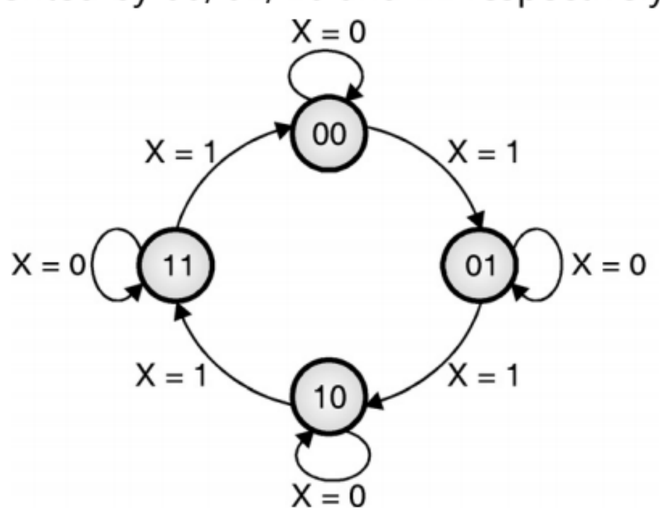
Dec. 19, 6 Marks

TechKnowledge Publications

Soln. :

Step 1 : Draw the state diagram :

– The state diagram is shown in Fig. P. 7.3.5(a) and the state table in Fig. P. 7.3.5(b). Let q_0, q_1, q_2 and q_3 are represented by 00, 01, 10 and 11 respectively.



(C-1596) Fig. P. 7.3.5(a) : State diagram

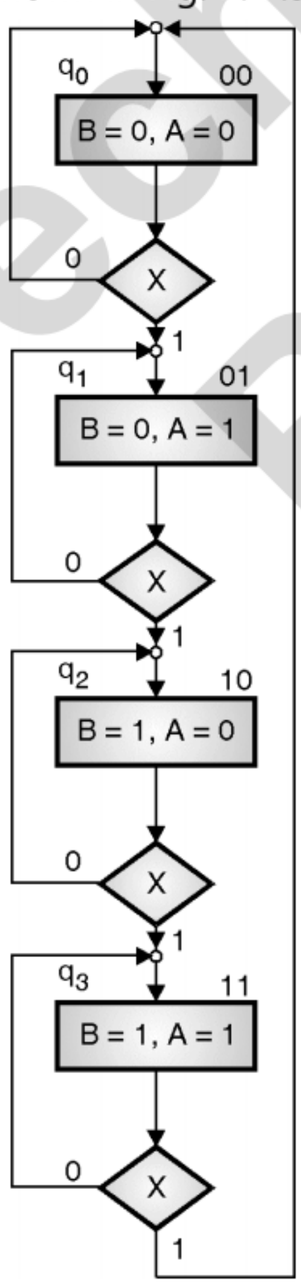
Present state	Input X = 0	Next state	Input X = 1	Next state	Output BA
q_0	q_0	00	q_1	01	
q_1	q_1	01	q_2	10	
q_2	q_2	10	q_3	11	
q_3	q_3	11	q_0	00	

(b) State table

(C-1596) Fig. P. 7.3.5

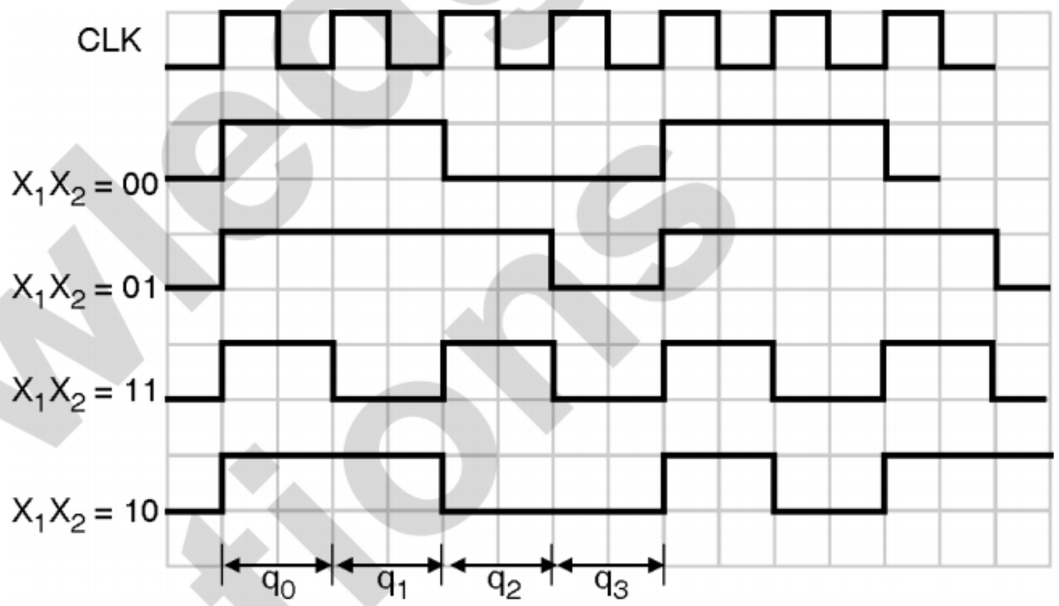
Step 2 : Draw the ASM chart :

– ASM chart is as shown in Fig. P. 7.3.5(c).



(C-1597) Fig. P. 7.3.5(c) : ASM chart

Ex. 7.3.6 : Develop an ASM chart and transition table for a controllable waveform generator that will output any one of the four waveforms given in Fig. P. 7.3.6(a) as determined by the values of its inputs X_1 and X_2. The period of the first two waveforms is four clock cycles, the period of the third is two and period of the final waveform is three clock periods. Change in input values will be infrequent. When an input change occur, the new waveform may begin at any point in it's period.



(C-1598) Fig. P. 7.3.6(a) : Given waveforms

Soln. :

From Fig. P. 7.3.6(a) it is clear that there are four states q_0, q_1, q_2 and q_3.

State q_0 :

– In this state irrespective of the values of X_1 and X_2 the generator will produce output z = 1 and go to the next state.

State q_1 :

In this state, the output Z = 1 for all the other combinations except X_1 X_2 = 11 and the generator will change its state to "q_2". Hence in the ASM chart of Fig. P. 7.3.6(b), a decision box has been included as $X_1 \cdot X_2$. If $X_1 \cdot X_2 = 1$ then Z = 0 otherwise Z = 1.

State q_2 :

– In this state output Z = 1 if X_2 = 1 and Z = 0 if X_2 = 0. A decision box accordingly appears in the ASM chart and the circuit goes to state q_3.

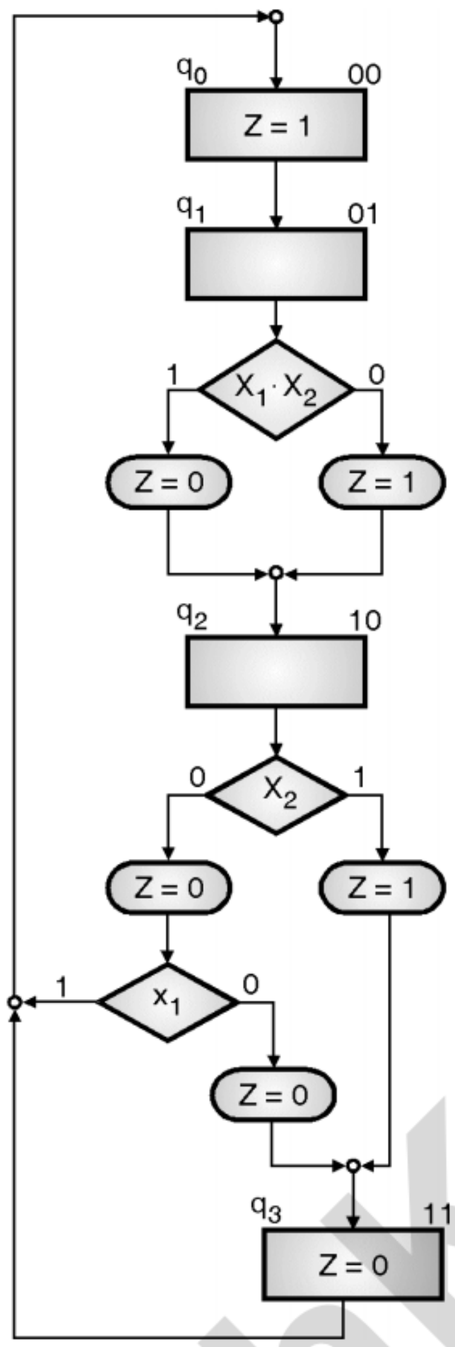
– If X_2 = 0 and X_1 = 1 i.e. X_1X_2 = 10, then the generator moves to state q_0 from q_2.

– For X_2 = 0 and X_1 = 0 i.e. X_1X_2 = 00, the output is 0 and the circuit goes to state q_3.

TechKnowledge Publications

State q_3 :

– In this state the output is 0 for all the combinations of X_1 X_2 and the circuit returns to state q_0.



(C-1599) **Fig. P. 7.3.6(b) : ASM chart**

Ex. 7.3.7 : State and explain basic components of ASM chart. Draw ASM chart for MOD 3 UP counter. **May 18, 6 Marks**

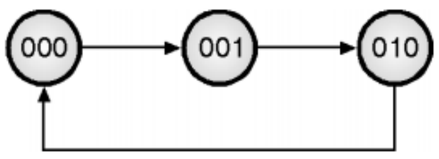
Soln. :

For the basic components of ASM refer sections 7.2, 7.2.1, 7.2.2 and 7.2.3.

Solution to Example :

Step 1 : State Diagram :

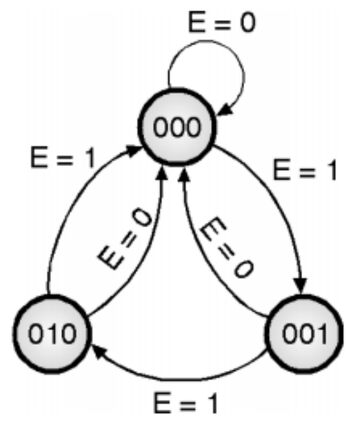
– With E = 1, the counting will take place in the following manner.



(C-7208) **Fig. P. 7.3.7(a) : MOD-3 counter**

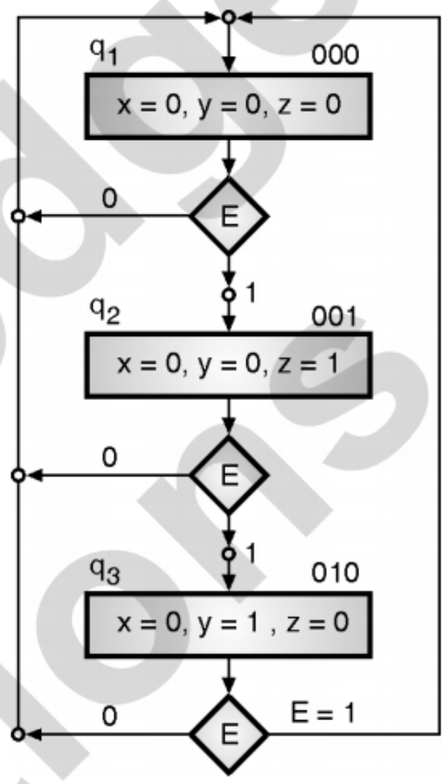
– But if E = 0, let us assume that the counting gets disabled and the counter is reset to 000.

– The state diagram is shown in Fig. P. 7.3.7(b) and the ASM chart is shown in Fig. P. 7.3.7(c).



(C-7209) **Fig. P. 7.3.7(b) : State diagram**

Step 2 : Draw the ASM chart :



(C-7210) **Fig. P. 7.3.7(c) : ASM chart**

Ex. 7.3.8 : What is ASM chart ? Design ASM chart for 4-bit gray code sequence with up-down conditions. **Dec. 10, 8 Marks**

Soln. : Refer Section 7.1 for ASM chart.

ASM chart of 4 bit gray code sequence :

– The 4 bit gray sequence is as follows :

(C-6232) **Table P. 7.3.8 : Gray sequence**

Decimal	Gray	State
0	0 0 0 0	S_0
1	0 0 0 1	S_1
2	0 0 1 1	S_2
3	0 0 1 0	S_3
4	0 1 1 0	S_4
5	0 1 1 1	S_5
6	0 1 0 1	S_6
7	0 1 0 0	S_7
8	1 1 0 0	S_8
9	1 1 0 1	S_9
10	1 1 1 1	S_{10}
11	1 1 1 0	S_{11}
12	1 0 1 0	S_{12}
13	1 0 1 1	S_{13}
14	1 0 0 1	S_{14}
15	1 0 0 0	S_{15}

Up counting

Down counting

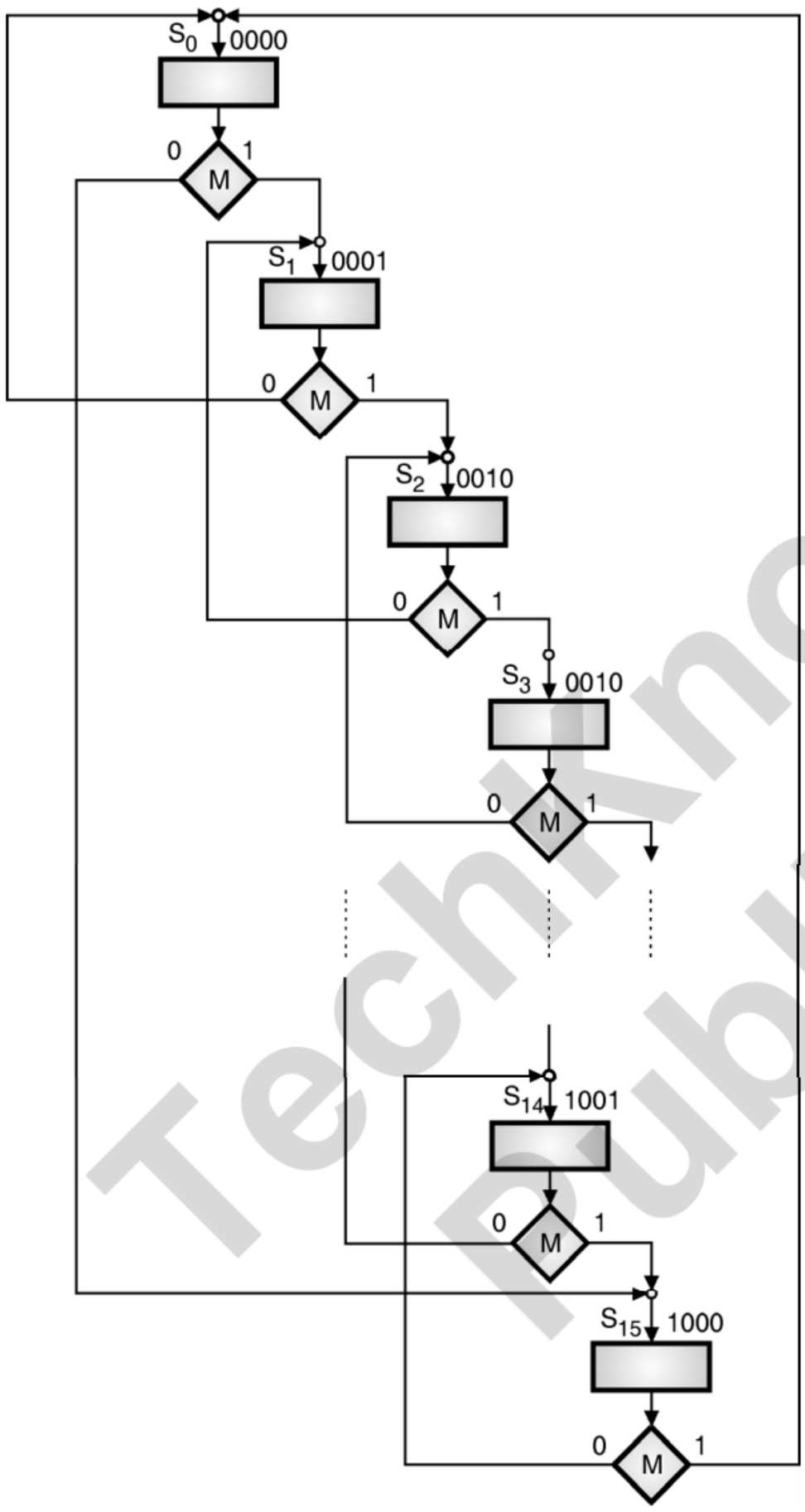
TechKnowledge Publications

– Let the mode control input be denoted by M and let the gray counter mode be related to M in the following manner.

M = 1 ...Up counting

M = 0 ...Down counting

– The ASM chart for this gray counter is as shown in Fig. P. 7.3.8.



(C-1746) **Fig. P. 7.3.8 : ASM chart for an up-down gray counter**

Ex. 7.3.9 : What is ASM chart ? Design ASM chart for 3-bit octal number sequence with up-down conditions. **May 11, 8 Marks**

Soln. :

– Refer section 7.1 for ASM chart. The 3-bit octal number sequence is as follows :

(C-6407) **Table P. 7.3.9 : 3-bit octal number sequence**

Decimal	Octal	State
0	0 0 0	S_0
1	0 0 1	S_1
2	0 1 0	S_2
3	0 1 1	S_3
4	1 0 0	S_4
5	1 0 1	S_5
6	1 1 0	S_6
7	1 1 1	S_7

Up condition (M = 1) ↓

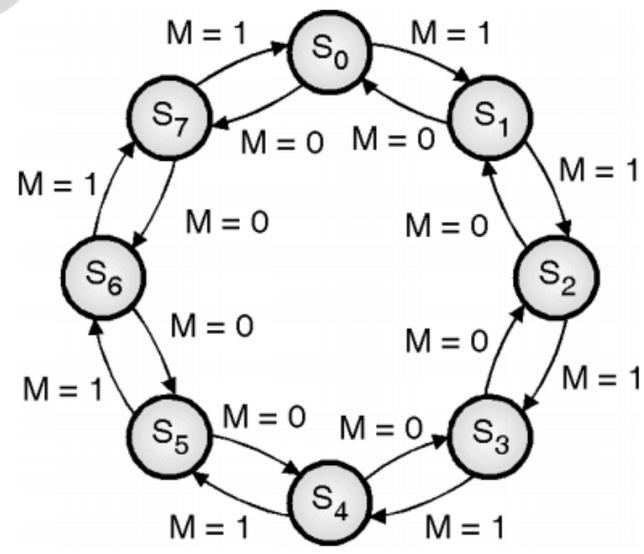
↑ Down condition (M = 0)

– Let the mode control input be denoted by M and let the up/down condition be related to M as follows :

M = 1 .....Up counting condition.

M = 0 .....Down counting condition.

– The state diagram is as shown in Fig. P. 7.3.9(a).

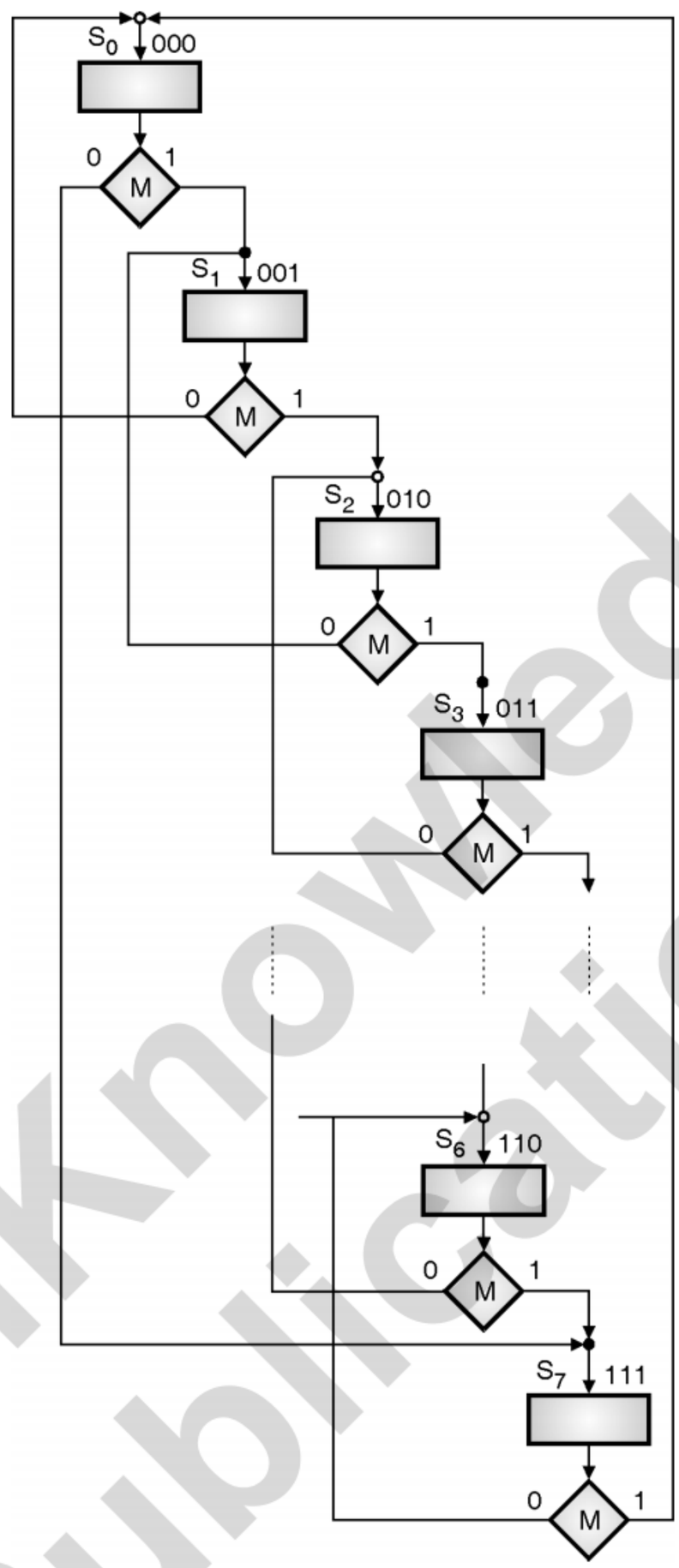


(C-1987) **(a) State diagram**

Fig. P. 7.3.9

– The ASM chart can be drawn from the state diagram, as shown in Fig. P. 7.3.9(b).

Tech Knowledge Publications



(C-1988) **(b) ASM chart for 3-bit octal number sequence with up-down conditions**

Fig. P. 7.3.9

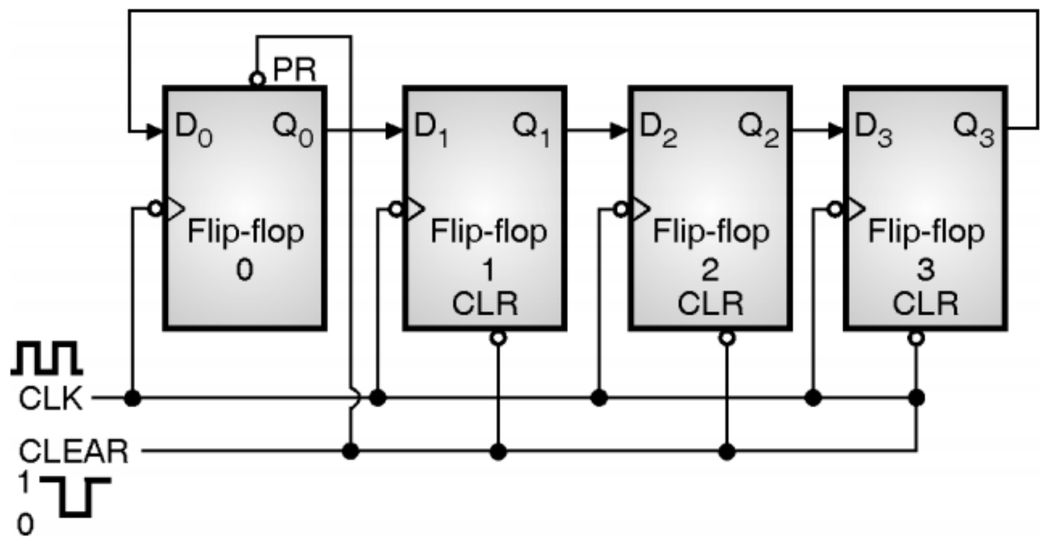
Ex. 7.3.10 : Draw an ASM chart and state table for a sequential ring counter with suitable present state and conditional input.

May 11, 8 Marks

Soln. :

Ring Counter :

– Fig. P. 7.3.10(a) shows a typical application of shift registers called Ring Counter.



(C-1748) **Fig. P. 7.3.10(a) : A four bit ring counter**

– Table P. 7.3.10(a) summarizes the operation of the ring counter.

TechKnowledge Publications

(C-1748(a)) Table P. 7.3.10(a) : Summary of operation of a ring counter

CLR	CLK	Q_0	Q_1	Q_2	Q_3
⊓	X	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0

← FF-0 Preset, others cleared

The presetted 1 follows a circular path to form a ring

State assignment :

– Let the state assignment for the distinct binary values of the output be as follows :

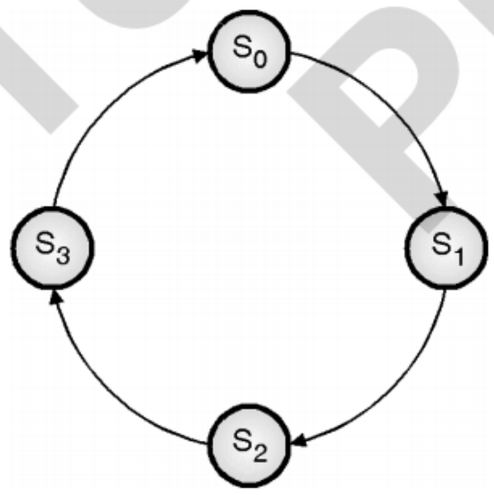
Output				State
Q_3	Q_2	Q_1	Q_0	
0	0	0	1	S_0
0	0	1	0	S_1
0	1	0	0	S_2
1	0	0	0	S_3

(C-8128)

State table and state diagram :

(C-8129) Table P. 7.3.10(b) : State table

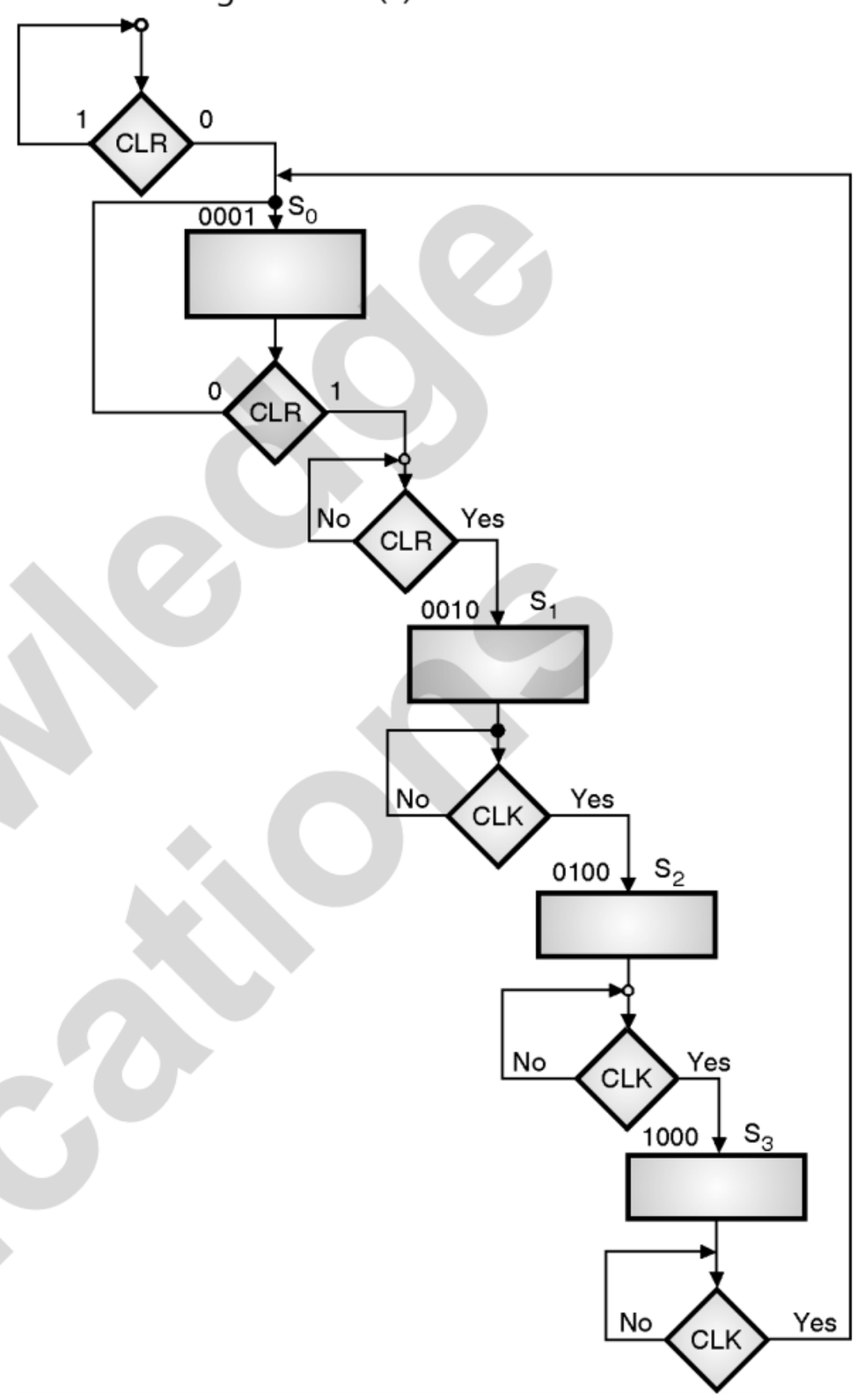
Present sate				Next sate			
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
0	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	0
1	0	0	0	0	0	0	1



(C-1749) Fig. P. 7.3.10(b) : State diagram

ASM chart :

– The ASM chart for the sequential 4-bit ring counter is as shown in Fig. P. 7.3.10(c).



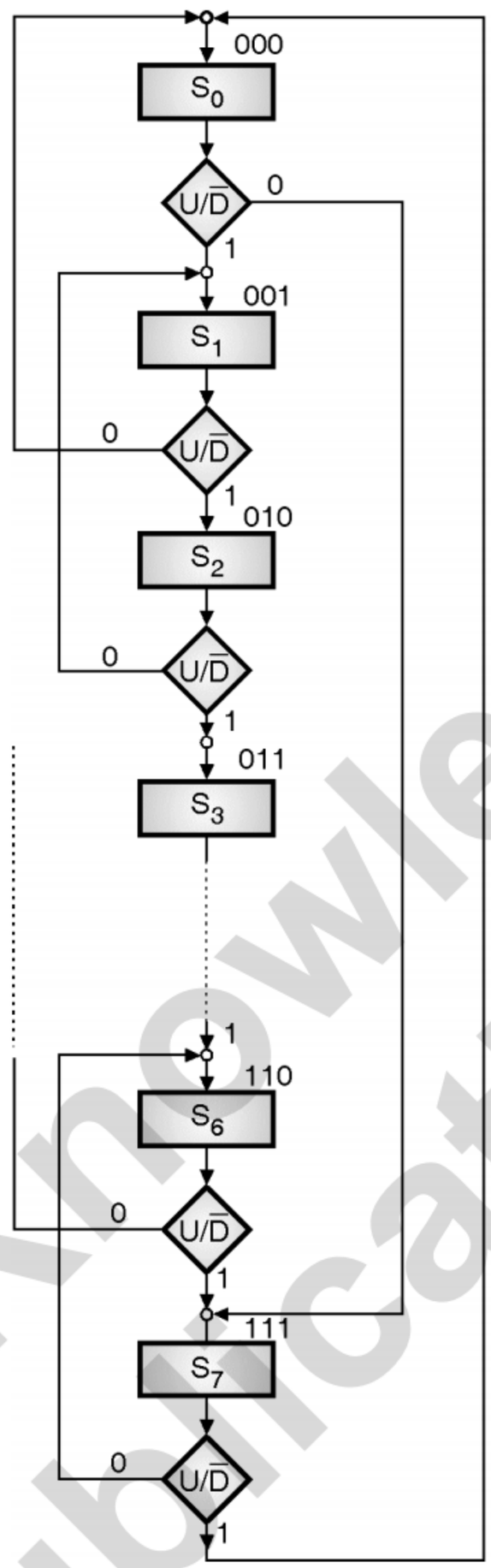
(C-1750) Fig. P. 7.3.10(c) : ASM chart for a 4 bit sequential ring counter

Ex. 7.3.11 : What is ASM chart ? Design ASM chart for 3-bit up-down counter. **Dec. 09, 8 Marks**

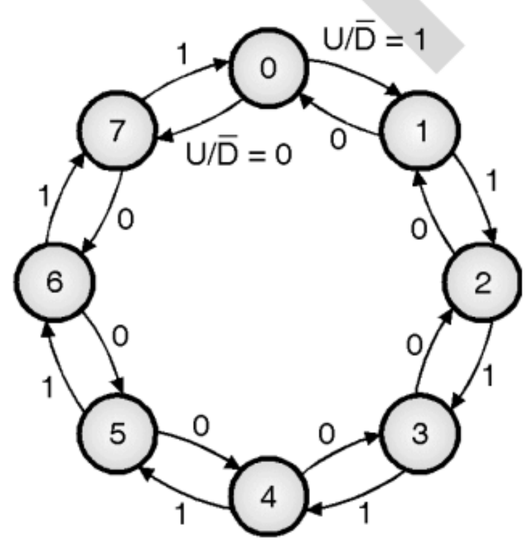
Soln. :

– For ASM charts refer section 7.1. The ASM chart for a 3 bit up down counter is as shown in Fig. P. 7.3.11(a).

TechKnowledge Publications



(C-3591) **Fig. P. 7.3.11(a) : ASM chart for a 3 bit up down counter**



(C-3591) **Fig. P. 7.3.11(b) : State diagram**

7.4 Case Study : Waveform Generator using MUX Controller Method :

SPPU : Dec. 08, May 11, Dec. 12, Dec. 15, May 19

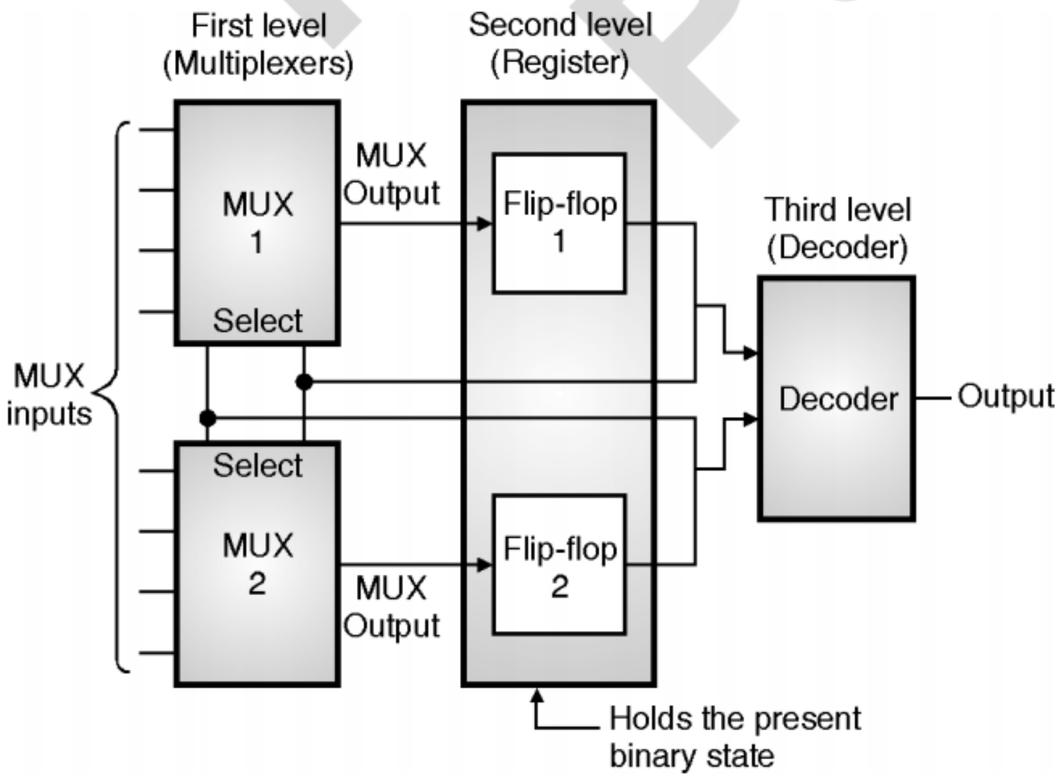
University Questions

Q. 1 Explain MUX controller method using suitable example. **(Dec. 08, May 11, Dec. 12, Dec. 15, May 19, 6 Marks)**

– In the first method of realizing a logic circuit, we started with the ASM chart and used the K-map simplification. We have discussed this method in the preceding section.

TechKnowledge Publications

- The other method of realizing a logic circuit is by using the multiplexers. We will discuss this method in this section.
- In chapter 2 we have shown that a combinational circuit can be implemented with the help of multiplexers.
- Therefore now we are going to replace the combinational circuit by a multiplexer.
- The multiplexer method of design has three levels of components, as shown in Fig. 7.4.1. The multiplexers are being used at the first level.
- The multiplexer outputs are applied to the flip flops which form the register at second level.
- The second level register is used for holding the present binary state.
- The multiplexers decide the next state of the register because MUX output has been connected to the FF inputs i.e. the register inputs as shown in Fig. 7.3.1.
- The third level has a decoder which provides a separate output for each control state.
- Sometimes the decoder is replaced by a combinational circuit.
- The inputs to this decoder come from the outputs of the register flip flops and the inputs X_1, X_2 etc.
- In order to understand the design with multiplexers let us use the same waveform generator example which we have discussed earlier.



(C-1604) **Fig. 7.4.1 : Block schematic for a 3-level scheme for multiplexer design**

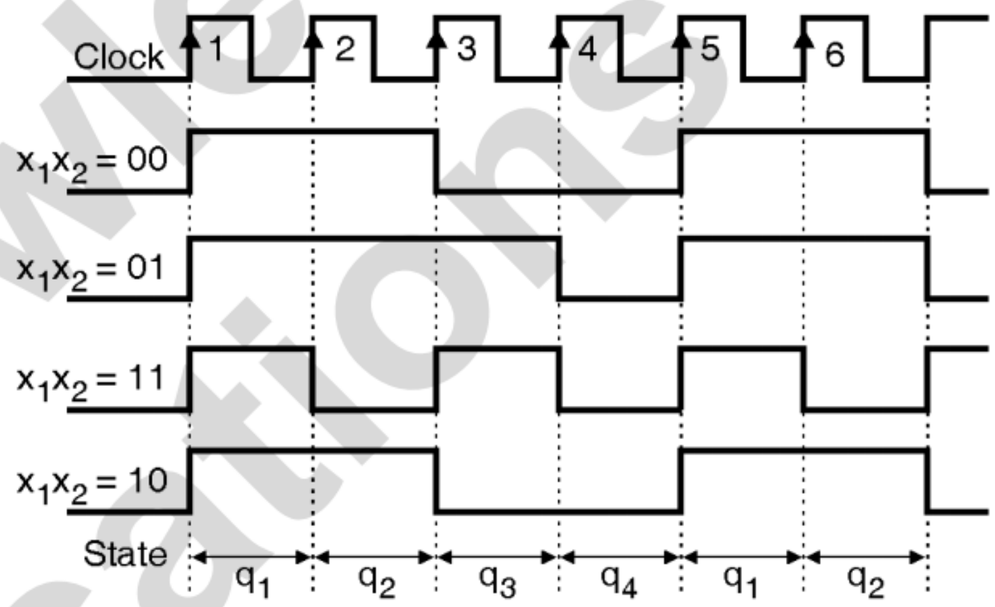
Ex. 7.4.1 : Realize the waveform generator of Ex. 7.2.3, using multiplexer design method.

Dec. 08, 4 Marks

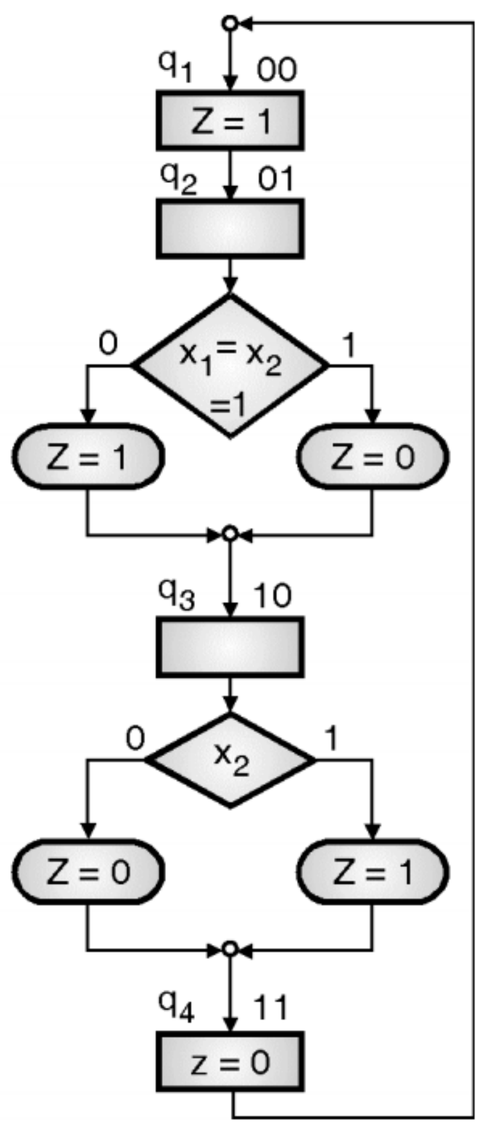
Soln. :

Step 1 : Draw the ASM chart :

- The waveforms and ASM chart have been redrawn for the reader's convenience in Fig. P. 7.4.1(a) and (b) respectively.
- This example consists of four states q_1, q_2, q_3 and q_4 and two control inputs X_1 and X_2. The three level implementation of this signal generator is shown in Fig. P. 7.4.1(c).



(a) Output waveforms of a signal generator

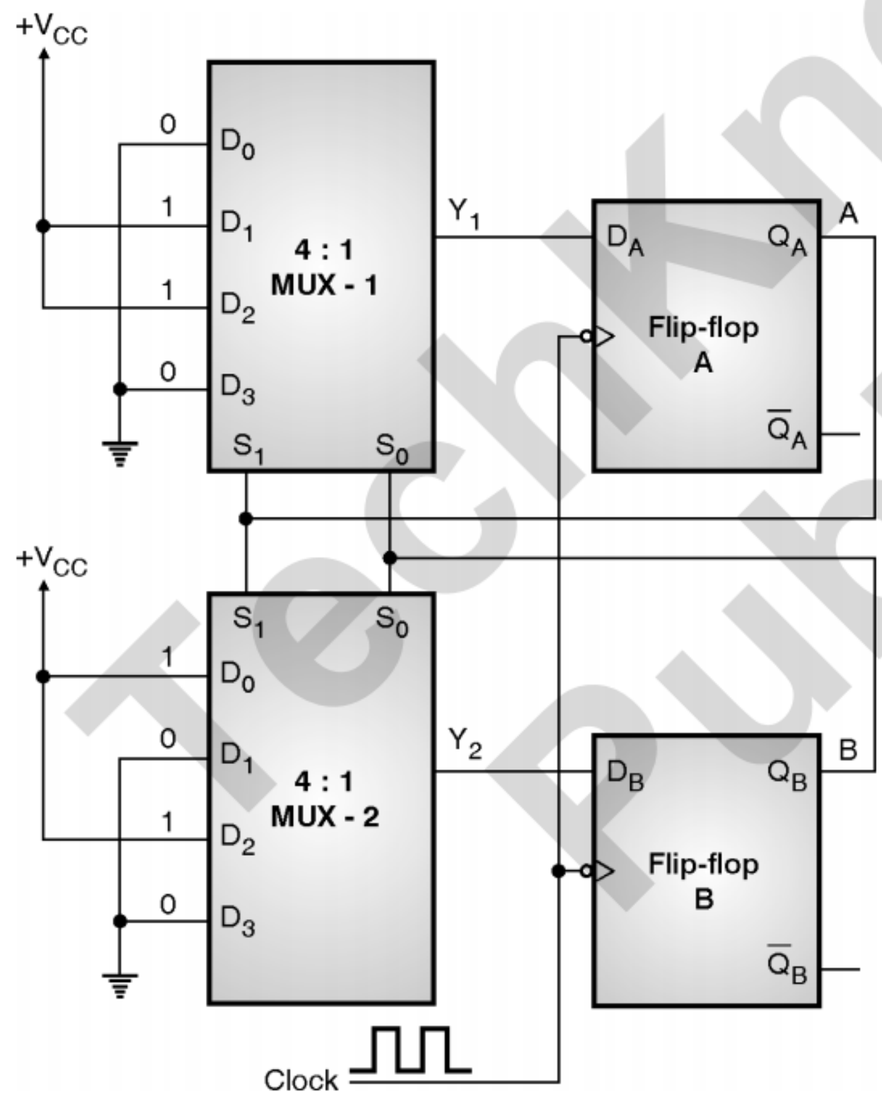


(b) Complete ASM chart

(C-1605) **Fig. P. 7.4.1**

Description :

- The logic diagram of Fig. P. 7.4.1(c) consists of two 4:1 multiplexers. Note that we have used 4 : 1 multiplexers because there are four states.
- The MUX outputs are applied to the "D" inputs of the two flip flops forming the register.
- Note that to handle four states we need two flip flops and to drive each FF we need one MUX. Therefore two multiplexers are used.
- The decoder/combinational circuit produces the Z output. i.e. One of the four waveforms is available here.
- The outputs of the register are applied to the decoder inputs as well as to the select inputs of the multiplexers.
- Thus FF outputs A, B will select one input out of four to the multiplexer.
- The outputs of the two multiplexers are applied at the inputs of the flip flops i.e. D_A and D_B respectively.



(C-1606) Fig. P. 7.4.1(c) : Logic diagram

- Thus each MUX should produce an input to the corresponding FF which equal to the binary value of next state.

∴ MUX output = FF input = Binary value of next state.

MUX inputs :

- The MUX inputs are decided by the decision boxes and state transitions given in the ASM chart of Fig. P. 7.3.9(b).
- To evaluate the MUX inputs, a table is prepared which shows the input condition for each possible transition in the ASM chart. Such a table is shown in Table P. 7.4.1.

(C-1606(a)) Table P. 7.4.1 : Multiplexer input conditions

	Present State		Next State		Input Conditions	Multiplexer inputs	
	S_1	S_0	S_1	S_0		MUX - 1	MUX - 2
Row 1	(q_1) 0	0	0	1 (q_2)	No conditions	$D_0 = 0$	$D_0 = 1$
Row 2	(q_2) 0	1	1	0 (q_3)	No conditions	$D_1 = 1$	$D_1 = 0$
Row 3	(q_3) 1	0	1	1 (q_4)	No conditions	$D_2 = 1$	$D_2 = 1$
Row 4	(q_4) 1	1	0	0 (q_1)	No conditions	$D_3 = 0$	$D_3 = 0$

- The input conditions column in Table P. 7.3.9 has been obtained from the decision boxes of the ASM chart.
- In this example the circuit always goes to the next state irrespective of the values of X_1 and X_2. Hence in the input condition column we have written "No condition".
- The two columns under "multiplexer inputs" in Table P. 7.4.1 specify the input values that must be applied to the inputs of MUX-1 and MUX-2.
- The multiplexer input for each present state is determined from the input conditions when the next state of flip flop is equal to 1.
- This can be explained by referring to ROW-1 of Table P. 7.4.1.
- It shows that S_1 changes from 0 → 0. Hence the D_0 input of MUX-1 should be 0 and S_0 changes from 0 → 1 so D_0 input of MUX-2 should be 1. This can be explained as follows :

When present state $S_1S_0 = 00$ i.e. q_1.

↓

The D_0 data input of both multiplexers are selected.

↓

∴ Outputs of the multiplexers will be equal to their respective D_0 inputs.

↓

So D_0 inputs of multiplexers should be such that they will force the D flip flop outputs so as to get the next desired state which is $S_1S_0 = 01(q_2)$ here.

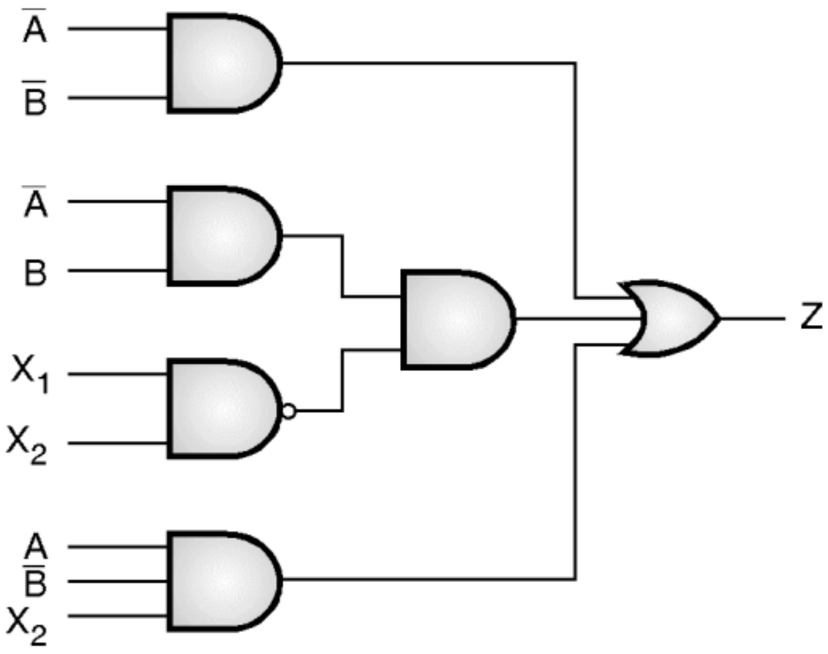
(C-6519)

- Similarly the other entries in the table have been done.

Realization :

- The signal generator is realized as shown in Fig. P. 7.4.1(d).

TechKnowledge Publications



(C-1607) Fig. P. 7.4.1(d) : Signal generator

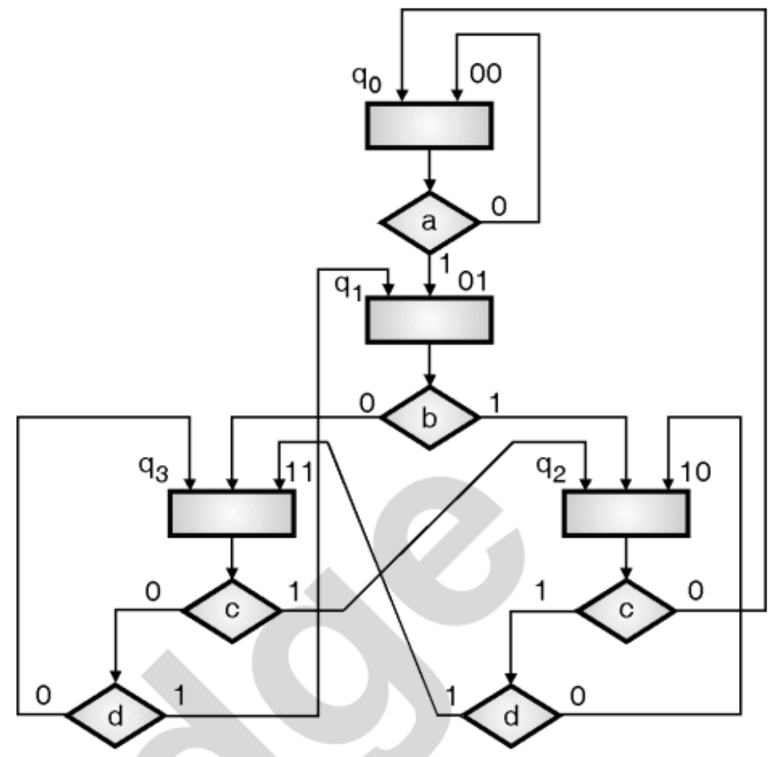
- The expression for output Z can be written by referring to the ASM chart of Fig. P. 7.4.1(b).
- We have to take into account the conditions for which Z = 1.
- For example in state q_1, output Z = 1. Therefore q_1 appears in the expression for Z. Then Z = 1 if state is q_2 and $X_1 X_2 \neq 11$, so another term in the expression for Z will be $q_2\left(\overline{X_1 X_2}\right)$. The last term in the expression for Z will be $q_3 X_2$.

$$\therefore \quad Z = q_1 + q_2\left(\overline{X_1 X_2}\right) + q_3 X_2 \qquad ...(1)$$

- Substitute $q_1 = \bar{A}\,\bar{B}$, $q_2 = \bar{A}B$, $q_3 = A\bar{B}$ to get

$$Z = \bar{A}\,\bar{B} + \bar{A}B\,\overline{X_1 X_2} + A\bar{B}X_2 \qquad ...(2)$$

Ex. 7.4.2 : For the ASM chart shown in Fig. P. 7.4.2(a), implement the logic circuit.



(C-1608) Fig. P. 7.4.2(a) : Given ASM chart

Soln. :

- From the given ASM chart it is evident that the number of states is 4 i.e. q_0 to q_3 and there are three control inputs.
- We have to use the three level implementation. In the first level, two 4:1 multiplexers are being used. We need 2 flipflops and to drive them, we require 2 multiplexers.
- MUX outputs are applied to the inputs of two D flipflops. The outputs of these flipflops are applied to the MUX select lines (S_0 and S_1).

Step 1 : Write the table to evaluate MUX inputs :

- Table P. 7.4.2 contains all the information about the multiplexer inputs.

(C-1608(a)) Table P. 7.4.2 : Multiplexer input conditions

	Present State		Next State		Input Conditions	Inputs		
	S_1	S_0	S_1	S_0		MUX - 1	MUX - 2	
Row 1	0	0	0	0	a = 0 i.e. $\bar{a}$	$D_0 = 0$	$D_0 = 0$	$D_0 = a$
	0	0	0	1	a = 1 i.e. a	$D_0 = 0$	$D_0 = 1$	
						$D_0 = 0$		
Row 2	0	1	1	0	b = 1 i.e. b	$D_1 = 1$	$D_1 = 0$	$D_1 = \bar{b}$
	0	1	1	1	b = 0 i.e. $\bar{b}$	$D_1 = 1$	$D_1 = 1$	
						$D_1 = 1$		
Row 3	1	0	0	0	c = 0 i.e. $\bar{c}$			Only consider the input condition that corresponds to D =1
	1	0	1	0	c = 1, d = 0 i.e. $c\bar{d}$	$c\bar{d} + cd = c$	cd	
	1	0	1	1	c = 1, d = 1 i.e. cd			
Row 4	1	1	0	1	c = 0, d = 1 i.e. $\bar{c}d$			
	1	1	1	0	c = 1 i.e. c	$c + \bar{c}\bar{d} = c + \bar{d}$	$\bar{c}d + \bar{c}\bar{d} = \bar{c}$	
	1	1	1	1	c = 0, d = 0 i.e. $\bar{c}\bar{d}$			

TechKnowledge Publications

– To understand the entries in Table P. 7.4.2 let us consider Row - 2 of Table P. 7.4.2.

	Present State		Next State		Input conditions	Inputs	
	S_1	S_0	S_1	S_0		MUX - 1	MUX - 2
Row 2	0	1	1	0	b	$D_1 = 1$	$D_1 = \overline{b}$
	0	1	1	1	$\overline{b}$		

(C-1608(b))

– The present state 01 goes to next state 10 if b = 1 or to the next state 11 if b = 0. So we have marked the input conditions as b and $\overline{b}$ in ROW-2 of Table P. 7.3.10.

– Now consider the column named MUX-1 in the above table after the present state 01, the next state of S_1 is always 1 and the next state of S_0 is equal to the complement of b so we have written MUX-1 input $D_1 = 1$ and MUX-2 input $D_1 = \overline{b}$.

– As an another example consider ROW 3 of Table P. 7.4.2.

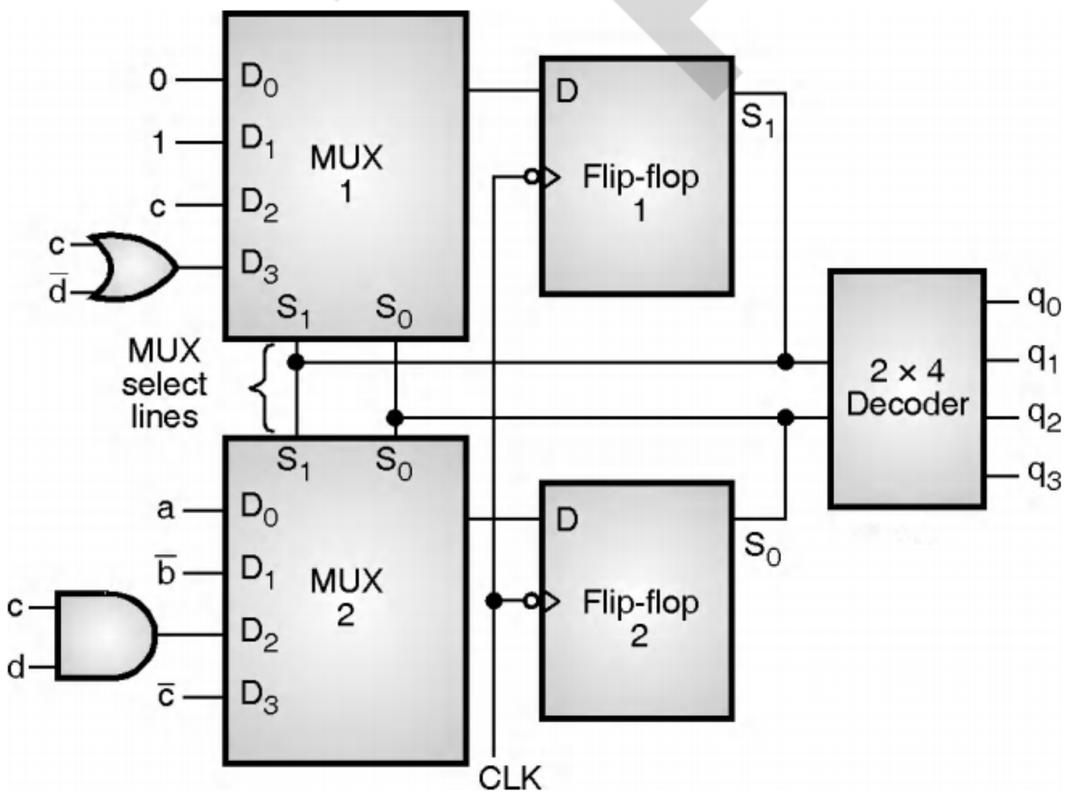
– Here, after the present state 10, the next state of S_1 is equal to 1 only if the input condition is $c\overline{d}$ or cd. When $c\overline{d}$ and cd are ORed, we get only c. So MUX-1 input is c.

– For ROW 3 the next state of S_0 is equal to 1 if cd = 11. Therefore MUX-2 input is cd.

– Similarly the entries in the other rows are done.

Step 2 : Draw the logic diagram :

– The logic diagram of the required controller is shown in Fig. P. 7.4.2(b).



(C-1609) **Fig. P. 7.4.2(b) : Control implementation using multiplexers**

Ex. 7.4.3 : A sequential circuit has to count DOWN from '111' to '100'. The circuit also has input 'X'. If X = 0 then the circuit will count DOWN and if X = 1 then they will remain in the current state. Draw an ASM chart and state table for this circuit and design the circuit to generate the output using MUX controller method.

Dec. 06, 16 Marks

Soln. :

Given : If X = 1 then count down from 111 to 100

If X = 0 then remain in the current state.

To do :
1. ASM chart
2. State table
3. Design using MUX controller method.

Part I : ASM chart and state table :

Step 1 : Draw the state diagram and state table :

– Let the four states through which the circuit goes be denoted by q_0, q_1, q_2 and q_3.

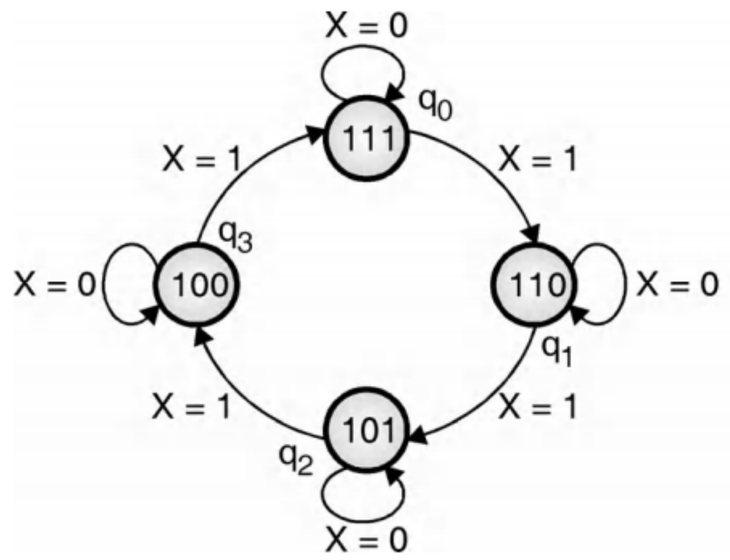
– The relation between these states and the outputs is as follows :

– Now draw the state diagram as shown in Fig. P. 7.4.3(a) and then write the state table as shown in Table P. 7.4.3(a).

State	Output
q_0	1 1 1
q_1	1 1 0
q_2	1 0 1
q_3	1 0 0

(C-8126)

TechKnowledge Publications



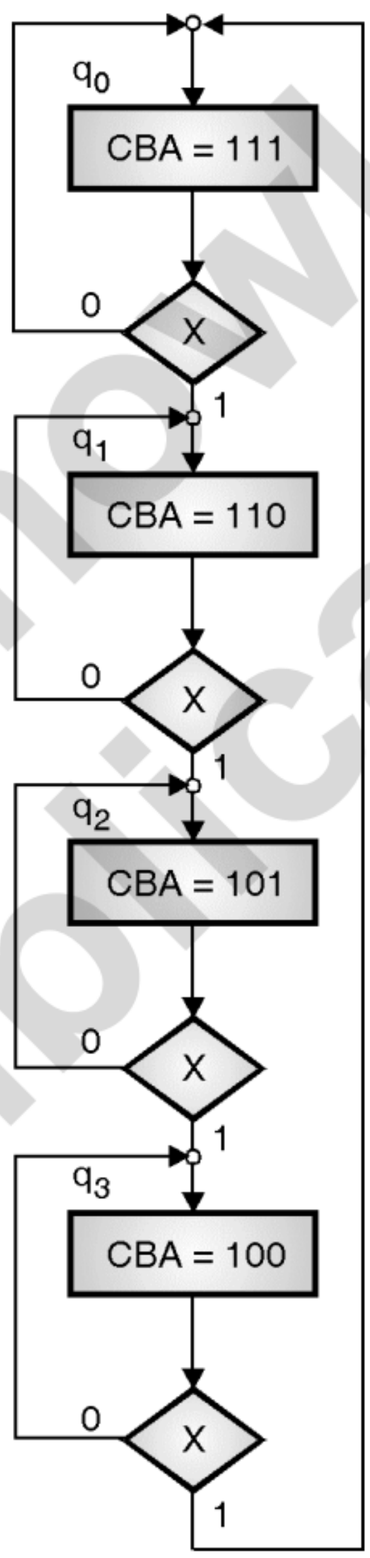
(C-1610) Fig. P. 7.4.3(a) : State diagram

(C-1610) Table P. 7.4.3(a) : State table

Present state	Input X = 0	Next state	Input X = 1	Next state	Output
q_0	q_0	111	q_1	110	
q_1	q_1	110	q_2	101	
q_2	q_2	101	q_3	100	
q_3	q_3	100	q_0	111	

Step 2 : Draw the ASM chart :

– The ASM chart is as shown in Fig. P. 7.4.3(b).



(C-1611) Fig. P. 7.4.3(b) : ASM chart

Part II : Design the circuit using multiplexers :

– We have to use three flip-flops because the output is 3 bit (111 to 110).

– So we have to use three 8 : 1 multiplexers.

– It is not necessary to use any decoder or combinational circuit after flip-flops.

– We will use the D flip-flops. MUX outputs are connected to the D F/F inputs as shown in Fig. P. 7.4.3(c).

– The F/F outputs A, B and C are connected to the select inputs S_2, S_1 and S_0 as shown in Fig. P. 7.4.3(c).

TechKnowledge Publications

Step 3 : Write the table for deciding MUX input :

– Refer Table P. 7.4.3(b) which shows the inputs to be applied to the inputs D_0 to D_7 of all the three 8 : 1 multiplexers.

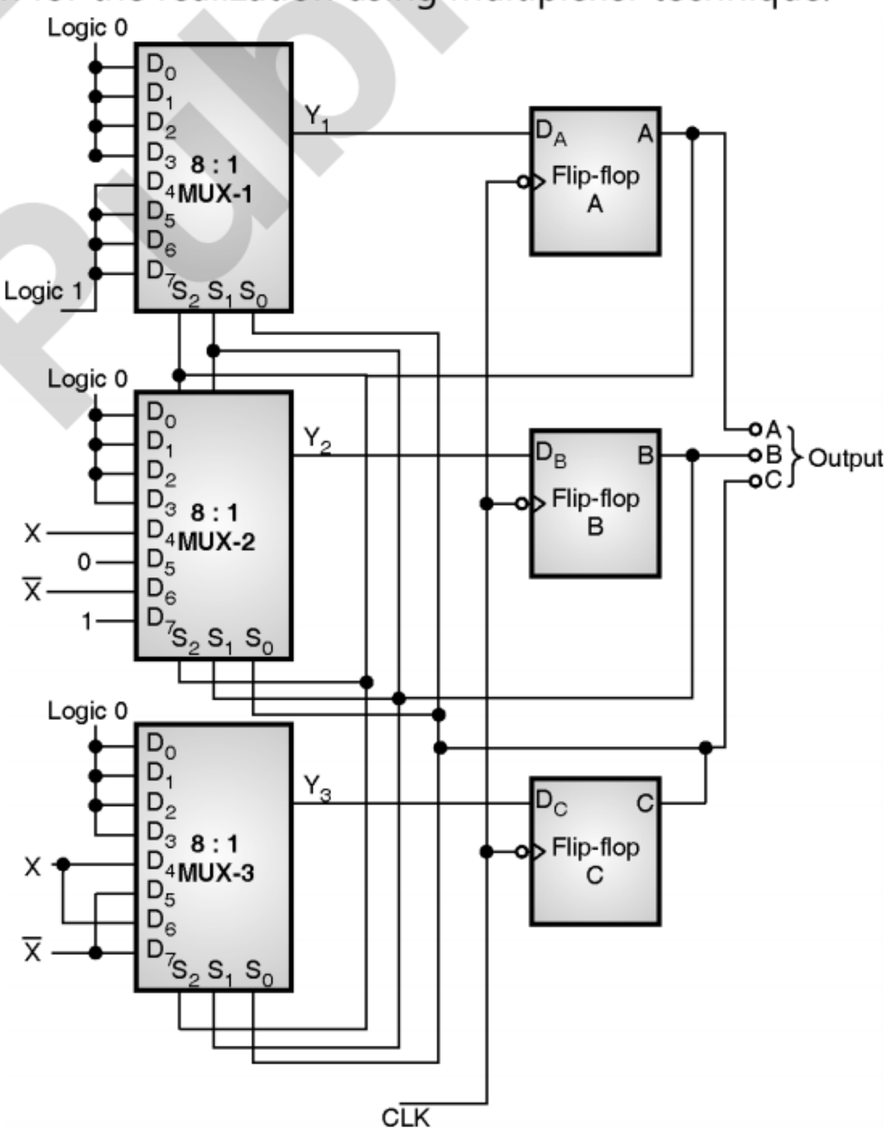
Note that the unused present states i.e. 000, 001, 010 and 011 are terminated into next state of 000.

(C-1611(a)) Table P. 7.4.3(b) : Multiplexer input conditions

	Present state			Next state			Input	MUX inputs		
	S_2	S_1	S_0	S_2	S_1	S_0	X	MUX 1	MUX 2	MUX 3
Unused states terminated to 000	0	0	0	0	0	0	0 or $\overline{X}$	$D_0=0$, $D_0=0$ | $D_0=0$	$D_0=0$, $D_0=0$ | $D_0=0$	$D_0=0$, $D_0=0$ | $D_0=0$
	0	0	0	0	0	0	1 or X			
	0	0	1	0	0	0	0 or $\overline{X}$	$D_1=0$, $D_1=0$ | $D_1=0$	$D_1=0$, $D_1=0$ | $D_1=0$	$D_1=0$, $D_1=0$ | $D_1=0$
	0	0	1	0	0	0	1 or X			
	0	1	0	0	0	0	0 or $\overline{X}$	$D_2=0$, $D_2=0$ | $D_2=0$	$D_2=0$, $D_2=0$ | $D_2=0$	$D_2=0$, $D_2=0$ | $D_2=0$
	0	1	0	0	0	0	1 or X			
	0	1	1	0	0	0	0 or $\overline{X}$	$D_3=0$, $D_3=0$ | $D_3=0$	$D_3=0$, $D_3=0$ | $D_3=0$	$D_3=0$, $D_3=0$ | $D_3=0$
	0	1	1	0	0	0	1 or X			
q_4	1	0	0	1	0	0	0 or $\overline{X}$	$D_4=1$, $D_4=1$ | $D_4=1$	$D_4=0$, $D_4=1$ | $D_4=X$	$D_4=0$, $D_4=1$ | $D_4=X$
	1	0	0	1	1	1	1 or X			
q_3	1	0	1	1	0	1	0 or $\overline{X}$	$D_5=1$, $D_5=1$ | $D_5=1$	$D_5=0$, $D_5=0$ | $D_5=0$	$D_5=1$, $D_5=0$ | $D_5=\overline{X}$
	1	0	1	1	0	0	1 or X			
q_2	1	1	0	1	1	0	0 or $\overline{X}$	$D_6=1$, $D_6=1$ | $D_6=1$	$D_6=1$, $D_6=0$ | $D_6=\overline{X}$	$D_6=0$, $D_6=1$ | $D_6=X$
	1	1	0	1	0	1	1 or X			
q_1	1	1	1	1	1	1	0 or $\overline{X}$	$D_7=1$, $D_7=1$ | $D_7=1$	$D_7=1$, $D_7=1$ | $D_7=1$	$D_7=1$, $D_7=0$ | $D_7=\overline{X}$
	1	1	1	1	1	0	1 or X			

Step 4 : Draw the logic diagram :

Fig. P. 7.4.3(c) shows the logic diagram for the realization using multiplexer technique.



(C-1612) Fig. P. 7.4.3(c) : Realization using multiplexer technique

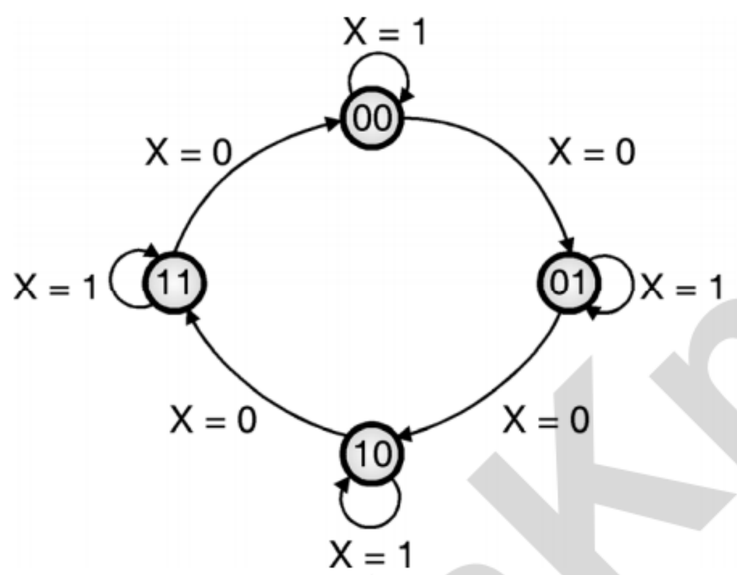
Ex. 7.4.4 : Draw ASM chart for the following state machine : A two bit UP counter with output 'Q_1Q_0' and enable signal 'X' is to be designed. If 'X' = 0, counter changes the state as '00-01-10-11-00'. If 'X' = 1, counter should remain in present state. Design your circuit using JK-FF and suitable MUXs.

May 07, 12 Marks, May 14, Dec. 14, 7 Marks, May 17, 8 Marks

Soln. :

Part I : ASM chart :

Step 1 : Draw the state diagram and state table :



(a) State diagram

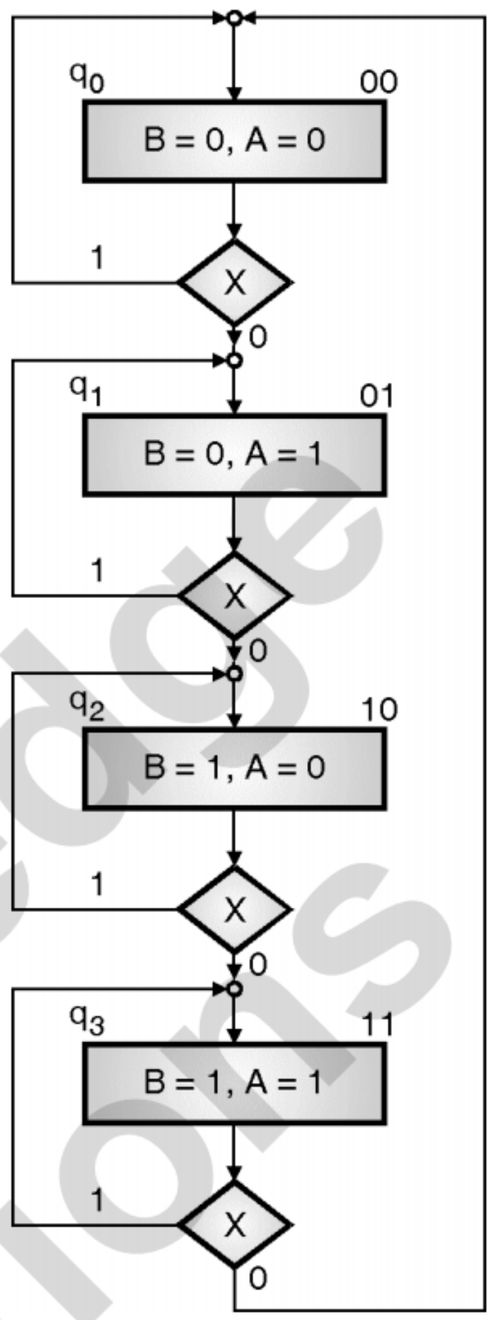
Present state	Input x = 0: Next state	Output BA	Input x = 1: Next state	Output BA
q_0	q_1	01	q_1	00
q_1	q_2	10	q_2	01
q_2	q_3	11	q_3	10
q_3	q_0	10	q_0	11

(b) State table

(C-1613) Fig. P. 7.4.4

Step 2 : Draw the ASM chart :

– The ASM chart is as shown in Fig. P. 7.4.4 (c).



(C-1614) Fig. P. 7.4.4(c) : ASM chart

Part II : Circuit design using JK FF and multiplexers :

– Since the output is 2 bit and there are 4 states, we have to use 2 JK flip flops and two 4 : 1 multiplexers.

– It is not necessary to use any decoder or combinational circuit.

– The FF outputs A and B are connected to the select inputs of MUX as shown in Fig. P. 7.4.4(d).

Step 3 : Write the table for deciding the MUX inputs :

– Refer Table P. 7.4.4 which shows the inputs to be applied to the inputs D_0 to D_3 of the two 4 : 1 multiplexers.

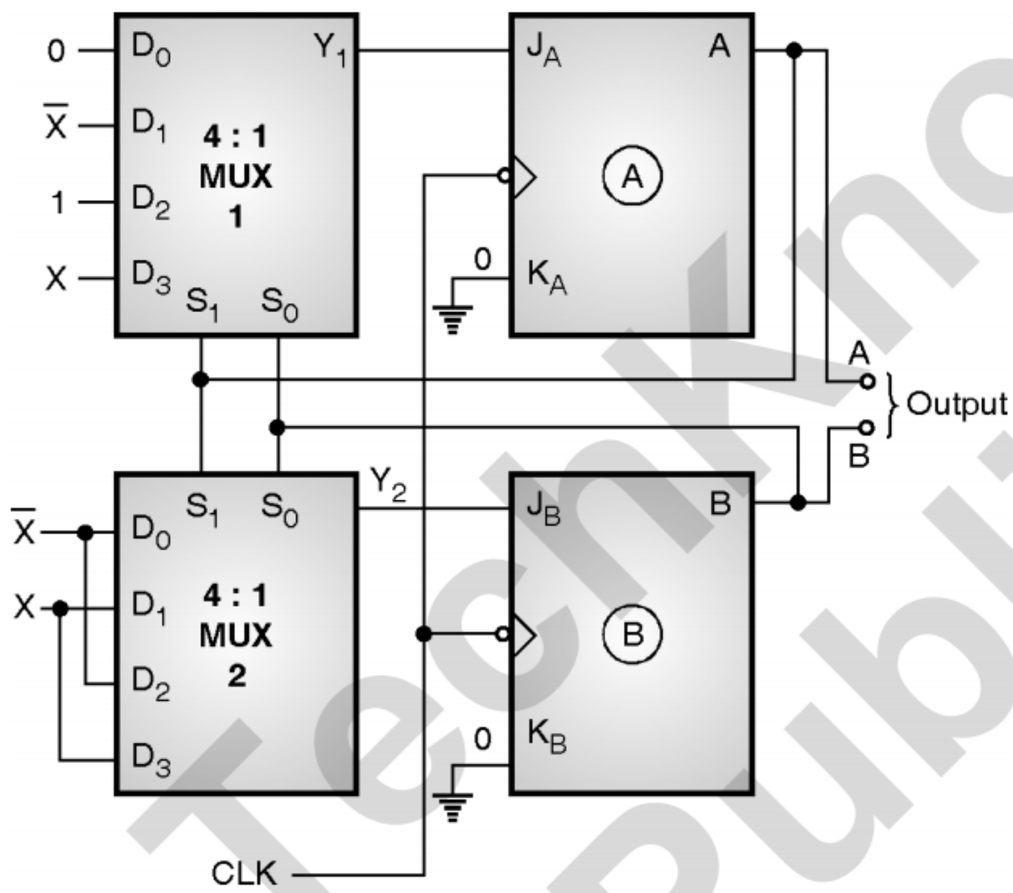
TechKnowledge Publications

(C-8127) Table P. 7.4.4 : Multiplexer input conditions

Present state		Next state		Input X	MUX input	
S_1	S_0	S_1	S_0		MUX 1	MUX 2
0	0	0	1	0 or $\overline{X}$	$D_0 = 0$	$D_0 = 1$
0	0	0	0	1 or X	$D_0 = 0$	$D_0 = 0$
					$D_0 = 0$	$D_0 = \overline{X}$
0	1	1	0	0 or $\overline{X}$	$D_1 = 1$	$D_1 = 0$
0	1	0	1	1 or X	$D_1 = 0$	$D_1 = 1$
					$D_0 = \overline{X}$	$D_1 = X$
1	0	1	1	0 or $\overline{X}$	$D_2 = 1$	$D_2 = 1$
1	0	1	0	1 or X	$D_2 = 1$	$D_2 = 0$
					$D_2 = 1$	$D_2 = \overline{X}$
1	1	0	0	0 or $\overline{X}$	$D_3 = 0$	$D_3 = 0$
1	1	1	1	1 or X	$D_3 = 1$	$D_3 = 1$
					$D_3 = X$	$D_3 = X$

Step 4 : Draw the logic diagram :

– Fig. P. 7.4.4(d) shows the logic diagram for the realization using JK FF and multiplexer.



(C-1615) Fig. P. 7.4.4(d) : Realization using JK FF and multiplexers

Ex. 7.4.5 : A sequential 'ring counter' with present state '0001'. The circuit also have an input 'X'. If X = 0, then circuit will show next output else for X = 1 it will show initial state '0001'. Draw an ASM chart and state table for this circuit to generate the output using mux controller method. **Dec. 09, 16 Marks**

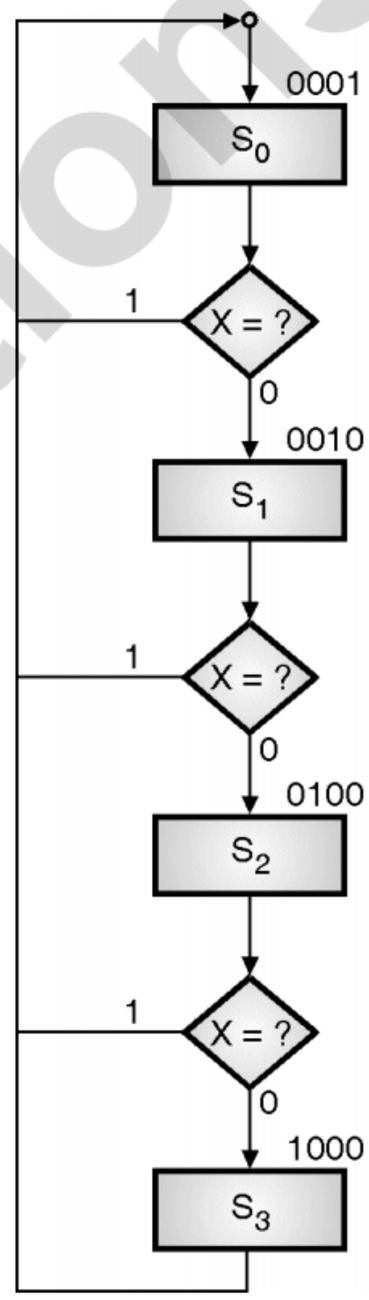
Soln. :

Given : Ring counter with initial state 0001.

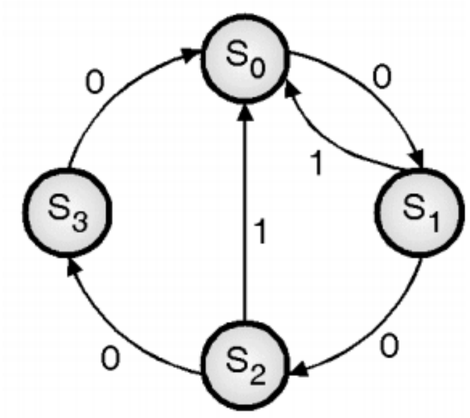
Input X if X = 0 ...Counter goes to next state

If X = 1 ...Counter shows initial state

– The required ASM chart and state diagram are as shown in Fig. P. 7.4.5(a) and (b).



(a) ASM chart



(b) State diagram

(C-3592) Fig. P. 7.4.5

Technical Knowledge Publications

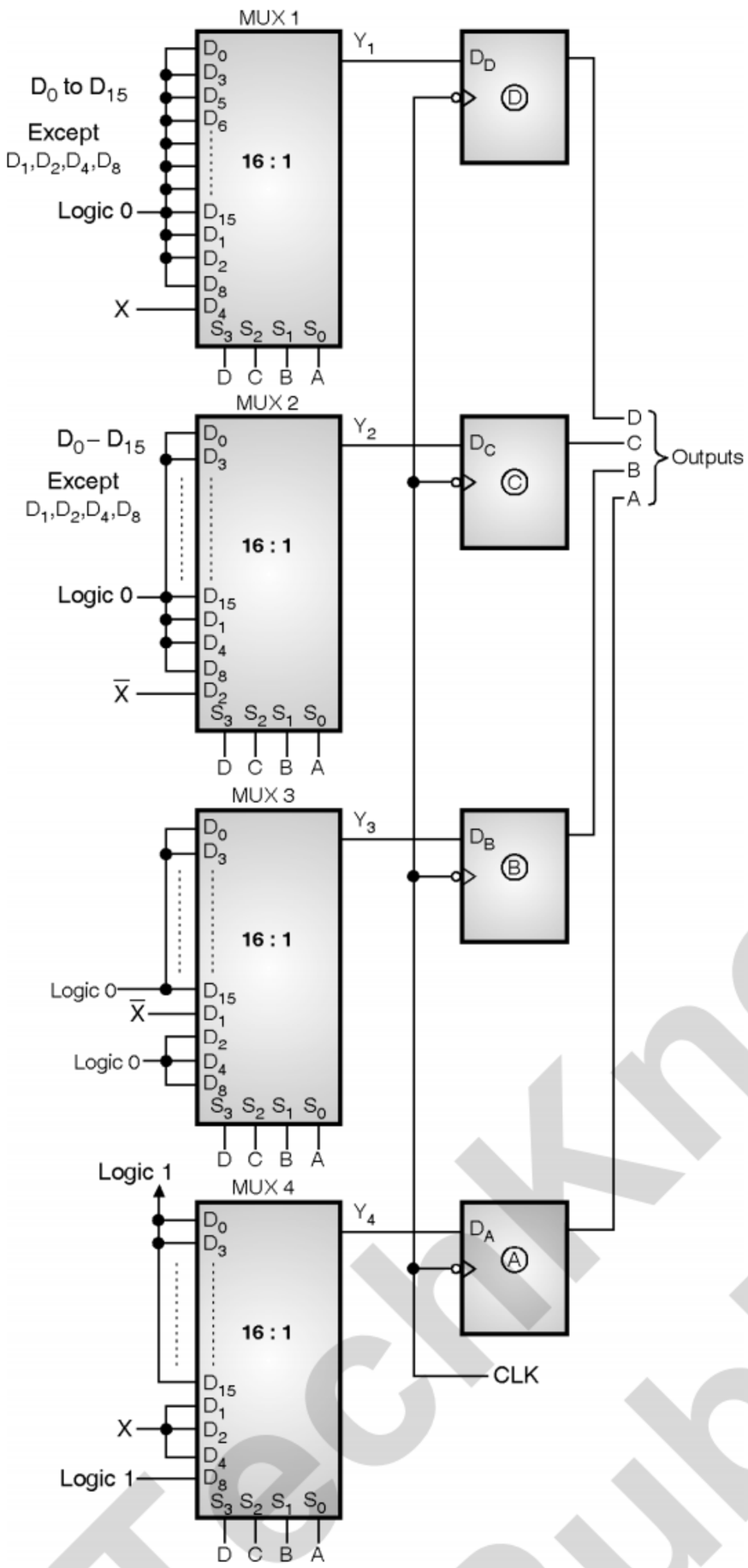
(C-3593) Table P. 7.4.5 : State table

	Present state				Input X	Next state				MUX inputs			
	D	C	B	A		D	C	B	A	MUX 1	MUX 2	MUX 3	MUX 4
Used states	0	0	0	1	0	0	0	1	0	$D_1=0$ | $D_1=0$	$D_1=0$	$D_1=1$ | $D_1=0$	$D_1=0$ | $D_1=1$
	0	0	0	1	1	0	0	0	1			$D_1=\overline{X}$	$D_1=X$
	0	0	1	0	0	0	1	0	0	$D_2=0$ | $D_2=0$	$D_2=1$ | $D_2=0$	$D_2=0$	$D_2=0$ | $D_2=1$
	0	0	1	0	1	0	0	0	1	$D_2=0$	$D_2=\overline{X}$		$D_2=X$
	0	1	0	0	0	1	0	0	0	$D_4=1$ | $D_4=0$	$D_4 = 0$	$D_4 = 0$	$D_4=0$ | $D_4=1$
	0	1	0	0	1	0	0	0	1	$D_4=X$			$D_4=X$
	1	0	0	0	0	0	0	0	1	$D_8=0$ | $D_8=0$	$D_8 = 0$	$D_8 = 0$	$D_8 = 1$
	1	0	0	0	1	0	0	0	1				
Unused states	0	0	0	0	0 or 1	0	0	0	1	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 1$
	0	0	1	1	0 or 1	0	0	0	1	$D_3 = 0$	$D_3 = 0$	$D_3 = 0$	$D_3 = 1$
	0	1	0	1	0 or 1	0	0	0	1	$D_5 = 0$	$D_5 = 0$	$D_5 = 0$	$D_5 = 1$
	0	1	1	0	0 or 1	0	0	0	1	$D_6 = 0$	$D_6 = 0$	$D_6 = 0$	$D_6 = 1$
	0	1	1	1	0 or 1	0	0	0	1	$D_7 = 0$	$D_7 = 0$	$D_7 = 0$	$D_7 = 1$
	1	0	0	1	0 or 1	0	0	0	1	$D_9 = 0$	$D_9 = 0$	$D_9 = 0$	$D_9 = 1$
	1	0	1	0	0 or 1	0	0	0	1	$D_{10} = 0$	$D_{10} = 0$	$D_{10} = 0$	$D_{10} = 1$
	1	0	1	1	0 or 1	0	0	0	1	$D_{11} = 0$	$D_{11} = 0$	$D_{11} = 0$	$D_{11} = 1$
	1	1	0	0	0 or 1	0	0	0	1	$D_{12} = 0$	$D_{12} = 0$	$D_{12} = 0$	$D_{12} = 1$
	1	1	0	1	0 or 1	0	0	0	1	$D_{13} = 0$	$D_{13} = 0$	$D_{13} = 0$	$D_{13} = 1$
	1	1	1	0	0 or 1	0	0	0	1	$D_{14} = 0$	$D_{14} = 0$	$D_{14} = 0$	$D_{14} = 1$
	1	1	1	1	0 or 1	0	0	0	1	$D_{15} = 0$	$D_{15} = 0$	$D_{15} = 0$	$D_{15} = 1$

All the unused states are terminated to initial state i.e. 0001

Tech Knowledge Publications

Logic diagram :



(C-3594) **Fig. P. 7.4.5(c) : Realization using multiplexers**

Ex. 7.4.6 : With the help of an ASM chart design a modulo 6 up-down counter.

May 10, 10 Marks

Soln. :

– The solution of this example is similar to that of Ex. 7.4.3 except one change. This counter will count upto 101 instead of 111.

Ex. 7.4.7 : A sequential ring counter with present state '01011'. The circuit also have an input 'Z'. If Z = 0, circuit shows next output (right shift) else for Z = 1, it shows initial state. Draw an ASM chart and state stable for this circuit to generate the output mux controller method.

Dec. 10, 16 Marks

Soln. : Very similar to Ex. 7.4.5.

Ex. 7.4.8 : Design a sequence generator circuit to generate the sequence 1-3-5-7 using multiplexer controller based ASM approach. Consideration :

1. If control input C = 0, the sequence generator circuit in the same state.
2. If control input C = 1, the sequence generator circuit goes into next state.

Dec. 13, 7 Marks

Soln. :

Part 1 : ASM chart and state table :

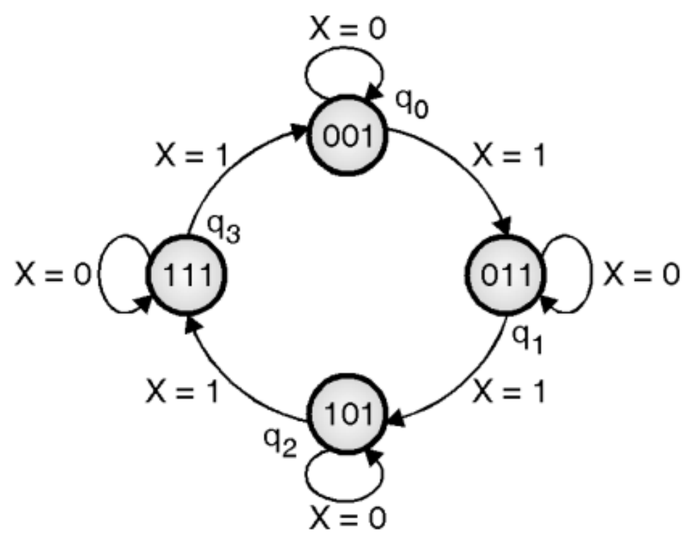
Step 1 : Draw the state diagram and state table :

– Let the four states through which the circuit goes be denoted by q_0, q_1, q_2 and q_3.

– The relation between these states and the outputs is as follows,

– Now draw the state diagram as shown in Fig. P. 7.4.8(a) and then write the state table as shown in Table P. 7.4.8(a). **(C-8130)**

State	Output
q_0	0 0 1
q_1	0 1 1
q_2	1 0 1
q_3	1 1 1



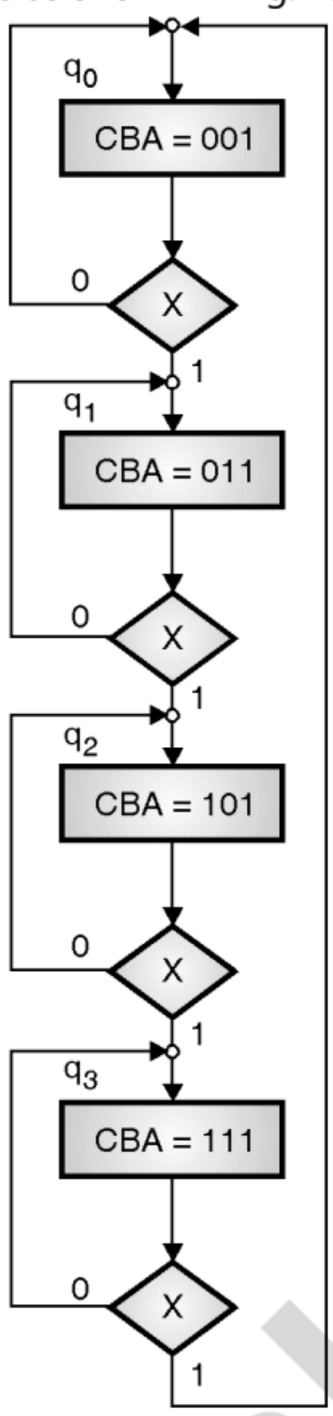
(C-4194) **Fig. P. 7.4.8(a) : State diagram**

Table P. 7.4.8(a) : State table

Present state	Input X = 0		Input X = 1	
	Next state	Output	Next state	Output
q_0	q_0	001	q_1	011
q_1	q_1	011	q_2	101
q_2	q_2	101	q_3	111
q_3	q_3	111	q_0	001

Step 2 : Draw the ASM chart :

– The ASM chart is as shown in Fig. P. 7.4.8(b).



(C-4195) **Fig. P. 7.4.8(b) : ASM chart**

Part 2 : Design the circuit using multiplexers :

– We have to use three flip-flops because the output is 3 bit (001 to 111).

– So we have to use three 8 : 1 multiplexers.

– It is not necessary to use any decoder or combinational circuit after flip-flops.

– We will use the D flip-flops. MUX outputs are connected to the D flip-flop inputs as shown in Fig. P. 7.4.8(c).

– The flip-flop outputs A, B and C are connected to the select inputs S_2, S_1 and S_0 as shown in Fig. P. 7.4.8(c).

Step 3 : Write the table for deciding MUX input :

– Refer Table P. 7.4.8(b) which shows the inputs to be applied to the inputs D_0 to D_7 of all the three 8 : 1 multiplexers.

– Note that the unused present states i.e. 000, 010, 100 and 110 are terminated into next state of 000.

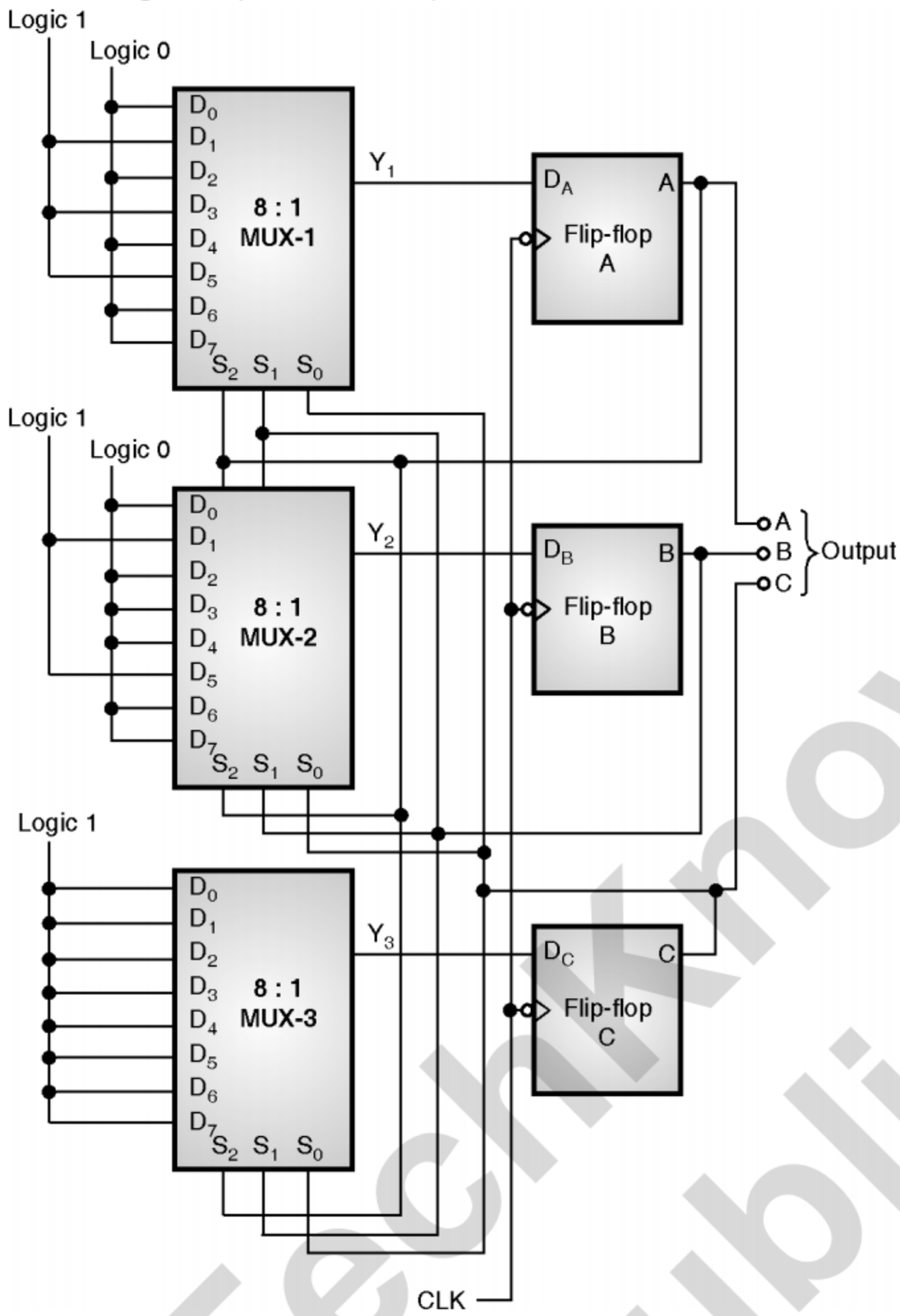
(C-6274) **Table P. 7.4.8(b) : Multiplexer input conditions**

	Present state			Next state			Input	MUX inputs					
	C	B	A	C	B	A	X	MUX 1		MUX 2		MUX 3	
Used states	0	0	1	0	0	1	0	$D_1 = 0$	$D_1 = 1$	$D_1 = 0$	$D_1 = 1$	$D_1 = 1$	$D_1 = 1$
	0	0	1	1	1	1	1	$D_1 = 1$		$D_1 = 1$		$D_1 = 1$	
	0	1	1	0	1	1	0	$D_3 = 0$	$D_3 = 1$	$D_3 = 1$	$D_3 = 0$	$D_3 = 1$	$D_3 = 1$
	0	1	1	1	0	1	1	$D_3 = 1$		$D_3 = 0$		$D_3 = 1$	
	1	0	1	1	0	1	0	$D_5 = 1$	$D_5 = 1$	$D_5 = 0$	$D_5 = 1$	$D_5 = 1$	$D_5 = 1$
	1	0	1	1	1	1	1	$D_5 = 1$		$D_5 = 1$		$D_5 = 1$	
	1	1	1	1	1	1	0	$D_7 = 1$	$D_7 = 0$	$D_7 = 1$	$D_7 = 0$	$D_7 = 1$	$D_7 = 1$
	1	1	1	0	0	1	1	$D_7 = 0$		$D_7 = 0$		$D_7 = 1$	
Unused states	0	0	0	0	0	1	0 OR	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 1$
	0	0	0	0	0	1	1	$D_0 = 0$		$D_0 = 0$		$D_0 = 1$	
	0	1	0	0	0	1	0 OR	$D_2 = 0$	$D_2 = 0$	$D_2 = 1$	$D_2 = 0$	$D_2 = 0$	$D_2 = 1$
	0	1	0	0	0	1	1	$D_2 = 0$		$D_2 = 0$		$D_2 = 1$	
	1	0	0	0	0	1	0 OR	$D_4 = 1$	$D_4 = 0$	$D_4 = 0$	$D_4 = 0$	$D_4 = 0$	$D_4 = 1$
	1	0	0	0	0	1	1	$D_4 = 0$		$D_4 = 0$		$D_4 = 1$	
	1	1	0	0	0	1	0 OR	$D_6 = 1$	$D_6 = 0$	$D_6 = 1$	$D_6 = 0$	$D_6 = 0$	$D_6 = 1$
	1	1	0	0	0	1	1	$D_6 = 0$		$D_6 = 0$		$D_6 = 1$	

Tech Knowledge Publications

Step 4 : Draw the logic diagram :

– Fig. P. 7.4.8(c) shows the logic diagram for realization using multiplexer technique.



(C-4196) **Fig. P. 7.4.8(c)**

Ex. 7.4.9 : Design a sequence generator circuit to generate the sequence 1-2-3-7-1 using multiplexer controller based ASM approach. Consideration :

1. If control input C = 0, the sequence generator circuit in the same state.
2. If control input C = 1, the sequence generator circuit goes into next state.

May 15, 7 Marks

Soln. :

Part 1 : ASM chart and state table :

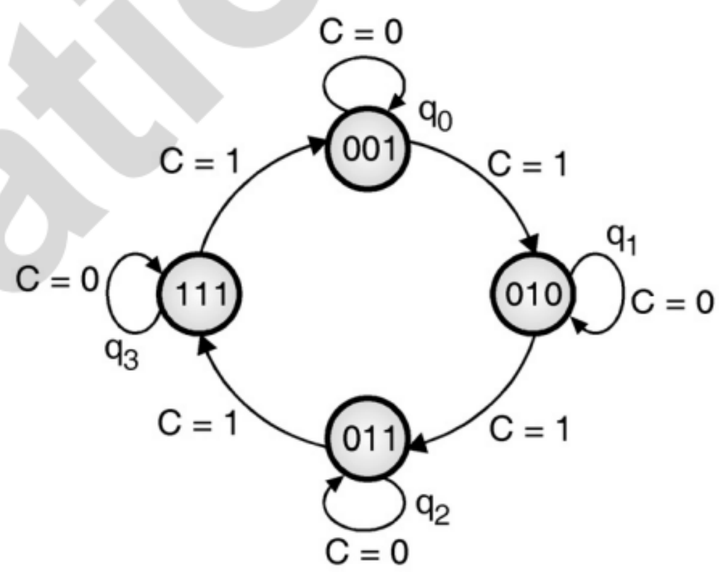
Step 1 : Draw the state diagram and state table :

– Let the four states through which the circuit goes be denoted by q_0, q_1, q_2 and q_3. The relation between these states and the outputs is as follows.

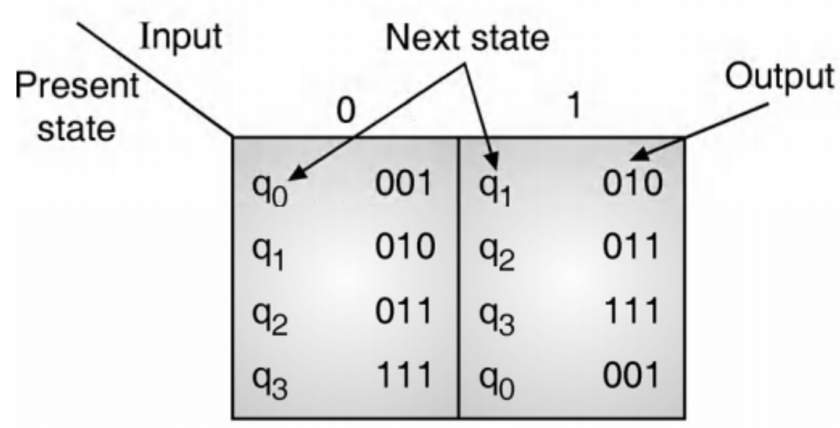
– Now draw the state diagram as shown in Fig. P. 7.4.9(a) and then write the state table as shown in Table P. 7.4.9(a).

(C-8131)**Table P. 7.4.9(a)**

State	Output
q_0	0 0 1
q_1	0 1 0
q_2	0 1 1
q_3	1 1 1



(a) : State diagram

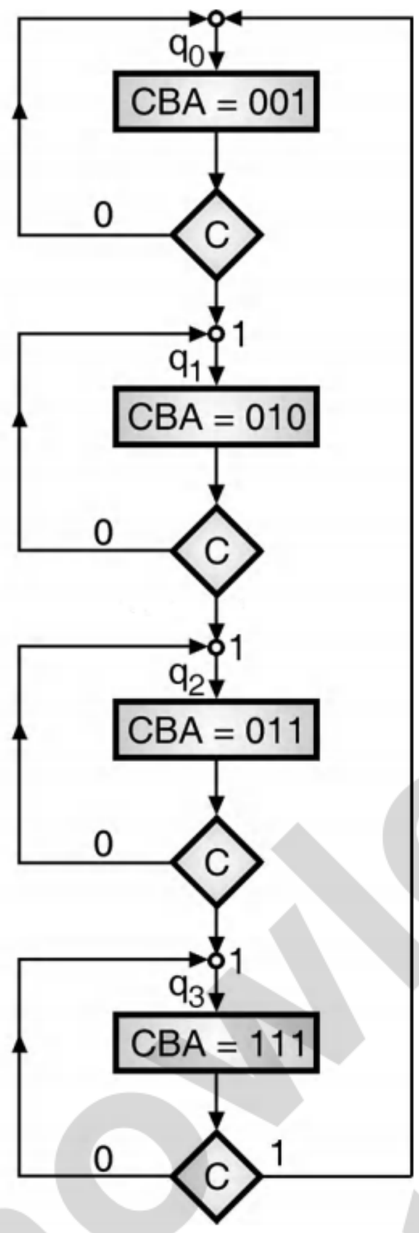


(b) : State table

Fig. P. 7.4.9

Step 2 : Draw the ASM chart :

– The ASM chart is as shown in Fig. P. 7.4.9(b).



(C-5105) **Fig. P. 7.4.9(b) : ASM chart**

Part 2 : Design the circuit using multiplexers :

- To use three flip-flops because the output is 3 bit (001 to 111).
- So have to use three 8 : 1 multiplexers. It is not necessary to use any decoder or combinational circuit after flip-flops. That have to use the D flip-flops.
- MUX outputs are connected to the D flip-flop inputs as shown in Fig. P. 7.4.9(c).
- The flip-flop outputs A, B and C are connected to the select inputs S_2, S_1 and S_0 as shown in Fig. P. 7.4.9(c).

Step 3 : Write the table for deciding MUX input :

- Refer Table P. 7.4.9(c) which shows the inputs to be applied to the inputs D_0 to D_7 of all the three 8 : 1 multiplexers.
- Note that the unused present states i.e. 000, 100, 101, 110 are terminated into next state of 000.

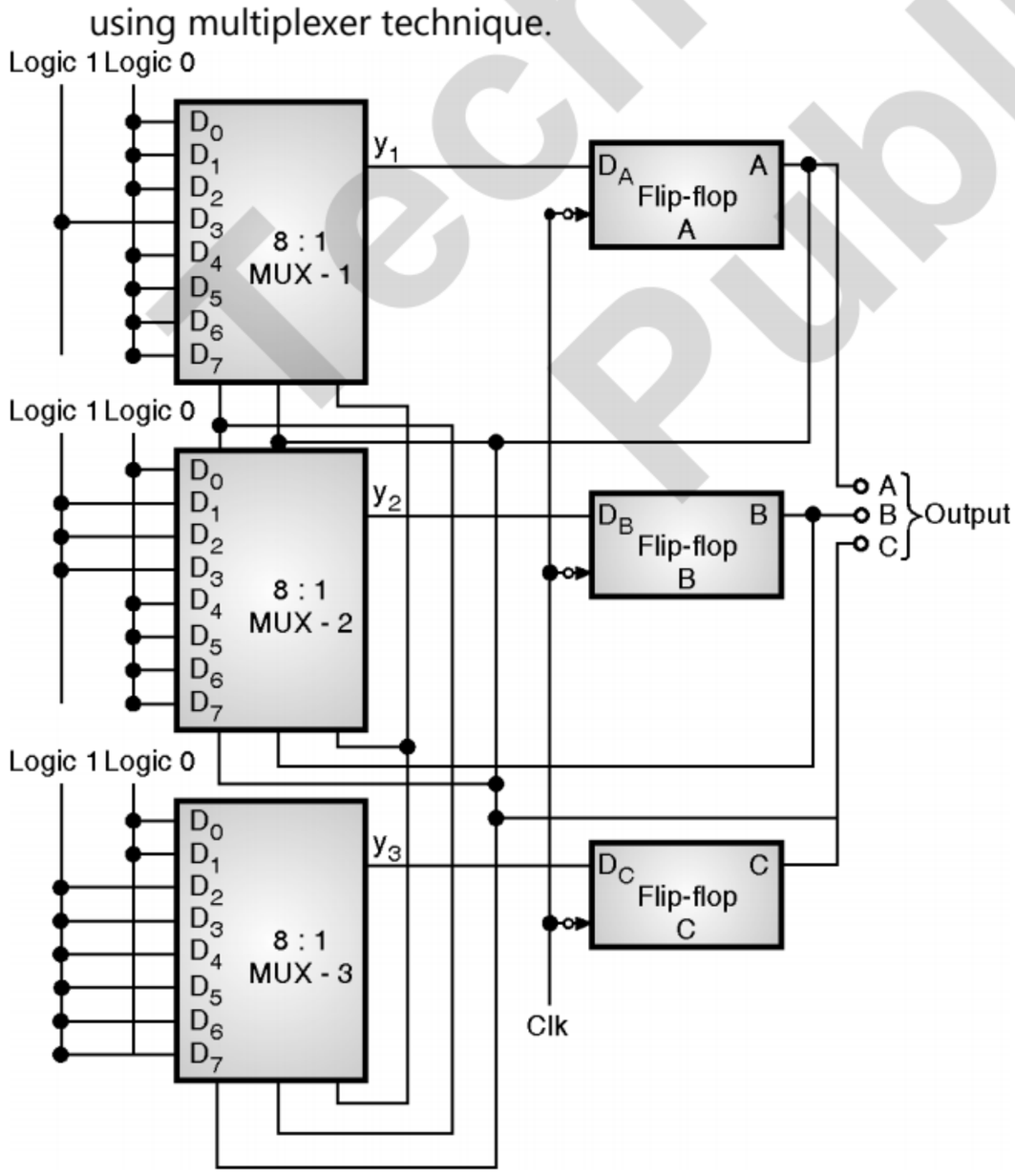
Tech Knowledge Publications

(C-6275) Table P. 7.4.9(c) : Multiplexer input conditions

	Present state			Next state			Input	MUX inputs					
	C	B	A	C	B	A	X	MUX 1		MUX 2		MUX 3	
Used states	0	0	1	0	0	1	0	$D_1 = 0$	$D_1 = 0$	$D_1 = 0$	$D_1 = 1$	$D_1 = 1$	$D_1 = 0$
	0	0	1	0	1	0	1	$D_1 = 0$		$D_1 = 1$		$D_1 = 0$	
	0	1	0	0	1	0	0	$D_2 = 0$	$D_2 = 0$	$D_2 = 1$	$D_2 = 1$	$D_2 = 0$	$D_2 = 1$
	0	1	0	0	1	1	1	$D_2 = 0$		$D_2 = 1$		$D_2 = 1$	
	0	1	1	0	1	1	0	$D_3 = 0$	$D_3 = 1$	$D_3 = 1$	$D_3 = 1$	$D_3 = 1$	$D_3 = 1$
	0	1	1	1	1	1	1	$D_3 = 1$		$D_3 = 1$		$D_3 = 1$	
	1	1	1	1	1	1	0	$D_7 = 1$	$D_7 = 0$	$D_7 = 1$	$D_7 = 0$	$D_7 = 1$	$D_7 = 1$
	1	1	1	0	0	1	1	$D_7 = 0$		$D_7 = 0$		$D_7 = 1$	
Unused states	0	0	0	0	0	1	0 OR 1	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 0$	$D_0 = 1$
	0	0	0	0	0	1		$D_0 = 0$		$D_0 = 0$		$D_0 = 1$	
	1	0	0	0	0	1	0 OR 1	$D_2 = 1$	$D_4 = 0$	$D_2 = 0$	$D_4 = 0$	$D_2 = 0$	$D_4 = 1$
	1	0	0	0	0	1		$D_2 = 0$		$D_2 = 0$		$D_2 = 1$	
	1	0	1	0	0	1	0 OR 1	$D_4 = 1$	$D_5 = 0$	$D_4 = 0$	$D_5 = 0$	$D_4 = 1$	$D_5 = 1$
	1	0	1	0	0	1		$D_4 = 0$		$D_4 = 0$		$D_4 = 1$	
	1	1	0	0	0	1	0 OR 1	$D_6 = 1$	$D_6 = 0$	$D_6 = 1$	$D_6 = 0$	$D_6 = 0$	$D_6 = 1$
	1	1	0	0	0	1		$D_6 = 0$		$D_6 = 0$		$D_6 = 1$	

Step 4 : Draw the logic diagram :

– Fig. P. 7.4.9(c) shows the logic diagram for realization using multiplexer technique.



(C-5106) Fig. P. 7.4.9(c)

Review Questions

Q. 1 Define ASM.

Q. 2 State various types of boxes used for drawing the ASM charts.

Q. 3 Define and explain about the state box.

Q. 4 Define and explain the decision and conditional box.

Q. 5 Compare the ASM chart and state diagram.

Q. 6 Demonstrate step by step use of ASM charts for designing sequential circuits.

Q. 7 Explain the K-map method for realizing the logic circuit from the ASM chart.

Q. 8 Explain the multiplexer method to realize the logic circuit from the ASM chart.

❑❑❑

TechKnowledge Publications

Unit IV

Chapter 8

Programmable Logic Devices

TechKnowledge Publications

Syllabus

PLD, ROM as PLD, Programmable Logic Array (PLA), Programmable Array Logic (PAL), Designing combinational circuits using PLDs.

Chapter Contents

8.1	Introduction to PLDs
8.2	ROM used as PLD
8.3	Programmable Logic Array (PLA)
8.4	Programmable Array Logic (PAL)
8.5	Comparison of PROM, PLA and PAL

8.1 Introduction to PLDs :

SPPU : May 06, Dec. 06, May 08, Dec. 08, May 12, Dec. 18

University Questions

Q. 1 What is PLD ? What do you mean by designing using PLDs ? Explain with suitable examples. **(May 06, 8 Marks)**

Q. 2 What is PLD ? **(Dec. 06, May 08, Dec. 08, 2 Marks)**

Q. 3 Define PLD. Mention different types of PLD. **(May 12, Dec. 18, 2 Marks)**

– We have discussed the combinational and sequential circuits.

– We have also discussed different ICs such as multiplexers, comparators, adders, code converters, demultiplexers etc.

– Such ICs are called as **fixed function ICs**, because they perform only a specific function.

Advantages of fixed function ICs :

– The advantages of using fixed function ICs for designing logic circuits are :

1. It is easy to test the circuits.
2. The cost of development is low.

Disadvantages of fixed function ICs :

– But there are disadvantages of fixed function ICs such as :

1. They need large board space.
2. They consume large power.
3. They do not provide any security.
4. Additions costs need to be incurred if the logic circuit is modified.

– The remedy to this problem is to use the Application Specific Integrated Circuits (ASICs).

– These ICs are designed by keeping in mind the specific requirements of a user or application. But the design of ASICs is too complex.

Advantages of fixed function ASICs :

– The advantages of ASICs are as follows :

1. They reduce the cost of design.
2. They need less space.
3. They provide better security. 4. They need less power.

Disadvantages of fixed function ICs :

– But the disadvantages of ASIC are :

1. Their initial development cost is very high.
2. The cost and efforts will increase further because new testing methods need to be used.

Use of PLDs :

– Using fixed function ICs and using ASIC were the two approaches discussed earlier.

– This third approach to the problem of digital circuit design is to use the Programmable Logic Devices (PLDs).

– This approach has the advantages of both the approaches discussed earlier.

What is a PLD ?

– PLD are special type of ICs which can be programmed by the users as per their requirements.

– Therefore it is possible to implement almost any combinational or sequential circuit using the PLD ICs.

8.1.1 Advantages of PLDs : SPPU : May 06

University Questions

Q. 1 What are the advantages of designing using PLDs over designing discrete LSI components ? Explain. **(May 06, 2 Marks)**

– Following are some of the advantages of PLDs :

1. They need less amount of space.
2. They need less power for operation.
3. Their speed of switching is high.
4. They ensure higher security (copying is less likely to take place).
5. PLDs have higher component density.
6. PLDs can be produced at lower costs.
7. PLDs have lower development cost as well.
8. The time required to implement the design is short.
9. PLDs provide more flexibility to the designer.
10. It is possible to reprogram PLDs very quickly.

8.1.2 Types of PLDs :

SPPU : May 12, Dec. 13, Dec. 18

University Questions

Q. 1 Mention different types of PLD. **(May 12, 2 Marks)**

Q. 2 What are the different types of PLDs ? Design 3 : 8 decoder using PLD. **(Dec. 13, 7 Marks)**

Q. 3 Define PLD. Mention different types of PLD. **(Dec. 18, 2 Marks)**

– Various types of PLDs are as follows :

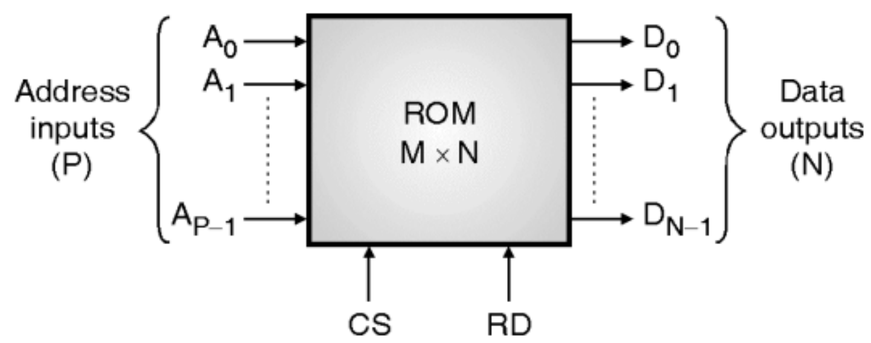
1. Read Only Memories (ROMs).
2. Programmable Logic Array (PLAs).
3. Programmable Array Logic (PALs).
4. Simple Programmable Logic Devices (SPLDs).

TechKnowledge Publications

5. Complex Programmable Logic Devices (CPLDs).
6. Field Programmable Gate Arrays (FPGAs).

8.2 ROM used as PLD :

- ROM stands for Read-Only-Memory. It can be programmed only once by the manufacturer as per the requirements of the user.
- A ROM of size M × N is shown in Fig. 8.2.1. It has M number of locations and each location can store an N bit word.



(C-510) Fig. 8.2.1 : Block diagram of a ROM

- There are P number of address lines which we can use to access M locations and the relation between P and M is given by,

  Number of locations $M = 2^P$
- There are N number of data lines, on which the stored data will appear, when the read (RD) input is active.
- CS is the chip select pin which has to be made active in order to enable to ROM.
- We can use the ROM to implement combinational circuits. This will be clear after solving the following example.

Ex. 8.2.1 : Draw the logic diagram of a circuit having the truth table of Table P. 8.2.1, using a 16 × 1 ROM. **Dec. 13, 7 Marks**

(C-6276) Table P. 8.2.1 : Truth table

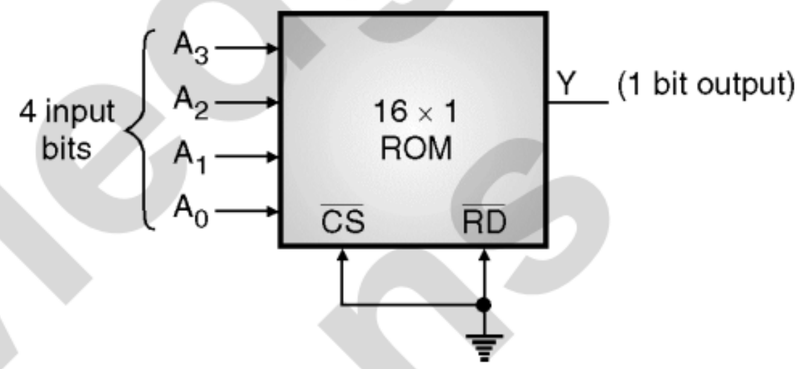
Inputs				Output
A_3	A_2	A_1	A_0	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Treat these as addresses → Store the bits in this sequence into the ROM

Soln. :

- Use a ROM of 16 × 1 capacity i.e. the one with M = 16, N = 1 and P = 4.
- The four address inputs A_3 A_2 A_1 A_0 can be treated as the four inputs of the required combinational logic and the bit pattern of Table P. 8.2.1 should be stored in the ROM.
- Hence the logic function corresponding to this is given by,

  $$Y = \sum m\,(0, 3, 4, 5, 8, 9, 10, 13)$$
- The realization using a ROM is shown in Fig. P. 8.2.1.



(C-511) Fig. P. 8.2.1

- In general if we want to implement a P variable, N output logic function, then we have to use a ROM of size (2^P × N).
- This will generate all the possible 2^P minterms.

8.2.1 Types of ROMs :

- Read only memories can be of different types. The different types of ROMs are as given below :

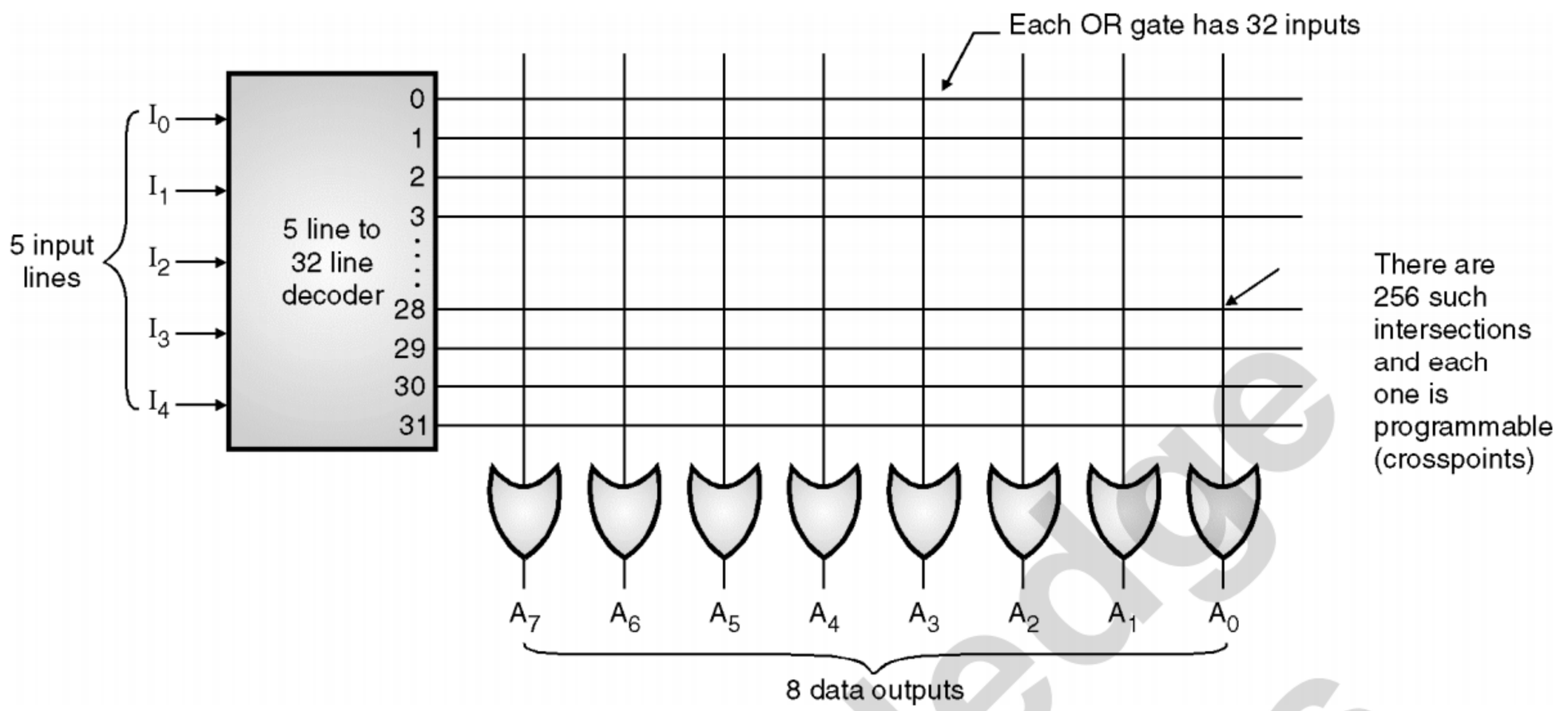
  1. PROM. 2. EPROM. 3. E^2PROM.
- At this juncture we can say that user can program the ROM if it is PROM, EPROM or E^2PROM.
- As we can use the programmable ROMs for logic design, it is called as a programmable logic device.

8.2.2 Internal Logic of a ROM :

Logic diagram :

- Let us understand the internal logic (organization) of a 32 × 8 ROM.
- A 32 × 8 ROM consists of 32 locations and at each location we can store an 8-bit word.
- To access these 32 locations, we have to use 5 input lines (address lines) to which we apply the binary words equivalent to 0 to 31 i.e. 00000 to 11111.
- Fig. 8.2.2 shows the internal structure of a ROM.
- The five bit input (00000 to 11111) is decoded into 32 lines with the help of a 5 – 32 line decoder. Each output of the decoder represents a memory address.

TechKnowledge Publications



(C-513) Fig. 8.2.2 : Internal structure of a 32 × 8 ROM

- The 32 outputs of the decoder are connected to each of the eight OR gates. That means each OR gate has 32 inputs.
- Since each OR gate has 32 connections and there are 8 OR gates, there will be 32 × 8 = 256 internal connections.
- All the 256 intersections (cross points) are programmable. What is meant by a programmable intersection ?
- It means that each intersection (see Fig. 8.2.2) is equivalent to a switch which can be either turned ON or OFF.
- An ON switch represents connection between the two intersecting lines whereas an OFF switch represents that there is no connection between those lines.
- The programmable intersection is also called as a crosspoint.

Implementation of the crosspoint switches :

- The crosspoints can be implemented with the help of various physical devices. But the simplest method is to use a **fuse link** that is normally intact and hence connects the two lines. Therefore it is equivalent to a closed switch under the normal operating conditions.
- Whenever the switch is to be open circuited, we have to blow this fuse.

ROM truth table :

- The ROM truth table is shown in Table 8.2.1.
- The data stored in the ROM is specified by the truth table. This table gives us the 8 bit words stored at each location.
- The truth table shows that there are 5 inputs and 8 outputs.
- Then there are 32 locations (addresses) and at each location an 8 bit word has been stored.

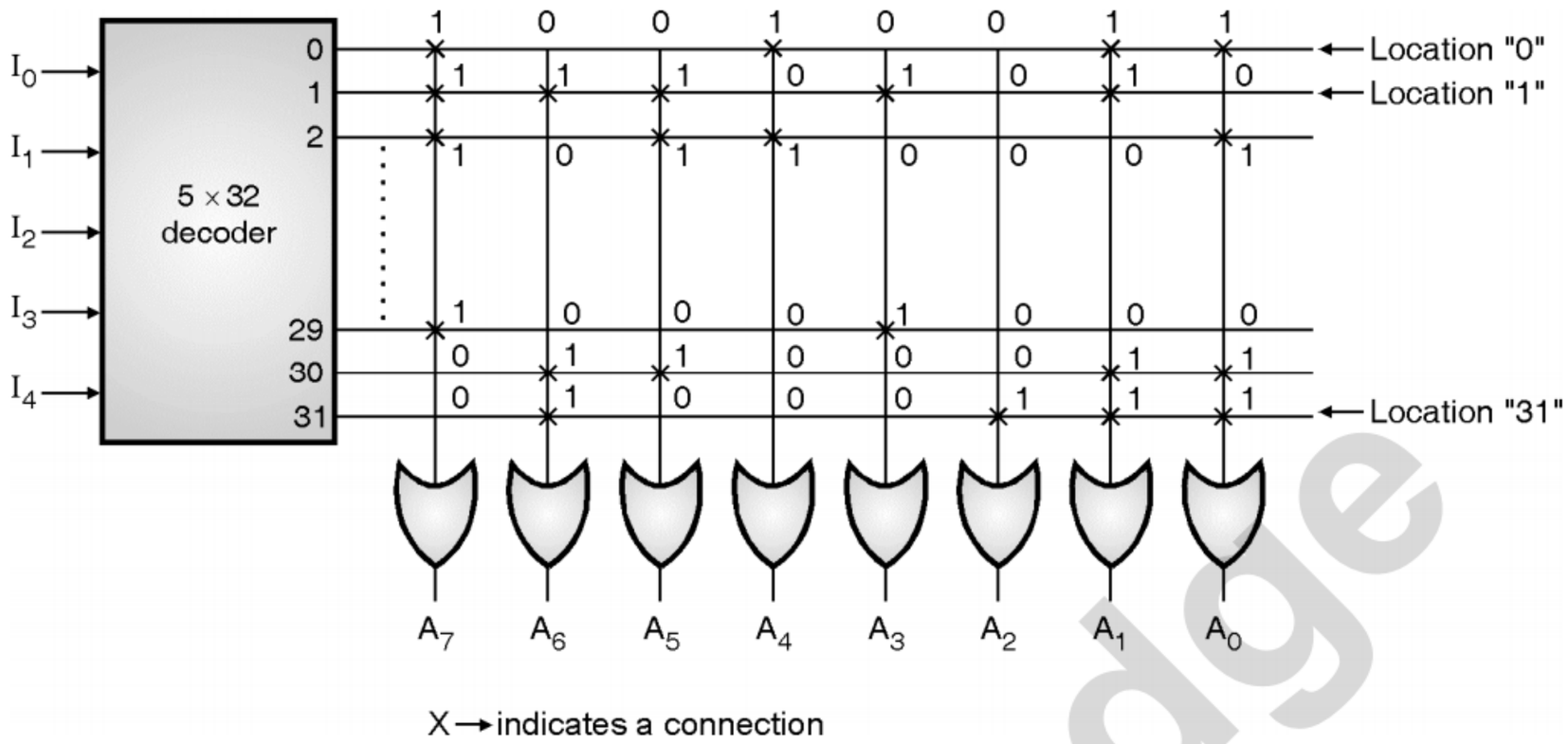
(C-6408) Table 8.2.1 : ROM truth table

Location number	Inputs					Outputs								
	I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
0	0	0	0	0	0	1	0	0	1	0	0	1	1	← Data in location 0
1	0	0	0	0	1	1	1	1	0	1	0	1	0	
2	0	0	0	1	0	1	0	1	1	0	0	0	1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
29	1	1	1	0	1	1	0	0	0	1	0	0	0	
30	1	1	1	1	0	0	1	1	0	0	0	1	1	
31	1	1	1	1	1	0	1	0	0	0	1	1	1	← Data in location 31

Programming a ROM :

- In the process of ROM programming, the fuse links need to be blown according to the truth table of ROM.
- Every 0 in the truth table indicates a no connection and every 1 indicates a connection.
- The ROM programmed according to truth table of Table 8.2.1 results in the configuration shown in Fig. 8.2.3.

TechKnowledge Publications



(C-514) Fig. 8.2.3 : ROM programmed according to Table 8.2.1

– Consider location "0" of Fig. 8.2.3. The word 10010011 has been stored in this location. The "0"s are stored by blowing off the fuse links between output "0" of the decoder and the input lines of OR gates, associated with A_6, A_5, A_3 and A_2. The four "1"s are marked by "×".

8.2.3 Implementation of a Combinational Circuit (Generating the Boolean Function) :

– To implement a combinational circuit using PROM, each output (A_7 through A_0) should be considered separately because the output of a Boolean function is expressed in SOP form (See Fig. 8.2.3)

– That means the ROM can be considered as a combinational circuit with 5-input variables and 8 outputs.

– The outputs can be expressed as

$$A_7 (I_4, I_3, I_2, I_1, I_0) = \Sigma m (0, 1, 2, \ldots 29) \text{ etc.}$$

– The mark "×" produces one minterm (See Fig. 8.2.3).

– The following example illustrates the use of ROM for implementation of a combinational circuit.

Ex. 8.2.2 : Design using PLD a 3 : 8 decoder.

May 10, Dec. 11, 8 Marks, Dec. 13, 4 Marks

Soln. :

Step 1 : Truth table :

– We shall use ROM as PLD. The truth table of a 3:8 decoder is as follows :

(C-6425) Table P. 8.2.2 : Truth table for 3 to 8 line decoder

Inputs			Outputs								
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \bar{A}\bar{B}C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \bar{A}B\bar{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \bar{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\bar{B}C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = ABC$

Step 2 : ROM program table :

– The required ROM is a 3 input variable ROM. They are A, B, C.

– Hence there will be 2^3 = 8 memory locations. Each location can store an 8 bit word.

– There will be 8 outputs say D_7, D_6 ... D_0.

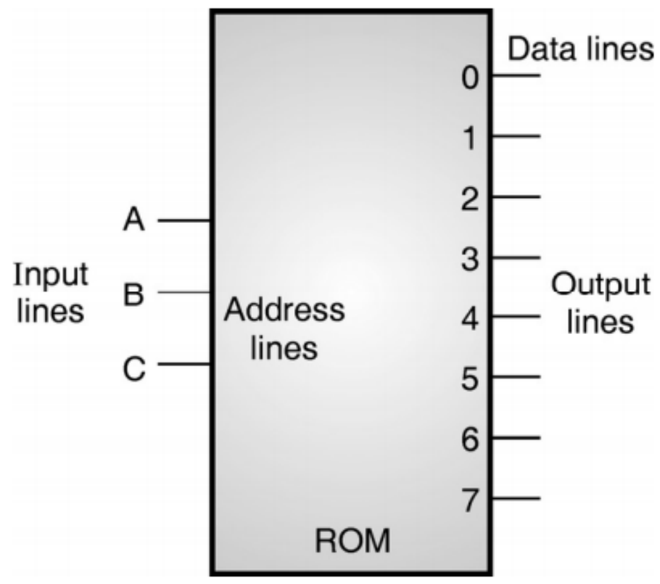
(C-8143) Table P. 8.2.2(a) : ROM programming table

Location no.	Inputs			Outputs							
	A	B	C	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	0	**1**
1	0	0	1	0	0	0	0	0	0	**1**	0
2	0	1	0	0	0	0	0	0	**1**	0	0
3	0	1	1	0	0	0	0	**1**	0	0	0
4	1	0	0	0	0	0	**1**	0	0	0	0
5	1	0	1	0	0	**1**	0	0	0	0	0
6	1	1	0	0	**1**	0	0	0	0	0	0
7	1	1	1	**1**	0	0	0	0	0	0	0

Step 3 : Logic diagram :

– The logic diagram is as shown in Fig. P. 8.2.2(a).

TechKnowledge Publications



(C-1752) **Fig. P. 8.2.2(a) : 3 : 8 decoder using PROM**

8.2.4 Advantages of ROM as PLD :

– Following are some of the advantages of ROM used as PLD :

1. It is extremely easy to implement and design a combinational circuit using ROM.
2. It is possible to change or modify the design quickly.
3. The cost of implementing a circuit using ROM is very low.
4. We can modify the implemented circuit very quickly.

8.2.5 Disadvantages of ROM as PLD :

– Some of the disadvantages of ROM as PLD are as follows :

1. The power requirement of ROM is high.
2. Complete circuit is not utilized.
3. If there is an increase in number of input variables, then the size of ROM becomes impracticably large.

8.3 Programmable Logic Array (PLA) :

SPPU : Dec. 07, Dec. 08, Dec. 09, May 14, May 17

University Questions

Q. 1 Draw and explain the general structure of PLA. **(Dec. 07, Dec. 08, Dec. 09, 4 Marks)**

Q. 2 What is PLA ? Explain the input buffer AND and OR matrix in PLA. **(May 14, 6 Marks)**

Q. 3 Draw block diagram of PLA device and explain. **(May 17, 6 Marks)**

Definition :

– A programmable logic array (PLA) is a type of logic device that can be programmed to implement various kinds of combinational logic circuits.

– The device has a number of AND and OR gates which are linked together to give output or further combined with more gates or logic circuits.

– A PLD generally consists of a group of logic gates. Such a group is called as an array. The arrays of logic gates used in PLDs are programmable arrays.

Block diagram :

– The block diagram of a PLA device is shown in Fig. 8.3.1.

– The logic gates used in a PLA can be the two level logic configurations of AND-OR, NAND-NAND or NOR-NOR types.

– Sometimes AND-OR-EXOR configuration is also used. In Fig. 8.3.1 the two level AND-OR configuration is being used.

– There are two types of PLDs namely the PLA i.e. programmable logic arrays and PAL i.e. the programmable array logic.

– As shown in Fig. 8.3.1 the AND matrix is followed by OR matrix. Therefore, we can use PLA or PAL for the implementation of logic functions that are expressed in the SOP form.

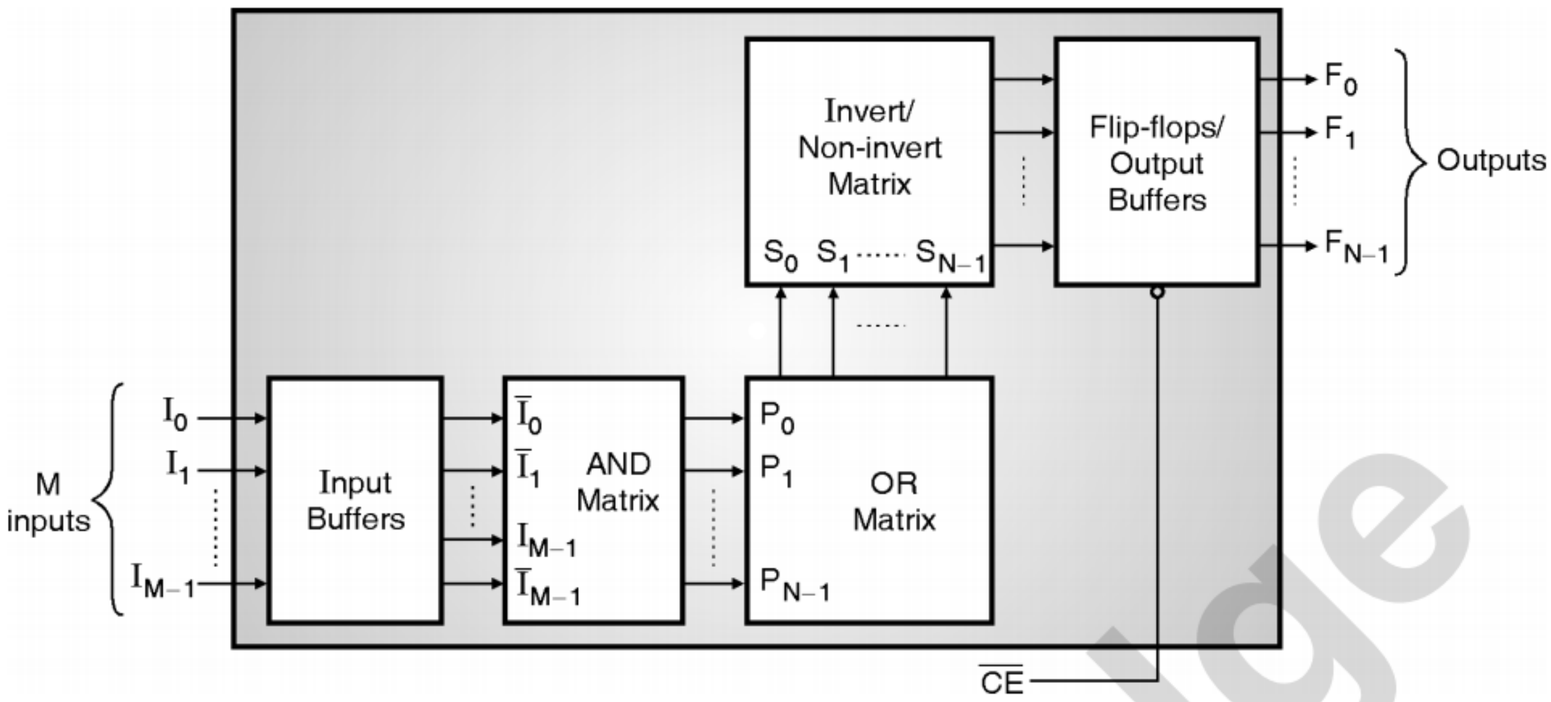
– A PLA as shown in Fig. 8.3.1, consists of two level AND-OR configuration.

– The inputs are applied to the AND matrix through input buffers and the output of AND matrix is applied to the OR matrix.

– The AND matrix can be used to implement the product terms in the SOP form and the OR matrix is used for implementing the sum of the product terms.

– Let us understand each block of a PLA device in detail.

Tech Knowledge Publications



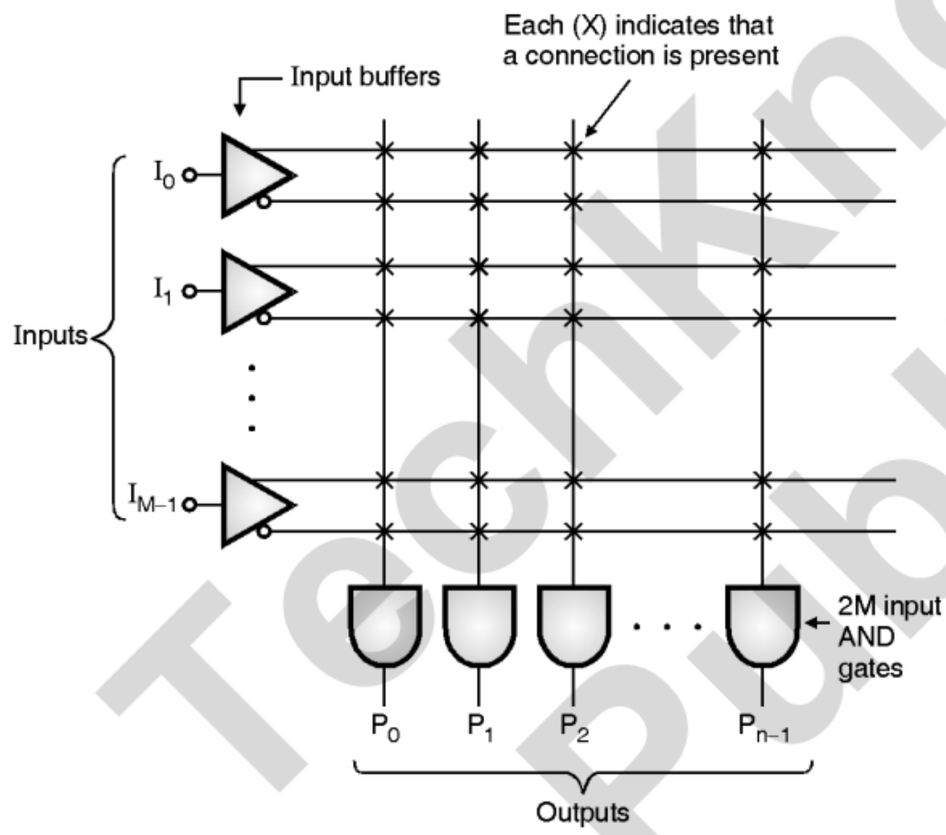
(C-518) Fig. 8.3.1 : Block diagram of a PLA device

1. Input Buffers :

– Input buffer in a PLA is used to minimize the loading of sources connected at the inputs.

2. And Matrix :

– Fig. 8.3.2 shows the simplified presentation of the AND matrix.



(C-520) Fig. 8.3.2 : Simplified representation of AND matrix

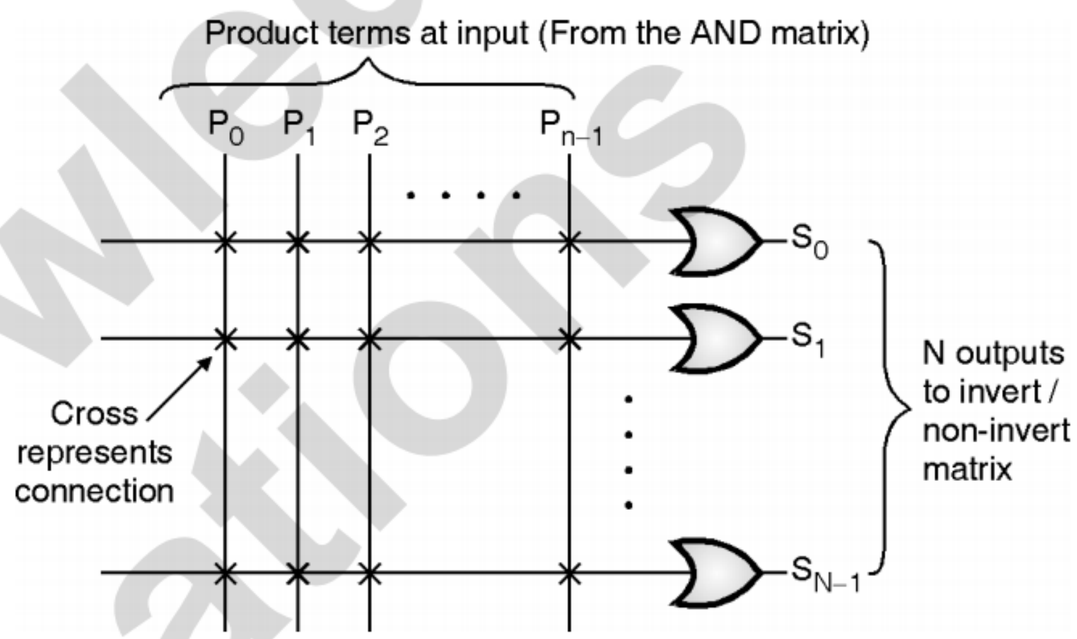
– The (×) marks indicate that a connection is present. Each AND gate has 2M inputs and all these inputs are shown only by a single line.

– When a logic function presented in the SOP form is to be implemented, we have to program the array.

– In the process of programming the desired connections are left with the (×) marks and such mark is not used when a connection is not required.

3. The OR Matrix :

– Fig. 8.3.3 shows the simplified presentation of the OR matrix.



(C-521) Fig. 8.3.3 : Simplified presentation of the OR matrix

– It is possible to program the OR matrix, by open circuiting the unwanted fuse links.

– The open circuited fuse links are equivalent to a "0" at the input of corresponding OR gate.

4. Invert/Non-invert Matrix :

– In Fig. 8.3.1, the block following the OR matrix is the inverting/non inverting matrix which is basically a programmable buffer.

– It can invert its input if active low output is required. Its input is passed without any inversion if active high output is required.

– Thus the sum of product terms obtained at the outputs of the OR matrix can be either in the inverted form or passed through as it is (i.e. in the non-inverted form).

5. Output Buffer :

– The current sourcing capability of the PLA can be increased by using the output buffer.

– The PLA outputs are generally made compatible with the TTL family.

Tech Knowledge Publications

8.3.1 The Programming Procedure for PLA :

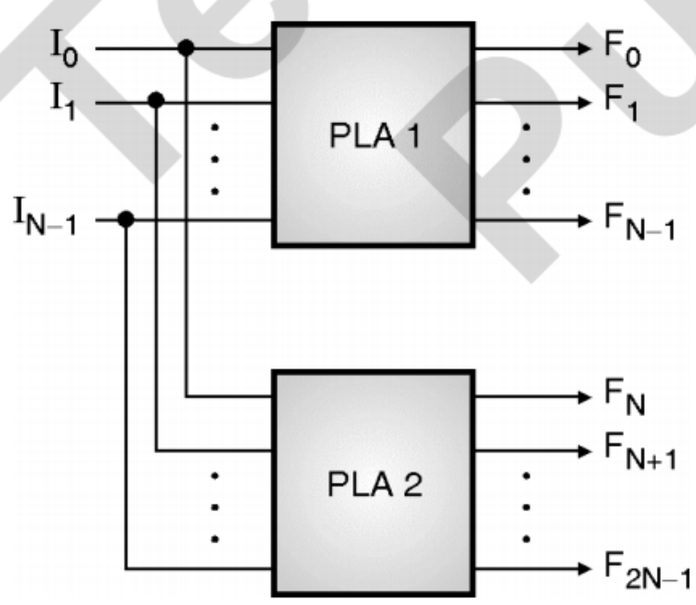
- The programming of a PLA is similar to the ROM programming.
- Like ROM, it is possible to use the mask programming technique for the PLA devices.
- According to the requirement specified by the user, the manufacturer prepares a mask and the desired data patterns are stored on the PLA.
- In a special type of PLA called Field PLA (FPLA), all the nichrome fuse elements are intact at the time of manufacturing.
- At the time of programming, all the unwanted links are open circuited electrically (by applying adequately high voltage).
- It is not possible to reprogram an FPLA.

8.3.2 Expansion of PLA Capacity :

- We may have to increase the capacity of a PLA device depending on the applications.

Increasing the number of outputs :

- Sometimes number of outputs are required to be increased.
- Then the inputs of two or more PLA devices should be connected individually in parallel as shown in Fig. 8.3.4.

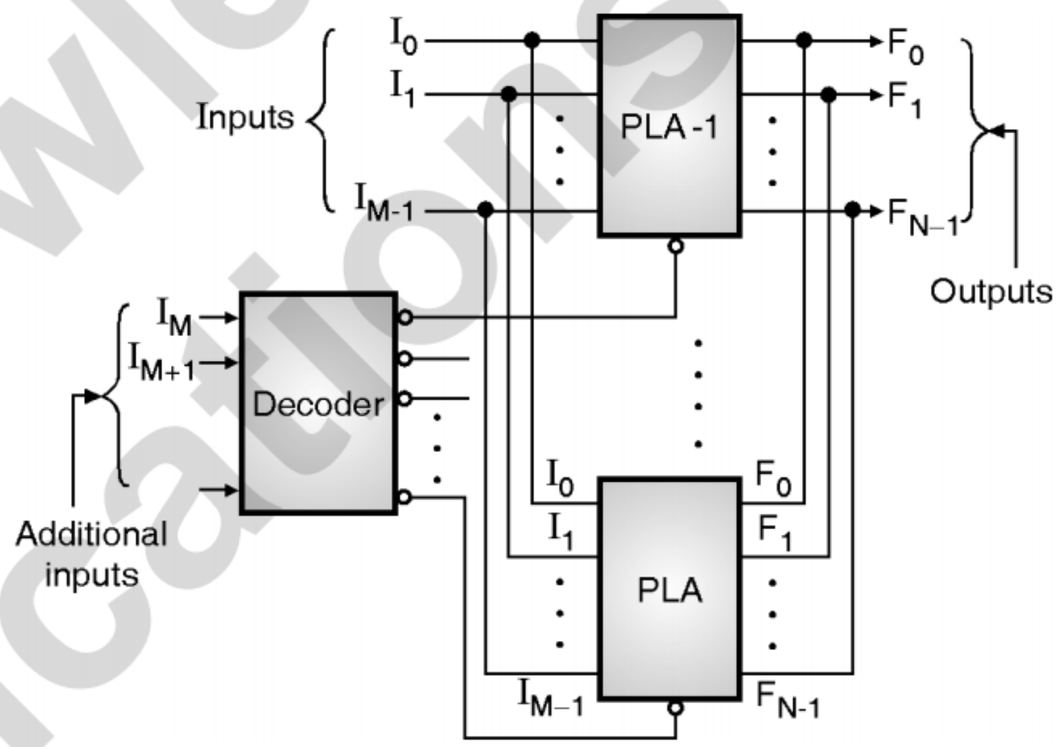


(C-1637) Fig. 8.3.4 : PLA connections for increasing the number of outputs

- Note that with such a connection the number of inputs and the number of product terms will remain unchanged but the number of output lines is increased.

Increasing the number of product terms :

- In order to increase the number of product terms keeping the number of inputs and outputs unchanged, we have to make connections as shown in Fig. 8.3.5.
- The inputs I_0 through I_{M-1} of all PLA devices are connected individually in parallel with each other.
- Similarly the individual outputs of all the PLA devices also are connected in parallel to each other.
- The number of inputs are increased by using a decoder for increased input lines I_M through I_{M+Q-1} as shown in Fig. 8.3.5.
- This type of connection will increase the number of product terms as well.



(C-1638) Fig. 8.3.5 : Increasing the number of product terms of PLA

8.3.3 Application Areas of PLA :

1. We can implement the combinational circuits using PLA.
2. We can also implement the sequential circuits using PLA.

- For implementing the combinational circuits, the PLA devices with only output buffers are used whereas to implement the sequential circuits we use the PLA devices with flip flop and buffers included in the output stage.

8.3.4 Designing of Combinational Circuit using PLA : SPPU : Dec. 10

University Questions

Q. 1 Explain in brief design model of PLA for any code conversion example. **(Dec. 10, 8 Marks)**

TechKnowledge Publications

– In order to implement combinational circuits using PLA, we have to follow the procedure given below.

Steps to be followed :

Step 1 : Write the truth table for the combinational circuit to be designed.

Step 2 : Write the Boolean expression in SOP form using the truth table.

Step 3 : Obtain the minimized SOP expression in order to reduce the number of product terms to a minimum.

Step 4 : Decide the input connections to the AND matrix so as to generate the required product terms.

Step 5 : Then decide the input connections of OR matrix to generate the required sum terms.

Step 6 : Decide the connections of invert/non-invert matrix.

Step 7 : Last step is the programming of the PLA.

– Solve the following examples to understand the use of PLA for designing the combinational logic.

Ex. 8.3.1 : Draw a combinational circuit for a PLA with three inputs, three product terms and two outputs.

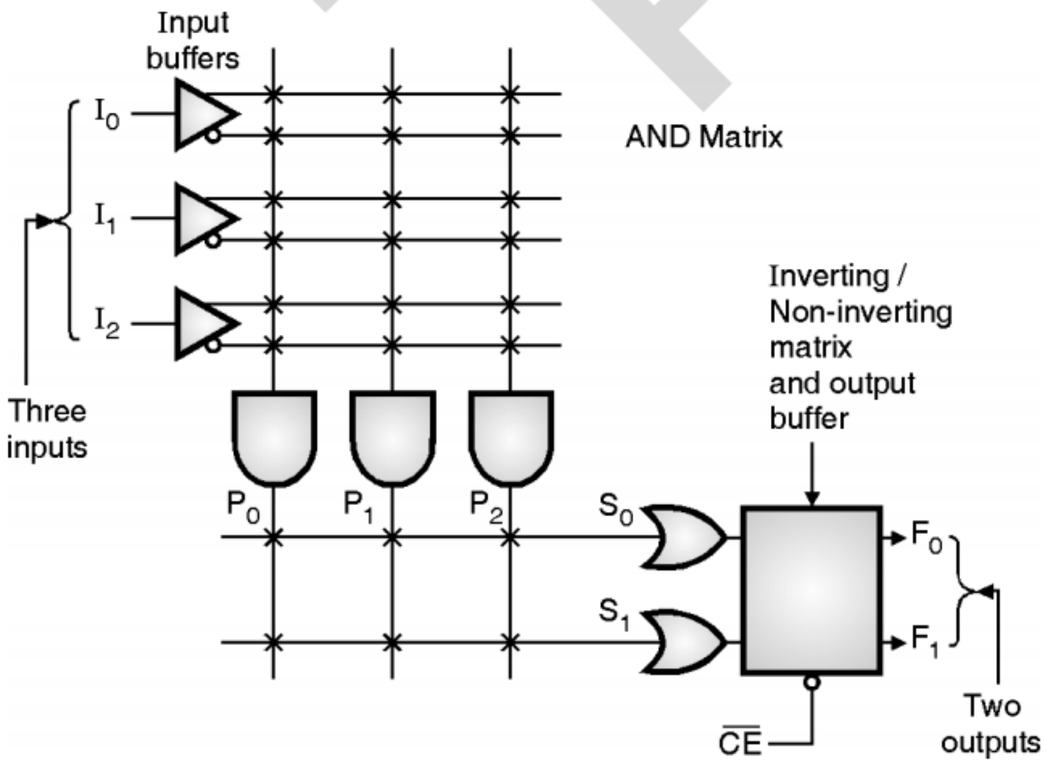
Soln. :

Given, number of inputs = 3 = I_0, I_1, I_2.

Number of outputs = 2 = Number of OR gates.

Number of product terms = Number of AND gates.

– Hence this PLA device will have 3, six input AND gates and two OR gates as shown in Fig. P. 8.3.1.



(C-525) **Fig. P. 8.3.1 : Three input, two output PLA**

8.3.5 How to Specify the Size of a PLA ?

– The size of a PLA is specified as M × P × N. When M denotes the number of inputs, P corresponds to the number of product terms and N denotes the number of outputs.

– For example, FPLA 823100 is a 16 × 48 × 8 PLA. That means it has 16 inputs, 48 product terms and 8 outputs.

Ex. 8.3.2 : Implement the following function using PLA :

$f_1 = m(0, 3, 4, 7)$, $f_2 = m(1, 2, 5, 7)$

Dec. 08, 8 Marks, Dec. 14, 7 Marks

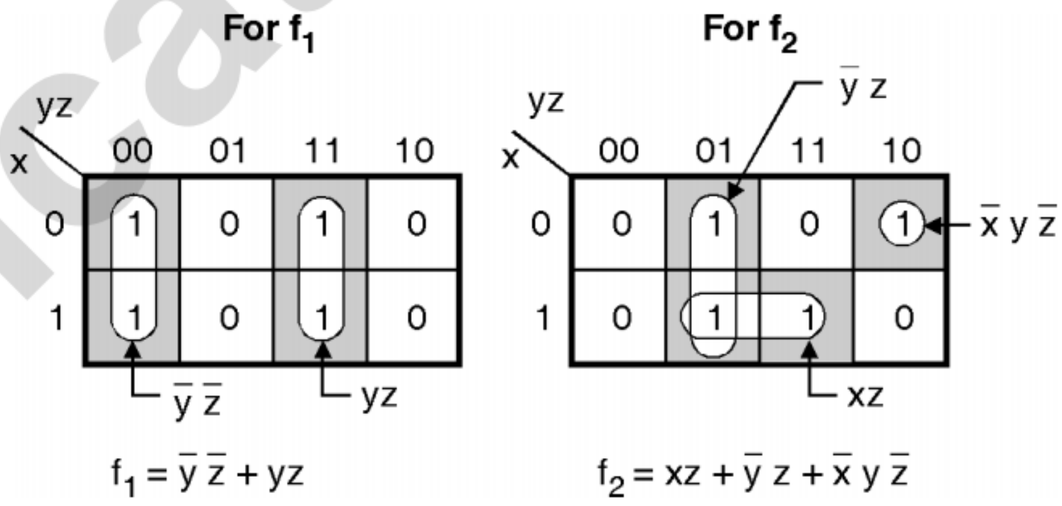
Soln. :

Assume x, y and z as the inputs,

$f_1(x, y, z) = \sum m(0, 3, 4, 7)$,

$f_2(x, y, z) = \sum m(1, 2, 5, 7)$

Step 1 : K-maps and simplification : (C-3572)



Step 2 : Prepare the PLA programming table :

$f_1 = \underset{(1)}{\overline{y}\overline{z}} + \underset{(2)}{yz}$ $\qquad f_2 = \underset{(3)}{xz} + \underset{(4)}{\overline{y}z} + \underset{(5)}{\overline{x}y\overline{z}}$

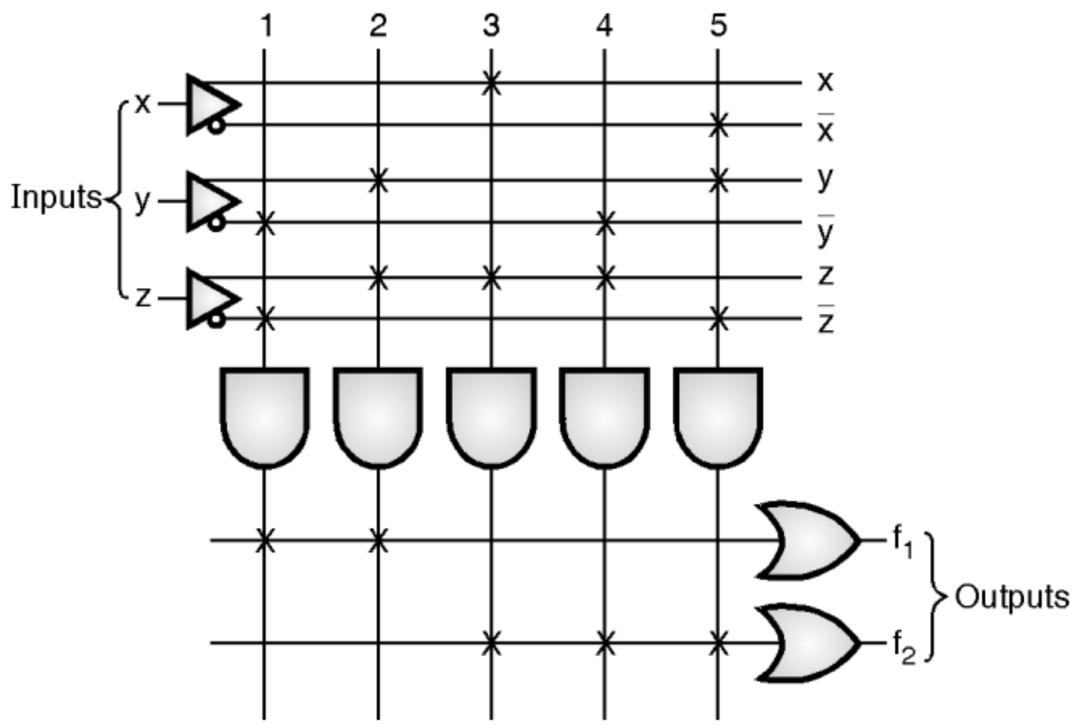
(C-6498)

Programming table for PLA : (C-8144)

Product term no.	Product term	Inputs			Outputs	
		x	y	z	f_1	f_2
1	$\bar{y}\,\bar{z}$	–	0	0	1	–
2	yz	–	1	1	1	–
3	xz	1	–	1	–	1
4	$\bar{y}\,z$	–	0	1	–	1
5	$\bar{x}y\bar{z}$	0	1	0	–	1

TechKnowledge Publications

Step 3 : Implement the logic circuit :



(C-3573) **Fig. P. 8.3.2**

Ex. 8.3.3 : A combinational circuit is defined by the function F_1 (A, B, C) = $\sum$ m (4, 5, 7) F_2 (A, B, C) = $\sum$ m (3, 5, 7). Implement this circuit with a PLA having 3 inputs, 3 product terms and 2 outputs.

May 14, 6 Marks, Dec. 14, 7 Marks

Soln. :

– This is an example which illustrates the use of PLA device to implement the combinational circuit.

Step 1 : Write truth table :

– This step can be omitted for this example.

Step 2 : Write the Boolean expression in SOP form :

$$F_1 (A, B, C) = \sum m (4, 5, 7) = A \bar{B} \bar{C} + A \bar{B} C + A B C \quad ...(1)$$

$$\text{and } F_2 (A, B, C) = \sum m (3, 5, 7) = \bar{A} B C + A \bar{B} C + A B C \quad ...(2)$$

Step 3 : Minimum SOP form :

$$F_1 (A, B, C) = A \bar{B} \bar{C} + A \bar{B} C + A B C$$

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

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

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

$$= A (\bar{B} + C) = A \bar{B} + A C \quad ...(3)$$

$$\text{And } F_2 (A, B, C) = \bar{A} B C + A \bar{B} C + A B C$$

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

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

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

$$= C [B + A] = B C + A C \quad ...(4)$$

Step 4 : Decide inputs to AND matrix :

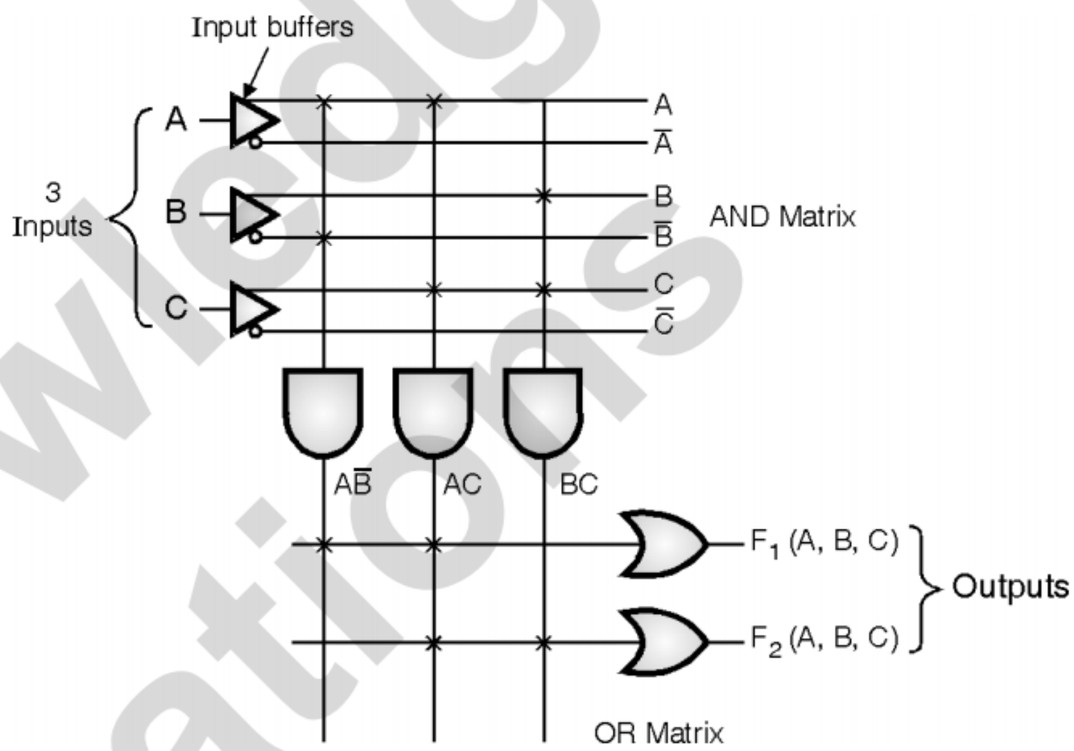
– The inputs to the AND matrix are decided from Equations (3) and (4). They are A, $\bar{B}$, B and C.

Step 5 : Decide inputs to OR matrix :

– The inputs to OR matrix are $A\bar{B}$, AC and BC.

Step 6 : Draw the logic diagram :

– The implementation is as shown in Fig. P. 8.3.3.



(C-526) **Fig. P. 8.3.3 : Implementation**

Ex. 8.3.4 : Implement 3 bit binary to gray converter using PLA :

1. Make a truth table
2. Simplify using K-map
3. Realize using PLA
4. Specify the size of PLA.

Dec. 16, May 18, 6 Marks

Soln. :

Step 1 : Write truth table :

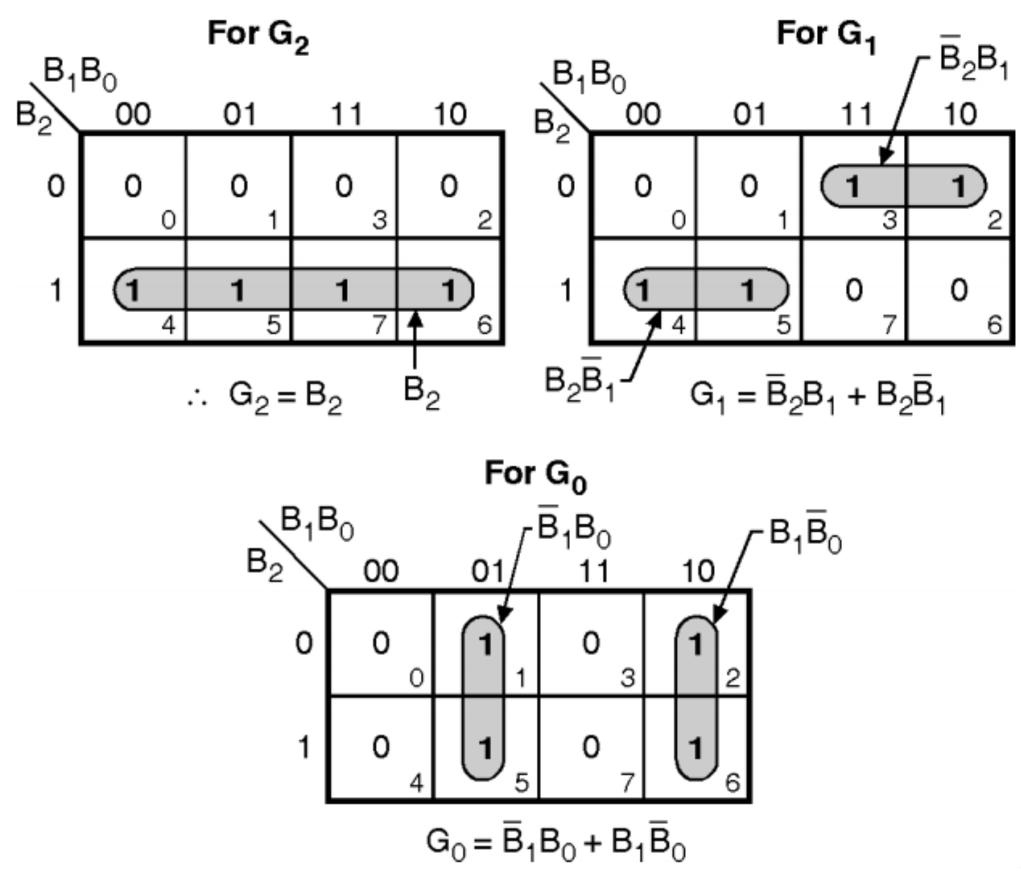
(C-8107) **Table P. 8.3.4**

Gray input			Binary output		
B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

TechKnowledge Publications

Step 2 : Simplification using K-map :

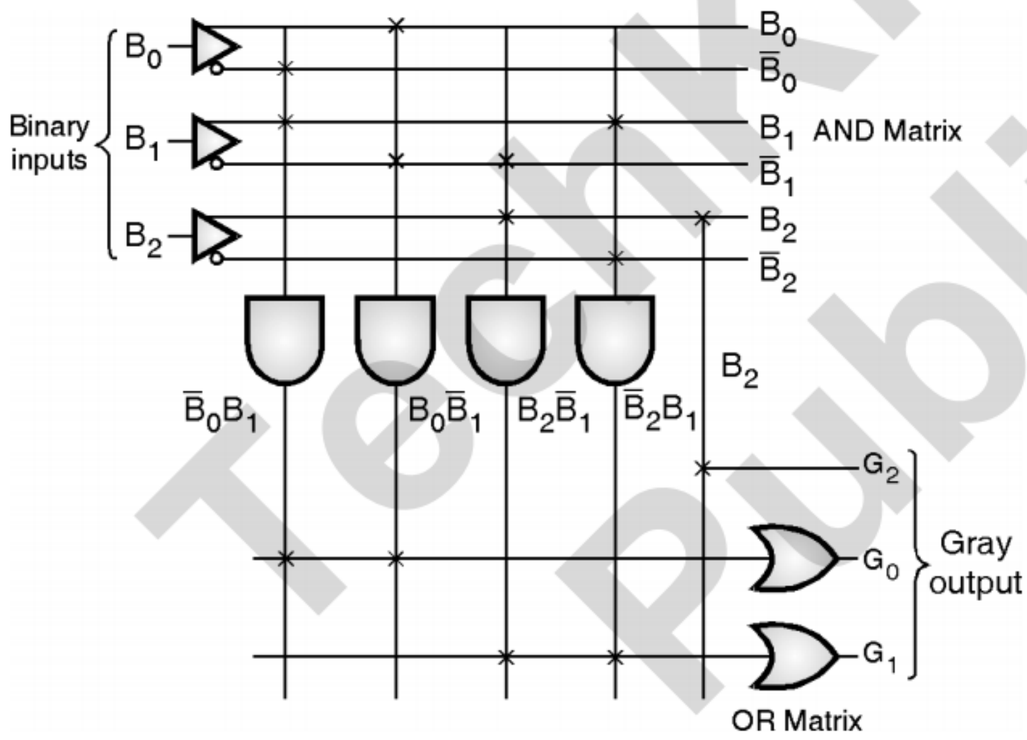
– The K maps for the gray outputs are as shown in Fig. P. 8.3.4(a).



(C-528) Fig. P. 8.3.4(a) : K maps and simplification

Step 3 : Implementation :

– Fig. P. 8.3.4(b) shows the implementation using PLA.



(C-529) Fig. P. 8.3.4(b) : 3 bit Binary to Gray code converter

Ex. 8.3.5 : Design a Gray to Binary code converter using a PLA. **OR**

Design and implement using FPLA : a 3 bit Gray to binary code converter.

Soln. :

Step 1 : Write the truth table relating the Gray and Binary codes :

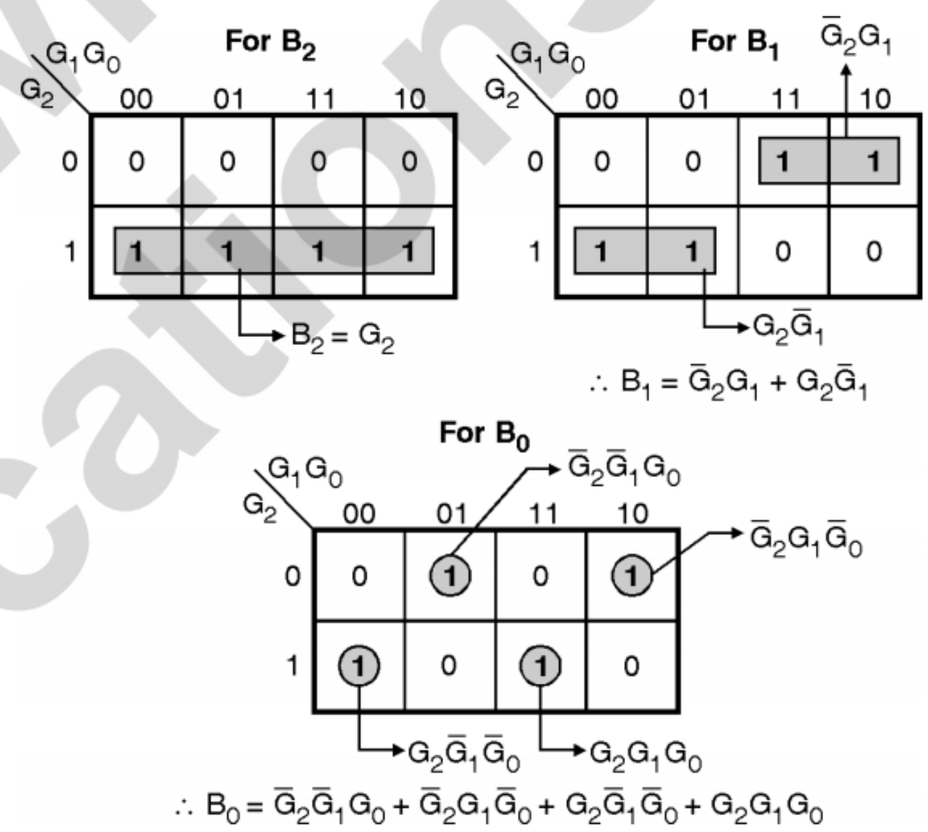
– The truth table relating the gray and binary codes is shown in Table P. 8.3.5.

(C-8108) Table P. 8.3.5 : Truth table relating Gray and Binary codes

Decimal	Gray input			Binary output		
	G_2	G_1	G_0	B_2	B_1	B_0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
3	0	1	1	0	1	0
2	0	1	0	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1
5	1	0	1	1	1	0
4	1	0	0	1	1	1

Step 2 : Write K-map and simplify :

– The K-maps for outputs B_2, B_1 and B_0 are shown in Fig. P. 8.3.5(a).



(C-530) Fig. P. 8.3.5(a) : K-map and simplification

– Hence the simplified equations are as follows :

$$B_2 = G_2,$$

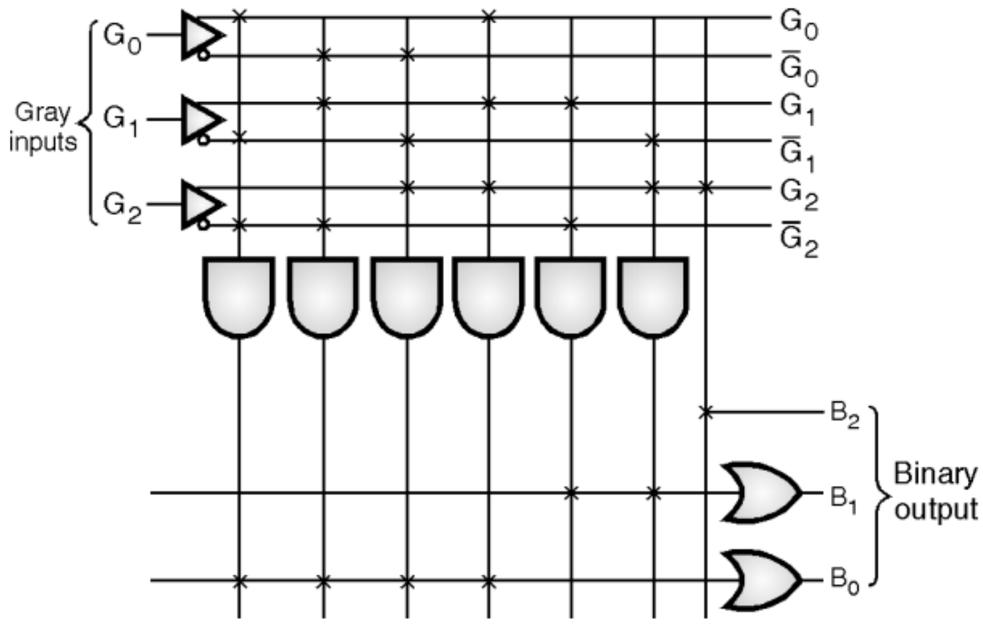
$$B_1 = \bar{G}_2 G_1 + G_2 \bar{G}_1$$

$$B_0 = \bar{G}_2 \bar{G}_1 G_0 + \bar{G}_2 G_1 \bar{G}_0 + G_2 \bar{G}_1 \bar{G}_0 + G_2 G_1 G_0$$

Step 3 : Implementation :

– Fig. P. 8.3.5(b) shows the gray to binary converter implemented using PLA.

TechKnowledge Publications



(C-531) Fig. P. 8.3.5(b) : Gray to binary converter

Ex. 8.3.6 : Design a BCD to Excess-3 code converter and implement it using a suitable PLA.

Dec. 15, 7 Marks, May 18, Dec. 18, 6 Marks

Soln. :

Step 1 : Write the truth table :

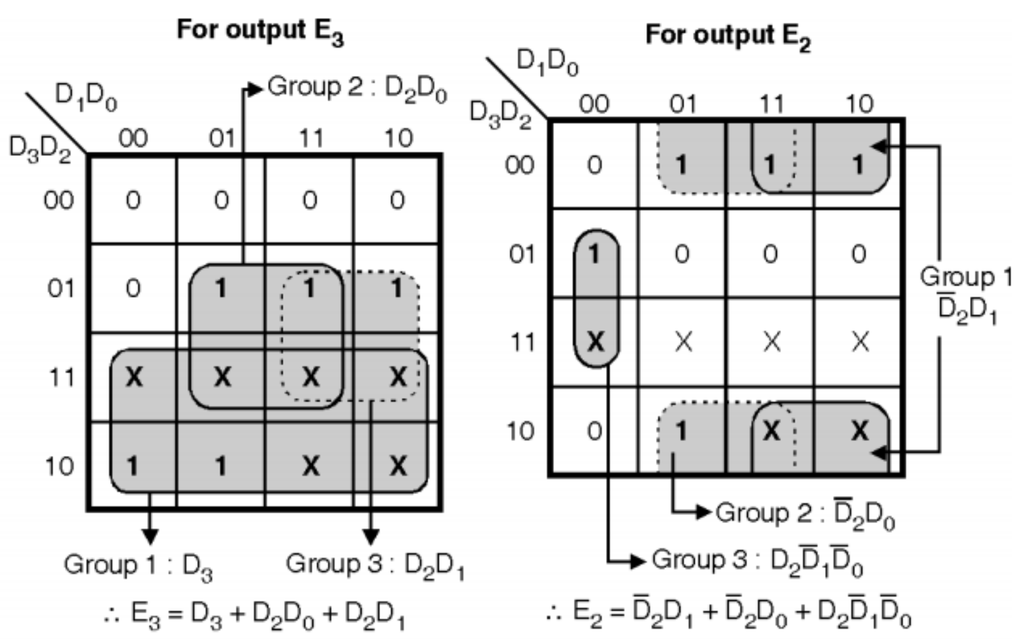
– The truth table relating the BCD and Excess 3 codes is shown in Table P. 8.3.6.

(C-8109) Table P. 8.3.6 : Truth table relating BCD and Excess-3 codes

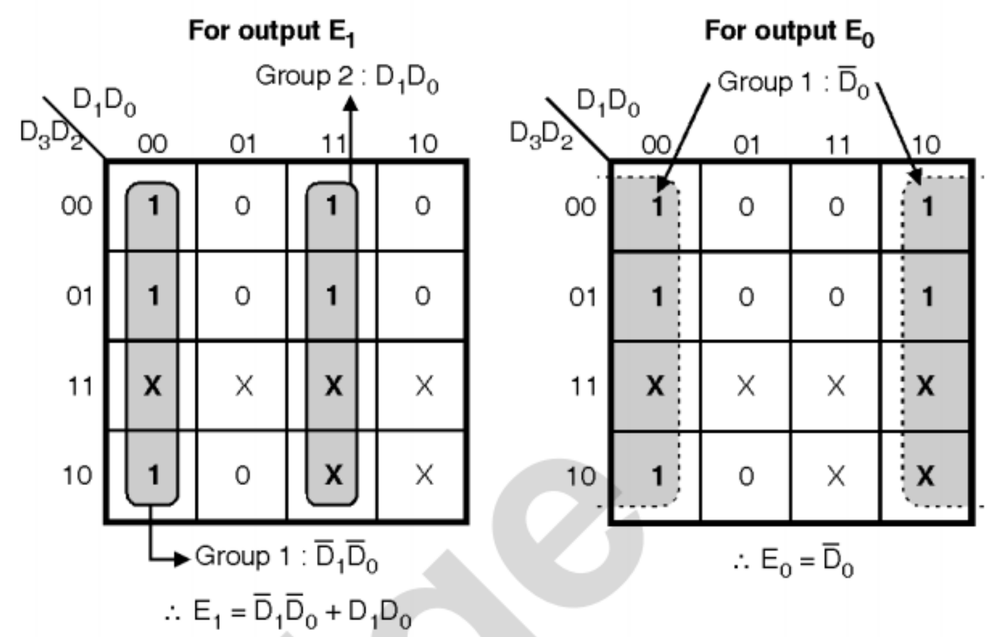
Decimal	BCD inputs				Excess-3 output			
	D_3	D_2	D_1	D_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Step 2 : K-maps and simplified expressions :

– Refer Figs. P. 8.3.6(a) to (d) for K-maps and simplified expressions.



(a) K-map for E_3 **(b) K-map for E_2**



(c) K-map for E_1 **(d) K-map for E_0**

(C-532) Fig. P. 8.3.6

– The four simplified expressions and associated product terms are as follows :

$$E_3 = D_3 + D_2D_0 + D_2D_1$$

Product term → (1) (2) (3)

$$E_2 = \overline{D}_2D_1 + \overline{D}_2D_0 + D_2\overline{D}_1\overline{D}_0$$

Product term → (4) (5) (6)

$$E_1 = \overline{D}_1\overline{D}_0 + D_1D_0$$

Product term → (7) (8)

$$E_0 = \overline{D}_0$$

Product term → (9) **(C-6419)**

Step 3 : Prepare the PLA program table :

– Prepare the PLA program table as shown in Table P. 8.3.6(a).

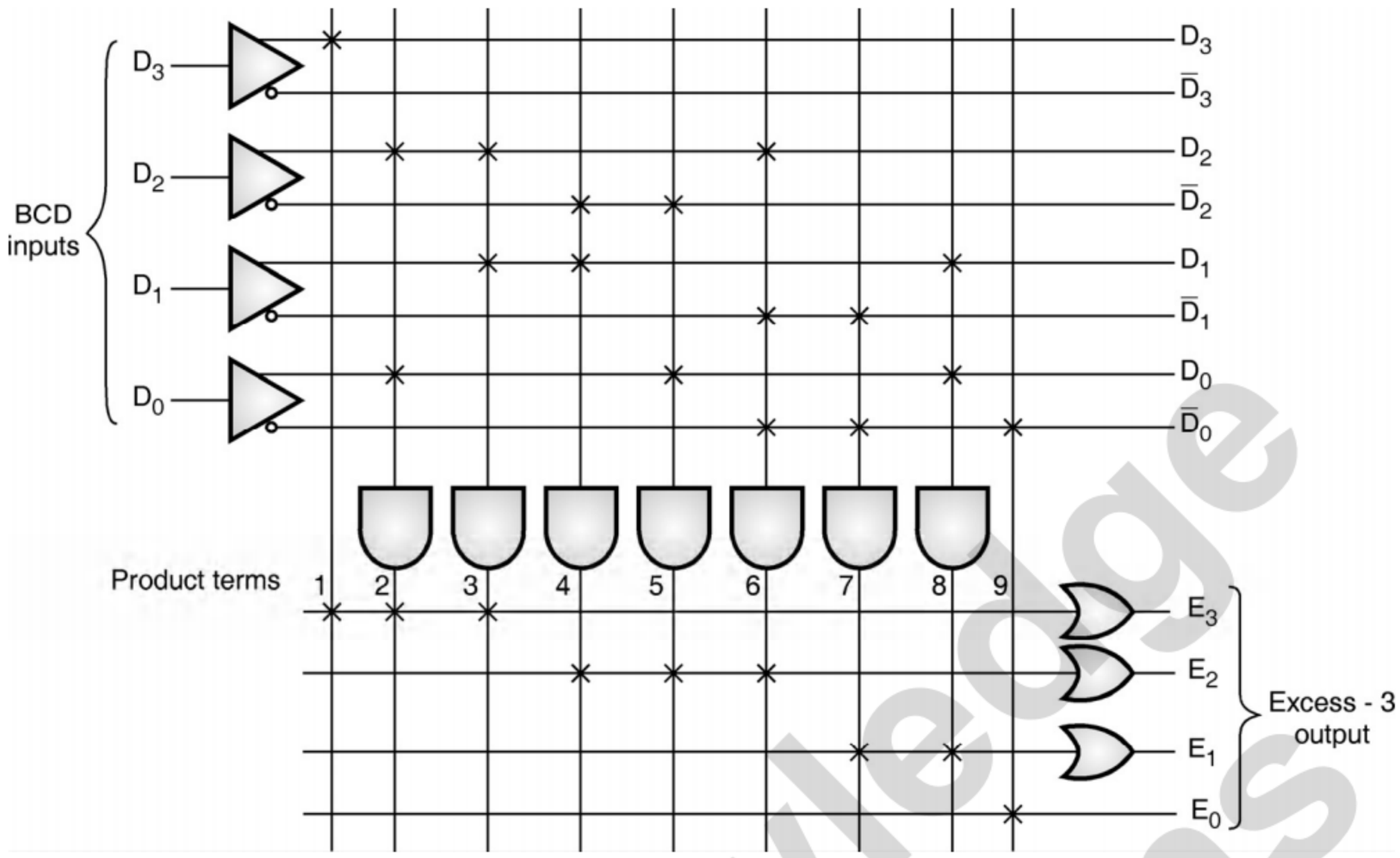
(C-8110) Table P. 8.3.6(a) : Program table for PLA

Product term number	Product term	Inputs				Outputs			
		D_3	D_2	D_1	D_0	E_3	E_2	E_1	E_0
1	D_3	1	–	–	–	1	–	–	–
2	D_2D_0	–	1	–	1	1	–	–	–
3	D_2D_1	–	1	1	–	1	–	–	–
4	$\overline{D}_2D_1$	–	0	1	–	–	1	–	–
5	$\overline{D}_2D_0$	–	0	–	1	–	1	–	–
6	$D_2\overline{D}_1\overline{D}_0$	–	1	0	0	–	1	–	–
7	$\overline{D}_1\overline{D}_0$	–	–	0	0	–	–	1	–
8	D_1D_0	–	–	1	1	–	–	1	–
9	$\overline{D}_0$	–	–	–	0	–	–	–	1

Step 4 : Implement the circuit :

– The logic circuit for BCD to Excess–3 converter using a PLA device is shown in Fig. P. 8.3.6(e).

TechKnowledge Publications



(C-533) **Fig. P. 8.3.6(e) : Implementation of BCD to Excess 3-code converter**

Ex. 8.3.7 : Design BCD to gray code converter and implement it using PLA. **May 19, 5 Marks**

Soln. :

Step 1 : Write the truth table :

(C-8145) **Table P. 8.3.7**

Decimal	BCD inputs				Gray code			
	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

Step 2 : K-maps and simplified expressions :

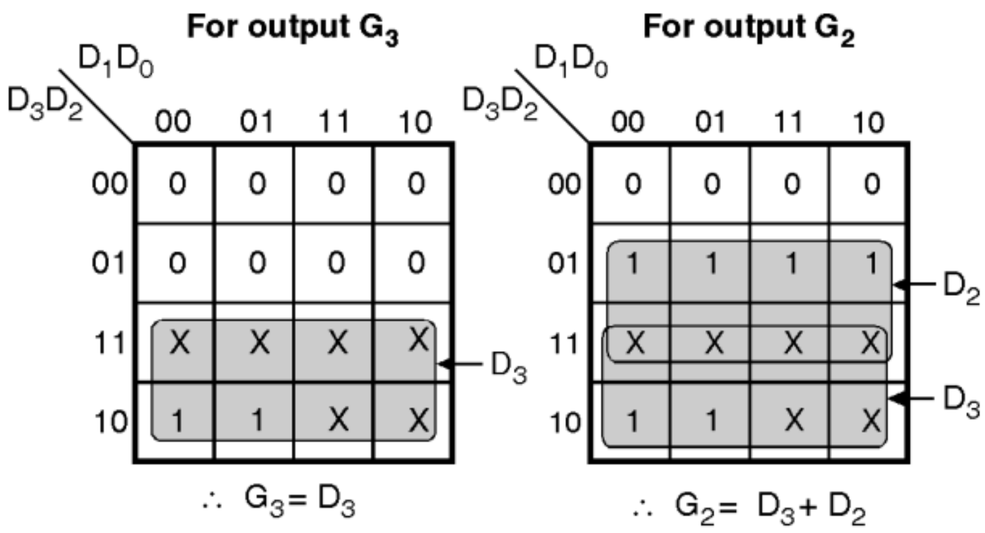
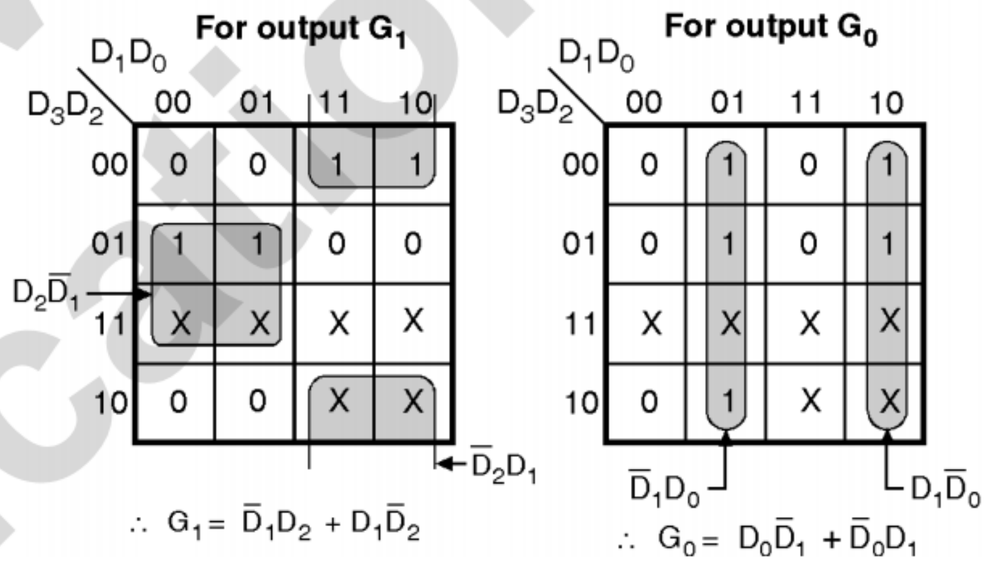


Fig. P. 8.3.7(Contd...)



(C-1642) **Fig. P. 8.3.7**

Step 3 : Prepare the PLA programming table :

– The four simplified expression and associated product terms are as follows :

$G_3 = D_3$ (1) $\qquad G_2 = D_3 + D_2$ (2) (3)

$G_1 = \overline{D}_1D_2 + D_1\overline{D}_2$ (4) (5) $\qquad G_0 = D_0\overline{D}_1 + \overline{D}_0D_1$ (6) (7)

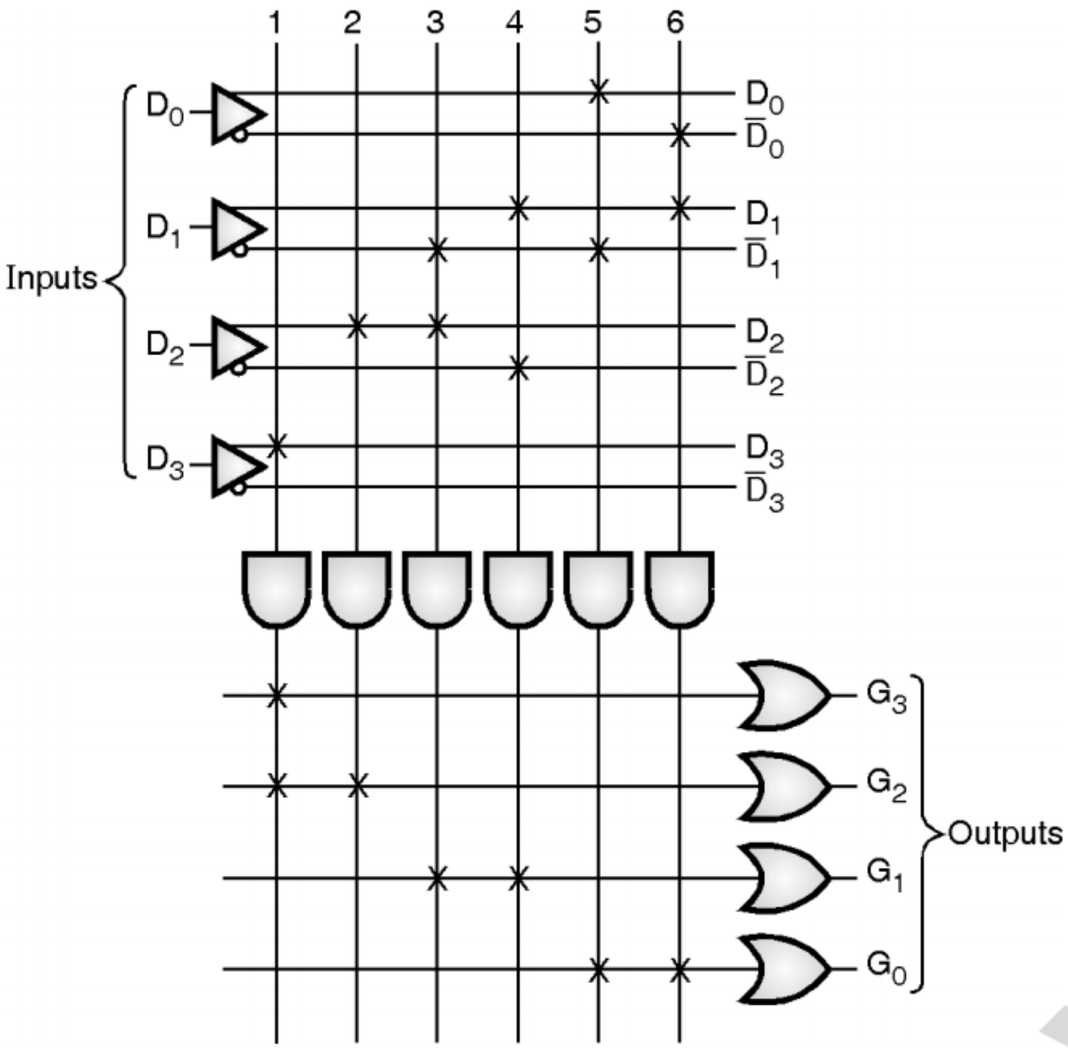
(C-6420)

– But the product terms (1) and (2) are identical.

(C-8146)

Product term number	Product term	Inputs				Outputs			
		D_3	D_2	D_1	D_0	G_3	G_2	G_1	G_0
1	D_3	1	–	–	–	1	1	–	–
2	D_2	–	1	–	–	–	1	–	–
3	$\overline{D}_1 D_2$	–	1	0	–	–	–	1	–
4	$D_1 \overline{D}_2$	–	0	1	–	–	–	1	–
5	$D_0 \overline{D}_1$	–	–	0	1	–	–	–	1
6	$\overline{D}_0 D_1$	–	–	1	0	–	–	–	1

Step 4 : Implement the logic circuit :



(C-1643) **Fig. P. 8.3.7(e)**

Ex. 8.3.8 : A combinational circuit is defined by the function :

f_1 (A, B, C) = Σ m (0, 1, 2, 4),

f_2 (A, B, C) = Σ m (1, 3, 5, 6)

Implement this circuit with PLA.

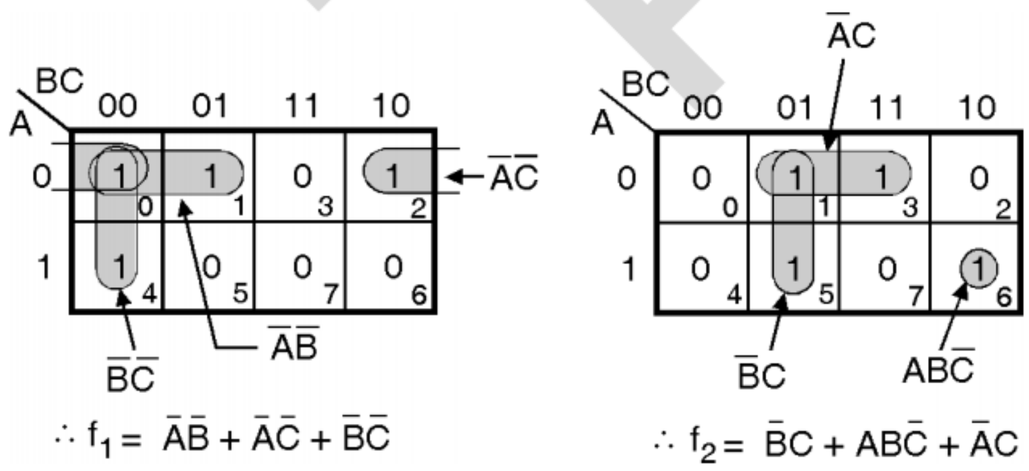
Dec. 13, 7 Marks

Soln. :

f_1 (A, B, C) = Σ m (0, 1, 2, 4),

f_2 (A, B, C) = Σ m (1, 3, 5, 6)

Step 1 : K-maps and simplification :



$\therefore f_1 = \overline{A}\overline{B} + \overline{A}\overline{C} + \overline{B}\overline{C}$ $\quad$ $\therefore f_2 = \overline{B}C + AB\overline{C} + \overline{A}C$

(C-4190) **Fig. P. 8.3.8(a) : K-maps**

Step 2 : Prepare the PLA programming table :

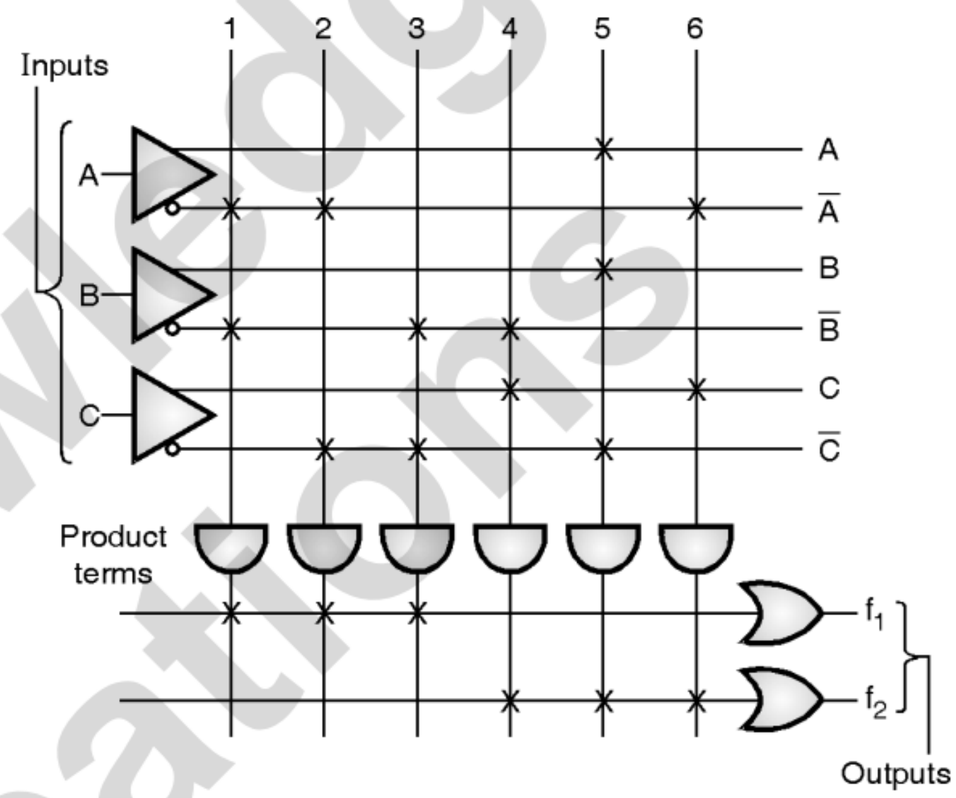
(C-6499) $f_1 = \overline{A}B + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}$, $\quad f_2 = \overline{B}C + AB\overline{C} + \overline{A}C$

(1) (2) (3) $\quad$ (4) (5) (6)

Programming table for PLA : (C-8147)

Product term no.	Product term	Inputs			Outputs	
		A	B	C	f_1	f_2
1.	$\overline{A}\,\overline{B}$	0	0	–	1	–
2.	$\overline{A}\,\overline{C}$	0	–	0	1	–
3.	$\overline{B}\,\overline{C}$	–	0	0	1	–
4.	$\overline{B}C$	–	0	1	–	1
5.	$AB\overline{C}$	1	1	0	–	1
6.	$\overline{A}C$	0	–	1	–	1

Step 3 : Implement logic circuit :



(C-4191) **Fig. P. 8.3.8(b) : Implementation using PLA**

Ex. 8.3.9: A combinational circuits is defined by the function :

F_1 (A, B, C) = Σ m (0, 1, 3, 7)

F_2 (A, B, C) = Σ m (1, 2, 5, 6)

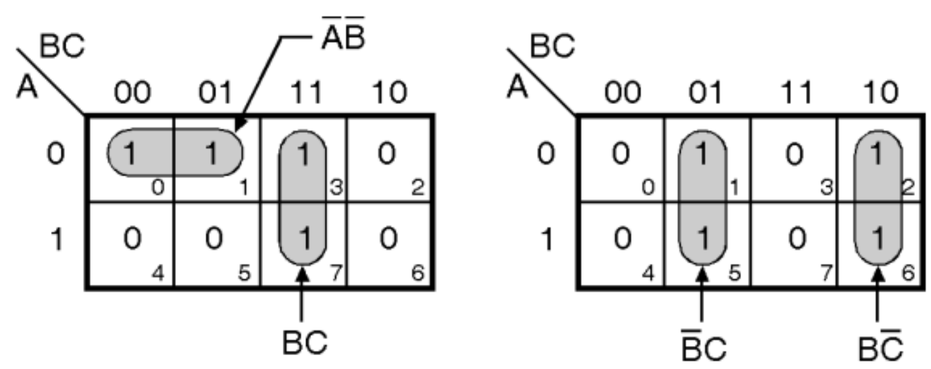
Implement this circuit with PLA.

May 15, 7 Marks, Dec. 17, 6 Marks

Soln. :

Step 1 : K-maps and simplification :

$F_1 = \overline{A}\,\overline{B} + BC$ $\quad$ $F_2 = \overline{B}C + B\overline{C}$



(C-5107) **Fig. P. 8.3.9(a)**

Step 2 : Prepare the PLA programming table :

(C-6500) $F_1 = \overline{A}\,\overline{B} + BC$ $\quad$ $F_2 = \overline{B}C + B\overline{C}$

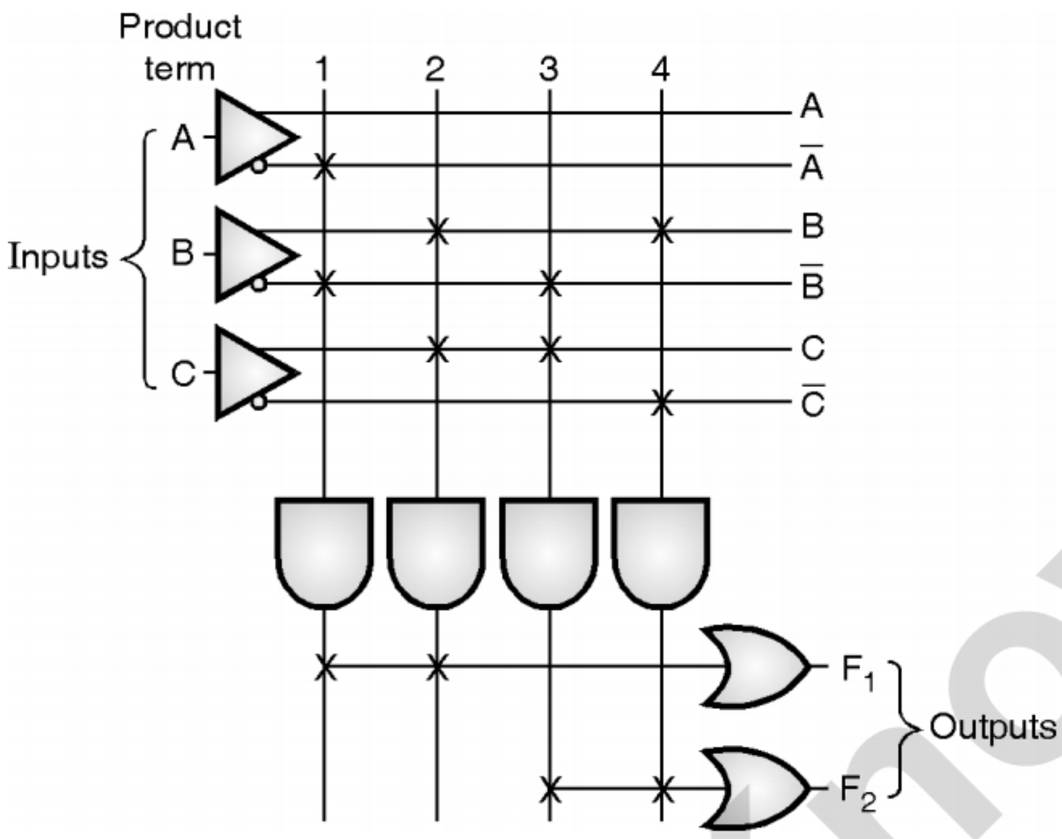
(1) (2) $\quad$ (3) (4)

TechKnowledge Publications

(C-8148) **Table P. 8.3.9 : Programming table for PLA**

Product term no.	Product term	Inputs			Outputs	
		A	B	C	f_1	f_2
1.	$\overline{A}\,\overline{B}$	0	0	–	1	–
2.	BC	–	1	1	1	–
3.	$\overline{B}\,C$	–	0	1	–	1
4.	$B\overline{C}$	–	1	0	–	1

Step 3 : Implement the logic circuit using PLA :



(C-5108) **Fig. P. 8.3.9(b)**

Ex. 8.3.10 : Design 4 input and 6 output combinational circuit using PLA. The input variables are A, B, C and D :

$Y_1 = \sum m\,(0, 3, 5, 6, 9, 10, 12, 15)$,

$Y_2 = \sum m\,(0, 1, 2, 3, 11, 12, 14, 15)$

$Y_3 = \sum m\,(0, 4, 8, 12)$,

$Y_4 = \sum m\,(0, 2, 3, 5, 7, 8, 12, 13)$

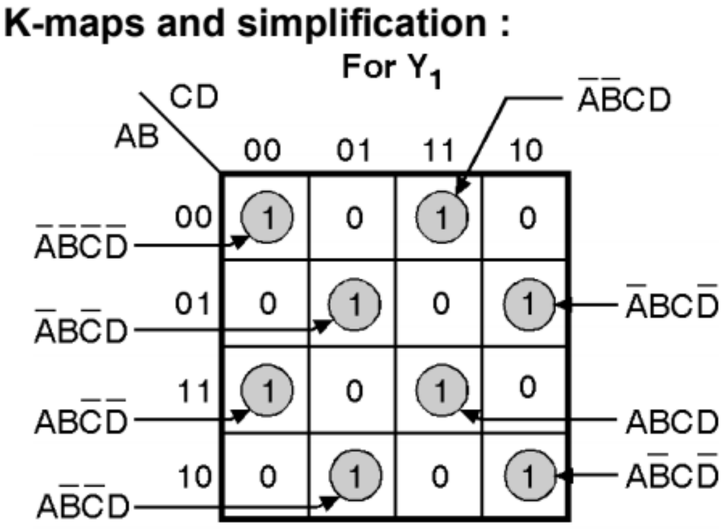
$Y_5 = \sum m\,(0, 1, 3, 4, 5, 6, 11, 13, 14, 15)$,

$Y_6 = \sum m\,(1, 2, 6, 8, 15)$. **May 17, 6 Marks**

Soln. :

Step 1 : K-maps and simplification :

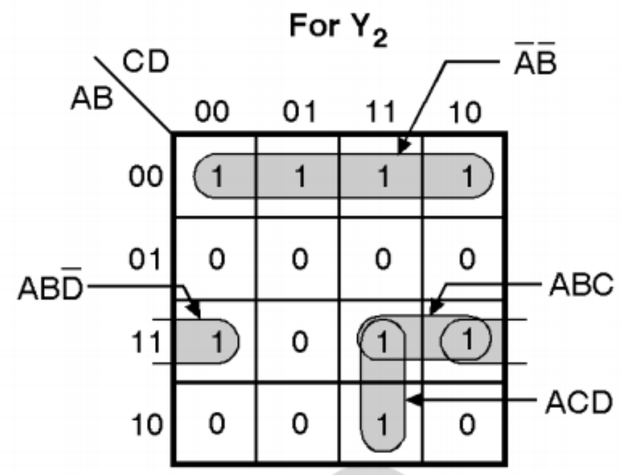
For Y_1



$\therefore\ Y_1 = \overline{A}\overline{B}CD + \overline{A}BC\overline{D} + ABCD + A\overline{B}C\overline{D}$
$+ \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}\overline{D} + A\overline{B}\overline{C}D$

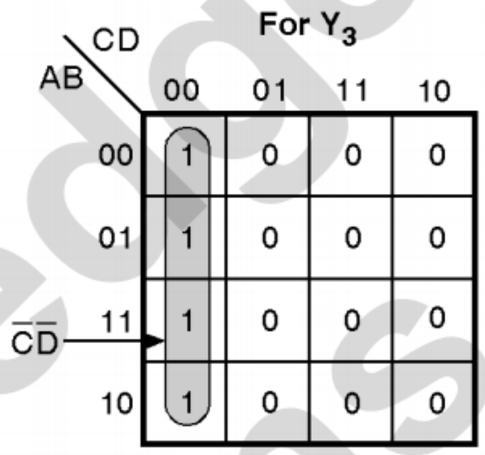
(C-5912) **(Contd...)**

For Y_2



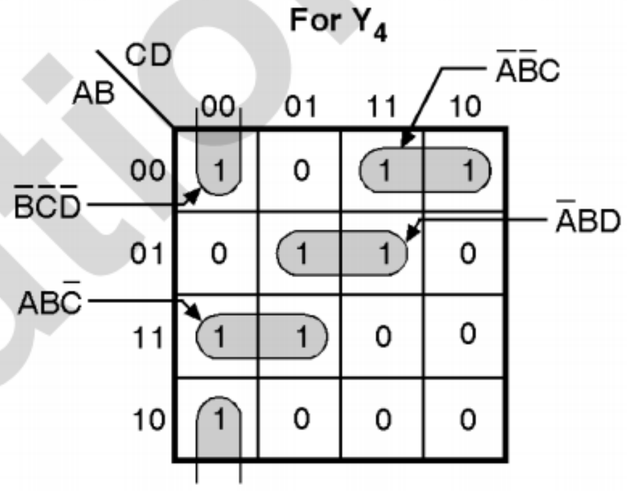
$\therefore\ Y_2 = \overline{A}\overline{B} + AB\overline{D} + ABC + ACD$

For Y_3



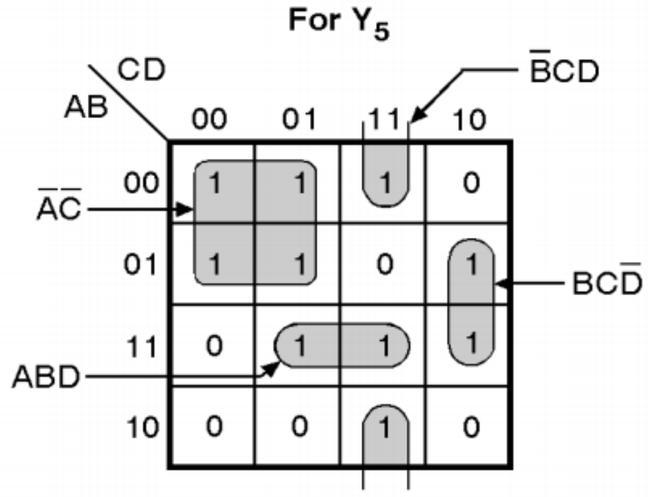
$\therefore\ Y_3 = \overline{C}\overline{D}$

For Y_4



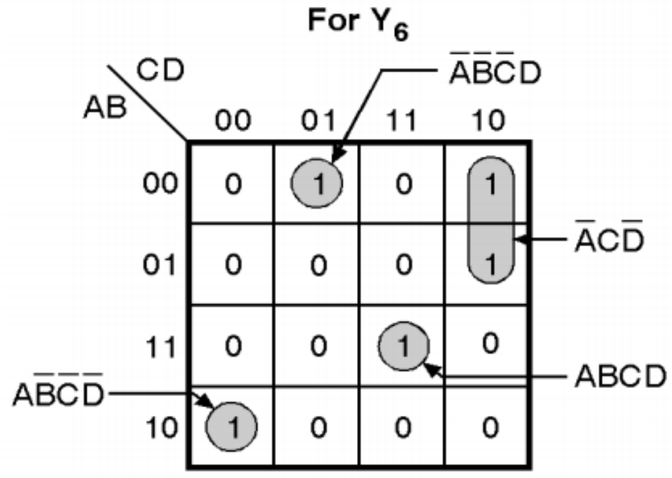
$\therefore\ Y_4 = \overline{A}\overline{B}C + \overline{A}BD + AB\overline{C} + \overline{B}\overline{C}\overline{D}$

For Y_5



$\therefore\ Y_5 = \overline{A}\overline{C} + ABD + BC\overline{D} + \overline{B}CD$

For Y_6



$\therefore\ Y_6 = \overline{A}\overline{B}\overline{C}D + \overline{A}C\overline{D} + ABCD + A\overline{B}\overline{C}\overline{D}$

(C-5912)

TechKnowledge Publications

Step 2 : Prepare the PLA programming table :

$$Y_1 = \bar{A}\bar{B}CD + \bar{A}BC\bar{D} + ABCD + A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + AB\bar{C}\bar{D} + A\bar{B}\bar{C}D$$
(1) (2) (3) (4) (5) (6) (7) (8)

$$Y_2 = \bar{A}\bar{B} + AB\bar{D} + ABC + ACD$$
(9) (10) (11) (12)

$$Y_3 = \bar{C}\bar{D}$$
(13)

$$Y_4 = \bar{A}\bar{B}C + \bar{A}BD + AB\bar{C} + \bar{B}\bar{C}\bar{D}$$
(14) (15) (16) (17)

$$Y_5 = \bar{A}\bar{C} + ABD + BC\bar{D} + \bar{B}CD$$
(18) (19) (20) (21)

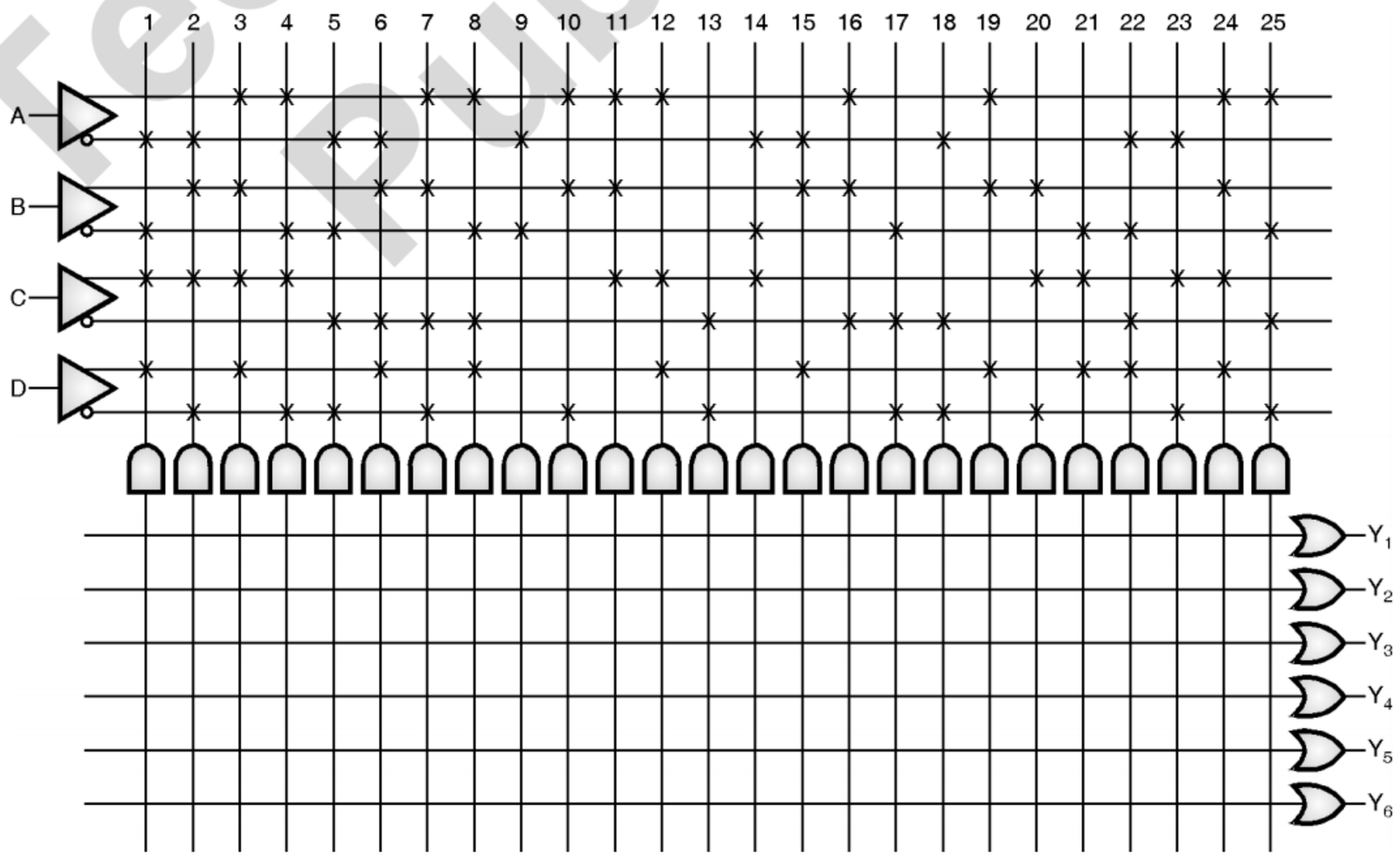
$$Y_6 = \bar{A}\bar{B}\bar{C}D + \bar{A}C\bar{D} + ABCD + A\bar{B}\bar{C}\bar{D}$$
(22) (23) (24) (25)

(C-6428)

(C-8149) Table P. 8.3.10 : Programming table for PLA

Product term no.	Product term	Inputs				Outputs					
		A	B	C	D	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
1	$\bar{A}\bar{B}CD$	0	0	1	1	1	–	–	–	–	–
2	$\bar{A}BC\bar{D}$	0	1	1	0	1	–	–	–	–	–
3	$ABCD$	1	0	1	1	1	–	–	–	–	–
4	$A\bar{B}C\bar{D}$	1	0	1	0	1	–	–	–	–	–
5	$\bar{A}\bar{B}\bar{C}\bar{D}$	0	0	0	0	1	–	–	–	–	–
6	$\bar{A}B\bar{C}D$	0	1	0	1	1	–	–	–	–	–
7	$AB\bar{C}\bar{D}$	1	1	0	0	1	–	–	–	–	–
8	$A\bar{B}\bar{C}D$	1	0	0	1	1	–	–	–	–	–
9	$\bar{A}\bar{B}$	0	0	–	–	–	1	–	–	–	–
10	$AB\bar{D}$	1	1	–	0	–	1	–	–	–	–
11	ABC	1	1	1	–	–	1	–	–	–	–
12	ACD	1	–	1	1	–	1	–	–	–	–
13	$\bar{C}\bar{D}$	–	–	0	0	–	–	1	–	–	–
14	$\bar{A}\bar{B}C$	0	0	1	–	–	–	–	1	–	–
15	$\bar{A}BD$	0	1	–	1	–	–	–	1	–	–
16	$AB\bar{C}$	1	1	0	–	–	–	–	1	–	–
17	$\bar{B}\bar{C}\bar{D}$	–	0	1	0	–	–	–	1	–	–
18	$\bar{A}\bar{C}$	0	–	0	0	–	–	–	–	1	–
19	ABD	1	1	–	1	–	–	–	–	1	–
20	$BC\bar{D}$	–	1	1	0	–	–	–	–	1	–
21	$\bar{B}CD$	–	0	1	1	–	–	–	–	1	–
22	$\bar{A}\bar{B}\bar{C}D$	0	0	0	1	–	–	–	–	–	1
23	$\bar{A}C\bar{D}$	0	–	1	0	–	–	–	–	–	1
24	$ABCD$	1	1	1	1	–	–	–	–	–	1
25	$A\bar{B}\bar{C}\bar{D}$	1	0	0	0	–	–	–	–	–	1

Step 3 : Implement the logic circuit :



(C-5913) Fig. P. 8.3.10 : Implementation using PLA

TechKnowledge Publications

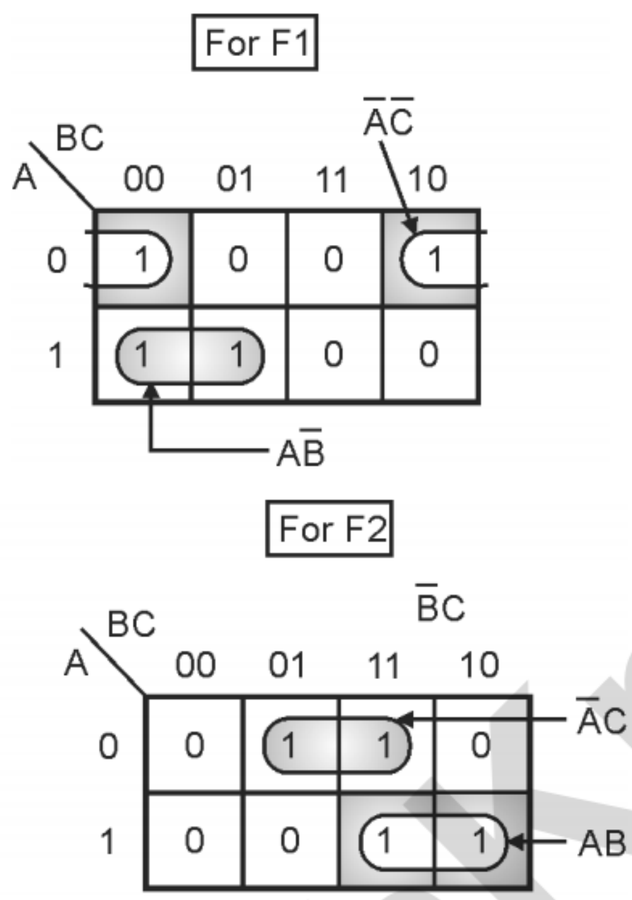
Ex. 8.3.11 : A combinational circuit is defined by following functions :

F1 (A, B, C) = ∑m (0, 2, 4, 5),

F2 (A, B, C) = ∑m (1, 3, 6, 7). Implement this using PLA. **May 19, 6 Marks**

Soln. :

Step 1 : K-maps and simplification :



(C-7930) **Fig. P. 8.3.11(a) : K-map**

$\therefore$ F1 = $A\overline{B} + \overline{A}\,\overline{C}$ $\therefore$ F2 = $AB + \overline{A}C$

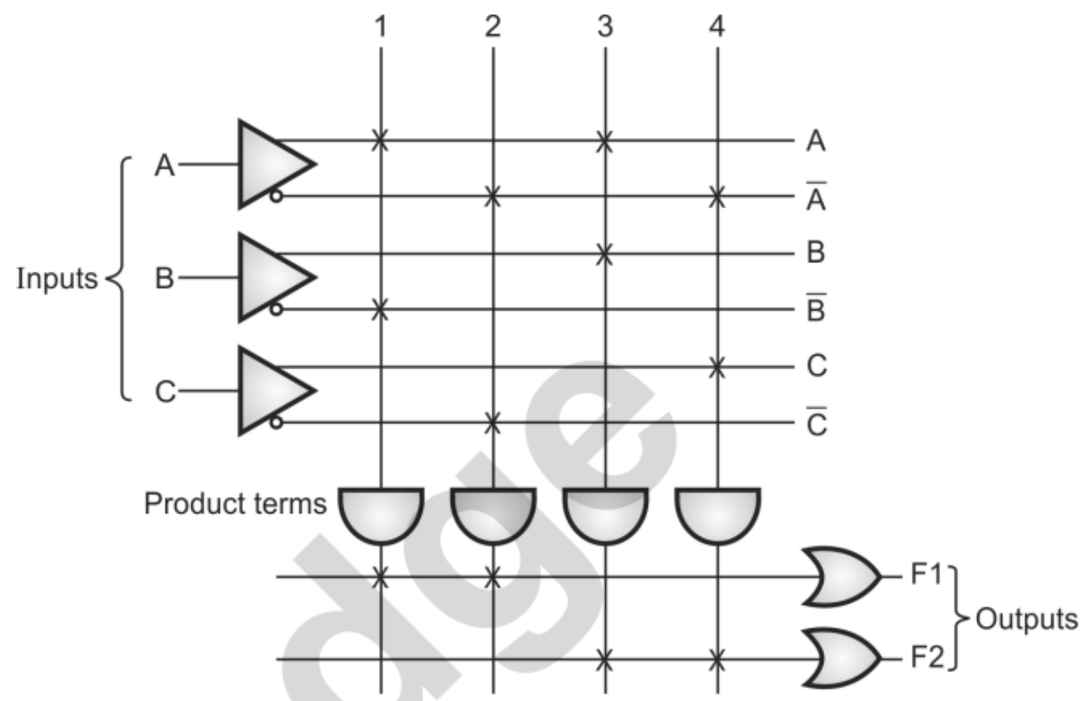
(1) (2) (3) (4)

Step 2 : Prepare the PLA programming table :

(C-7931) **Table P. 8.3.11 : PLA programming table**

Product term no.	Product term	Inputs			Outputs	
		A	B	C	F1	F2
1	$A\overline{B}$	1	0	–	1	–
2	$\overline{A}\,\overline{C}$	0	–	0	1	–
3	AB	1	1	–	–	1
4	$\overline{A}C$	0	–	1	–	1

Step 3 : Implementation using PLA :



(C-7932) **Fig. P. 8.3.11(b) : Implementation using PLA**

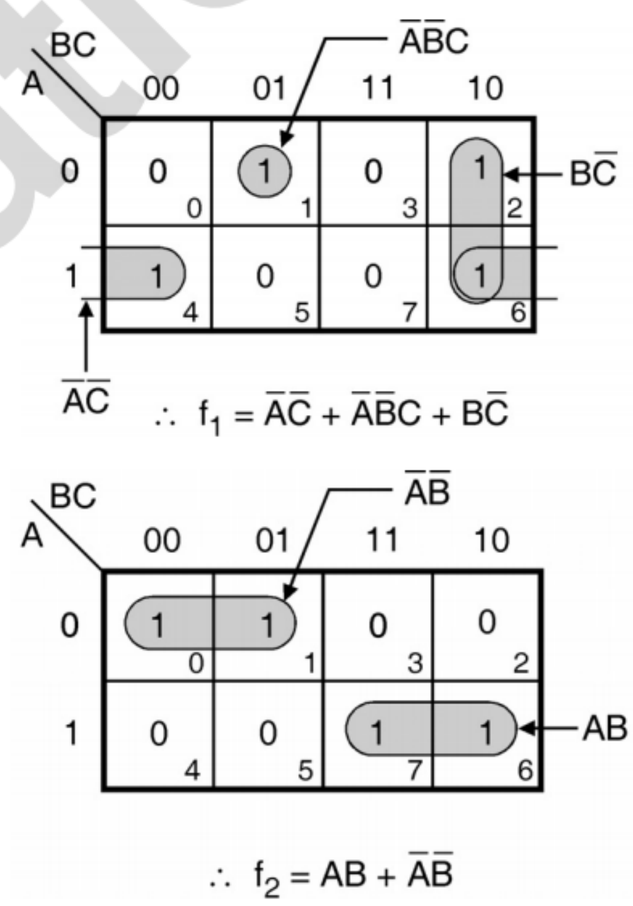
Ex. 8.3.12 : Implement the following functions using PLA :

F1 = Σ m (1, 2, 4, 6)

F2 = Σ m (0, 1, 6, 7) **Dec. 19, 6 Marks**

Soln. :

Step 1 : K-maps and simplification :



(C-4184) **Fig. P. 8.3.12 : K-maps**

Step 2 : Prepare the PLA programming table :

$f_1 = \overline{A}\,\overline{C} + \overline{A}\,\overline{B}C + B\overline{C}$, $f_2 = \overline{A}\,\overline{B} + AB$

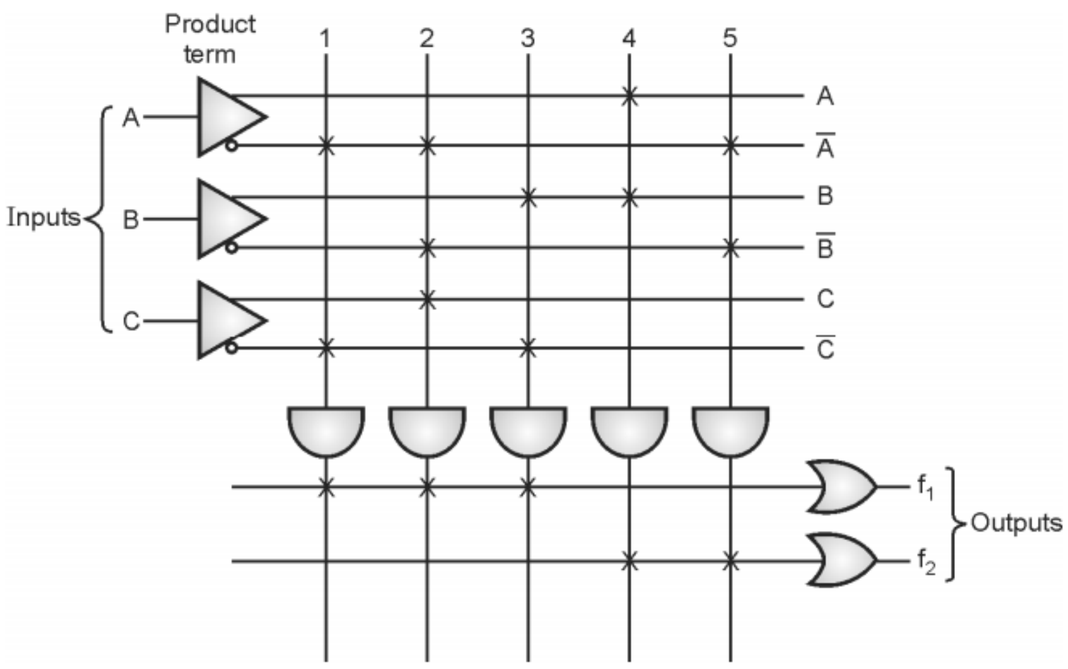
(1) (2) (3) (4) (5) **(C-6421)**

(C-8150) **Table P. 8.3.12 : Programming table for PLA**

Product term no.	Product term	Inputs			Outputs	
		A	B	C	f_1	f_2
1.	$\overline{A}\,\overline{C}$	0	–	0	1	–
2.	$\overline{A}\,\overline{B}C$	0	0	1	1	–
3.	$B\overline{C}$	–	1	0	1	–
4.	AB	1	1	–	–	1
5.	$\overline{A}\,\overline{B}$	0	0	–	–	1

TechKnowledge Publications

Step 3 : Implement the Logic circuit :



(CT-1) **Fig. P. 8.3.12 : Logic circuit**

8.4 Programmable Array Logic (PAL) :

SPPU : Dec. 04, May 05, Dec. 07, Dec. 08, May 11

University Questions

Q. 1 Draw neat diagrams of the following output arrangements of a PAL :

1. Active high 2. Active low
3. Programmable 4. Registered

(Dec. 04, 6 Marks)

Q. 2 With the help of suitable example explain the basic fixed OR and programmable AND logic of PAL.

(May 05, 4 Marks)

Q. 3 Draw and explain general structure of PAL.

(Dec. 07, Dec. 08, May 11, 4 Marks)

Definition :

- Programmable Array Logic (PAL) is a commonly used programmable logic device (PLD). It has programmable AND array and fixed OR array.
- Because only the AND array is programmable, it is easier to use but not flexible as compared to Programmable Logic Array (PLA).
- PAL is the most commonly used type of PLD. The long form of PAL is a programmable array of logic gates.
- The logic gates are used in the form of AND-OR configuration. This will allow us to use the PAL to implement expressions in the SOP form.
- The special feature of PAL is that it has a programmable AND array and a fixed OR array.
- That means the array of OR gates is not programmable.
- Also note that each OR gate in the OR array gets inputs from some of the AND gates.
- That means outputs of all the AND gates are not applied to any of the OR gates.

Configuration :

- Fig. 8.4.1 shows the configuration for a PAL with 5-inputs, 8-AND gate (programmable) and 4-fixed OR gates.
- There are 8-AND gets capable of producing 8-product terms.

1. Input Buffers :

- Input buffers are used for reducing the loading on inputs.

2. AND Matrix :

- There are 8 programmable AND gates in the AND matrix.
- These AND gates can be programmed by opening the suitable fuse links, to generate 8 product terms.

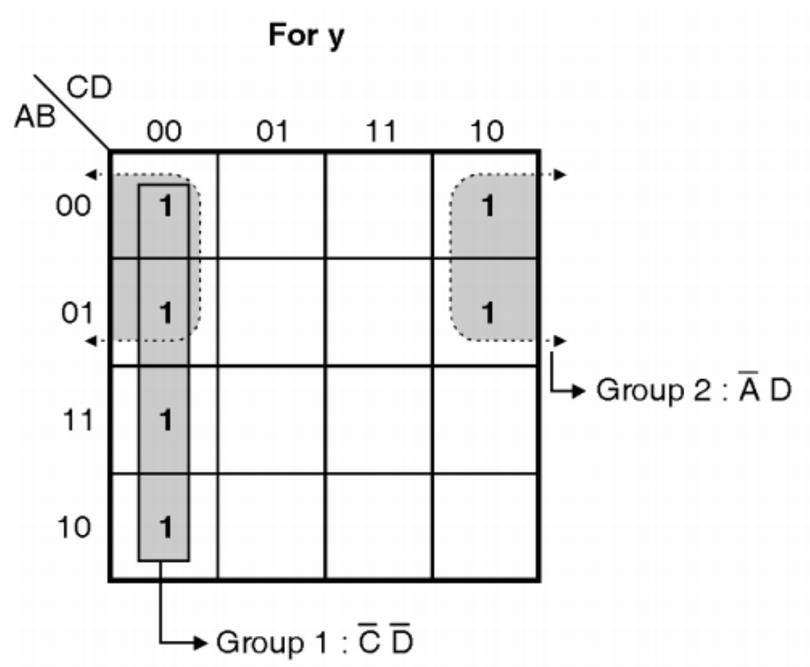
3. OR Matrix :

- The OR matrix consists of four fixed OR gates. That means we cannot programme them.
- Each OR gate receives inputs from the outputs of only two AND gates as shown in Fig. 8.4.1.

4. Input and Output Circuits :

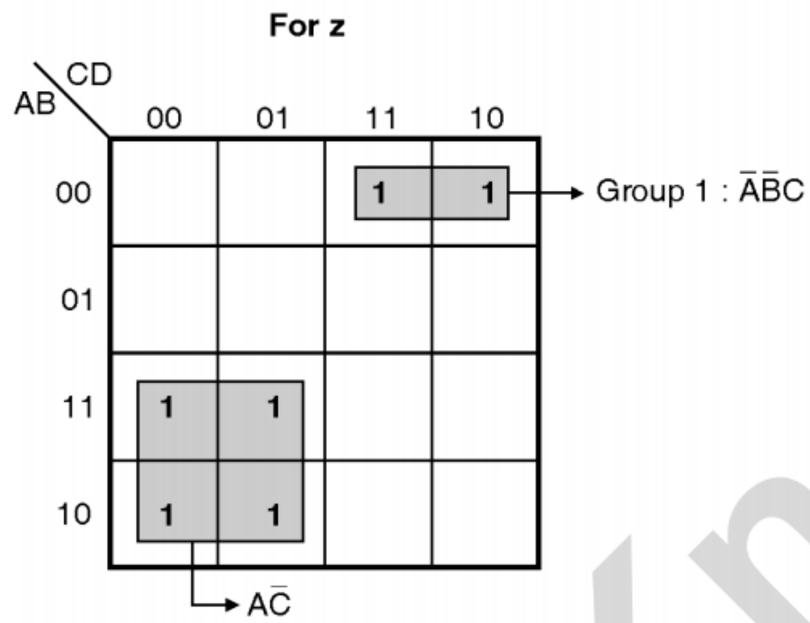
- The input and output circuits of PALs are similar to those of PLAs.
- The number of nichrome fuse links in PAL is equal to 2M × n where M = Number of available inputs and n corresponds to number of product terms.
- There are tristate buffers used between the output of each OR gate and the device output.

TechKnowledge Publications



$\therefore$ y = $\bar{C}\bar{D} + \bar{A}D$

(c)



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

(d)

(C-540) Fig. P. 8.4.1 : K-maps and simplified expressions for PAL

Step 2 : Write the PAL programming table :

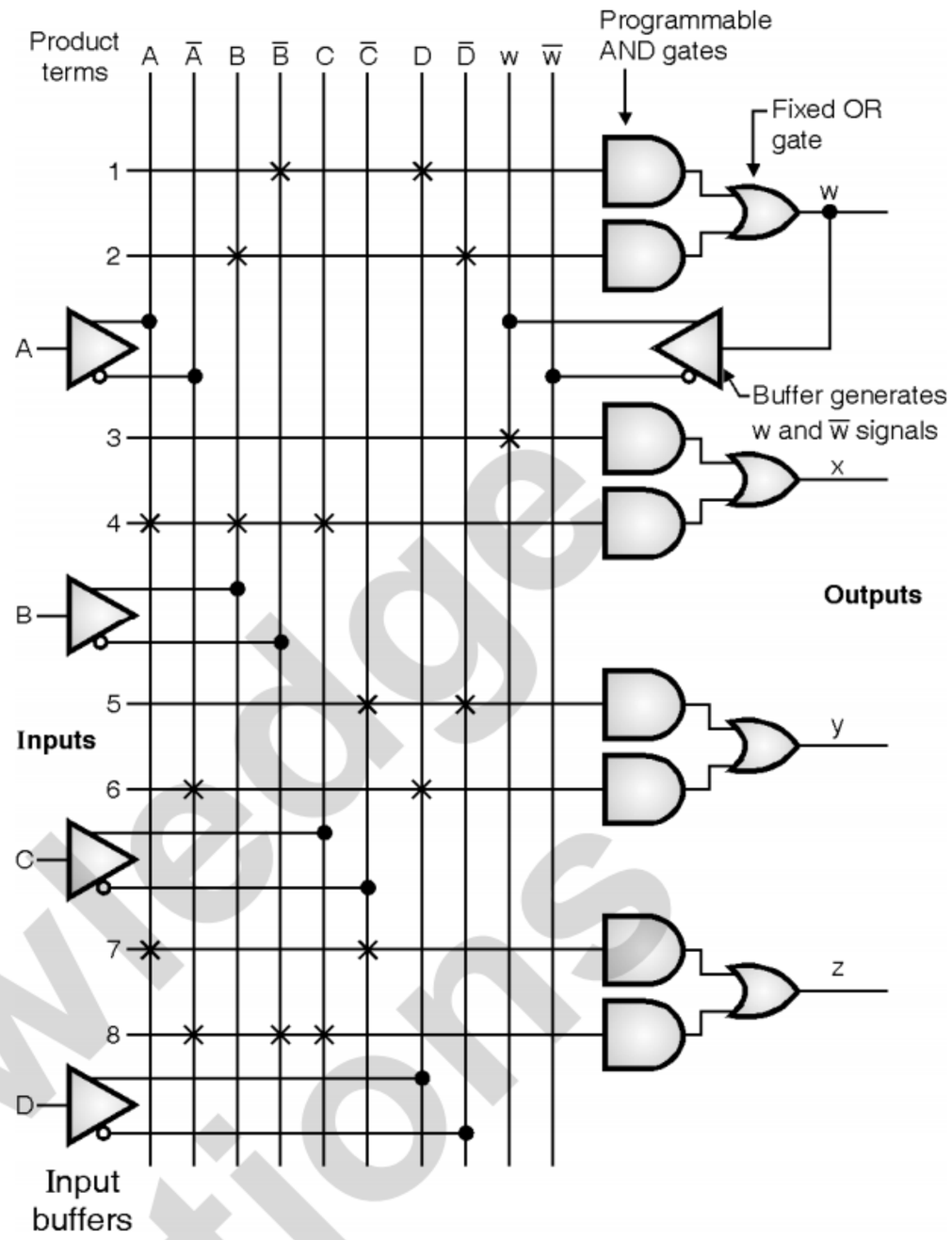
– Table P. 8.4.1 shows the PAL programming table.

(C-8151) Table P. 8.4.1 : PAL programming table

Product term no.	Product term	AND inputs					Outputs
		A	B	C	D	w	
1	$\bar{B}D$	–	0	–	1	–	$w = \bar{B}D + B\bar{D}$
2	$B\bar{D}$	–	1	–	0	–	
3	w	–	–	–	–	1	$x = w + ABC$
4	ABC	1	1	1	–	–	
5	$\bar{C}\bar{D}$	–	–	0	0	–	$y = \bar{C}\bar{D} + \bar{A}D$
6	$\bar{A}D$	0	–	–	1	–	
7	$A\bar{C}$	1	–	0	–	–	$z = A\bar{C} + \bar{A}\bar{B}C$
8	$\bar{A}\bar{B}C$	0	0	1	–	–	

Step 3 : Logic diagram :

– The logic diagram is as shown in Fig. P. 8.4.1(e).



(C-541) Fig. P. 8.4.1(e) : Implementation using PAL

Ex. 8.4.2 : Design a BCD to excess-3 code converter and implement it using PAL.

Soln. :

Step 1 : Write the truth table :

– The truth table relating the BCD and excess 3 codes is shown in Table P. 8.4.2.

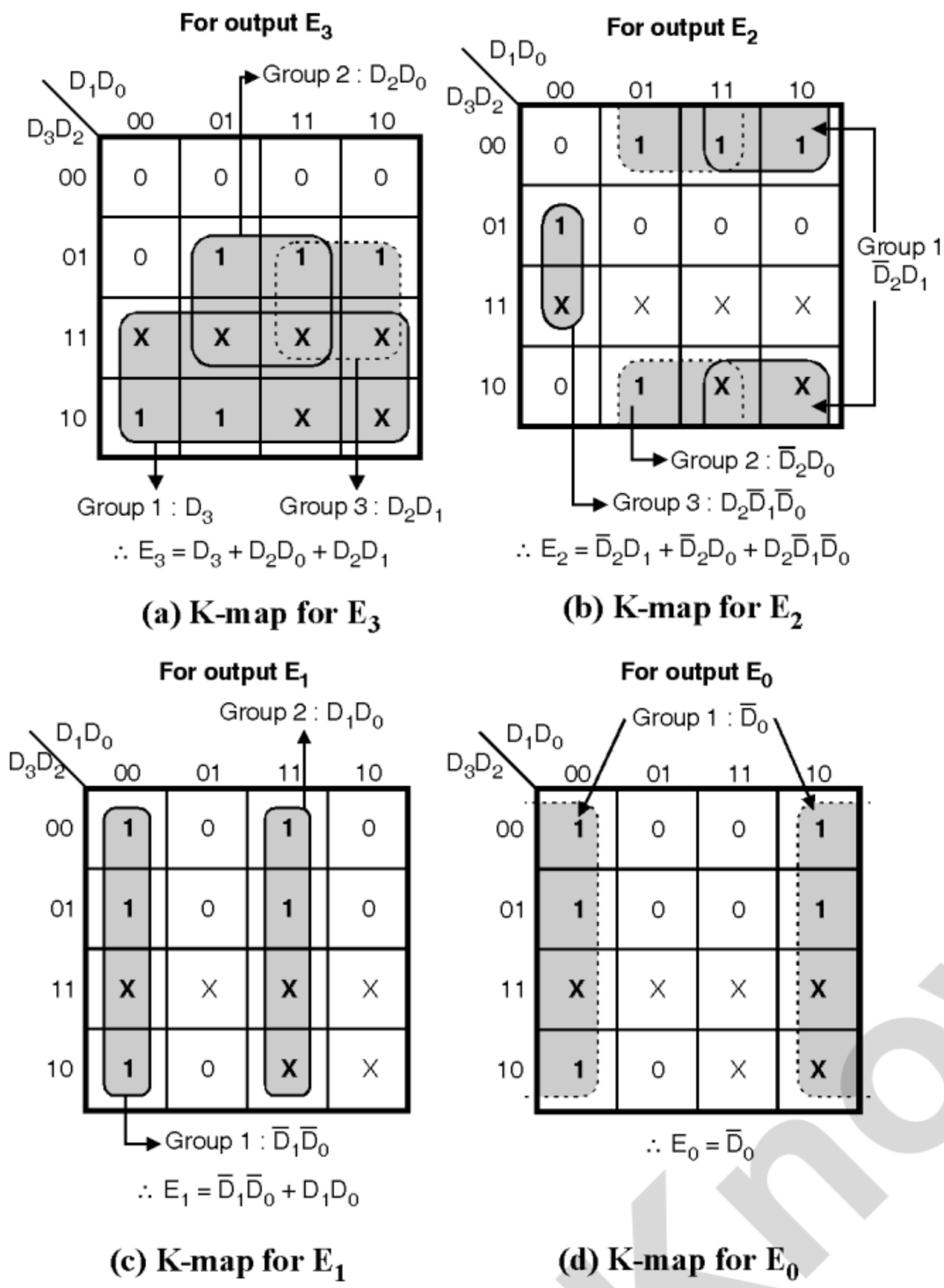
(C-8109) Table P. 8.4.2 : Truth table relating BCD and Excess-3 codes

Decimal	BCD inputs				Excess-3 output			
	D_3	D_2	D_1	D_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

Step 2 : K-maps and simplified expressions :

– Refer Figs. P. 8.4.2(a) to (d) for K-maps and simplified expressions.

TechKnowledge Publications



(C-532) **Fig. P. 8.4.2**

– The four simplified expressions and associated product terms are as follows :

$E_3 = D_3 + D_2D_0 + D_2D_1$ — Product term → (1) (2) (3)

$E_2 = \overline{D}_2D_1 + \overline{D}_2D_0 + D_2\overline{D}_1\overline{D}_0$ — Product term → (4) (5) (6)

$E_1 = \overline{D}_1\overline{D}_0 + D_1D_0$ — Product term → (7) (8)

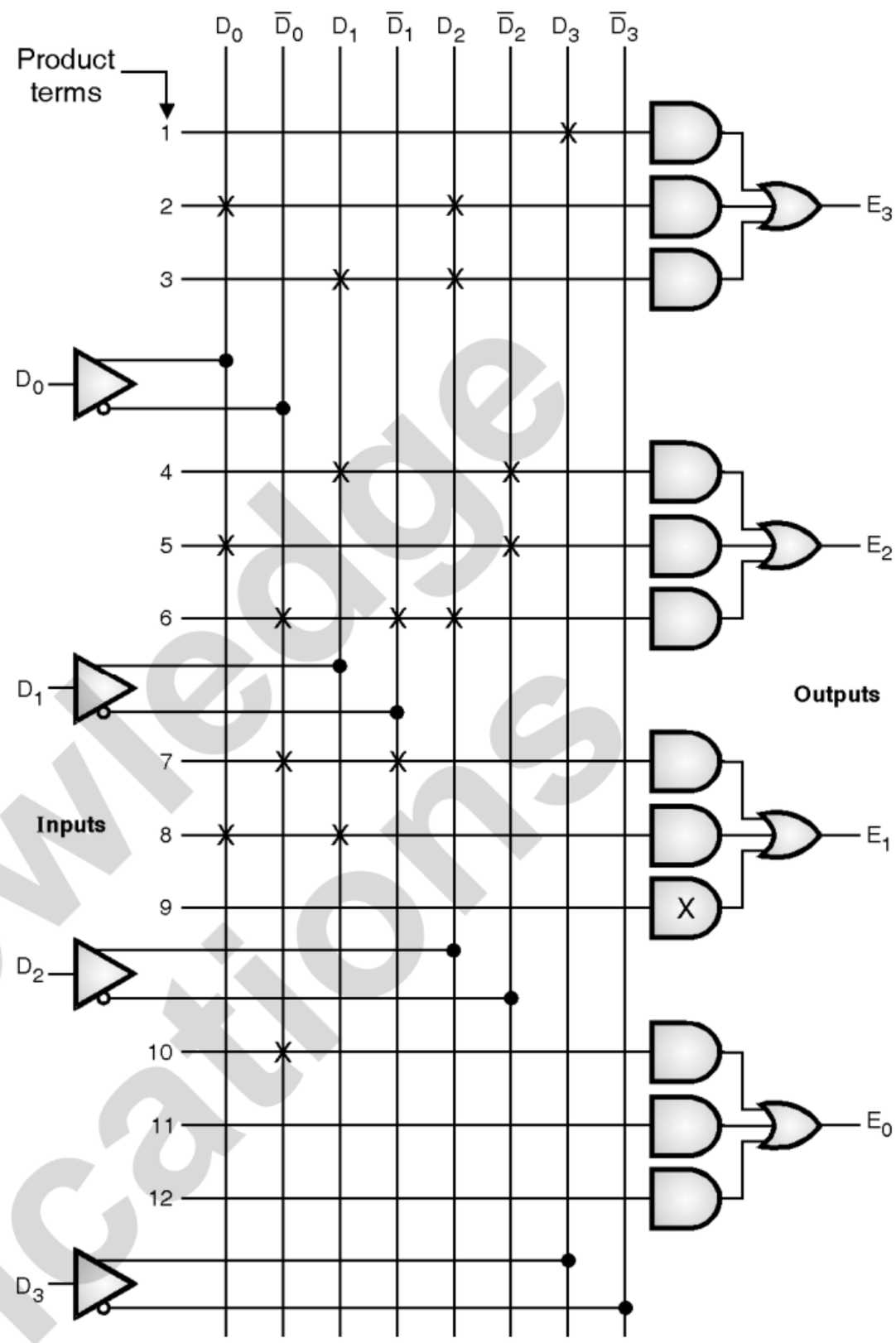
$E_0 = \overline{D}_0$ — Product term → (9)

(C-6419)

Step 3 : Write the PAL programming table : (C-6423)

Product term number	Product term	AND inputs D_3	D_2	D_1	D_0	Outputs
1.	D_3	1	–	–	–	$E_3 = D_3 + D_2D_0 + D_2D_1$
2.	D_2D_0	–	1	–	1	
3.	D_2D_1	–	1	1	–	
4.	$\overline{D}_2D_1$	–	0	1	–	$E_2 = \overline{D}_2D_1 + \overline{D}_2D_0 + \overline{D}_1\overline{D}_0D_2$
5.	$\overline{D}_2D_0$	–	0	–	1	
6.	$\overline{D}_1\overline{D}_0D_2$	–	1	0	0	
7.	$\overline{D}_1\overline{D}_0$	–	–	0	0	$E_1 = \overline{D}_1\overline{D}_0 + D_1D_0$
8.	D_1D_0	–	–	1	1	
9.	–	–	–	–	–	
10.	$\overline{D}_0$	–	–	–	0	$E_0 = \overline{D}_0$
11.	–	–	–	–	–	
12.	–	–	–	–	–	

Step 4 : Implementation :



(C-3427) **Fig. P. 8.4.2 : Implementation using PAL**

Ex. 8.4.3 : Design a combinational circuit using a suitable PAL considering the following Boolean expressions. Use a PAL with four inputs, four outputs and three wide AND-OR structure.

w (A, B, C, D) = Σ m (2, 12, 13)

x (A, B, C, D)

= Σ m (7, 8, 9, 10, 11, 12, 13, 14, 15)

y (A, B, C, D)

= Σ m (0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)

z (A, B, C, D) = Σ m (1, 2, 8, 12, 13)

Soln. :

Step 1 : K-maps and simplifications :

– The K-maps and simplified expressions are as shown in Fig. P. 8.4.3.

TechKnowledge Publications

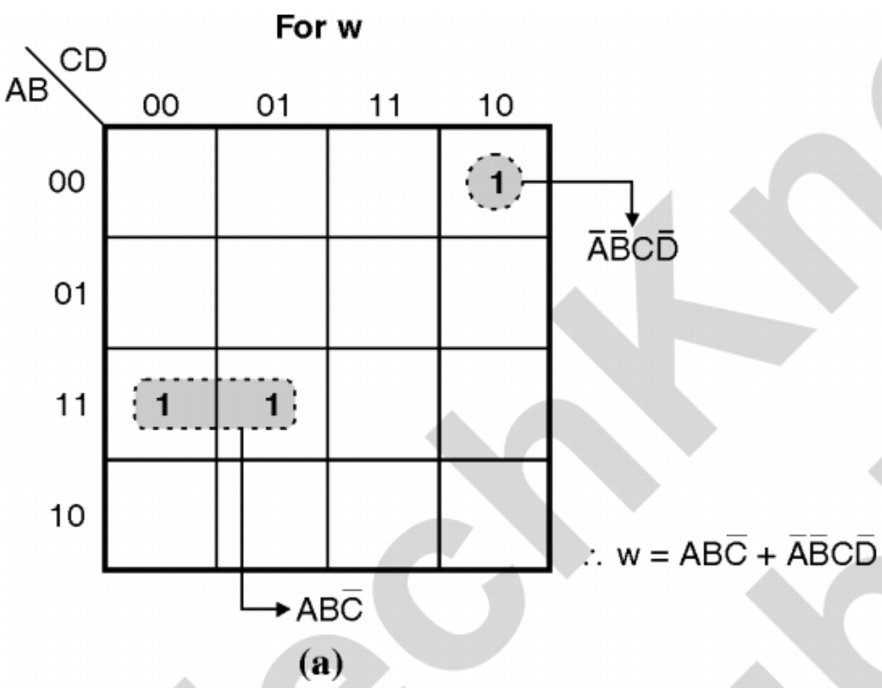
– As we have been told to use a 4 input, 4 output, 3 wide AND OR structured PAL, we must restrict the number of product terms in expression for each output to at the most 3.

– But z consists of 4-product terms. Hence as shown in Fig. P. 8.4.3(d), we have reduced them to 3 by substituting,

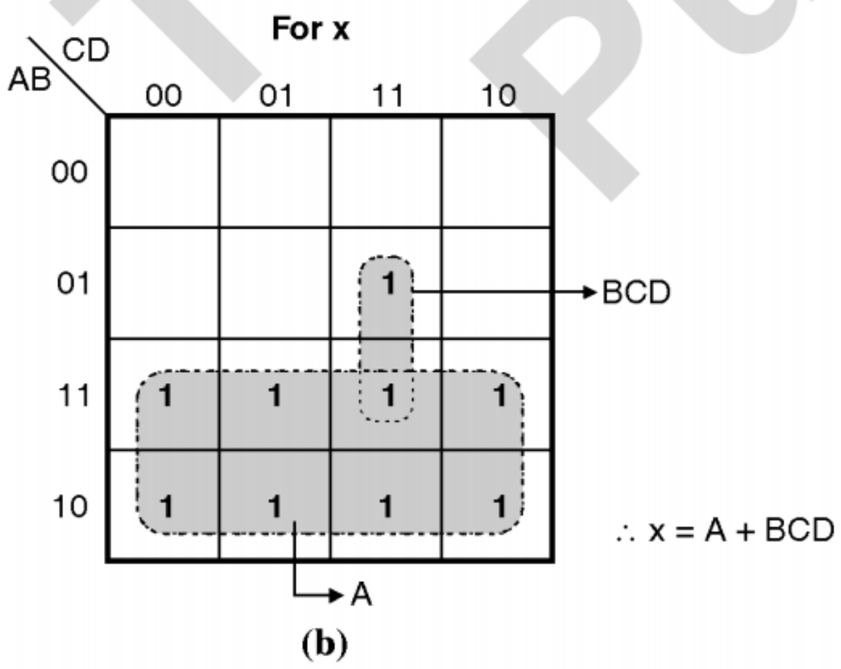
$$w = \bar{A}\,\bar{B}C\bar{D} + AB\bar{C}.$$

– The outputs w and x contain only two product terms each.

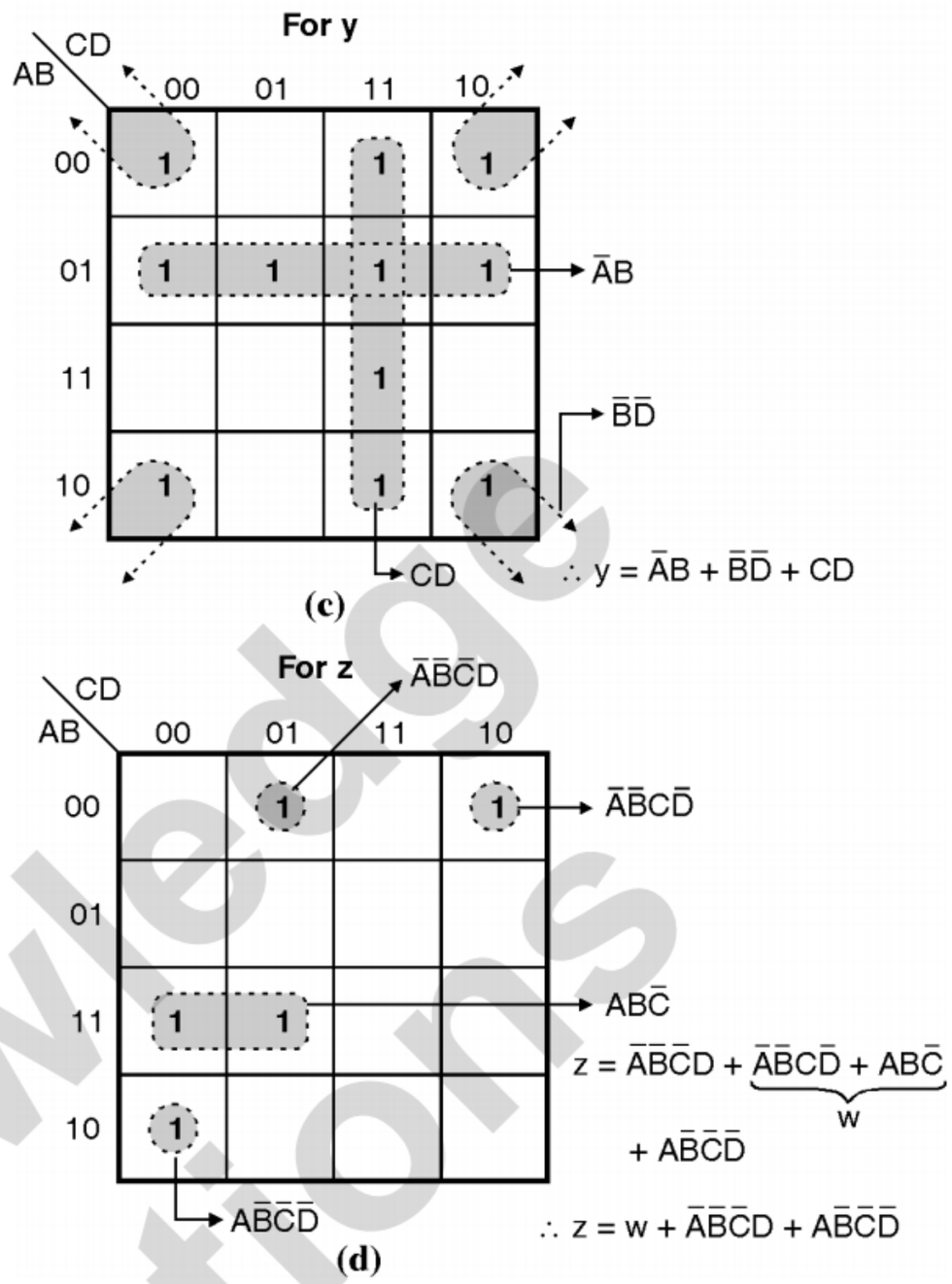
– So one AND gate for each one of them will remain unused. Then we have to leave all the fuse links intact for an unused AND gate intact.



(a)



(b)



(C-542) Fig. P. 8.4.3 : K-maps and simplifications

Step 2 : Prepare the PAL programming table :

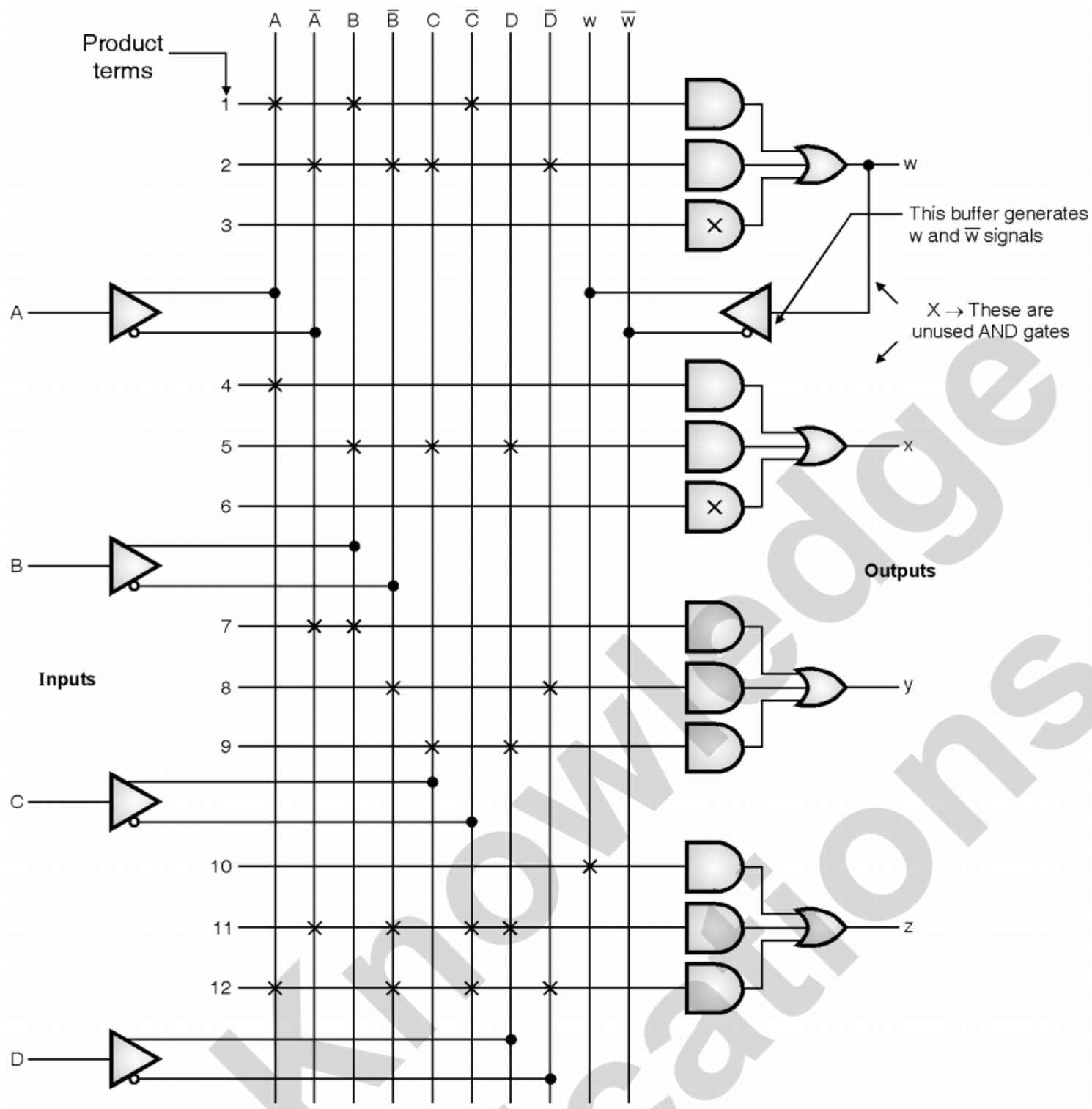
(C-8111) Table P. 8.4.3 : PAL programming table

Product term no.	Product term	AND inputs					Outputs
		A	B	C	D	w	
1	$AB\bar{C}$	1	1	0	–	–	$w = AB\bar{C} + \bar{A}\,\bar{B}\,C\bar{D}$
2	$\bar{A}\,\bar{B}C\bar{D}$	0	0	1	0	–	
3	–	–	–	–	–	–	
4	A	1	–	–	–	–	$x = A + BCD$
5	BCD	–	1	1	1	–	
6	–	–	–	–	–	–	
7	$\bar{A}B$	0	1	–	–	–	$y = \bar{A}B + \bar{B}\,\bar{D} + CD$
8	$\bar{B}\,\bar{D}$	–	0	–	0	–	
9	CD	–	–	1	1	–	
10	w	–	–	–	–	1	$z = w + \bar{A}\,\bar{B}\,\bar{C}\,D + A\bar{B}\,\bar{C}\,\bar{D}$
11	$\bar{A}\,\bar{B}\,\bar{C}\,D$	0	0	0	1	–	
12	$A\bar{B}\,\bar{C}\,\bar{D}$	1	0	0	0	–	

Step 3 : Implementation :

– Refer Fig. P. 8.4.3(e) for the implementation.

TechKnowledge Publications



(C-543) **Fig. P. 8.4.3(e) : Implementation using PAL**

Ex. 8.4.4 : Implement the following function using PAL :

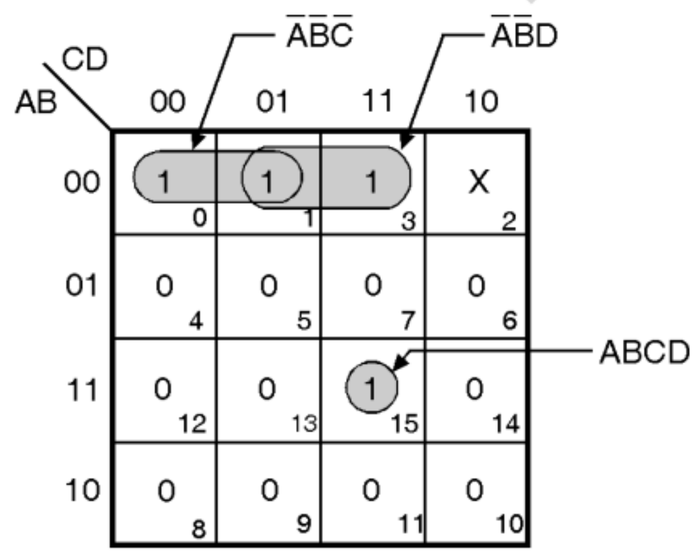
$F(A, B, C, D) = \sum m\,(0, 1, 3, 15)$

Dec. 07, 8 Marks, May 12, 4 Marks

Soln. :

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

Step 1 : Simplify the equation using K-map :



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

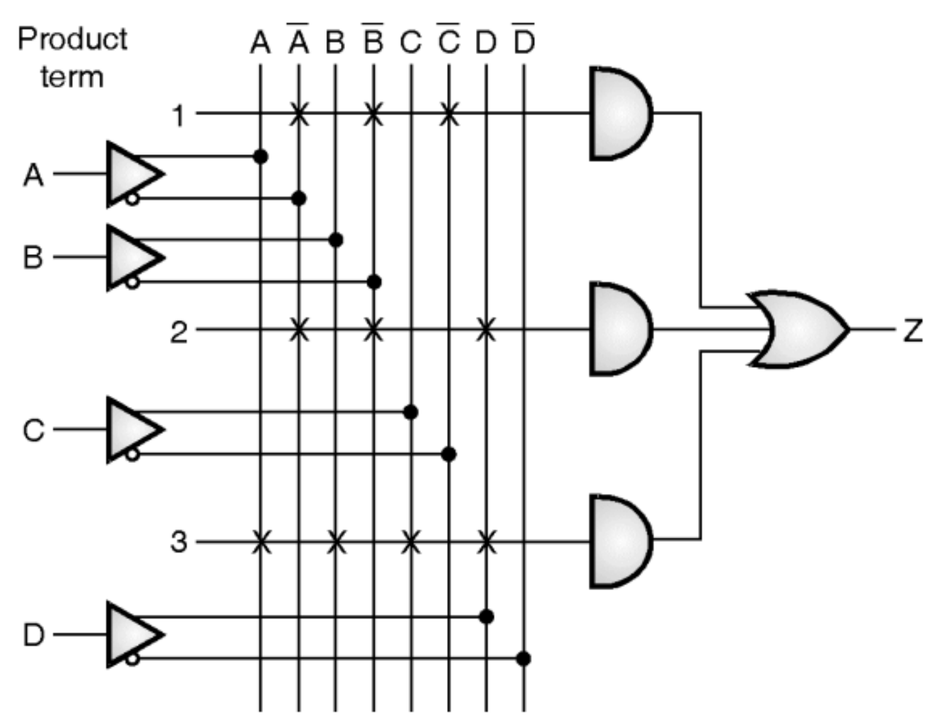
(C-3246) **Fig. P. 8.4.4(a)**

Step 2 : Write PAL programming table :

(C-8161) **Table P. 8.4.4**

Product term no.	Product term	And inputs				Output
		A	B	C	D	
1.	$\bar{A}\bar{B}\bar{C}$	0	0	0	–	$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}D + ABCD$
2.	$\bar{A}\bar{B}D$	0	0	–	1	
3.	ABCD	1	1	1	1	

Step 3 : Logic diagram :



(C-3247) **Fig. P. 8.4.4(b)**

Ex. 8.4.5 : Implement 4 : 1 multiplexer using suitable PAL. **Dec. 05, Dec. 06, May 07, 6 Marks, Dec. 14, 6 Marks**

Soln. :

Step 1 : Write the truth table :

(C-8152) Table P. 8.4.5(a)

Select inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

– Hence the logical expression for output is,

$$Y = \overline{S}_1\,\overline{S}_0\,D_0 + \overline{S}_1\,S_0\,D_1 + S_1\,\overline{S}_0\,D_2 + S_1\,S_0\,D_3$$

(1) (2) (3) (4)

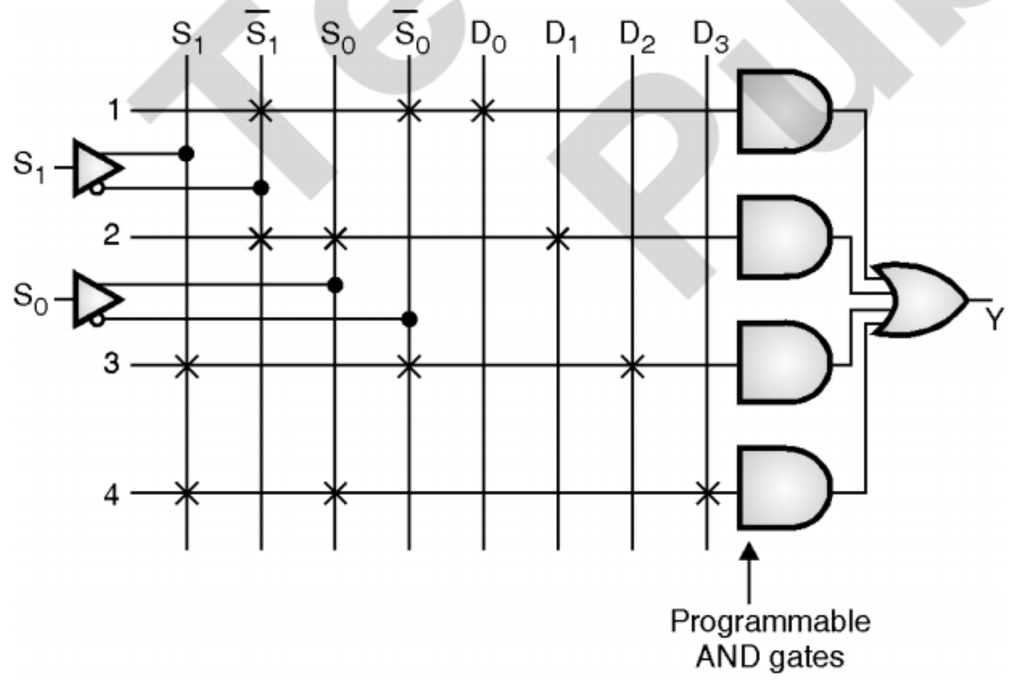
(C-6424)

Step 2 : PAL programming table :

(C-8153) Table P. 8.4.5(b)

Product term no.	Product term	AND inputs						Output
		S_1	S_0	D_0	D_1	D_2	D_3	
1	$\overline{S}_1\overline{S}_0D_0$	0	0	✓	–	–	–	$Y = \overline{S}_1\,\overline{S}_0\,D_0$
2	$\overline{S}_1S_0D_1$	0	1	–	✓	–	–	$+\,\overline{S}_1\,S_0\,D_1$
3	$S_1\overline{S}_0D_2$	1	0	–	–	✓	–	$+\,S_1\,\overline{S}_0\,D_2$
4	$S_1S_0D_3$	1	1	–	–	–	✓	$+\,S_1\,S_0\,D_3$

Step 3 : Logic diagram :



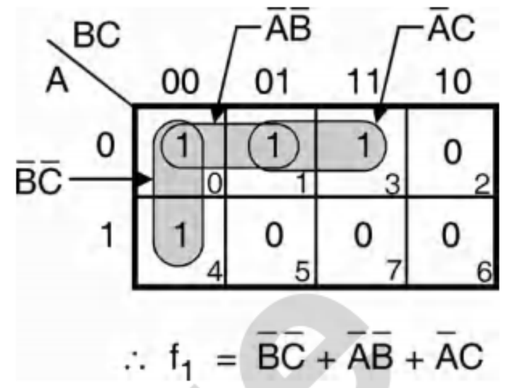
(C-3574) Fig. P. 8.4.5

Ex. 8.4.6 : A combinational circuit is defined by the function : f_1 (A, B, C) = Σ m (0, 1, 3, 4). Implement this circuit with PAL.

Dec. 13, 6 Marks, Dec. 17, 4 Marks

Soln. :

Step 1 : Simplify the equation using K-map :



$$\therefore\ f_1 = \overline{B}\,\overline{C} + \overline{A}\,\overline{B} + \overline{A}C$$

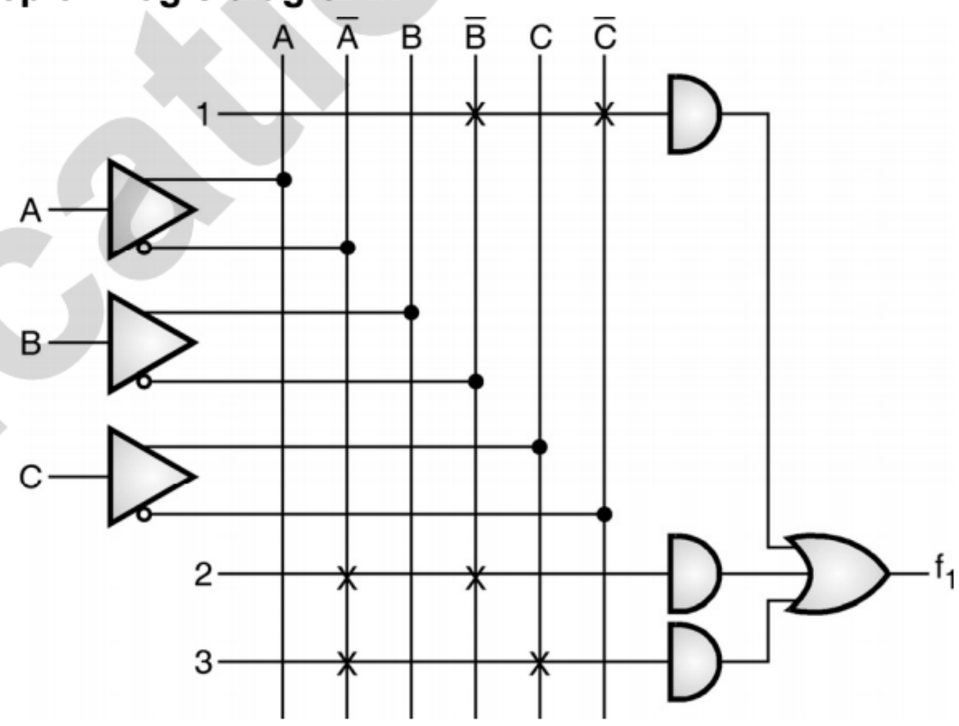
(C-4192) Fig. P. 8.4.6(a)

Step 2 : Write PAL programming table :

(C-8154) Table P. 8.4.6(a)

Product term no.	Product term	Inputs			Output
		A	B	C	
1	$\overline{B}\,\overline{C}$	–	0	0	$f_1 = \overline{B}\,\overline{C} + \overline{A}\,\overline{B} + \overline{A}C$
2	$\overline{A}\,\overline{B}$	0	0	–	
3	$\overline{A}C$	0	–	1	

Step 3 : Logic diagram :



(C-4193) Fig. P. 8.4.6(b) : Logic diagram using PAL

Ex. 8.4.7 : Implement the following Boolean function using PAL :

w (A, B, C, D) = ∑ m (0, 2, 6, 7, 8, 9, 12, 13)

x (A, B, C, D) = ∑ m (0, 2, 6, 7, 8, 9, 12, 13, 14)

y (A, B, C, D) = ∑ m (2, 3, 8, 9, 10, 12, 13)

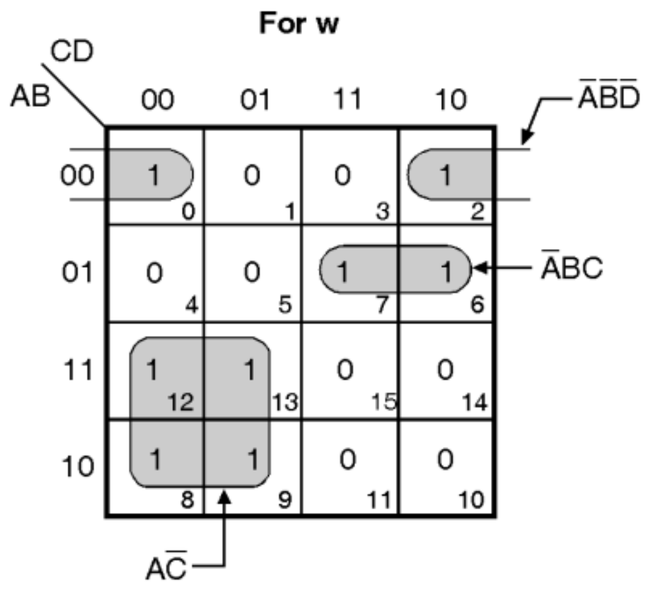
z (A, B, C, D) = ∑ m (1, 3, 4, 6, 9, 12, 14)

May 14, 7 Marks

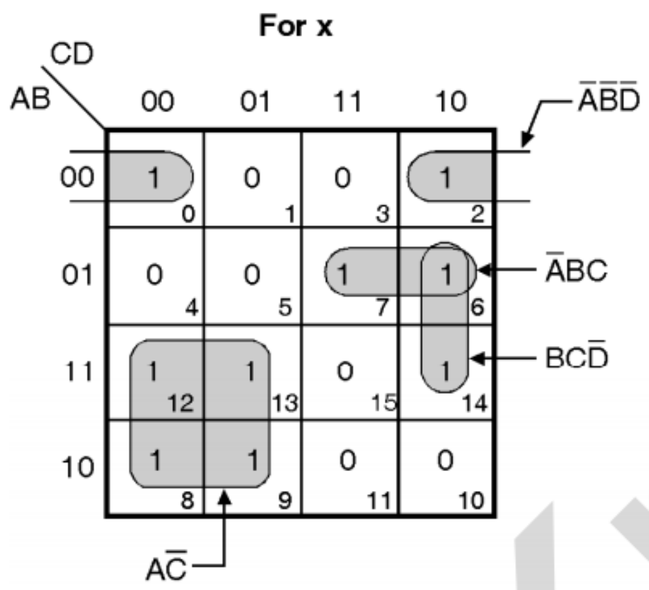
TechKnowledge Publications

Soln. :

Step 1 : Simplify the equations using K-map :

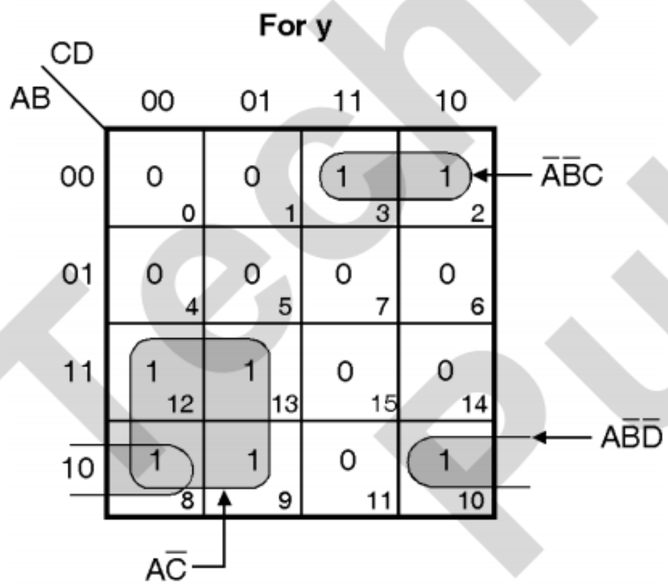


$\therefore\ w = A\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{D}$

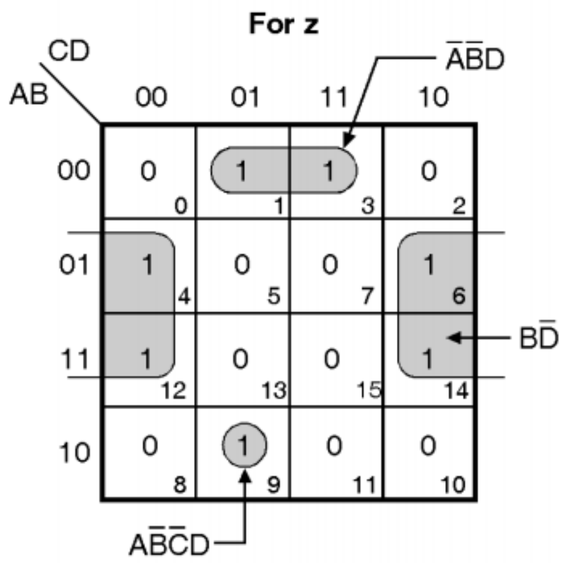


$\therefore\ x = A\bar{C} + BC\bar{D} + \bar{A}BC + \bar{A}\bar{B}\bar{D}$

$x = w + BC\bar{D}$



$\therefore\ y = A\bar{C} + A\bar{B}\bar{D} + \bar{A}\bar{B}C$



$\therefore\ z = \bar{A}\bar{B}\bar{D} + B\bar{D} + A\bar{B}\bar{C}D$

(C-4803) Fig. P. 8.4.7(a)

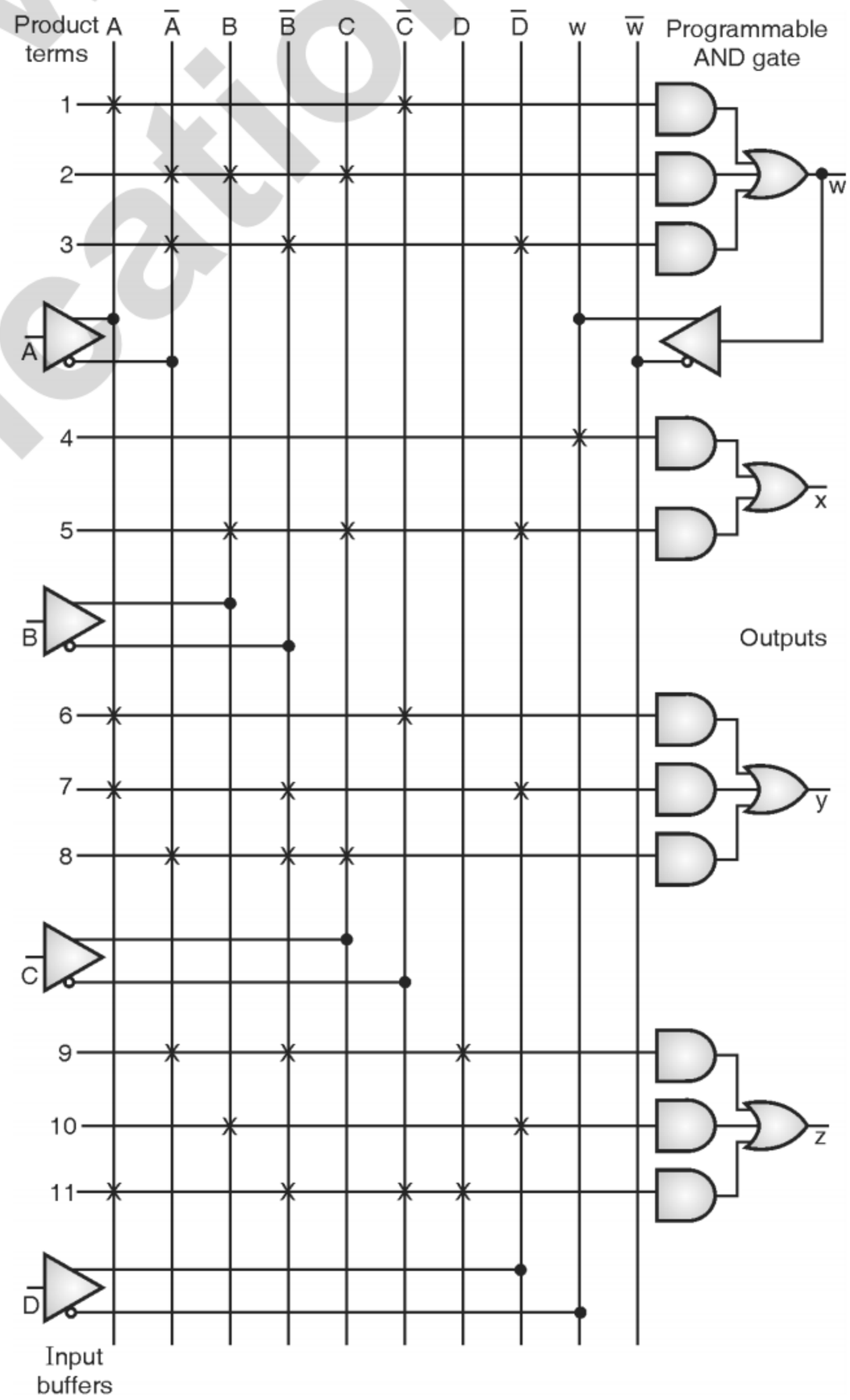
Step 2 : Write the PAL programming table :

– Table P. 8.4.7 shows the PAL programming table.

(C-8155) Table P. 8.4.7

Product term no.	Product term	AND inputs					Outputs
		A	B	C	D	w	
1.	$A\bar{C}$	1	–	0	–	–	$w = A\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{D}$
2.	$\bar{A}BC$	0	1	1	–	–	
3.	$\bar{A}\bar{B}\bar{D}$	0	0	–	0	–	
4.	w	–	–	–	–	1	$x = w + BC\bar{D}$
5.	$BC\bar{D}$	–	1	1	0	–	
6.	$A\bar{C}$	1	–	0	–	–	$y = A\bar{C} + A\bar{B}\bar{D} + \bar{A}\bar{B}C$
7.	$A\bar{B}\bar{D}$	1	0	–	0	–	
8.	$\bar{A}\bar{B}C$	0	0	1	–	–	
9.	$\bar{A}\bar{B}D$	0	0	–	1	–	$z = \bar{A}\bar{B}D + B\bar{D} + A\bar{B}\bar{C}D$
10.	$B\bar{D}$	–	1	–	0	–	
11.	$A\bar{B}\bar{C}D$	1	0	0	1	–	

Step 3 : Logic diagram :



(C-4809) Fig. P. 8.4.7(b) : Implementation using PAL

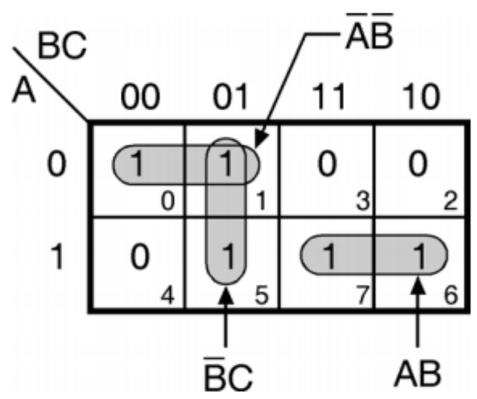
TechKnowledge Publications

Ex. 8.4.8 : A combinational circuits is defined by the function : F_1 (A, B, C) = Σ m (0, 1, 5, 6, 7). Implement this circuit with PAL.

May 15, 6 Marks

Soln. :

Step 1 : Simplify the equation using K-map :



(C-5109) Fig. P. 8.4.8(a)

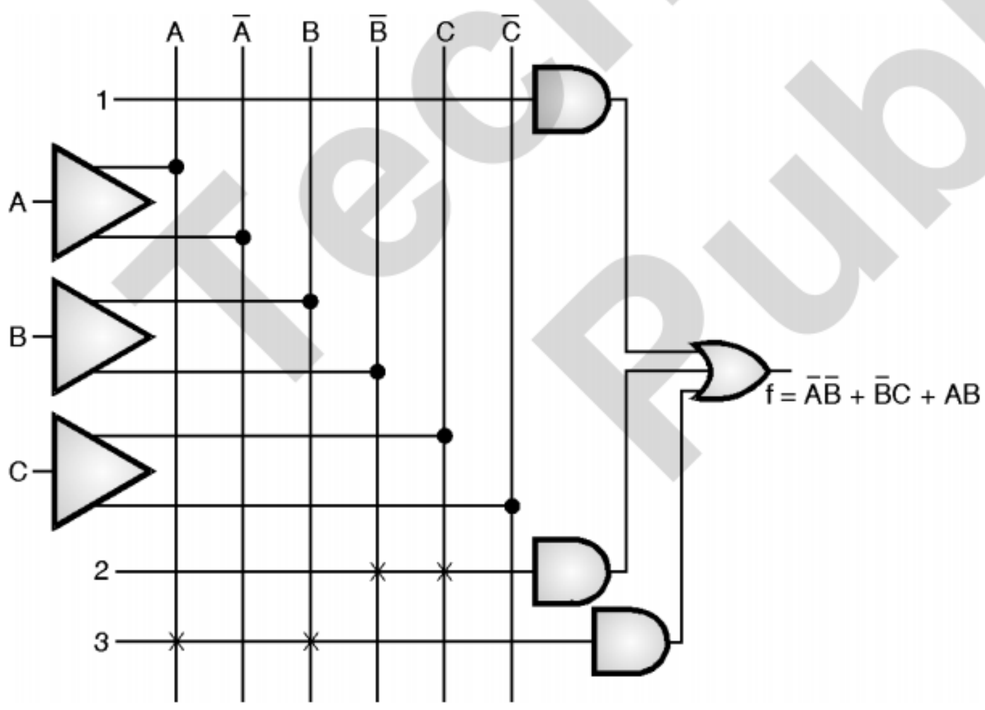
$$\therefore F_1 = \overline{A}\,\overline{B} + \overline{B}C + AB$$

Step 2 : Write PAL programming table :

(C-8156) Table P. 8.4.8

Product term no.	Product term	Inputs			Outputs
		A	B	C	
1	$\overline{A}\,\overline{B}$	0	0	–	$F_1 = \overline{A}\,\overline{B} + \overline{B}\,C + AB$
2	$\overline{B}C$	–	0	1	
3	AB	1	1	–	

Step 3 : Logic diagram :



(C-5110) Fig. P. 8.4.8(b) : Logic diagram

Ex. 8.4.9 : Show how PAL is programmed for the following 3 variable logic function :

1. $X = AB'C + A'BC' + A'B' + AC$
2. $Y = A'B'C + AB'C' + A + AB$.

Dec. 15, 6 Marks

Soln. :

1. $X = AB'C + A'BC' + A'B' + AC$:

Step 1 : Convert the given function into canonical form :

$$\therefore AB'C + A'BC' + A'B'(C + C') + AC(B + B')$$

$$= AB'C + A'BC' + A'B'C + A'B'C' + ABC + AB'C$$

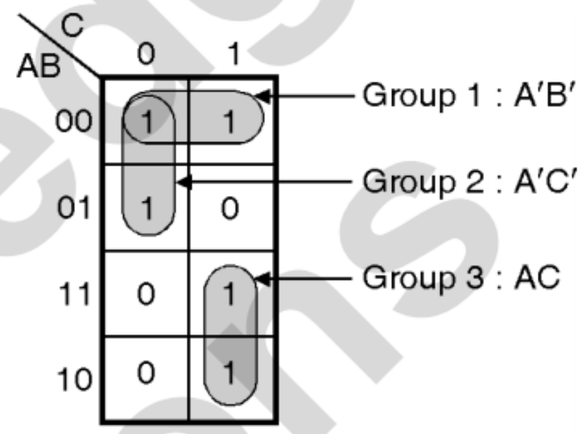
$$= AB'C + A'BC' + A'B'C + A'B'C' + ABC$$

$$[\because AB'C + AB'C = AB'C]$$

∴ Function X can be represented as,

$$X = \Sigma m (0, 1, 2, 5, 7)$$

Step 2 : Simplify using K-map :



$$\therefore X = A'B' + A'C' + AC$$

(C-1172(a))

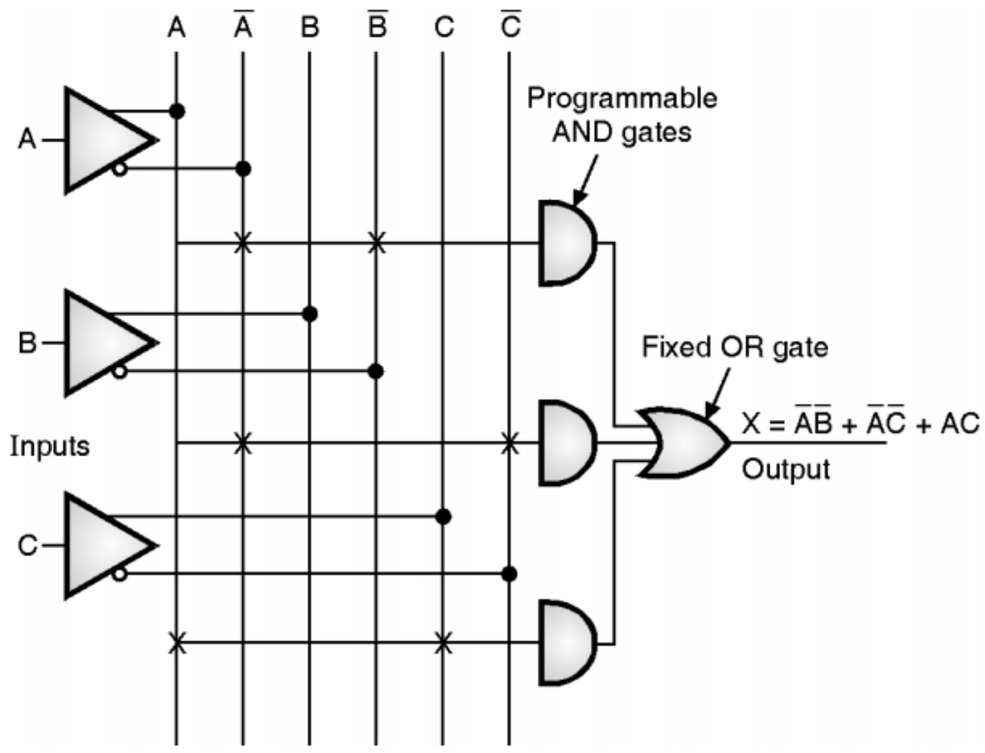
Step 3 : Write the PAL programming table :

– Table P. 8.4.9 shows the PAL programming table.

(C-8157) Table P. 8.4.9 : PAL programming table

Product term no.	Product term	Inputs			Output
		A	B	C	
1.	A' B'	0	0	–	X = A' B' + A' C' + AC
2.	A' C'	0	–	0	
3.	AC	1	–	1	

Step 4 : Logic diagram :



(C-1173) Fig. P. 8.4.9(a) : Implementation using PAL

TechKnowledge Publications

2. $Y = A'B'C + AB'C' + A + AB$:

Step 1 : Convert the given function into canonical form :

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

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

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

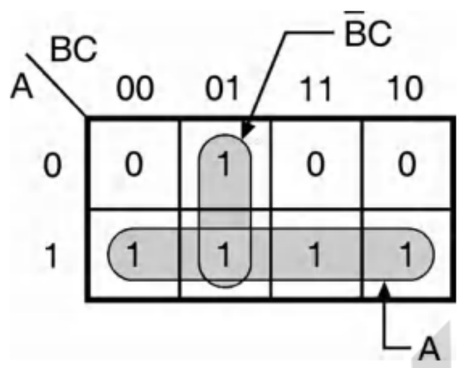
$$= \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC + AB\bar{C} + A\bar{B}C + ABC + AB\bar{C}$$

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

$$\ldots (\because \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} = A\bar{B}\bar{C},\ ABC + ABC = ABC,\ AB\bar{C} + AB\bar{C} = AB\bar{C})$$

$$Y = \sum m(1, 4, 5, 6, 7)$$

Step 2 : Simplify using K-map :



(C-5175) **Fig. P. 8.4.9(b) : K-map**

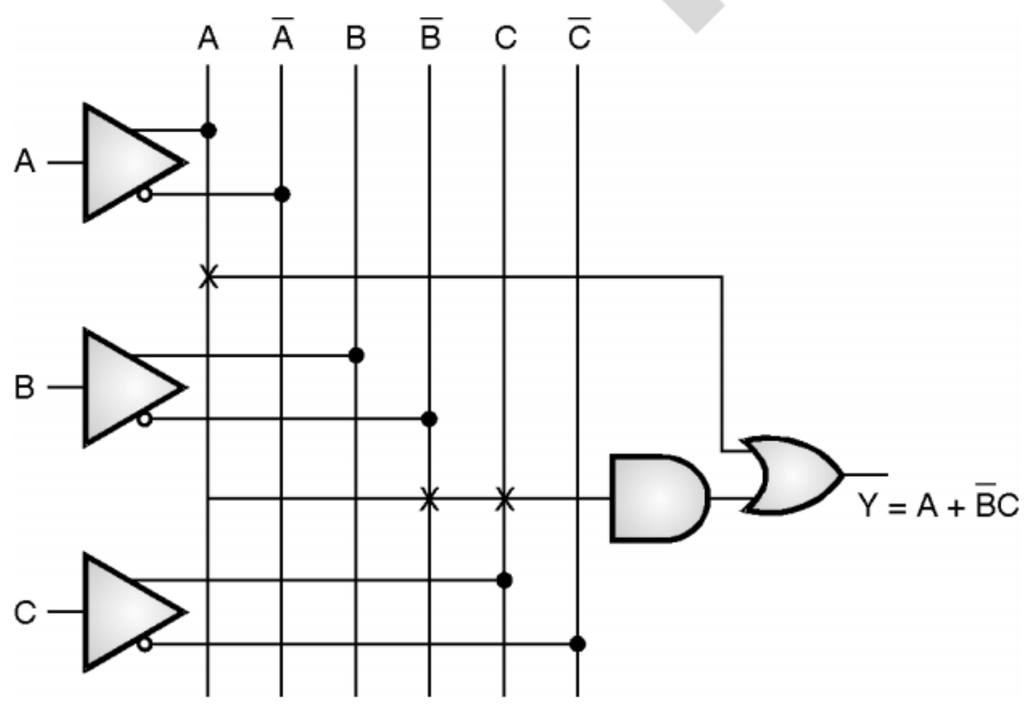
$$\therefore Y = A + \bar{B}C$$

Step 3 : Write the PAL programming table :

(C-8158) **Table P. 8.4.9(a) : PAL programming table**

Product term no.	Product term	AND inputs			Output
		A	B	C	
1.	A	1	–	–	$Y = A + \bar{B}C$
2.	$\bar{B}C$	–	0	1	

Step 4 : Logic diagram :



(C-5176) **Fig. P. 8.4.9(c) : Implementation using PAL**

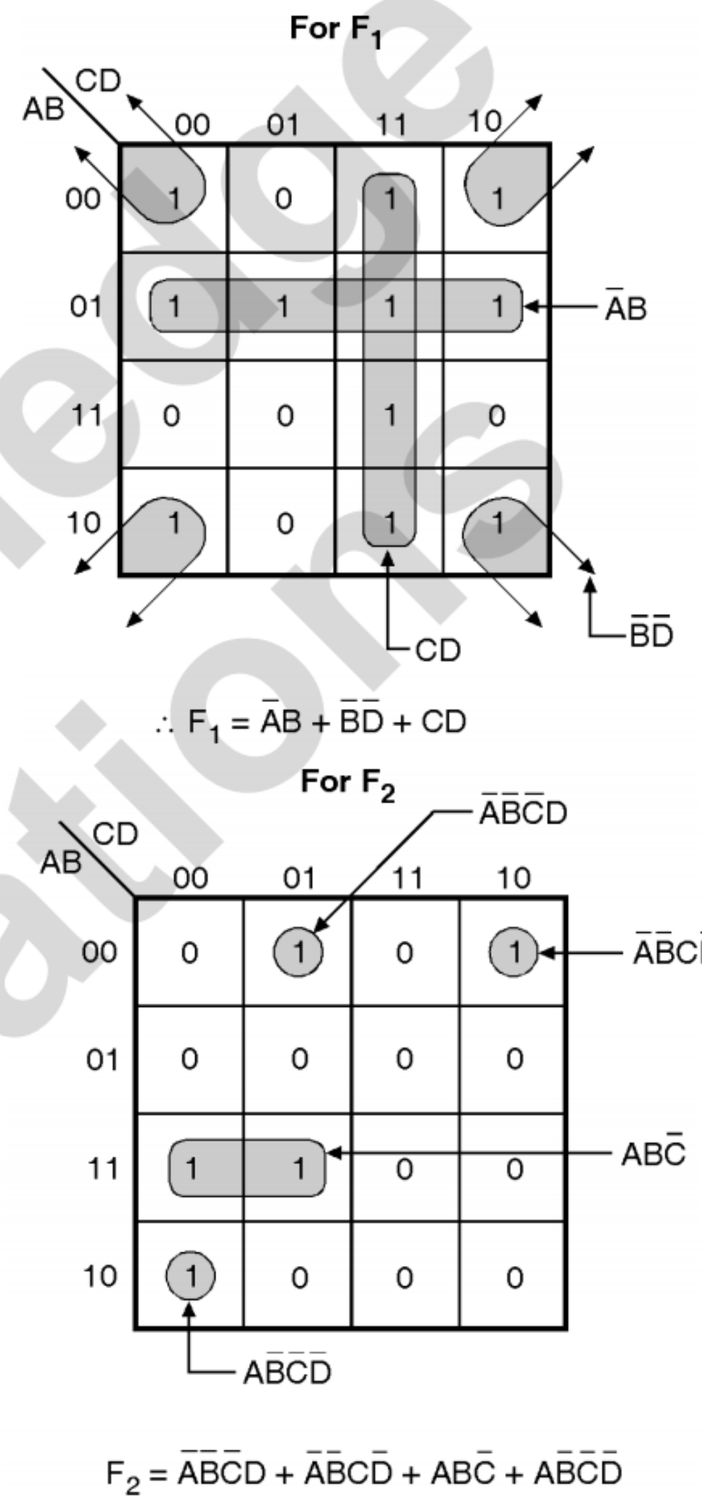
Ex. 8.4.10 : Implement the following Boolean function using PAL :

$F_1 = \sum m(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$

$F_2 = \sum(1, 2, 8, 12, 13)$ **Dec. 16, 6 Marks**

Soln. :

Step 1 : K-maps and simplifications :



(C-5700) **Fig. P. 8.4.10 : K-maps and simplifications**

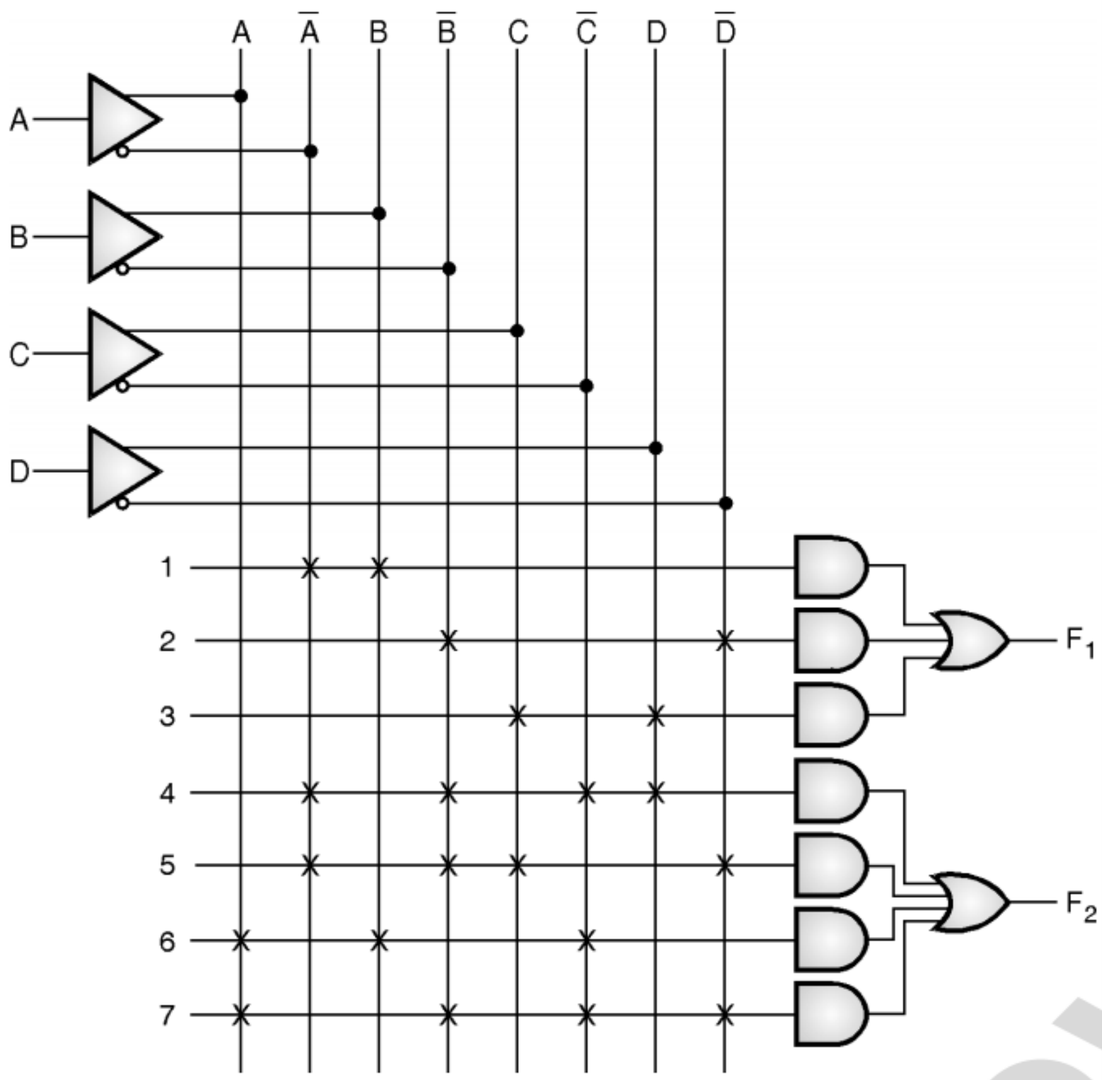
Step 2 : Prepare the PAL programming table :

(C-8159) **Table P. 8.4.10 : PAL programming table**

Product term no.	Product term	AND inputs				Outputs
		A	B	C	D	
1.	$\bar{A}B$	0	1	–	–	$F_1 = \bar{A}B + \bar{B}\bar{D} + CD$
2.	$\bar{B}\bar{D}$	–	0	–	0	
3.	CD	–	–	1	1	
4.	$\bar{A}\bar{B}\bar{C}D$	0	0	0	1	$F_2 = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + AB\bar{C} + A\bar{B}\bar{C}\bar{D}$
5.	$\bar{A}\bar{B}C\bar{D}$	0	0	1	0	
6.	$AB\bar{C}$	1	1	0	–	
7.	$A\bar{B}\bar{C}\bar{D}$	1	0	0	0	

TechKnowledge Publications

Step 3 : Logic diagram :



(C-5701) **Fig. P. 8.4.10(a) : Logic diagram using PAL**

Ex. 8.4.11 : Implement the following function using PAL :

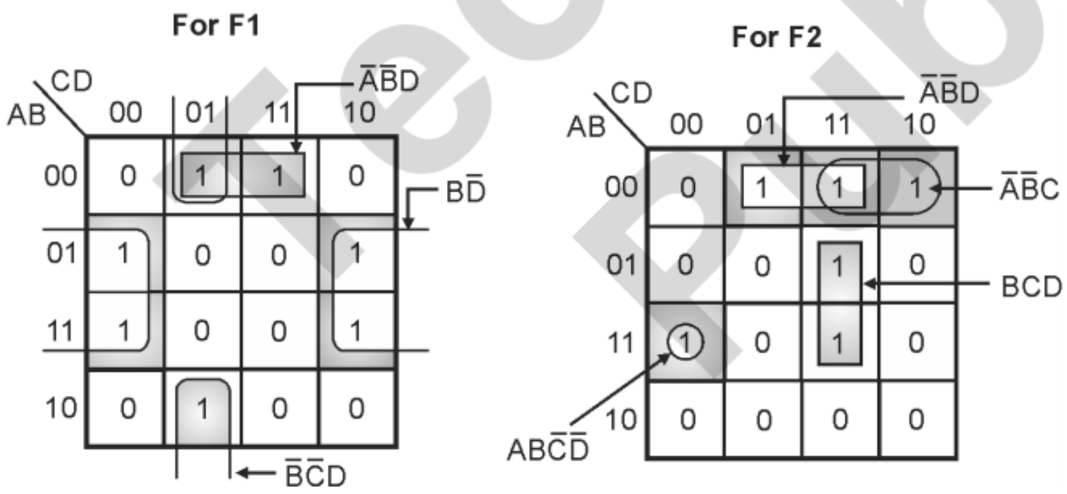
$F1(A, B, C, D) = \sum m\ (1, 3, 4, 6, 9, 12, 14)$

$F2(A, B, C, D) = \sum m\ (1, 2, 3, 7, 12, 15)$.

Dec. 18, 4 Marks

Soln. :

Step 1 : Simplify the equations using k-map :



(C-7458) **Fig. P. 8.4.11(a)**

$$\therefore \quad F_1 = \bar{A}\,\bar{B}D + B\,\bar{D} + \bar{B}\,\bar{C}\,D$$

$$\therefore \quad F_2 = AB\bar{C}\,\bar{D} + \bar{A}\,\bar{B}D + \bar{A}\,\bar{B}\,C + BCD$$

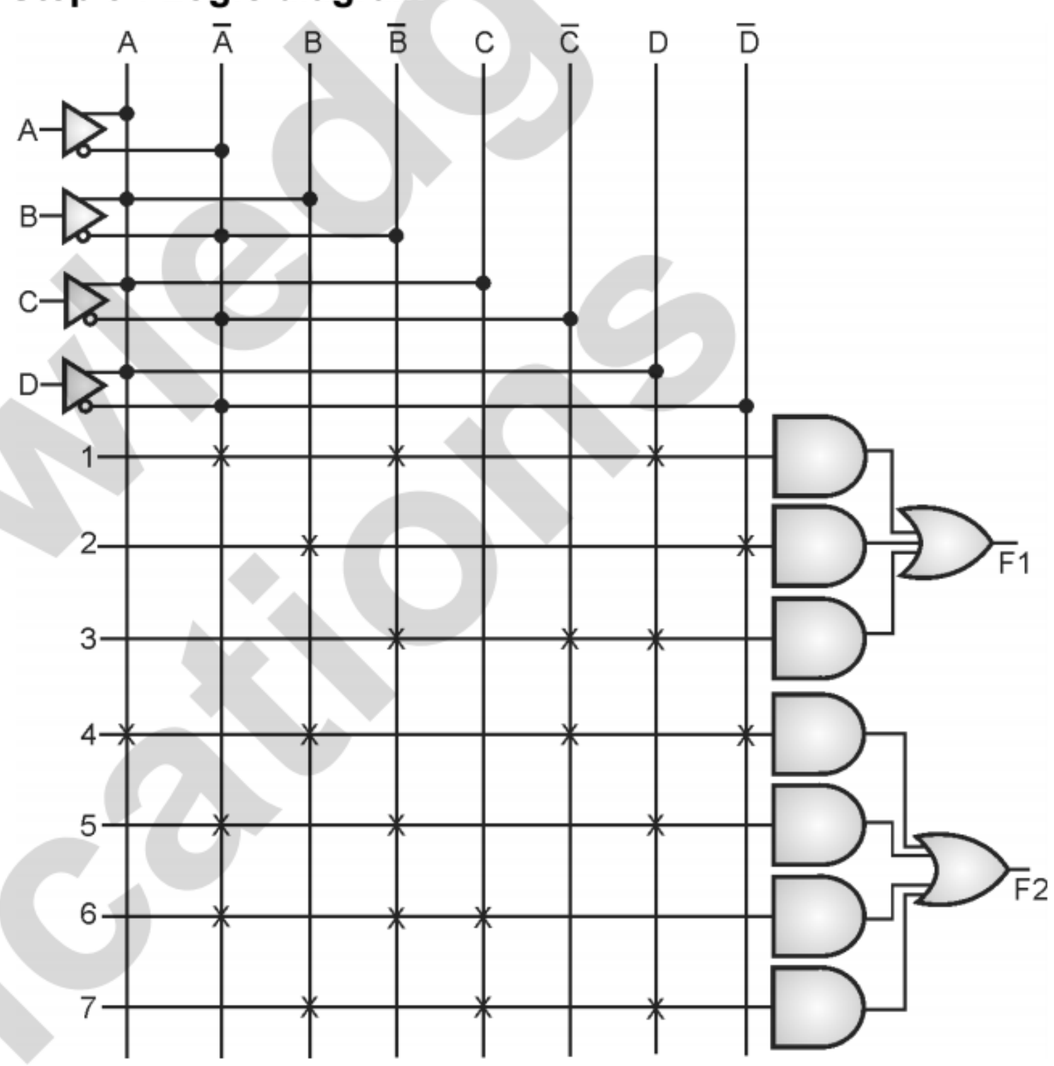
Step 2 : Write the PAL programming table :

– Table P. 8.4.11 shows the PAL programming table.

(C-7459) **Table P. 8.4.11**

Product term no.	Product term	AND inputs				Outputs
		A	B	C	D	
1	$\bar{A}\bar{B}D$	0	0	–	1	$F1 = \bar{A}\bar{B}D$ $B\bar{D} + \bar{B}\bar{C}D$
2	$B\bar{D}$	–	1	–	0	
3	$\bar{B}\bar{C}D$	–	0	0	1	
4	$AB\bar{C}\bar{D}$	1	1	0	0	$F2 = AB\bar{C}\bar{D} + \bar{A}\bar{B}D + \bar{A}\bar{B}C + BCD$
5	$\bar{A}\bar{B}D$	0	0	–	1	
6	$\bar{A}\bar{B}C$	0	0	1	–	
7	BCD	0	1	1	1	

Step 3 : Logic diagram :



(C-7460) **Fig. P. 8.4.11(b) : Logic diagram using PAL**

Ex. 8.4.12 : Implement following Boolean function using PAL.

$F1\ (A, B, C, D) = \Sigma\ m\ (7, 8, 9, 10, 11, 12, 13, 14, 15)$

$F2\ (A, B, C, D) = \Sigma\ m\ (2, 12, 13)$

Dec. 19, 6 Marks

Soln. :

Step 1 : K-maps and simplifications :

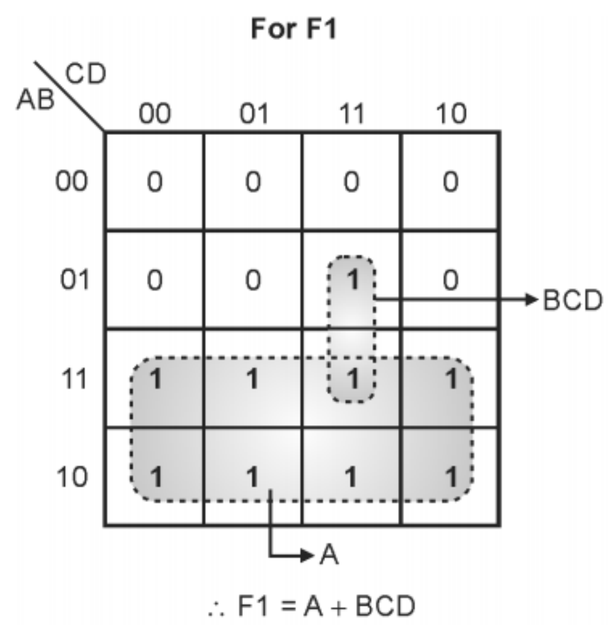
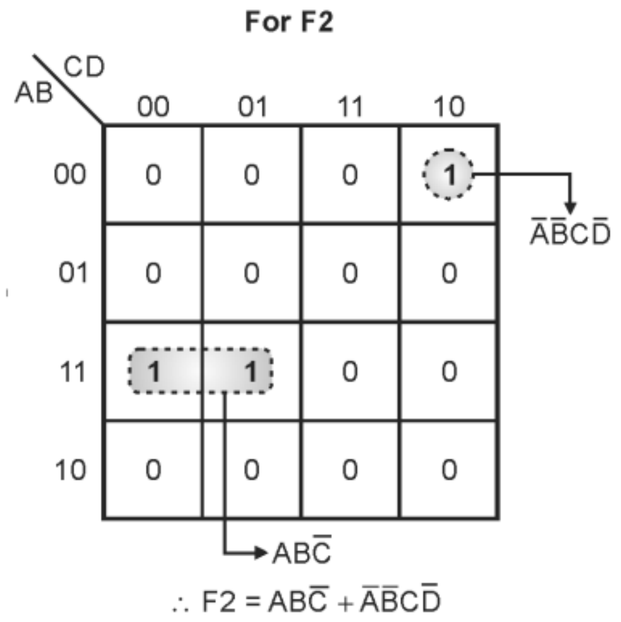


Fig. P. 8.4.12(Contd...)



(CT-2) Fig. P. 8.4.12 : K-maps and simplifications

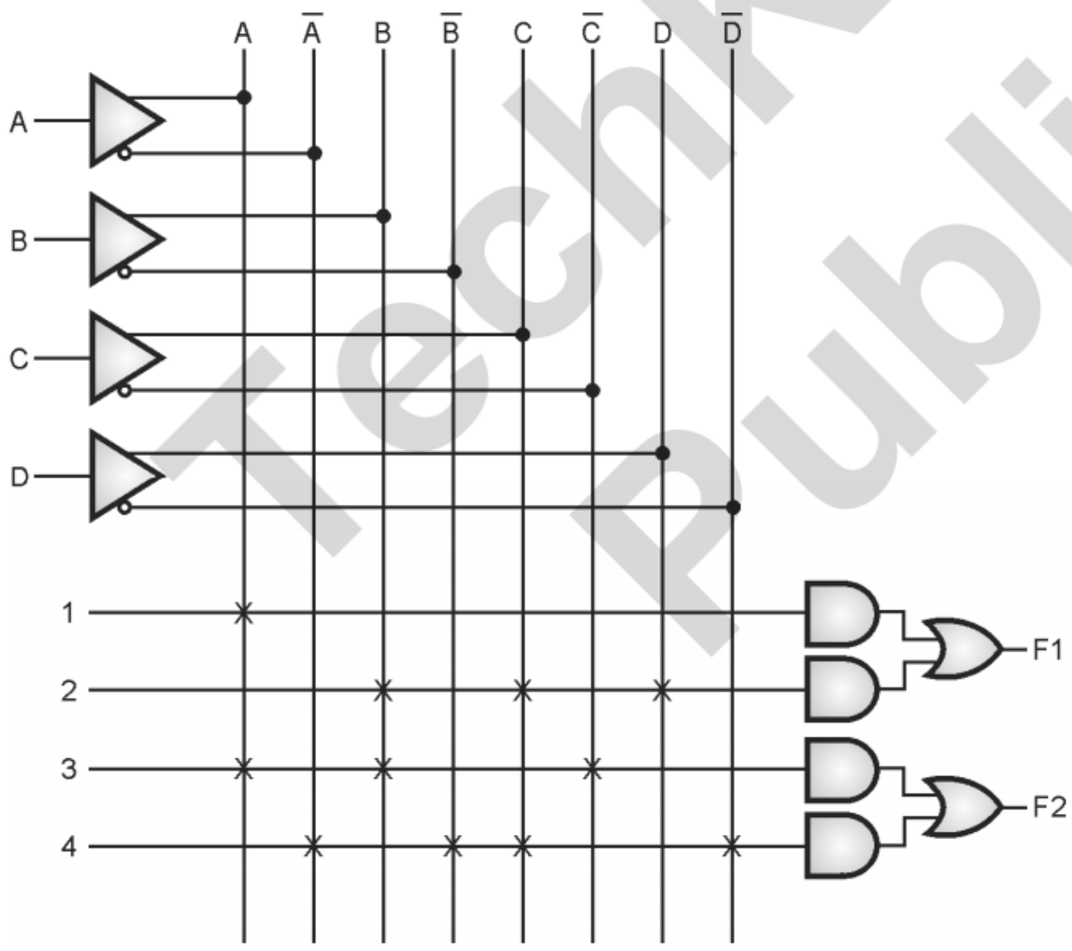
Step 2 : Prepare the PAL programming table :

(C-8160) Table P. 8.4.12 : PAL programming table

Product term no.	Product term	AND inputs				Outputs
		A	B	C	D	
1.	A	0	1	–	–	$F1 = A + BCD$
2.	BCD	–	1	1	1	
3.	$AB\bar{C}$	1	1	0	–	$F2 = AB\bar{C} + \bar{A}\,\bar{B}\,C\bar{D}$
4.	$\bar{A}\,\bar{B}C\bar{D}$	0	0	1	0	

Step 3 : Implementation :

– Refer Fig. P. 8.4.12(a) for the implementation.



(CT-4) Fig. P. 8.4.12(a) : Implementation using PAL

8.4.2 Types of PAL Devices :

– Some of the practically used PAL devices are as follows :

1. PAL 16 H8 2. PAL 16 P8

– PAL 16 H8 is identical to PAL 16L8 except one change. It does not have inverted output buffers.

– PAL 16 P8 has a programmable output polarity as it uses the EX-OR gates on the output side as buffers.

8.4.3 Registered PALs :

Definition :

– Registered PALs are the PALs in which flip flops have been added on the output side alongwith the output buffers.

– Registered PALs are preferred for implementation of sequential circuits. Examples of registered PALs are 16 R4, 16 R6 and 16 R8 etc. where R indicates that it is a registered PAL.

8.4.4 Configurable PAL : SPPU : May 07

University Questions

Q. 1 What do you mean by configurable devices ? **(May 07, 4 Marks)**

– Due to developments in the PAL, it has become possible to introduce configurable outputs. The output capabilities of such devices are much enhanced as compared to those of the conventional devices.

– These devices contain special circuitry called as output macrocells.

– Macro cell has a circuitry made of fuse links. We can configure it to obtain various output options. This will improve the device flexibility.

– A configurable PAL is so flexible and improved that it can replace a large number of simpler PAL devices.

– 22V10 is a typical configurable PAL which has 22 inputs, 10 outputs and 120 product terms.

– Out of 22 possible inputs 12 inputs are dedicated inputs and the remaining are input/output lines.

– Output stage consists of a positive edge triggered D FF. There are 10 outputs.

– One macrocell is used alongwith each output. Tri state drivers are also used in the macrocell.

TechKnowledge Publications

8.5 Comparison of PROM, PLA and PAL :

SPPU : May 08, May 15, Dec. 15

University Questions

Q. 1 What is the difference between PAL and PLA ? **(May 08, Dec. 15, 6 Marks)**

Q. 2 Comparison between PROM, PLA and PAL. **(May 15, 7 Marks)**

PROM	PLA	PAL
It is cheap and easy to use.	It is expensive than PAL and PROM and complicated to use.	Moderately expensive and moderately complicated.
There are fixed AND arrays and programmable OR arrays.	AND and OR arrays are programmable	Only the AND array is programmable. OR array is fixed.
SOP functions in the standard form only can be implemented.	Any SOP function can be implemented	Any SOP function can be implemented.
It is possible to decode any minterm.	We can get any desired minterm by programming the AND matrix.	We can get any desired minterm by programming the AND matrix.

Review Questions

Q. 1 Write a short note on concept of PLA and PAL.

Q. 2 Explain the concept of PLA with the help of a block diagram.

Q. 3 Draw the combinational circuit for a PLA with 3 inputs, 3 product terms and 2 outputs.

Q. 4 How is the size of a PLA specified ? What is the difference between a masked PLA and FPLA ?

Q. 5 Explain AND-OR structure of PLA and PAL.

Q. 6 What is a Programmable Logic Device (PLD) ? What is the advantage of using PLDs in the design of digital system ?

Q. 7 What are the basic configuration of the three PLDs ?

Q. 8 Describe with neat diagram, AND-OR structure of PLA and PAL.

Q. 9 What is a PLA ? Explain the architecture of a PLA.

Q. 10 What is the architectural difference between PROM, PLA and PAL ?

Q. 11 With neat diagram, show the internal structure of PLA.

Q. 12 Explain the difference between PLA and PAL devices with the help of internal logic diagrams.

Q. 13 Design a 3-bit Gray to Binary code converter. Implement it using suitable PROM. Draw the PROM table and logic diagram.

Q. 14 State two advantages of PLD over fixed function IC and application specific IC. What is the difference between PLA, PAL and PROM ?

Q. 15 Compare PROM, PAL and PLA.

Q. 16 What is CPLD ? Explain its basic architecture.

Q. 17 What is FPGA ? Explain its basic structure.

Q. 18 What is the difference between CPLD and FPGA ?

❑❑❑

TechKnowledge Publications

Unit V

Chapter 9

Logic Families

Syllabus

Classification of logic families : Unipolar and bipolar logic families, Characteristics of digital ICs : Fan-in, Fan-out, Current and voltage parameters, Noise immunity, Propagation delay, Power dissipation, Figure of Merits, Operating temperature range, Power supply requirements.

Transistor - Transistor logic : Operation of TTL NAND gate (Two input), TTL with active pull up, TTL with open collector output, Wired AND connection, Tristate TTL devices, TTL characteristics.

CMOS : CMOS inverter, CMOS characteristics, CMOS configurations - Wired logic, Open drain outputs.

Exemplar / Case study : To study the various basic gate design using TTL/CMOS logic family.

Technical Knowledge Publications

Chapter Contents

9.1	Introduction	9.10	Advantages and Disadvantages of TTL
9.2	Classification of Logic Families	9.11	MOS - Logic Family
9.3	Characteristics of Digital ICs	9.12	CMOS Logic
9.4	Transistor-Transistor Logic (TTL)	9.13	CMOS Characteristics
9.5	Sourcing and Sinking in TTL	9.14	Open Drain Outputs (CMOS)
9.6	Wired AND Connection (TTL)	9.15	Wired Logic (CMOS)
9.7	Open Collector Outputs (TTL)	9.16	Interfacing
9.8	Tristate (Three state) TTL Devices	9.17	Comparison of CMOS and TTL
9.9	Standard TTL Characteristics		

9.1 Introduction :

SPPU : May 14, Dec. 14, Dec. 16

University Questions

Q. 1 What is logic family ? Give the classification of logic family.

(May 14, 3 Marks, Dec. 14, 2 Marks, Dec. 16, 7 Marks)

Definition :

- Logic families are defined as the type of logic circuit used in an IC. Various digital ICs available in market belong to various types. Each type is known as a logic family.
- Various digital ICs available in market belong to various types. These types are known as "families".
- Based on the components and devices internally used, the digital IC families are named as RTL (Resistor Transistor Logic), TTL (Transistor Transistor Logic), DTL (Diode Transistor Logic), CMOS etc.

Features of logic families :

- Table 9.1.1 gives the comparison of some of the outstanding features of these logic families.

Table 9.1.1 : Comparison of important features of logic families

Sr. No.	Characteristics	TTL	CMOS	ECL
1.	Power input	Moderate	Low	Moderate to high
2.	Frequency limit	High	Moderate	Very high
3.	Circuit density	Moderate to high	High to very high	Moderate
4.	Circuit types per family	High	High	Moderate

- Thus TTL family has a great versatility. It has high speed as well (delay less than 1 nS for some TTL subfamilies).
- The CMOS family is popular because of its low input power and very high circuit density i.e. more circuits can be placed in the same volume of IC.
- ECL is used for very high speed digital circuits.
- But it needs more input power and less types of logic circuits are available in ECL than those available in TTL and CMOS.

9.1.1 Classification Based on Circuit Complexity :

- The logical circuits of different types are available in the integrated circuits. Depending on the level of complexity the integrated circuit are classified into four categories as follows :

1. SSI : Small Scale Integration.
2. MSI : Medium Scale Integration.
3. LSI : Large Scale Integration.
4. VLSI : Very Large Scale Integration.

- The number of components (diodes, transistors, gates etc.) in SSI will be the lowest and that in VLSI will be the highest.

1. Small Scale Integration (SSI) < 10 components.
2. Medium Scale Integration (MSI) < 100 components.
3. Large Scale Integration (LSI) > 100 components.
4. Very Large Scale Integration (VLSI) > 1000 components.

9.2 Classification of Logic Families :

9.2.1 Classification Based on Devices Used :

SPPU : May 14, Dec. 14, Dec. 16, Dec. 17, May 18

University Questions

Q. 1 What is logic family ? Give the classification of logic family.

(May 14, 3 Marks, Dec. 14, 2 Marks, Dec. 16, 7 Marks

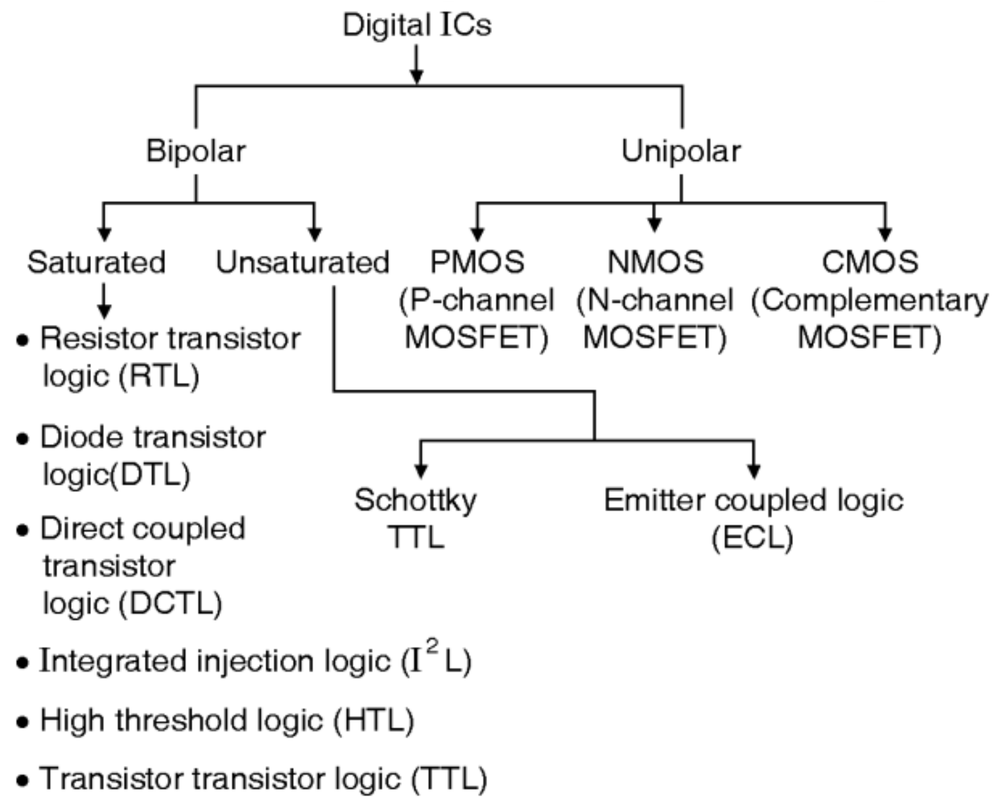
Q. 2 Give the classification of logic families.

(Dec. 17, 3 Marks, May 18, 4 Marks)

- The two basic techniques for manufacturing ICs are :

1. Bipolar technology.
2. Unipolar devices-Metal Oxide Semiconductor (MOS) technology.

- The classification of logic families is shown in Fig. 9.2.1.



(C-205) **Fig. 9.2.1 : Classification of logic families**

Bipolar families :

- The bipolar families of logic circuits use the bipolar transistors fabricated on the chip.
- That means all the gates belonging to the bipolar family use the transistorised circuits.
- In the bipolar category there are three basic families called, Diode Transistor Logic (DTL), Transistor Transistor Logic (TTL) and Emitter Coupled Logic (ECL).
- DTL uses diodes and transistors, TTL uses transistors as the main elements.
- TTL has become the most popular family in SSI (Small Scale Integration) and MSI (Medium Scale Integration) chips, while ECL is the fastest logic family which is used for high speed applications.
- In the "Bipolar saturated" logic families, the bipolar transistors are used as the main device.
- It is used as a switch and operated in the saturation or cutoff regions.
- TTL is an example of saturated bipolar logic.
- In the unsaturated bipolar logic, the bipolar transistors are not driven into hard saturation.
- This increases the speed of operation. So the unsaturated bipolar ICs such as Schottky TTL and ECL (Emitter Coupled Logic) are much faster as compared to TTL.
- All these ICs are fabricated on silicon chips using different fabrication technologies.

Unipolar families :

- The MOS family use fabricates MOS Field Effect Transistors (MOSFETs) fabricated on the chip. So all the gates belonging to the MOS family use the MOSFET based circuits.
- In the MOS category there are three logic families namely PMOS (p-channel MOSFETs) family, NMOS (n-channel MOSFETs) family and CMOS (Complementary MOSFETs) family.
- PMOS is the oldest and slowest type. NMOS is used for the LSI (large scale integration) field i.e. for the microprocessors and memories.
- CMOS which uses a push pull arrangement of n-channel and p-channel MOSFETs, is extensively used where low power consumption is needed such as in pocket calculators.

9.3 Characteristics of Digital ICs :

SPPU : Dec. 08, May 17

University Questions

Q. 1 Explain any four characteristics of digital ICs. **(Dec. 08, 2 Marks, May 17, 4 Marks)**

- Eventhough there are various logic families, the general characteristics, their definitions, and terminologies used for all of them have been standardized.
- So let us discuss some of the most important general characteristics that are applicable to all the digital ICs first and then discuss them for some particular families.
- The important characteristics of all the digital IC families are as follows :
  1. Voltage and current parameters
  2. Fan-in and fan-out
  3. Noise margin
  4. Propagation delay (speed)
  5. Power dissipation
  6. Operating temperature
  7. Invalid voltage levels
  8. Figure of merit (SPP).

9.3.1 Voltage and Current Parameters :

SPPU : Dec. 13, May 17, Dec. 18

University Questions

Q. 1 Define the following terms and mention its standard values for TTL family :
1. Voltage and current parameter.
2. Power dissipation.
3. Noise margin. **(Dec. 13, 6 Marks)**

Q. 2 Explain characteristics of digital ICs (any four). **(May 17, Dec. 18, 4 Marks)**

Voltage parameters (Threshold levels) :

- Ideally the input voltage levels of 0 V and + 5 V (for TTL) are called as logic 0 and 1 levels respectively.
- However practically we won't always observe or obtain the voltage levels matching exactly to these values.
- Therefore it is necessary to define the worst case input voltages.

1. $V_{IL\,(max)}$ - Worst case low level input voltage :

- This is the maximum value of input voltage which is to be considered as a logic 0 level.

– If the input voltage is higher than $V_{IL\,(max)}$, then it will not be treated as a low (0) input level.

2. $V_{IH\,(min)}$ - Worst case high level input voltage :

– This is the minimum value of the input voltage which is to be considered as a logic 1 level.

– If the input voltage is lower than $V_{IH\,(min)}$, then it will not be treated as a High (1) input.

3. $V_{OH\,(min)}$ - Worst case high level output voltage :

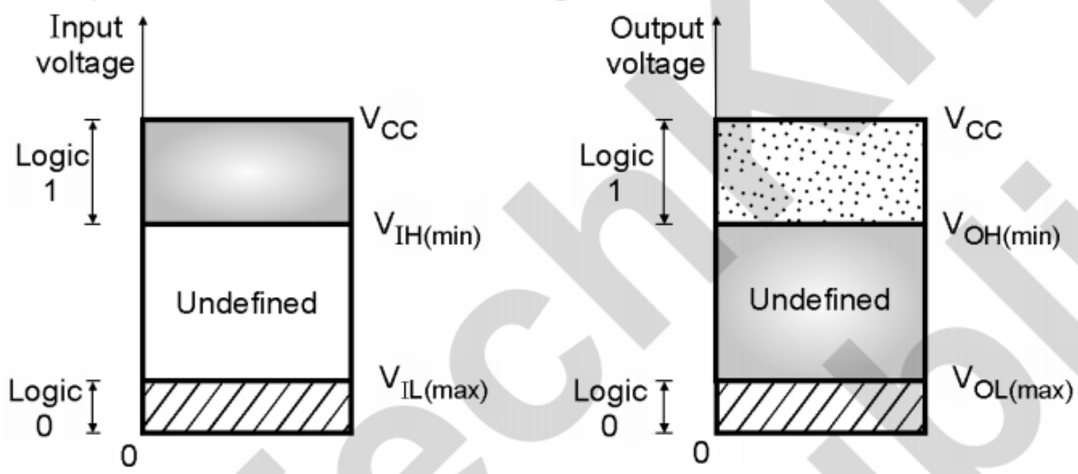
– This is the minimum value of the output voltage which will be considered as a logic HIGH (1) level.

– If the output voltage is lower than this level then it won't be treated as a HIGH (1) output.

4. $V_{OL\,(max)}$ - Worst case low level output voltage :

– This is the maximum value of the output voltage which will be considered as a logic LOW (0) level.

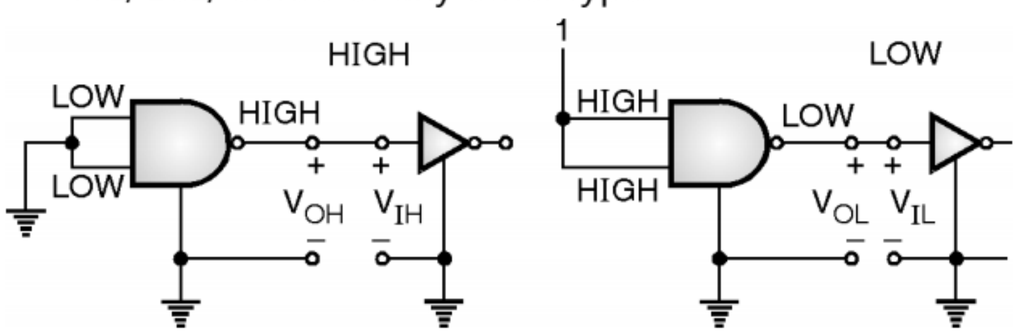
– If the output voltage is higher than this value then it won't be treated as a LOW (0) output. All the voltage parameters are shown in Fig. 9.3.1.



(C-7584) **Fig. 9.3.1 : Voltage parameters**

– The voltage parameters can be shown on the digital circuit consisting of gates as shown in Fig. 9.3.2.

– Note that, the NAND and NOT gates shown, can be of TTL, ECL, CMOS or any other type.



(C-207) **Fig. 9.3.2 : Voltage parameters on a logic circuit**

Current parameters :

(a) I_{IL} - Low level input current : It is the current that flows into the input terminals when a low level input voltage in the specified range is applied.

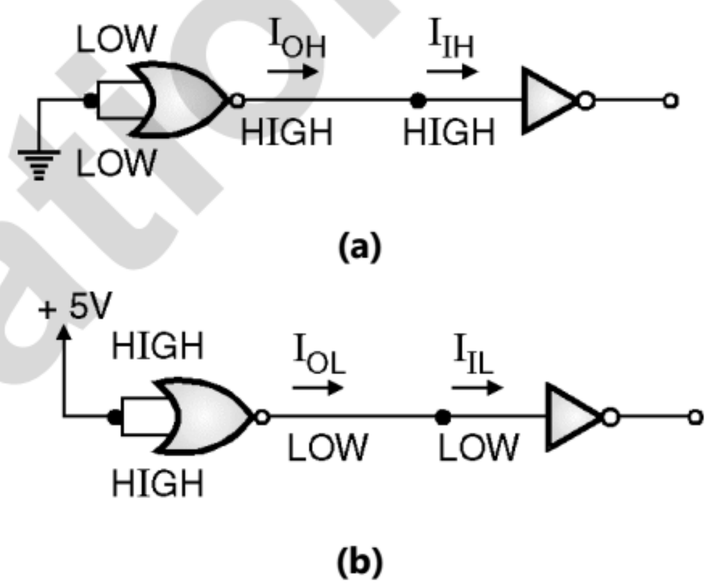
(b) I_{IH} - High level input current : It is the current that flows into the input terminals when a high level input voltage in the specified range is applied.

(c) I_{OL} - Low level output current : This is the current that flows out of the output when the output voltage happens to be in the specified low (0) voltage range and a specified load is applied.

(d) I_{OH} - High level output current : This is the current flowing from the output when the output voltage happens to be in the specified HIGH (1) voltage range and a specified load is applied.

– If the output current flows into the output terminal then it is called as a **sinking current** and if the output current flows out of the output terminal then it is called as a **sourcing current.**

– The current parameters are displayed on the logic circuit shown in Fig. 9.3.3.



(C-208) **Fig. 9.3.3 : Current parameters**

– Note that the actual current directions can be opposite to those shown in Fig. 9.3.3; depending on the logic family.

– Note that current flowing into a node or device is considered positive and current flowing out of node or device is considered negative.

9.3.2 Fan-in and Fan-out :

SPPU : Dec. 08, May 17, Dec. 18

University Questions

Q. 1 Explain fanout. **(Dec. 08, 2 Marks)**

Q. 2 Explain characteristics of digital ICs (any four). **(May 17, Dec. 18, 4 Marks)**

Fan-in :

– Fan-in is defined as the number of inputs a gate has. For example a two input gate will have a fan-in equal to 2.

TechKnowledge Publications

Fan-out :

- Fan-out is defined as the maximum number of inputs of the same IC family that a gate can drive without falling outside the specified output voltage limits.
- Higher fan out indicates higher the current supplying capacity of a gate.
- For example, a fan out of 5 indicates that the gate can drive (supply current to) at the most 5 inputs of the same IC family.

9.3.3 Noise Margin :

SPPU : Dec. 08, Dec. 13, Dec. 16, May 17, Dec. 18

University Questions

Q. 1 Explain noise margin. **(Dec. 08, 2 Marks)**

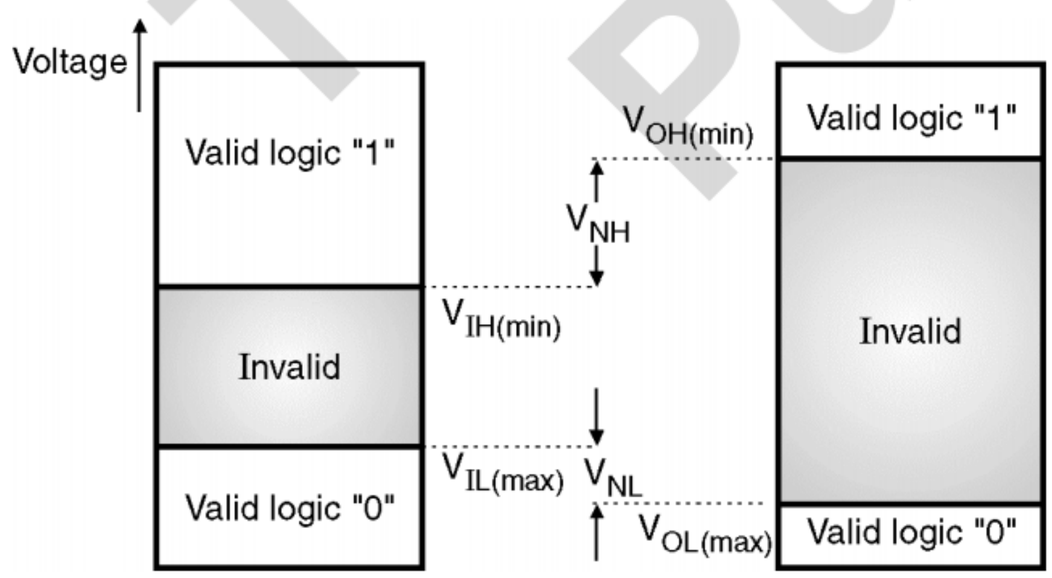
Q. 2 Define the following terms and mention its standard values for TTL family :

1. Voltage and current parameter
2. Power dissipation
3. Noise margin. **(Dec. 13, 6 Marks)**

Q. 3 State the following characteristics of digital TTL and CMOS ICs : Noise immunity. **(Dec. 16, 2 Marks)**

Q. 4 Explain characteristics of digital ICs (any four). **(May 17, Dec. 18, 4 Marks)**

- To understand the meaning of the term "Noise Margin" or "Noise Immunity", refer to the input and output voltage profiles shown in Fig. 9.3.4.



(a) Input profile **(b) Output profile**

(C-210) **Fig. 9.3.4**

- Noise is an unwanted electrical disturbance which may induce some voltage in the connecting wires used between two gates or from a gate output to load.

Definition :

- **Noise immunity** is defined as the ability of a logic circuit to tolerate the noise without causing the output to change undesirably.
- A quantitative measure of noise immunity of a logic family is known as **noise margin.**
- In order to avoid the effects of noise voltage, the designers adjust the voltage levels $V_{OH\ (min)}$ and $V_{IH\ (min)}$ to different levels with some difference between them as shown in Fig. 9.3.4.
- The difference between $V_{OH\ (min)}$ and $V_{IH\ (min)}$ is known as the **high level noise margin V_{NH}.**
- Similarly the difference between $V_{IL\ (max)}$ and $V_{OL\ (max)}$ is called as the **low level noise margin V_{NL}.**
- High level noise margin,

$$V_{NH} = V_{OH\ (min)} - V_{IH\ (min)}$$

- Low level noise margin,

$$V_{NL} = V_{IL\ (max)} - V_{OL\ (max)}$$

- When a high logic output is driving a logic circuit input, any negative noise spike greater than V_{NH} can force the voltage to reduce into the invalid range.
- Similarly when a low logic output is driving a logic circuit input, any positive noise spikes greater than V_{NL} can force the voltage to go into the invalid state.

9.3.4 Propagation Delay (Speed of Operation) :

SPPU : Dec. 08, Dec. 16, May 17, Dec. 18

University Questions

Q. 1 Explain propagation delay. **(Dec. 08, 2 Marks)**

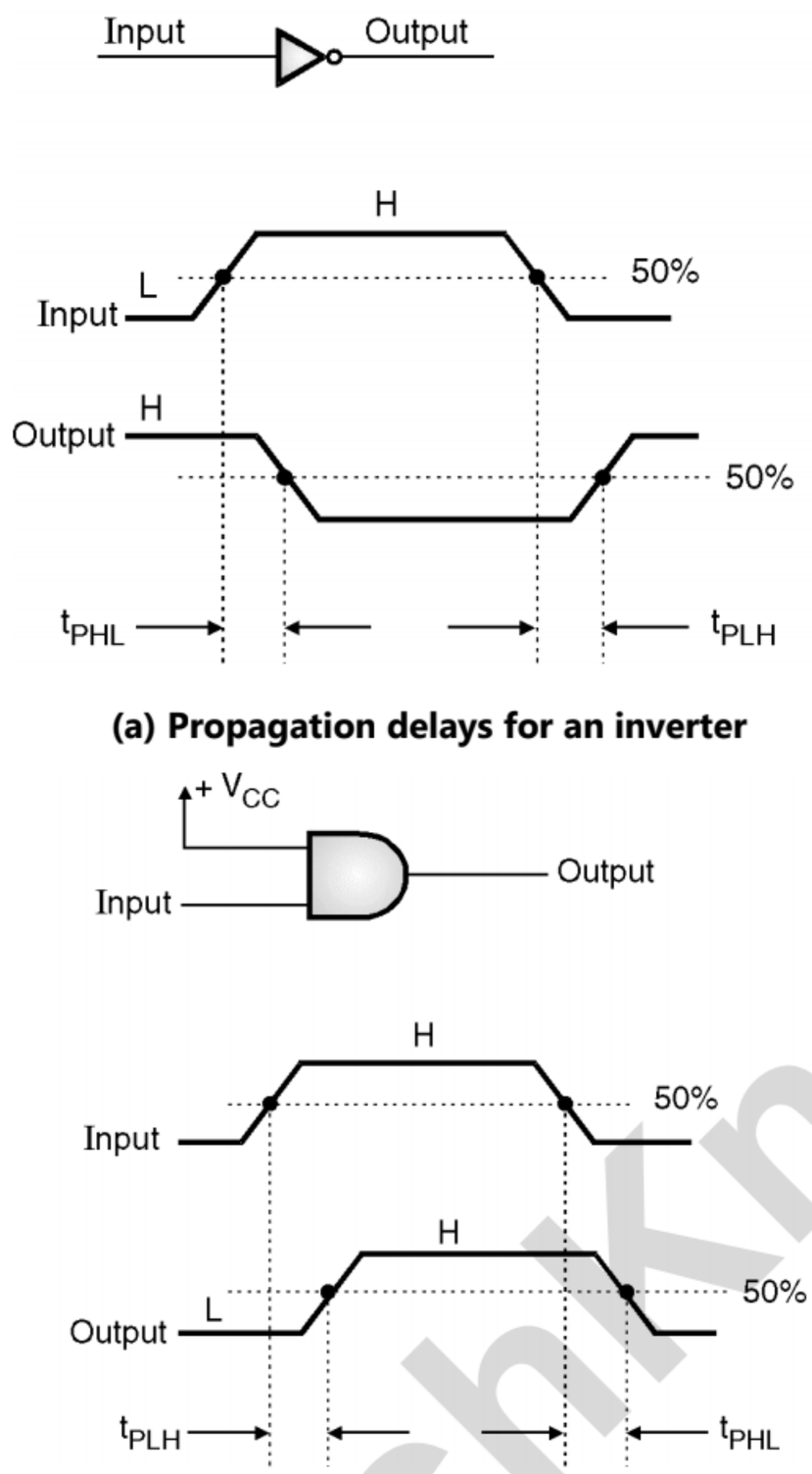
Q. 2 State the following characteristics of digital TTL and CMOS ICs : Speed of operation. **(Dec. 16, 2 Marks)**

Q. 3 Explain characteristics of digital ICs (any four). **(May 17, Dec. 18, 4 Marks)**

Definition :

- The output of a logic gate does not change its state instantaneously when the state of its input is changed.
- There is a time delay between these two time instants, which is called as the propagation delay.
- Thus **propagation delay** is defined as time delay between the instant of application of an input pulse and

the instant of occurrence of the corresponding output pulse. This is shown in Fig. 9.3.5.



(a) Propagation delays for an inverter

(b) Propagation delays for an AND gate

(C-211) Fig. 9.3.5

– From Fig. 9.3.5 it is observed that there are two propagation delays.

1. **t_{PHL} :** It is the propagation delay measured when the output is making a transition from HIGH (1) to LOW (0) state.
2. **t_{PLH} :** This is the propagation delay measured when the output makes a transition from LOW (0) to HIGH (1) state.

(a) The values of t_{PHL} and t_{PLH} are not always equal. If they are not equal then the one which is higher is considered as the propagation delay time of the gate.

(b) The propagation delays are measured between the points corresponding to 50% levels as shown in Fig. 9.3.5.

– Ideally propagation delay should be zero and practically it should be as short as possible.

– The values of propagation delays are used to measure how fast a logic circuits is.

– For example, a logic circuit with a propagation time of 5 nS will be a faster logic circuit than the one with 10 nS propagation time, under the specified load conditions.

9.3.5 Power Dissipation :

SPPU : Dec. 08, Dec. 13, Dec. 18

University Questions

Q. 1 Explain the power dissipation. **(Dec. 08, 8 Marks)**

Q. 2 Define the following terms and mention its standard values for TTL family :

1. Voltage and current parameter
2. Power dissipation
3. Noise margin. **(Dec. 13, 6 Marks)**

Q. 3 Explain the characteristics of digital IC. **(Dec. 18, 4 Marks)**

Definition :

– Due to applied voltage and currents flowing through the logic ICs, some power will be dissipated in it, in the form of heat.

– This power is in milliwatts. Care should be taken to reduce the power dissipation taking place in the logic IC in order to protect the IC against damage due to excessive temperature, to reduce the loading on power supplies etc.

– Another importance of power dissipation is that the product of power dissipation and propagation time is always constant.

– Therefore if we reduce the power dissipation may lead then the propagation delay will increase in order to keep their product constant.

– Usually there is only one power supply terminal on any IC. It is denoted by V_{CC} for the TTL ICs and V_{DD} for the CMOS ICs.

– The power drawn by an IC from the power supply is given by,

$$P = V_{CC} \times I_{CC}$$

where I_{CC} is the current drawn from the power supply.

9.3.6 Operating Temperature :

– The temperature range acceptable for the consumer and industrial applications is 0° to 70° C and that for the military applications is – 55° C to 125° C.

Tech Knowledge Publications

– The performance of gates will be in the specified limits only if the temperature is in these ranges.

9.3.7 Figure of Merit (Speed Power Product (SPP)) : SPPU : Dec. 16, Dec. 18

University Questions

Q. 1 State the following characteristics of digital TTL and CMOS ICs : Figure of merit. **(Dec. 16, 2 Marks)**

Q. 2 Explain the characteristics of digital IC. **(Dec. 18, 4 Marks)**

Definition :

– The figure of merit of a logical family is the product of power dissipation and propagation delay.

– It is called as the speed power product. The speed is specified in seconds and power is specified in W.

∴ Figure of merit = Propagation delay time × Power dissipation.

– Ideally the value of figure of merit is zero and practically, it should be as low as possible.

– Figure of merit is always a compromise between speed and power dissipation.

– That means if we try to reduce the propagation delay then the power dissipation will increase and vice-versa.

9.3.8 Invalid Voltage Levels :

– The operation of a logic circuit will be proper if and only if its input voltage levels are kept outside the invalid voltage range.

– That means the input voltage should be either lower than $V_{IL\ (max)}$ or higher than $V_{IH\ (min.)}$.

– The invalid input voltage will produce an unpredictable output response. Therefore it should be avoided.

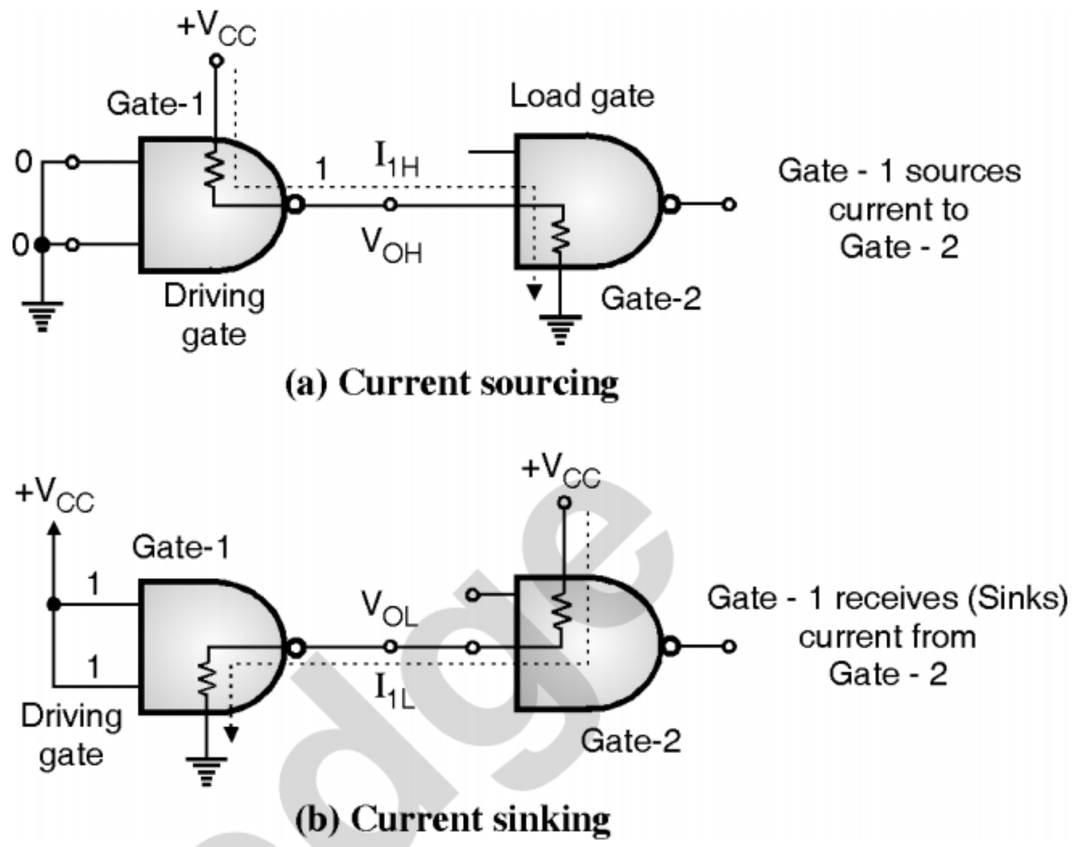
– When the output is overloaded, there is a possibility of output voltage going into the invalid range.

9.3.9 Current Sourcing and Current Sinking :

Current sourcing :

– The current sourcing action is illustrated in Fig. 9.3.6(a). Gate-1 acts as a source and Gate-2 acts as load.

– The output of Gate-1 is high. It supplies a current I_{IH} to the input of Gate-2. This is called as the current sourcing action.



(C-1000) **Fig. 9.3.6**

Current sinking :

– The current sinking action has been demonstrated in Fig. 9.3.6(b).

– Gate-2 which is the load gate acts as a load.

– As soon as the output of Gate-1 goes low, the current starts flowing into the output terminal of Gate-1 as shown in Fig. 9.3.6(b).

– Thus when gate-1 is accepting the current through its output terminal, it is said that the current sinking is taking place.

9.3.10 Power Supply Requirements :

– The supply voltage(s) and the amount of power required by an IC are important characteristics required to choose the proper power supply.

9.4 Transistor-Transistor Logic (TTL) :

– The long form of TTL is transistor logic. The digital ICs in the TTL family use only transistors as their basic building block.

– TTL ICs were first developed in 1965 and they were known as "standard TTL".

– This version of TTL circuits is not practically used now due to availability of advanced versions.

– The standard TTL has been improved to a great extent over the years.

– TTL devices are still used as "glue" logic which connects more complex devices in digital systems.

TechKnowledge Publications

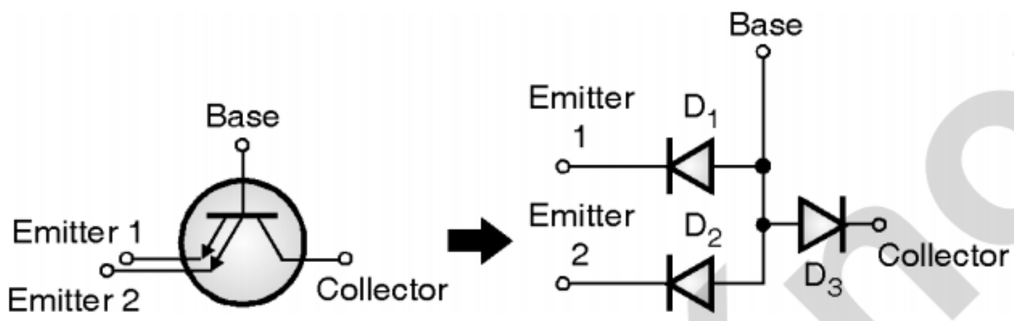
– The bipolar TTL family as a whole is now becoming obsolete, but we can study it in order to understand basic important concepts in general about the logic families.

9.4.1 The Multiple Emitter Transistor :

– In the TTL circuits that we are going to discuss, use a very special type of transistor.

– The normal transistor has only three terminals namely collector, base and emitter.

– But this special transistor has more than one emitters, as shown in Fig. 9.4.1(a) and its equivalent circuit is shown in Fig. 9.4.1(b). The number of emitters is equal to the number of inputs of the gate.



(a) Multiple emitter transistor **(b) Equivalent circuit**

(C-1011) **Fig. 9.4.1**

– The multiple emitter input transistor can have upto eight emitters, for an eight input NAND gate.

– In the equivalent circuit of Fig. 9.4.1(b), diodes D_1 and D_2 represent the two base to emitter junctions whereas D_3 represents the collector to base junction.

– In the forthcoming discussions, we are going to replace the multiple emitter transistor by its equivalent circuit shown in Fig. 9.4.1(b).

– This transistor can be turned ON by forward biasing either (or both) the diodes D_1 and D_2.

– This transistor will be in the OFF state if and only if both the base-emitter junctions (D_1 and D_2) are reverse biased.

Standard TTL :

– The 7400 TTL series is known as the standard TTL series. The TTL gates that we are going to discuss belong to this family.

9.4.2 Two Input TTL-NAND Gate (Totempole Output) :

SPPU : May 12, Dec. 12, May 17, Dec. 17, May 19

University Questions

Q. 1 Explain the working of two input TTL NAND gate with active pull up. Consider various input, output states for explanation. **(May 12, 8 Marks)**

Q. 2 Draw and explain 2-input NAND TTL logic gate with totem output driver. **(Dec. 12, 8 Marks)**

Q. 3 Explain operation of TTL NAND gate. **(May 17, 6 Marks)**

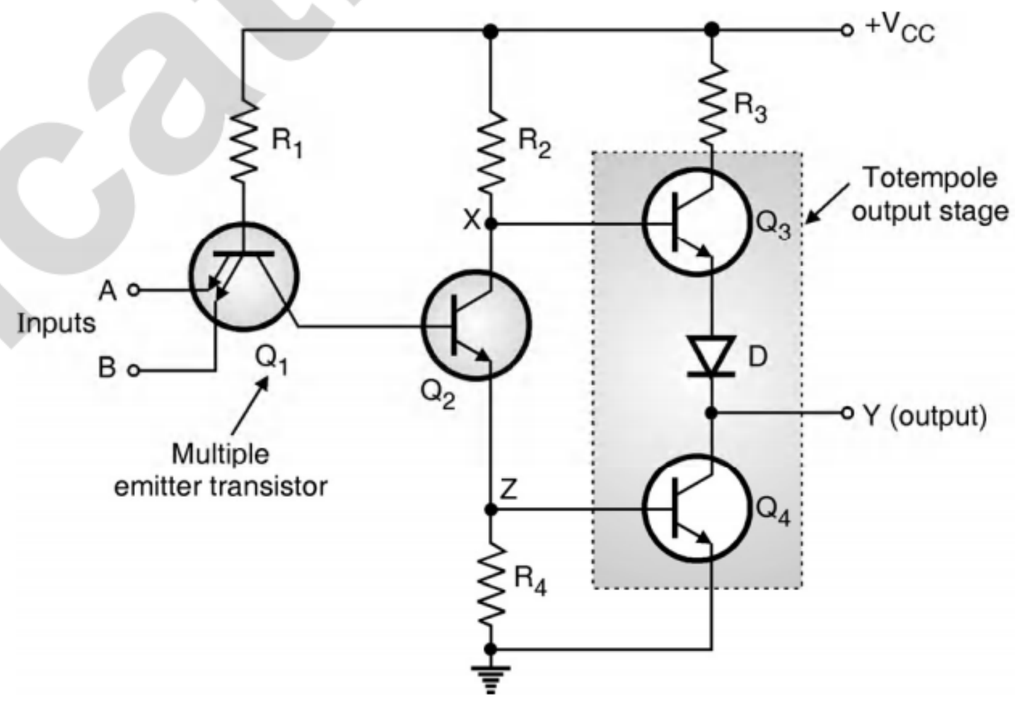
Q. 4 Draw and explain 2 input NAND TTL logic with totempole output driver. **(Dec. 17, 7 Marks)**

Q. 5 Draw 2-i/p standard TTL NAND gate with totem pole. Explain operation of transistor (ON/OFF) with suitable input conditions and truth table. **(May 19, 7 Marks)**

Circuit diagram :

– A two input TTL-NAND gate is shown in Fig. 9.4.2.

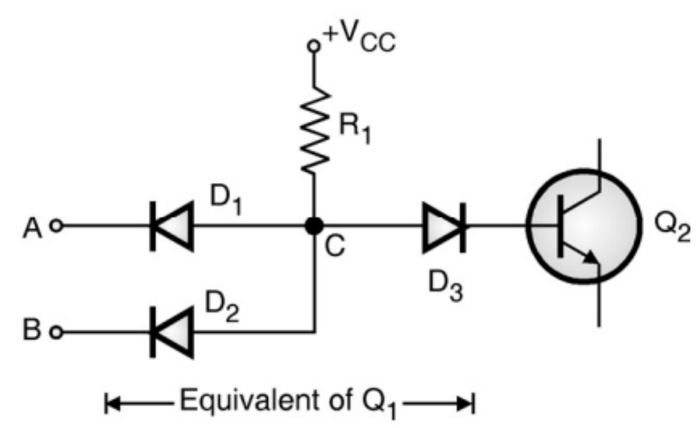
– A and B are the two inputs while Y is the output terminal of this NAND gate.



(C-1012) **Fig. 9.4.2 : Two input TTL NAND gate**

Operation :

– In order to understand the operation of this circuit, let us replace transistor Q_1 by its equivalent circuit as shown in Fig. 9.4.3.



(C-1012) **Fig. 9.4.3 : Transistor Q_1 is replaced by its equivalent**

1. A and B are the input terminals. The input voltages A and B can be either LOW (zero Volt ideally) or HIGH (+ V_{CC} ideally).

2. A and B both LOW (A = B = 0) :

– If A and B both are connected to ground, then both the B-E junctions of transistor Q_1 are forward biased.

– Hence diodes D_1 and D_2 in Fig. 9.4.3 will conduct to force the voltage at point C in Fig. 9.4.3 to 0.7 V.

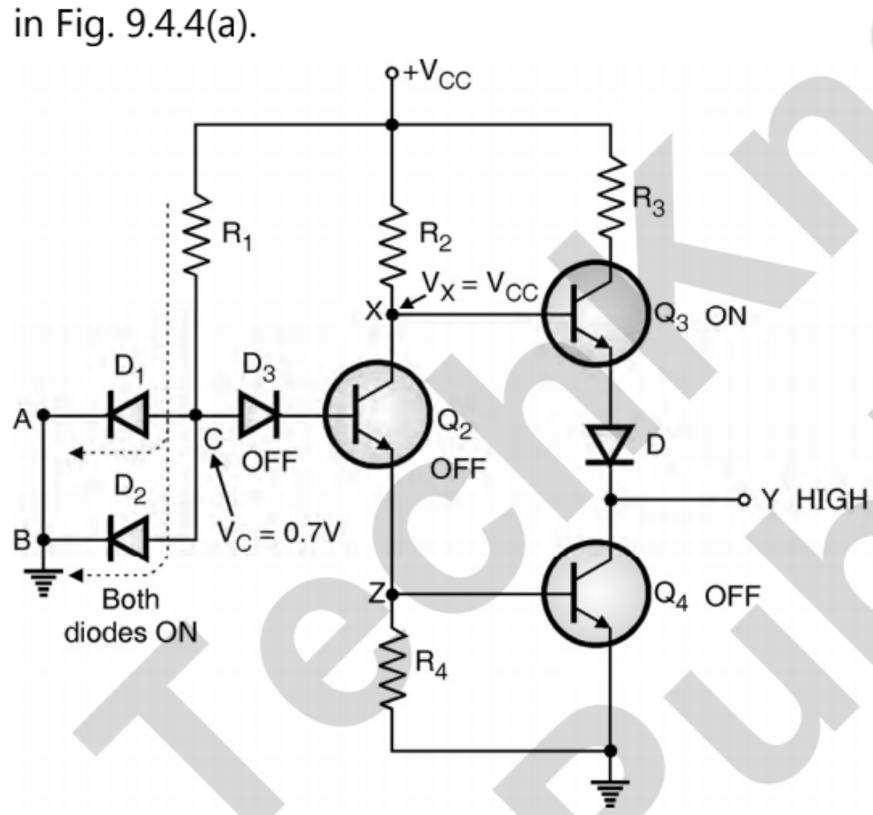
– This voltage is insufficient to forward bias base-emitter junction of Q_2 due to the presence of D_3. Hence Q_2 will remain OFF.

– Therefore its collector voltage V_X rises to V_{CC}.

– As transistor Q_3 is operating in the emitter follower mode, output Y will be pulled up to high voltage.

∴ Y = 1 (HIGH) ...For A = B = 0 (LOW)

– The equivalent circuit for this input condition is shown in Fig. 9.4.4(a).



(C-1013) Fig. 9.4.4(a) : Equivalent circuit for A = B = 0

3. Either A or B LOW (A = 0 or B = 0) :

– If any one input (A or B) is connected to ground with the other terminal left open or connected to + V_{CC}, then the corresponding diode (D_1 or D_2) will conduct.

– This will pull down the voltage at "C" to 0.7 V. (Fig. 9.4.3)

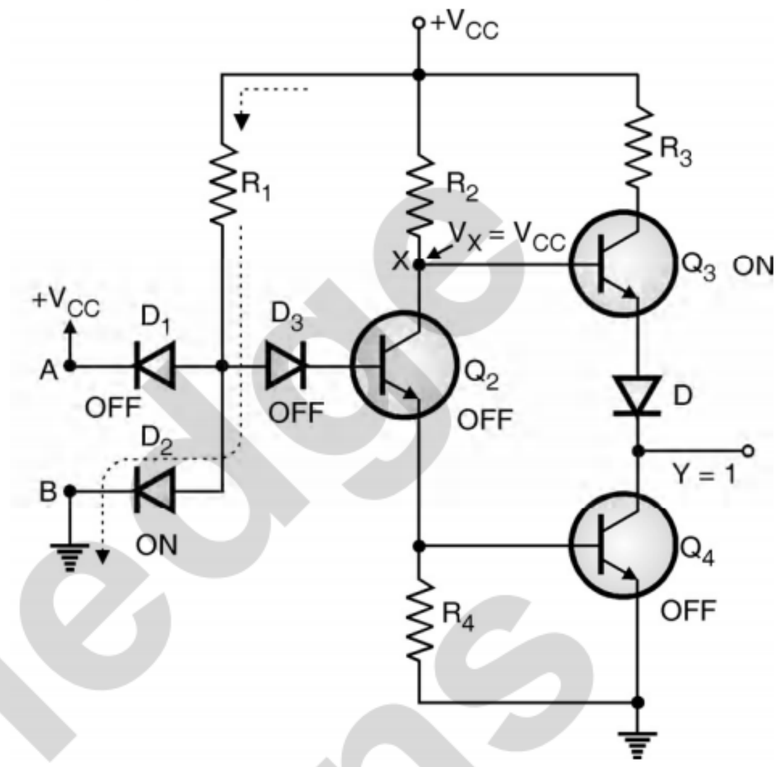
– This voltage is insufficient to turn ON D_3 and Q_2. So it remains OFF.

– So collector voltage V_X of Q_2 will be equal to V_{CC}. This voltage acts as base voltage for Q_3.

– As Q_3 acts as an emitter follower, output Y will be pulled to V_{CC}.

$$\therefore\ Y = 1 \begin{cases} \text{if } A = 0 \text{ and } B = 1 \\ \text{if } A = 1 \text{ and } B = 0 \end{cases}$$ **(C-6373)**

– The equivalent circuit for this mode is shown in Fig. 9.4.4(b).



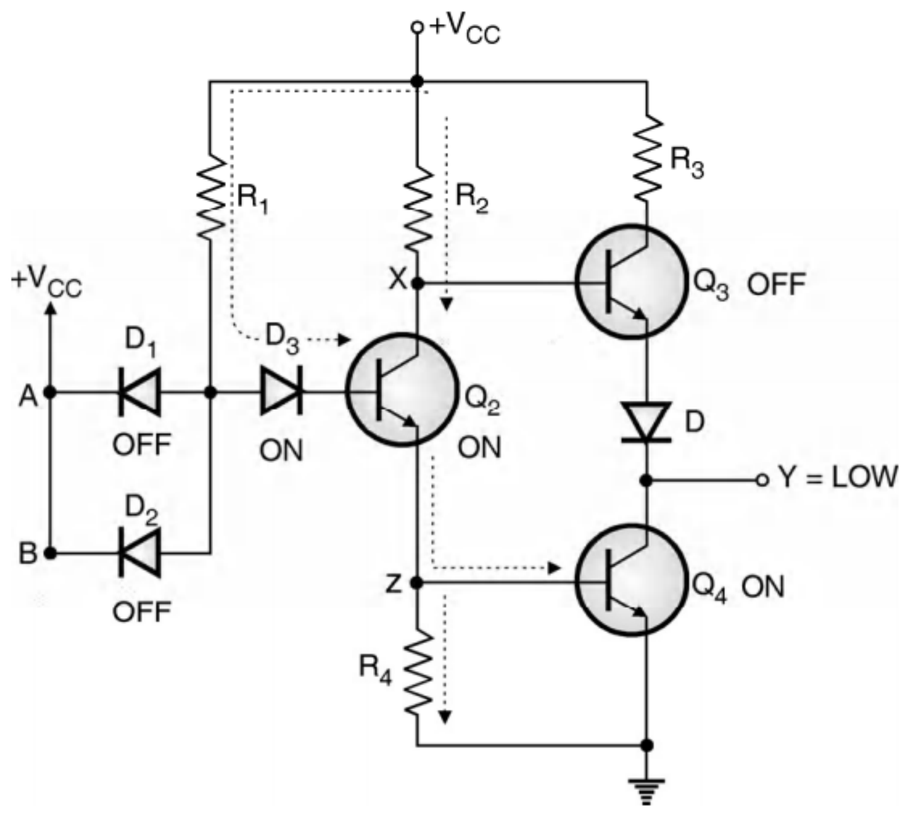
(C-1013) Fig. 9.4.4(b) : Equivalent circuit for A = 1, B = 0

4. A and B both HIGH (A = B = 1) :

– If A and B both are connected to + V_{CC}, then both the diodes D_1 and D_2 will be reverse biased and do not conduct.

– Therefore voltage at point "C" i.e. at the anode of D_3 increases to a sufficiently high value.

– Therefore diode D_3 is forward biased and base current is supplied to transistor Q_2 via R_1 and D_3 as shown in Fig. 9.4.4(c).



(C-1014) Fig. 9.4.4(c) : Equivalent circuit for A = B = 1

– As Q_2 conducts, the voltage at X will drop down and Q_3 will be OFF, whereas voltage at Z (across R_3) will increase to a sufficient level to turn ON Q_4.

TechKnowledge Publications

– As Q_4 goes into saturation, the output voltage Y will be pulled down to a low voltage.

∴ Y = 0 ...For A = B = 1

– The equivalent circuit for this mode of operation is shown in Fig. 9.4.4(c).

– This discussion reveals that the circuit operates as a NAND gate.

Truth Table :

– The Truth Table of Two input standard NAND gate is as follows.

(C-7538) Table 9.4.1 : Truth Table of a 2-input NAND gate

Inputs		Status of various transistors					Output Y
A	B	D_1	D_2	Q_2	Q_3	Q_4	
0	0	ON	ON	OFF	ON	OFF	1
0	1	ON	OFF	OFF	ON	OFF	1
1	0	OFF	ON	OFF	ON	OFF	1
1	1	OFF	OFF	ON	OFF	ON	0

9.4.3 Totem-pole (Active Pull up) Output Stage :

SPPU : May 05, Dec. 07

University Questions

Q. 1 Give advantages and disadvantages of totem-pole output-stage arrangement. **(May 05, 8 Marks)**

Q. 2 Will totempole output is suitable for wired-OR logic ? Justify your answer. **(Dec. 07, 5 Marks)**

– The arrangement of Q_3 and Q_4 on the output side of a TTL NAND gate is called as the totem-pole arrangement.

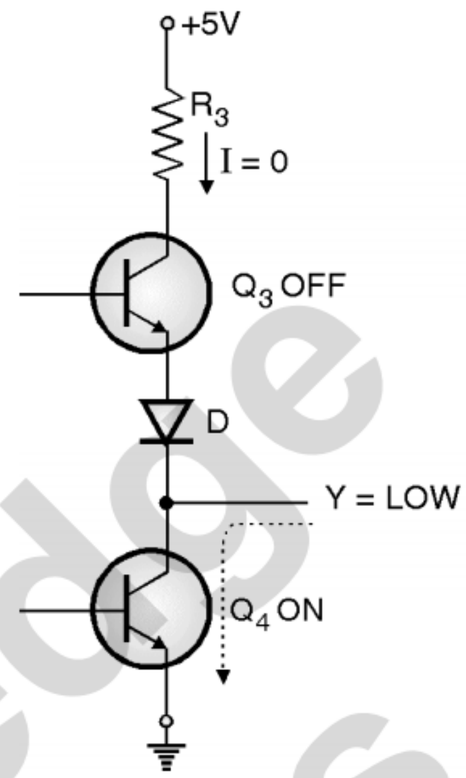
– It is possible in TTL gates to speed up the charging of output capacitance without corresponding increase in the power dissipation with help of **Totem-pole** output stage.

– The Totem-pole output is also known as **active pull-up.**

Advantages of totem-pole output stage :

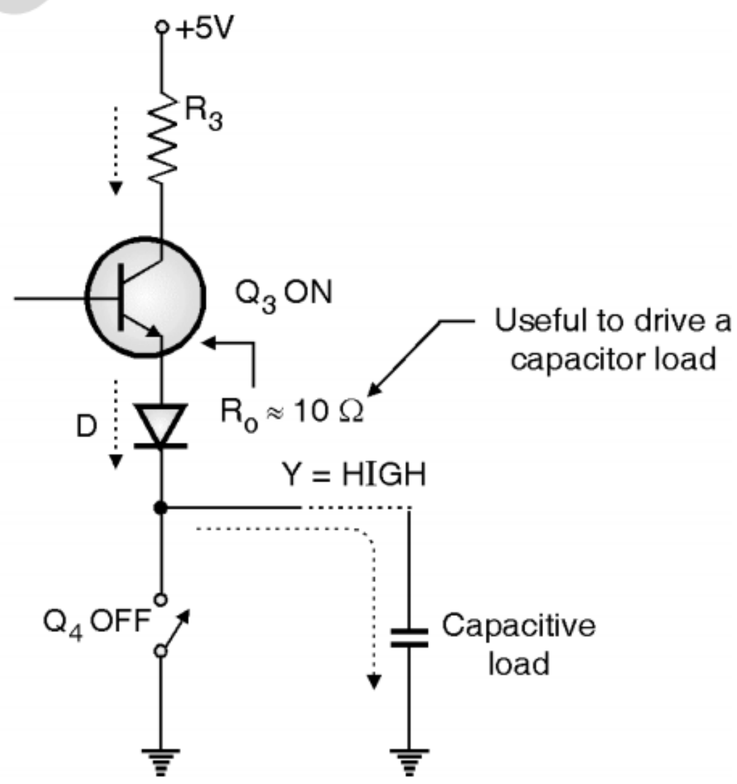
– The **advantages** of using the totem-pole output stage are as follows :

1. With Q_3 in the circuit, the current flowing through R_3 will be equal to zero when the output Y = 0, that means when Q_4 is ON, as shown in Fig. 9.4.5(a). This is important because it reduces the power dissipation taking place in the circuit.



(C-1015) Fig. 9.4.5(a) : No power dissipation in R_3

2. Another advantage of totem-pole arrangement is when the output Y is HIGH. Here Q_3 is ON and acting in the emitter follower mode. It will therefore have a very low output impedance (typically 10 Ω). Therefore the output time constant will be very short for charging up any capacitive load on the output as shown in Fig. 9.4.5(b).



(C-1015) Fig. 9.4.5(b) : Low output resistance

Disadvantages of Totem-pole output :

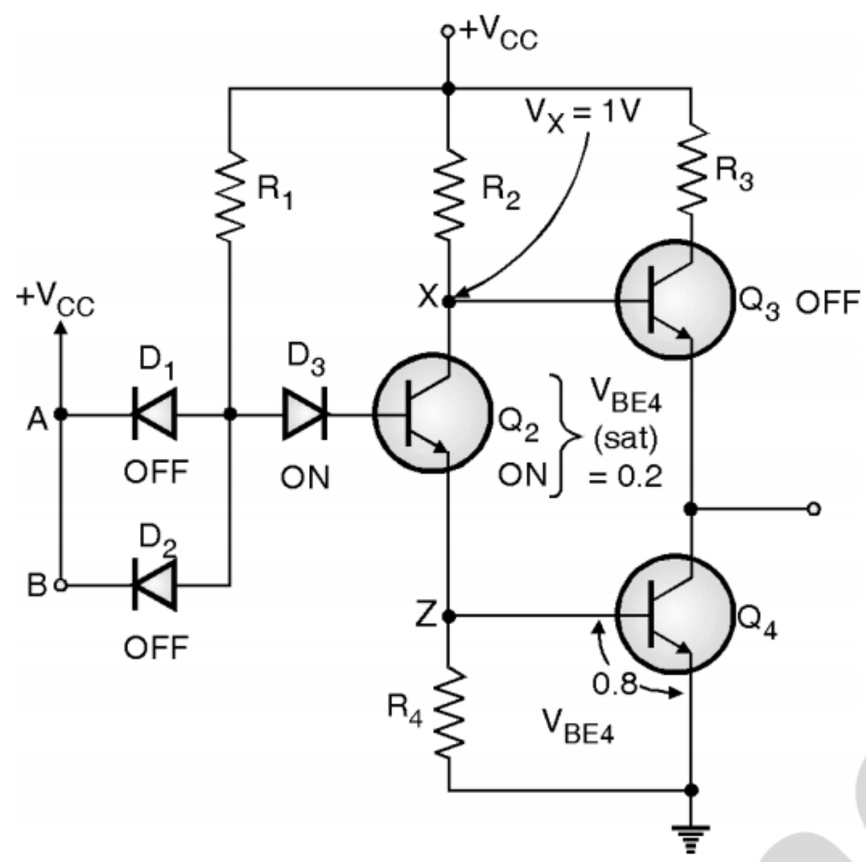
1. Q_4 in the totem-pole output turns OFF more slowly than Q_3 turns ON.
2. So before Q_4 is completely turned OFF, Q_3 will come into conduction. So for a very short duration of few nanoseconds, both the transistors will be simultaneously ON.

Tech Knowledge Publications

3. This is called as **cross conduction** and it will draw relatively large current (30 to 40 mA) from the 5 V supply.

Function of diode D :

– To understand the function of "D" consider the circuit without D, as shown in Fig. 9.4.6.



(C-1016) **Fig. 9.4.6 : Function of D**

– Assume that, A = B = 1. So D_1 and D_2 are OFF, D_3 ON and Q_2 is in saturation.

– Hence Q_4 is ON (saturation). Therefore voltage at the collector of Q_2 is given by,

$$V_X = V_{BE4\,(sat)} + V_{CE2\,(sat)}$$
$$= 0.8 + 0.2 = 1 \text{ Volt}$$

– V_X = 1 V will be sufficient to turn ON Q_3 which actually should not happen. So in order to keep Q_3 in the OFF state diode D is added in the circuit.

– It is important to avoid simultaneous conduction of Q_3 and Q_4, because it will lead to cross conduction and will increase power dissipation.

– Thus D_3 is used for successfully avoiding the cross conduction.

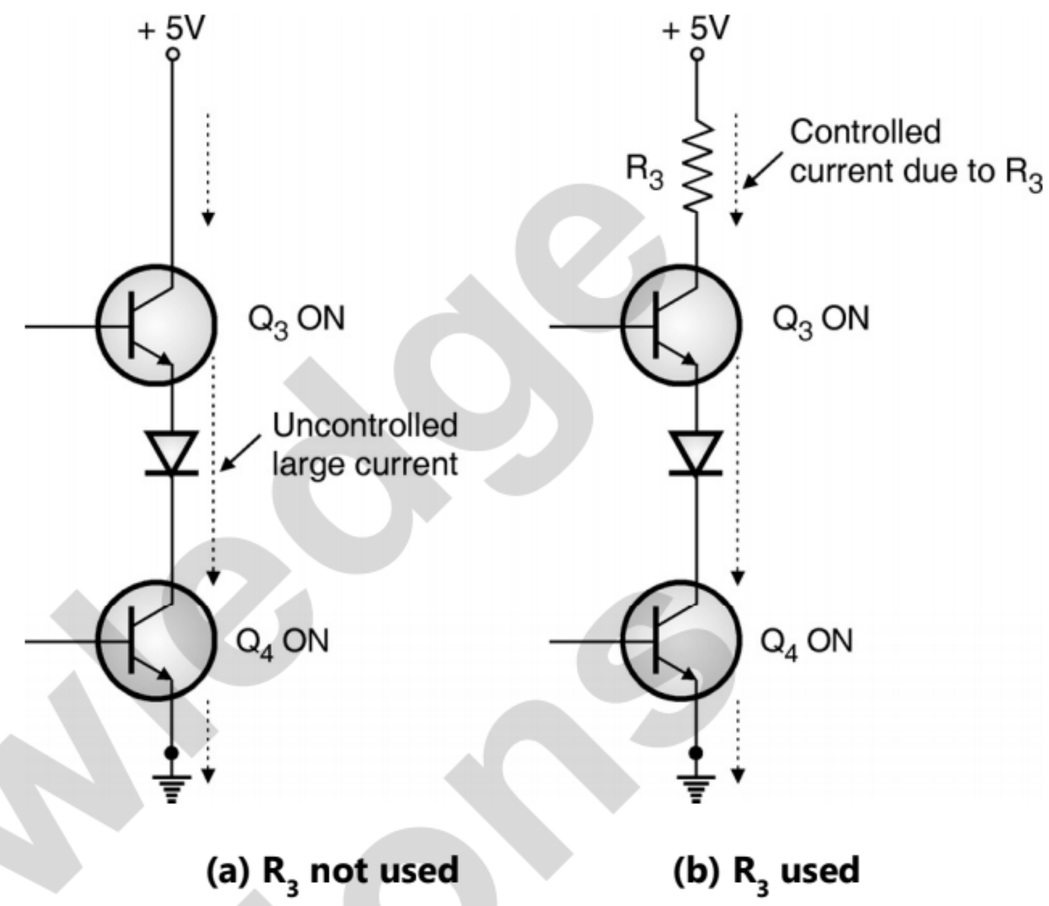
Function of R_3 :

– As discussed earlier, transistor Q_4 in the totem pole arrangement turns OFF more slowly than Q_3 turns ON.

– So for few nanoseconds both Q_3 and Q_4 will conduct simultaneously when the output Y is changing state from low to high. This is called as **cross conduction.**

– During the cross conduction, if R_3 is not used then there will be no current limiting element in series with Q_3 and Q_4 and a heavy current will be drawn from the source as shown in Fig. 9.4.7(a) which can damage the IC.

– This can be avoided by limiting the current by inserting resistor R_3 in series with Q_3, as shown in Fig. 9.4.7(b).



(a) R_3 not used **(b) R_3 used**

(C-1017) **Fig. 9.4.7 : Function of R_3**

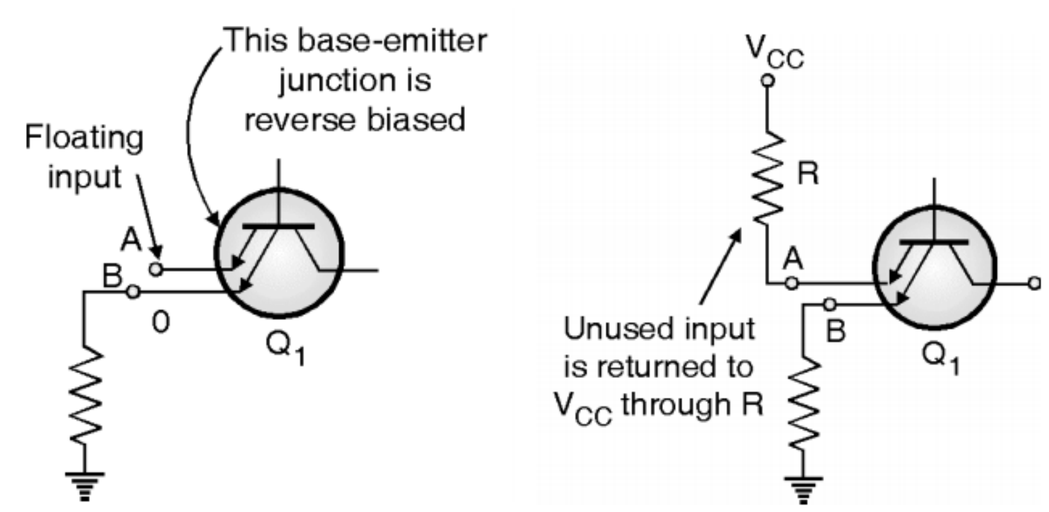
9.4.4 Unconnected Inputs :

– If any input of a TTL gate is left open, disconnected or floating, then the corresponding base emitter junction of the input transistor Q_1 is not forward biased as shown in Fig. 9.4.8(a).

– Therefore the open or floating input is equivalent to a logical 1 is applied to that input.

– Hence in TTL ICs all the unconnected inputs are treated as logical 1s.

– However, the unused inputs should either be connected to some used input (s) or returned to V_{CC} through a suitable resistor as shown in Fig. 9.4.8(b).



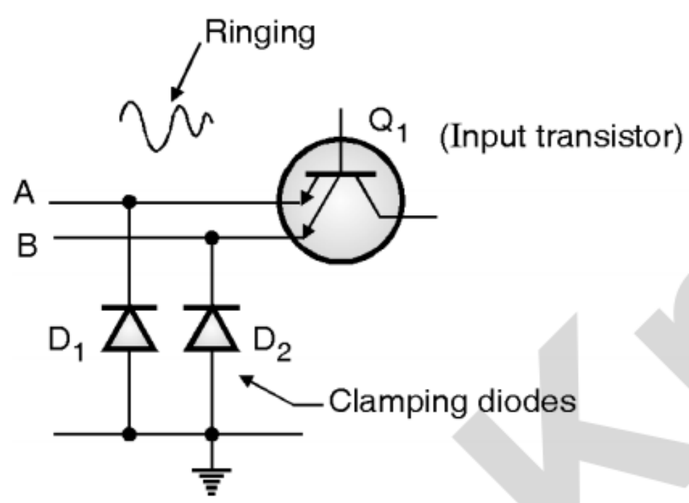
(a) Unused input **(b) Standard connection**

(C-1018) **Fig. 9.4.8**

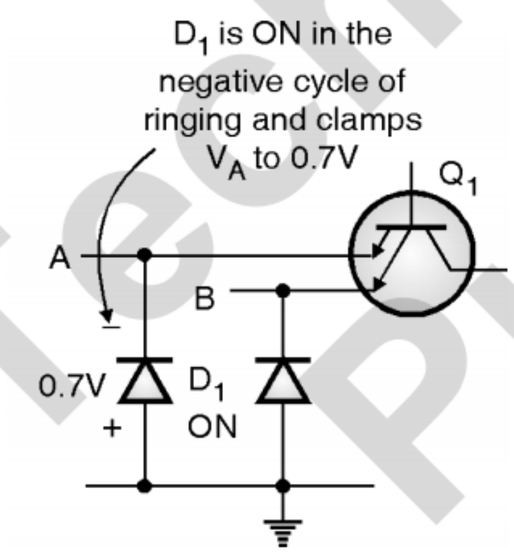
TechKnowledge Publications

9.4.5 Clamping Diodes :

- The TTL inputs should not be subjected to negative voltages. Under normal operating conditions this gets managed easily.
- But if fast voltage transitions are applied at the input, then there is a possibility of ringing (sinusoidal oscillations with positive and negative half cycles). Due to ringing, the inputs will be subjected to negative voltages.
- To suppress this ringing, clamping diodes are generally connected externally in all the TTL circuits as shown in Fig. 9.4.9(a).



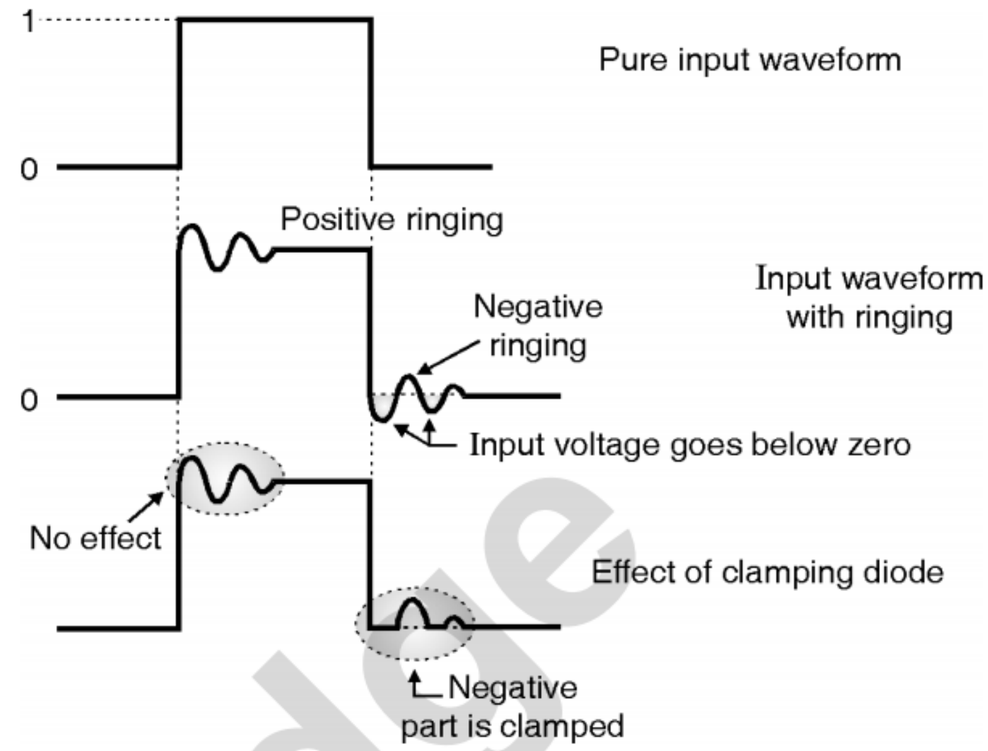
(a) Clamping diodes



(b) Effect of clamping diodes

(C-1019) Fig. 9.4.9

- These are fast recovery diodes. They are forward biased during the negative half cycles of the ringing sinusoidal waveform. Hence the negative input voltage will be restricted (clamped) to – 0.7 Volt approximately as shown in Fig. 9.4.9(b).
- The clamping action can be better understood by referring to Fig. 9.4.10.



(C-1020) Fig. 9.4.10 : Effect of clamping diodes

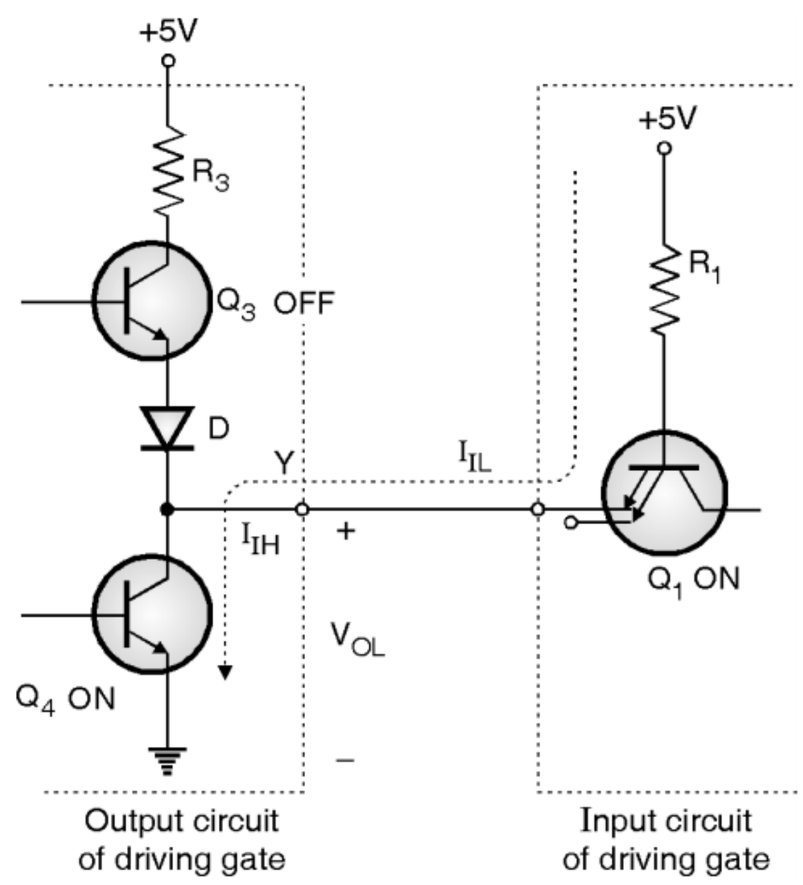
9.4.6 5400 Series :

- The temperature range for the devices in 7400 series is from 0 to 70°C, over a supply voltage range of 4.75 to 5.25 V.
- So 7400 series is used for the commercial applications.
- But 5400 TTL series is developed specially for the military applications.
- The devices of 5400 series operate over the temperature range of – 55 to 125°C and over the supply voltage range of 4.5 to 5.5 Volts.

9.5 Sourcing and Sinking in TTL :

9.5.1 Current Sinking Action :

- The current sinking will take place when output Y = 0 as shown in Fig. 9.5.1(a).



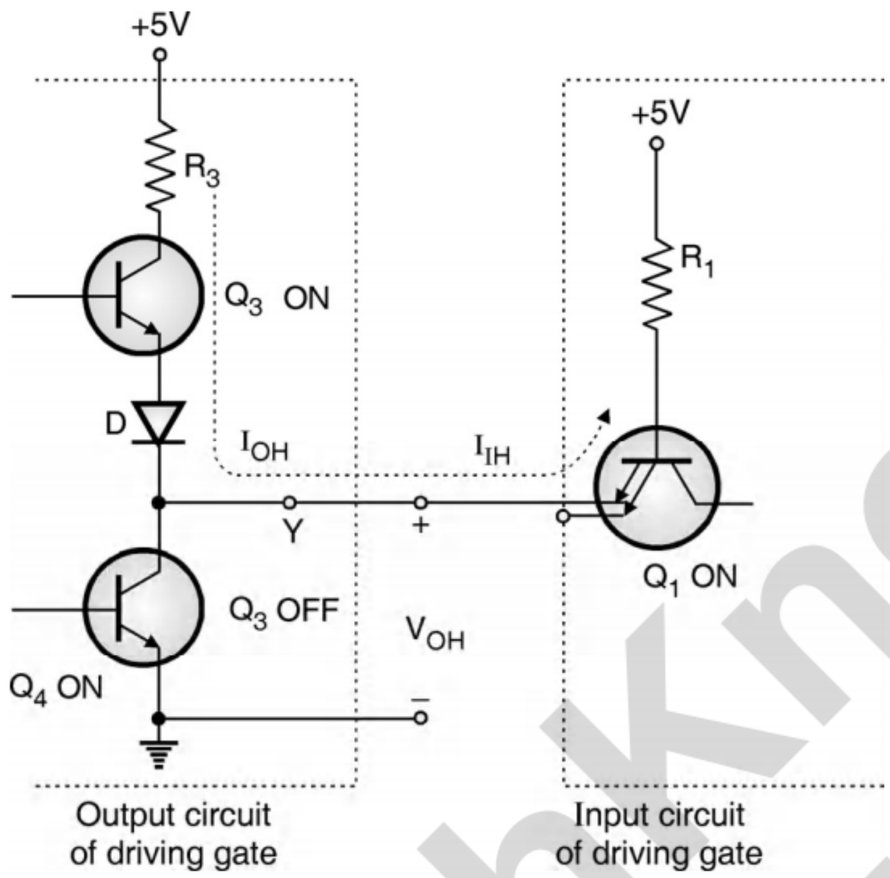
(C-1022) Fig. 9.5.1(a) : With Y = 0 the driving gate sinks current from the load gate

– The current flows from the input stage of load gate into the output stage of driving gate.

– Transistor Q_4 of the driving gate acts as a closed switch because it is in saturation. This will forward bias the base emitter junction of Q_1 of the load gate.

9.5.2 Current Sourcing Action :

– Current sourcing will take place when the output of the driving gate (Y) becomes HIGH, i.e. Y = 1. Refer Fig. 9.5.1(b).



(C-1023) Fig. 9.5.1(b) : Current sourcing in TTL

– Q_3 will be ON and Q_4 will be OFF. So the base emitter junction of Q_1 of the load gate will be reverse biased.

– Therefore the reverse leakage current of Q_1 will be supplied (sourced) by the driving gate as shown in Fig. 9.5.1(b).

– Note that this current is extremely small, typically of the order of 10 μA.

– Transistor Q_3 is called as current sourcing or pull up transistor and Q_4 is called as current sinking or pull down transistor.

9.5.3 Three Input TTL NAND Gate :

SPPU : Dec. 15, Dec. 16, May 18, Dec. 18, Dec. 19

University Questions

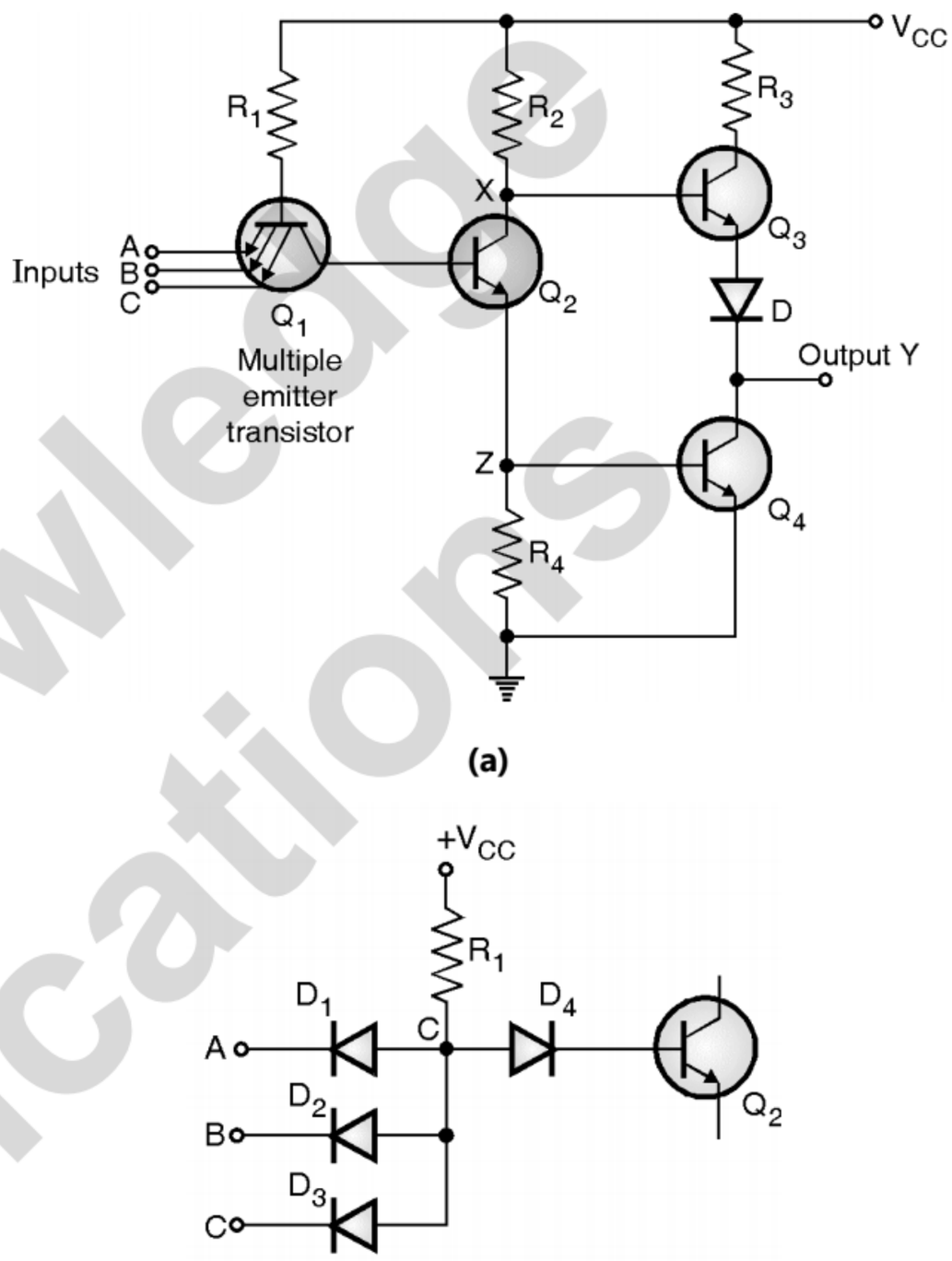
Q. 1 Draw three input standard TTL NAND gate circuit and explain its operation.

(Dec. 15, Dec. 16, 6 Marks, May 18, Dec. 18, 5 Marks, Dec. 19, 7 Marks)

Circuit diagram :

– The circuit diagram of a three input NAND gate is as shown in Fig. 9.5.2.

– Note that the multiple emitter transistor has three emitter terminals which act as the inputs A, B and C to the NAND gate.



(b) Equivalent circuit of the multiple emitter transistor Q_1

(C-1685) Fig. 9.5.2 : Three input TTL NAND gate

Operation :

– The principle of operation of this circuit is exactly same as that of the two input NAND gate discussed in section 9.4.2.

– If at least one of the inputs is low (0) then at least one of the diodes D_1, D_2, D_3 in Fig. 9.5.2(b) will be conducting. Hence the voltage at point C will be clamped to 0.7 V.

– Hence Q_2 will be off. So Q_3 will conduct and the output Y = 1 (HIGH).

∴ Y = 1 if at least one input is 0.

– If A = B = C = 1 then D_1, D_2, D_3 will be off. So Q_2 will be turned on. So Q_3 will be turned off and Q_4 will be turned on. So the output Y = 0 (LOW).

∴ Y = 0 if all the inputs are 1.

Tech Knowledge Publications

Truth Table :

– Table 9.5.1 shows the truth table and the status of various transistors in the circuit.

(C-6351) Table 9.5.1 : Truth table of the 3-input NAND gate

Input			Status of various transistors			Output Y
A	B	C	Q_2	Q_3	Q_4	
0	0	0	OFF	ON	OFF	1
0	0	1	OFF	ON	OFF	1
0	1	0	OFF	ON	OFF	1
0	1	1	OFF	ON	OFF	1
1	0	0	OFF	ON	OFF	1
1	0	1	OFF	ON	OFF	1
1	1	0	OFF	ON	OFF	1
1	1	1	ON	OFF	ON	0

9.6 Wired AND Connection (TTL) :

SPPU : Dec. 04, May 05, May 14

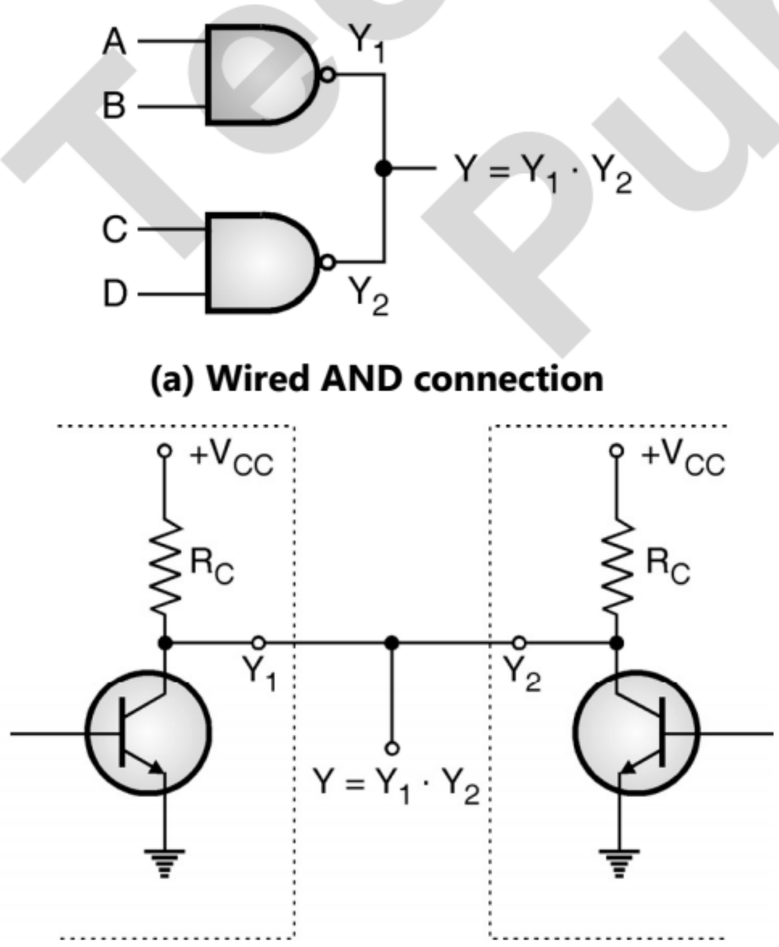
University Questions

Q. 1 Explain why two TTL totempole outputs cannot be connected together ? **(Dec. 04, May 05, 5 Marks)**

Q. 2 Explain the wired logic output of TTL with neat diagram. **(May 14, 3 Marks)**

– If we connect the outputs of two gates directly to each other as shown in Fig. 9.6.1, then they are said to be wired ANDed with each other.

– Wired ANDing is also called as wired ORing.



(a) Wired AND connection

(b) Wired AND connection with internal output stages shown

(C-1025) Fig. 9.6.1

– If the outputs are wired ANDed, then the final output is given by,

$$Y = Y_1 \cdot Y_2 = (\overline{A \cdot B}) \cdot (\overline{C \cdot D}) = \overline{AB + CD}$$

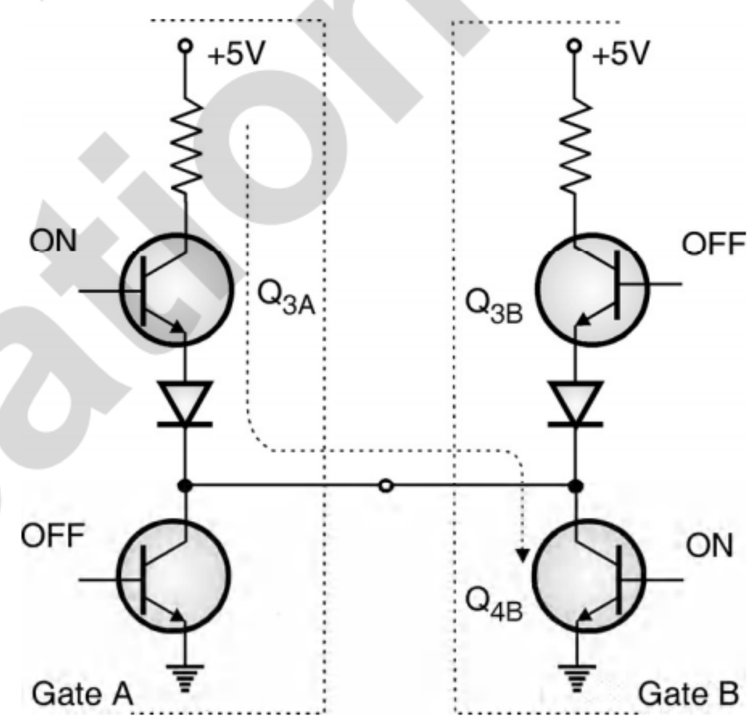
Can we wire AND the gates having Totempole output stage ?

– The answer is No. Wired AND connection should not be used for the totem-pole output circuits because of the current spike problem.

– TTL circuits with open collector outputs are available which can be used for wired AND connection.

Why cannot we connect the totempole outputs together ?

– The outputs of two TTL gates with totem pole output are shown to be connected together in Fig. 9.6.2.



(C-1026) Fig. 9.6.2 : Totempole outputs tied together

– There are two reasons why such connection should be avoided.

– Assume that output of Gate A is HIGH so Q_{3A} is ON and output of Gate B is LOW so Q_{4B} is conducting.

– Since both transistors operate in saturation, they are equivalent to small resistances. Hence Q_{4B} acts as a very low resistive load and draws a very large current which is higher than its rated current.

– Hence the transistor may get damaged due to overheating, after sometime.

– Second problem caused by these high currents is that the current through Q_{4B} will produce a voltage drop V_{CE4B} to raise V_{OL} between 0.5 to 1 V. This is greater than the allowable $V_{OL(max)}$.

– Due to these two reasons, the totempole outputs should never be directly connected together.

9.7 Open Collector Outputs (TTL) :

SPPU : Dec. 11, May 17, May 18

University Questions

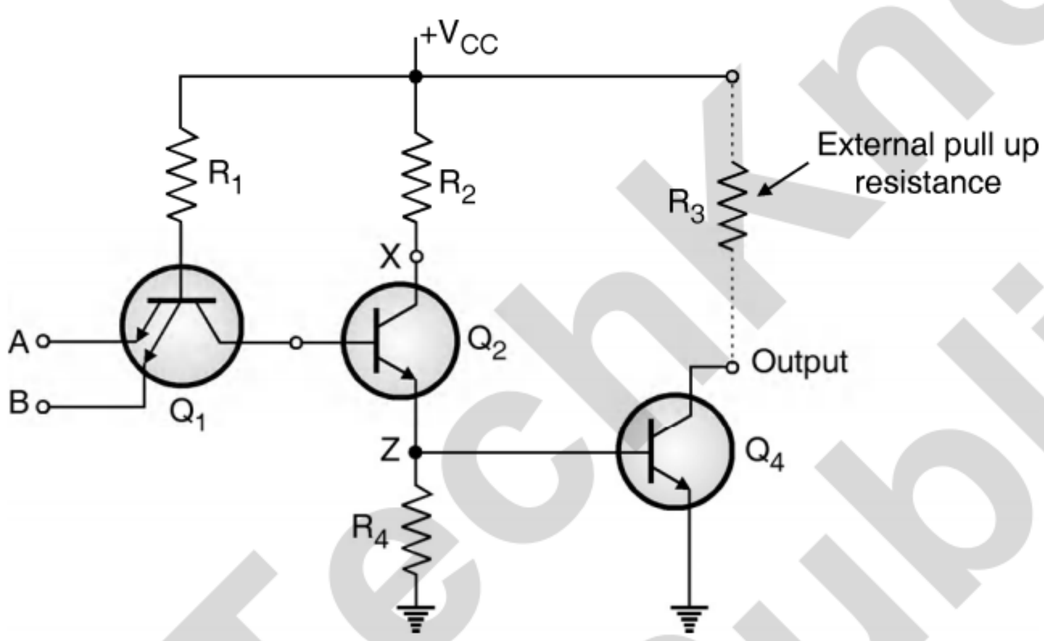
Q. 1 Explain the working of two input TTL NAND gate with open collector output. Consider various input, output states for explanation. **(Dec. 11, 8 Marks)**

Q. 2 Explain TTL open collector. **(May 17, 3 Marks)**

Q. 3 Explain TTL open collector logic. **(May 18, 5 Marks)**

Circuit diagram :

- We have seen that the gates having totem-pole output cannot be wired ANDed.
- Such a connection becomes possible if another type output stage called open collector output is used.
- The circuit diagram of a 2-input NAND gate is shown in Fig. 9.7.1. You will realize that this is the same TTL NAND gate which we have discussed earlier but with R_3 and Q_3 removed.



(C-1027) **Fig. 9.7.1 : Open collector 2 input NAND gate**

- The collector point of Q_4 is brought out as output as shown in Fig. 9.7.1, therefore it is called as open collector output.
- For proper operation it is necessary to connect an external resistance R_3 between V_{CC} and the open collector output as shown in Fig. 9.7.1. This resistance is called as pull up resistance.

Operation :

1. With A = B = 0 :

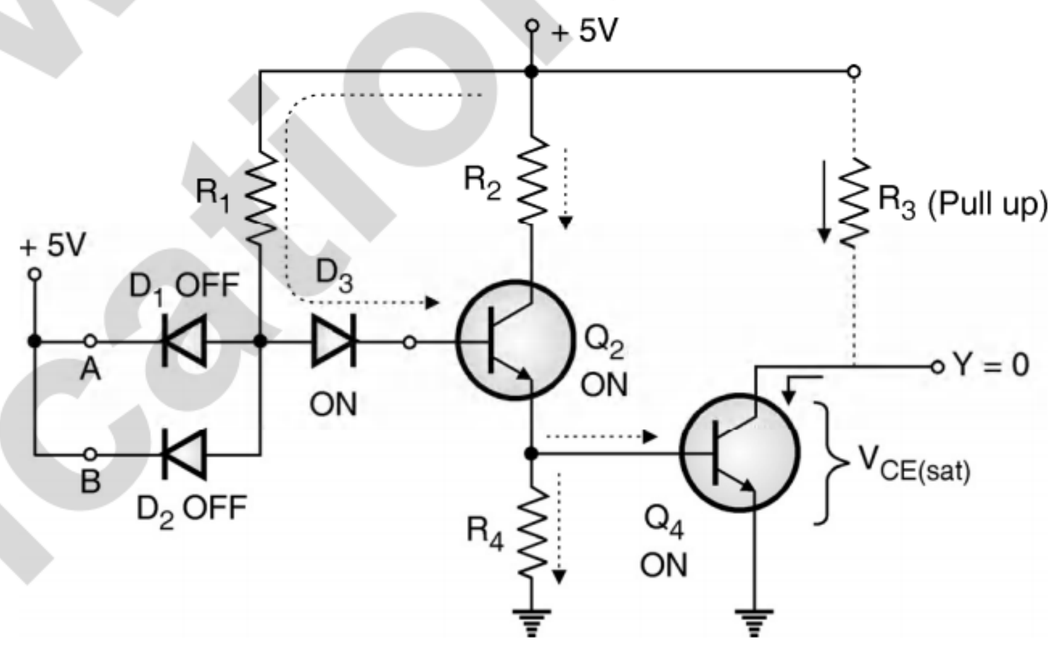
- With A = B = 0, both the BE junctions of Q_1 are forward biased. So Q_2 remains OFF.
- Hence no current flows through R_4. So $V_Z \approx 0$ V.
- Therefore Q_4 is OFF and its collector voltage is equal to V_{CC}. So Y = 1 when A = B = 0.

2. With A = 0, B = 1 OR A = 1, B = 0 :

- One of the BE junctions is forward biased (the one corresponding to 0 input).
- So Q_2 is OFF and Q_4 also is OFF. So its collector voltage is equal to V_{CC}.
- Therefore output Y = 1 when any one input is low.

3. With A = B = 1 :

- When both the inputs are high, transistor Q_1 is turned OFF.
- So Q_2 will be turned ON. Sufficient voltage is developed across R_4. Base current is applied to Q_4 and Q_4 goes into saturation.
- So the output voltage is equal to $V_{CE\,(sat)}$ of Q_4 which is very small. Thus Y = 0 when A = B = 1. The equivalent circuit for this mode is shown in Fig. 9.7.2.



(C-1028) **Fig. 9.7.2 : Equivalent circuit of open collector 2-input NAND gate with A = B = 1**

Calculation of pull up resistor R_3 :

- The value of R_3 depends on the amount of current the gate is supposed to supply (source). That means R_3 depends on the value of **Fan-out.**
- Let the fan out be 5.

$$\therefore \text{ Fan out } = \frac{I_{OL}}{I_{IL}}, \quad \therefore 5 = \frac{I_{OL}}{I_{IL}}$$

- For a standard TTL, the value of I_{IL} is 1.6 mA.

$$\therefore I_{OL} = 5 \times I_{IL} = 5 \times 1.6 \text{ mA} = 8 \text{ mA}$$

- This current will flow through R_3. When Q_4 goes into saturation. Apply KVL to the collector circuit of Q_4 to write,

$$V_{CC} - I_{OL} R_3 - V_{CE4\,(sat)} = 0$$

$$\therefore R_3 = \frac{V_{CC} - V_{CE4\,(sat)}}{I_{OL}} = \frac{5 - 0.2}{8 \times 10^{3}} = 600\ \Omega$$

9.7.1 Disadvantages of Open Collector Output :

1. The value of pull up resistance is high (few kΩ). Therefore if the load capacitance is large then the RC time constant (R_3C) becomes large. This slows down the switching speed of Q_4. Therefore the gates having an open collector output will be slow.
2. Second disadvantage is increased power dissipation. When Q_4 is ON, a large current flows through the pull up R_3. Hence power dissipation is increased. This problem is eliminated if we use totem-pole output arrangement.

9.7.2 Advantage of Open Collector Output :

SPPU : Dec. 15, Dec. 17

University Questions

Q. 1 Explain the advantages of open collector output. **(Dec. 15, 4 Marks, Dec. 17, 3 Marks)**

– The main advantage of open collector output is that **Wired ANDing** becomes possible.

9.7.3 Wired ANDing :

SPPU : Dec. 15, Dec. 17, Dec. 18

University Questions

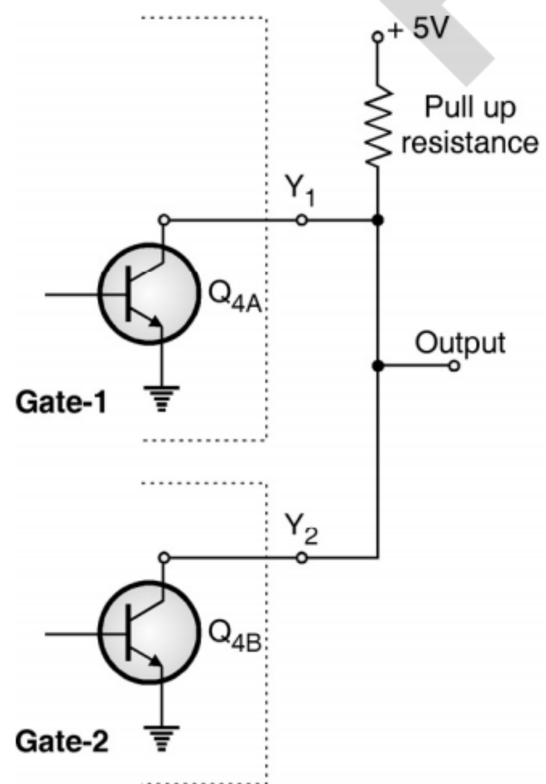
Q. 1 Explain the advantages of open collector output. **(Dec. 15, Dec. 17, 4 Marks)**

Q. 2 Draw and explain wired AND gate in detail. **(Dec. 18, 5 Marks)**

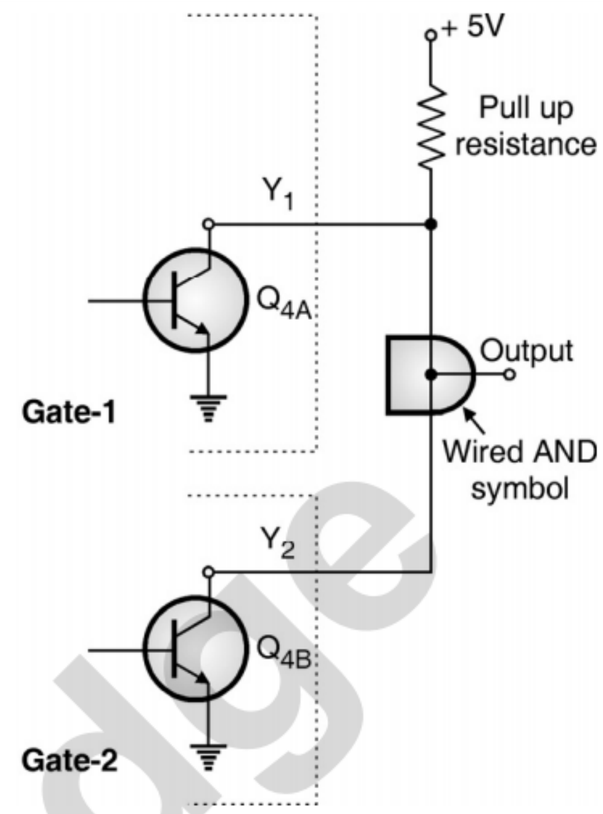
– Wired ANDing means connecting the outputs of gates together to obtain the AND function.

– It is possible to connect the outputs of two or more gates together as shown in Fig. 9.7.3(a).

– The wired ANDing is represented schematically as shown in Fig. 9.7.3(b).



(C-1029) **Fig. 9.7.3 (a) : Wired ANDing of two open collector TTL gates**



(C-1029) **Fig. 9.7.3 (b) : Symbol of wired ANDing**

– Q_{4A} and Q_{4B} are the output transistors of Gate-A and Gate-B respectively. A common pull up resistance is used for all the output transistors.

– The transistors Q_{4A} and Q_{4B} are operated as switches. They operate in either saturation or cut off regions.

– Therefore the wired ANDing equivalent circuit is as shown in Fig. 9.7.4(a).

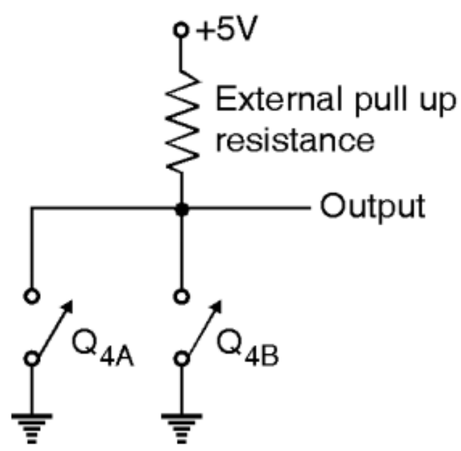
– The equivalent circuit of Fig. 9.7.4(a) indicates that the wired AND output will be "0" if any one or all the output transistor (s) are conducting.

– The output will be high (1) if and only if all the transistors are in their OFF state.

– This is the reason why such a connection is called as wired AND connection.

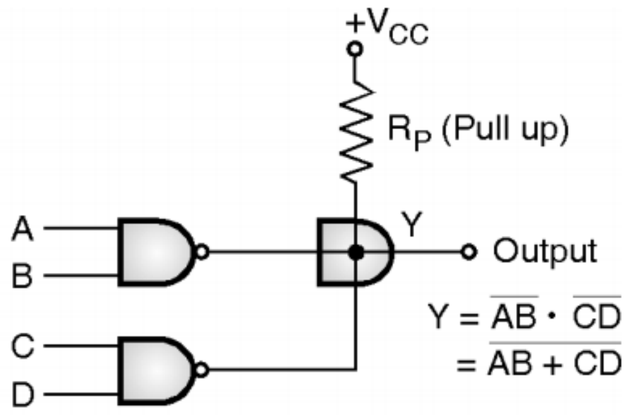
– Figs. 9.7.4(b), (c) and (d) shows that it is possible to wire AND the outputs of any gate such as NAND, NOR or inverter.

– The corresponding output Boolean equations are given alongwith the diagrams.

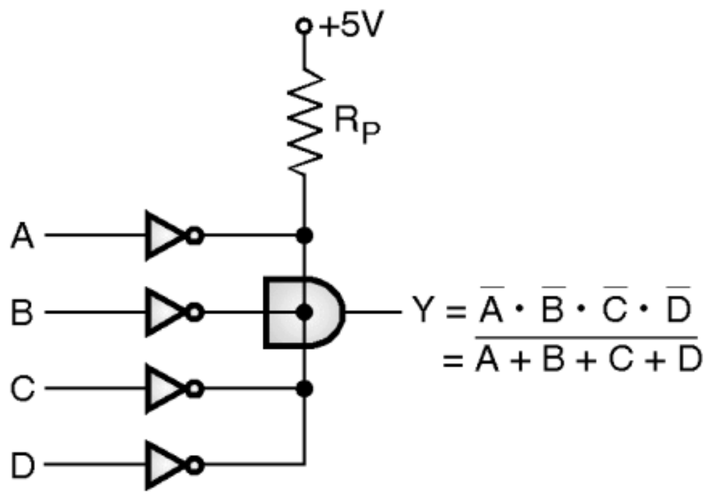


(C-1030) **Fig. 9.7.4(a) : Equivalent circuit of wired ANDing**

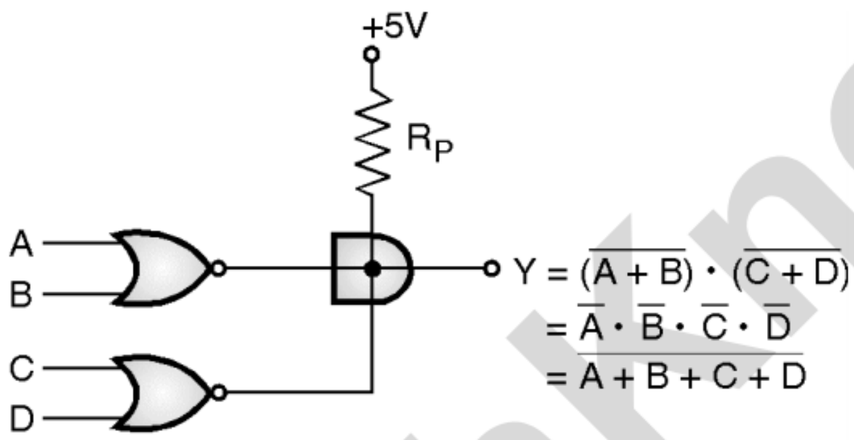
TechKnowledge Publications



(C-1031) Fig. 9.7.4(b) : Wired ANDing of NAND gate outputs



(c) Wired ANDing of inverter outputs



(d) Wired ANDing of the NOR outputs

(C-1031) Fig. 9.7.4

9.7.4 Comparison of Totem-pole and Open Collector Outputs : SPPU : Dec. 05, May 07

University Questions

Q. 1 With the help of circuit diagram explain the difference between totempole output and open collector output. **(Dec. 05, May 07, 10 Marks)**

– Table 9.7.1 gives the comparison of totem-pole and open collector outputs of TTL.

Table 9.7.1 : Comparison of totem-pole and open collector outputs

Sr. No.	Parameter	Totem-pole	Open collector
1.	Circuit components on the output side	Q_3 (pull up transistor), D and Q_4 (pull down transistor) are used.	Only the pull down transistor Q_4 is used.
2.	Wired ANDing	Cannot be done	Easily be done
3.	External pull up resistor	Not required	Required to be connected.
4.	Power Dissipation	Low due to presence of pull up transistor Q_3.	High due to current flowing through R_3.
5.	Speed	Operating speed is high.	Operating speed is low.

9.8 Tristate (Three state) TTL Devices :

SPPU : Dec. 06, May 07, May 08, May 19

University Questions

Q. 1 What do you mean by high impedance state ? Where is it used ? **(Dec. 06, 6 Marks)**

Q. 2 What is tri-state ? **(May 07, 2 Marks)**

Q. 3 What do you mean by high impedance state ? How is it used in single bus organization to connect various registers ? Explain with the help of suitable diagram. **(May 08, 8 Marks)**

Q. 4 Explain Tristate logic and Tristate TTL inverter. **(May 19, 6 Marks)**

– We have discussed the totem-pole and open collector output TTL configurations in previous sections.

– Now let us study the third type of output circuitry used in TTL as well as CMOS logic families. It is called as the **tristate configuration.**

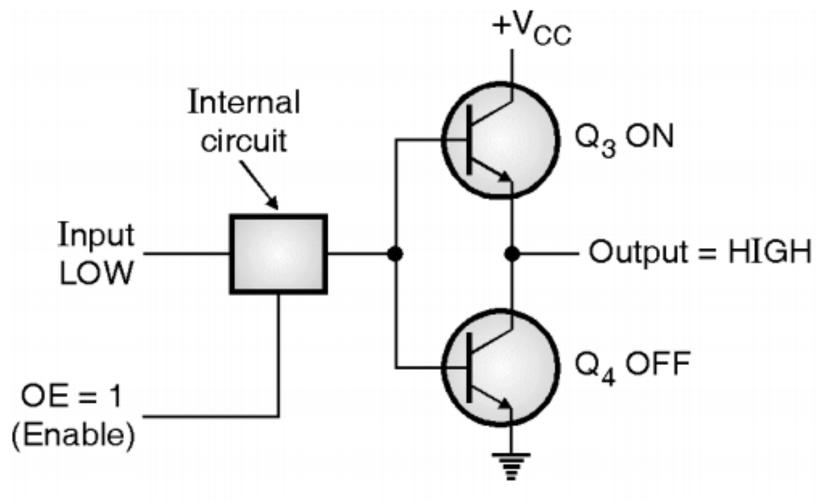
– It is called as tristate because it allows three possible output states namely :

1. HIGH 2. LOW and
3. High Impedance (Hi – Z) state.

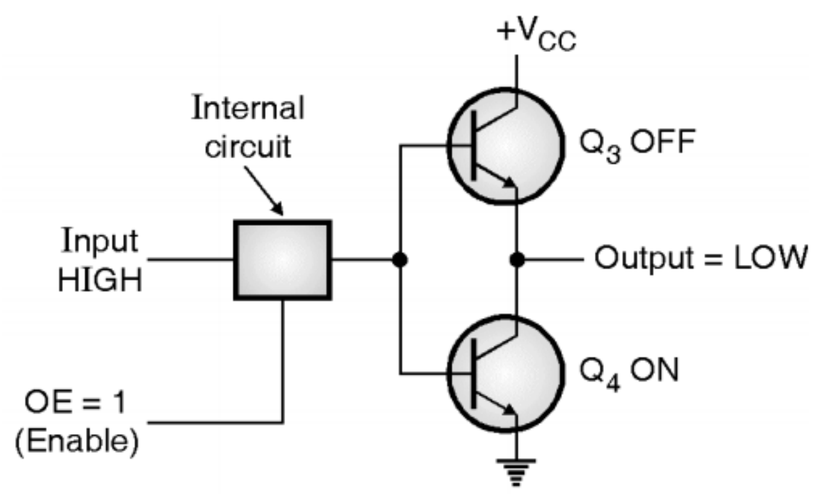
– An additional input called Output Enable (OE) is introduced. Both the pull up and pull down transistors (Q_3 and Q_4) are being used.

Block diagram :

– The output stage block diagram of a tri-state inverter is shown in Fig. 9.8.1(a). The enable signal OE decides the operation of transistors Q_3 and Q_4.



(a) State 1 : HIGH



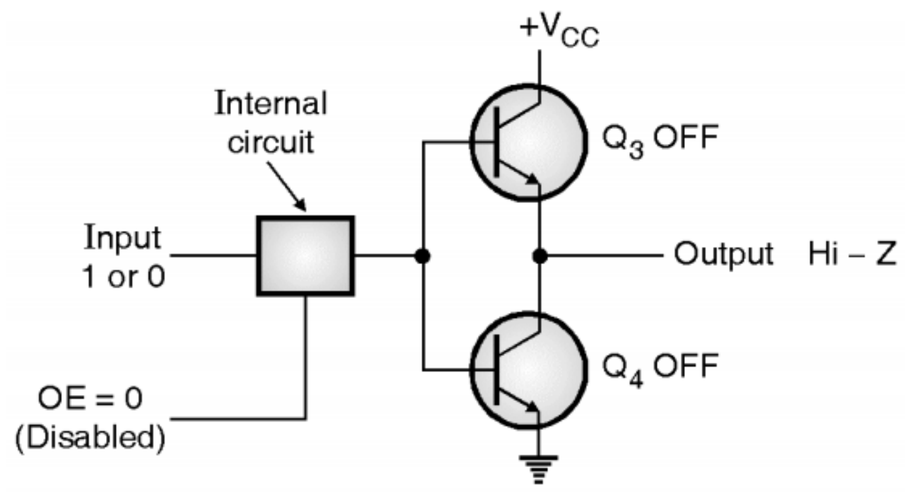
(b) State 2 : LOW

(C-1032) Fig. 9.8.1

- When OE = 1, then the inverter operates as a normal inverter. If input is 0 then output will be 1 (HIGH) as shown in Fig. 9.8.1(a). The pull up transistor Q_3 will be ON.
- Similarly with OE = 1 and input high (1), the output will be low because the pull down transistor Q_4 will be ON as shown in Fig. 9.8.1(b).

High impedance state :

- If the enable input OE = 0 (zero), then irrespective of the status of input, both the transistors will remain off as shown in Fig. 9.8.1(c).
- This state of operation is called as high impedance (Hi – Z) state. In this state the output terminal is essentially open circuit i.e. not connected anywhere.



(c) State 3 : High Impedance state

(C-1032) Fig. 9.8.1

9.8.1 Advantages of Tristate :

- The outputs of tristate ICs can be connected together i.e. they can make use of a common wire.
- But in doing so there is no reduction in switching speed (this problem is faced for open collector ICs).
- There is no reduction in speed because the tristate output when enabled, operates as a totem-pole TTL.
- So we get all the advantages of totem-pole output such as low output impedance and high speed.
- However note that when tristate outputs are connected together, we should enable only one of them at a time.
- Otherwise, two or more active outputs will fight to take control of the common line.
- This will result in very high currents which can damage the ICs and produce invalid output voltage levels.

9.8.2 Tristate Buffers : SPPU : May 05, May 07

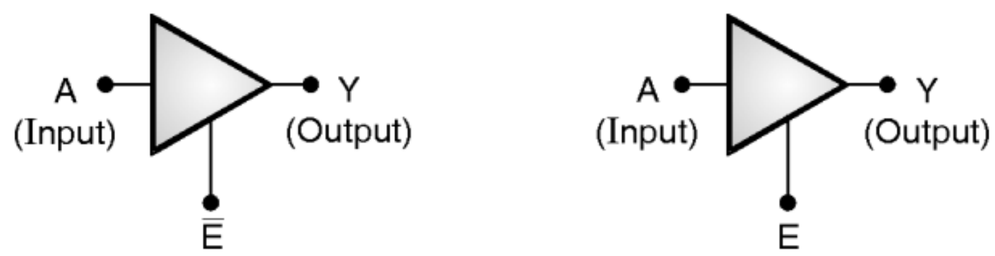
University Questions

Q. 1 What do you mean by tri-state buffer ? **(May 05, 8 Marks)**

Q. 2 What is the use of tri-state buffers ? Explain with suitable circuit diagram. **(May 07, 6 Marks)**

Definition :

- A tristate buffer is a circuit which either connects or isolates its input from its output.
- In other words it controls the flow of signal from input to output.
- Tristate buffers are of two types, namely the non-inverting buffers and inverting buffer.
- The inverting tristate buffers will invert the input data while passing it to the output.
- Fig. 9.8.2 shows the symbols and truth tables for a non-inverting buffer.



Truth table

$\overline{E}$	Y
0	A
1	Hi – Z

Truth table

E	Y
0	Hi – Z
1	A

(C-1033) Fig. 9.8.2 : Tristate non-inverting buffers, symbols and truth tables

TechKnowledge Publications

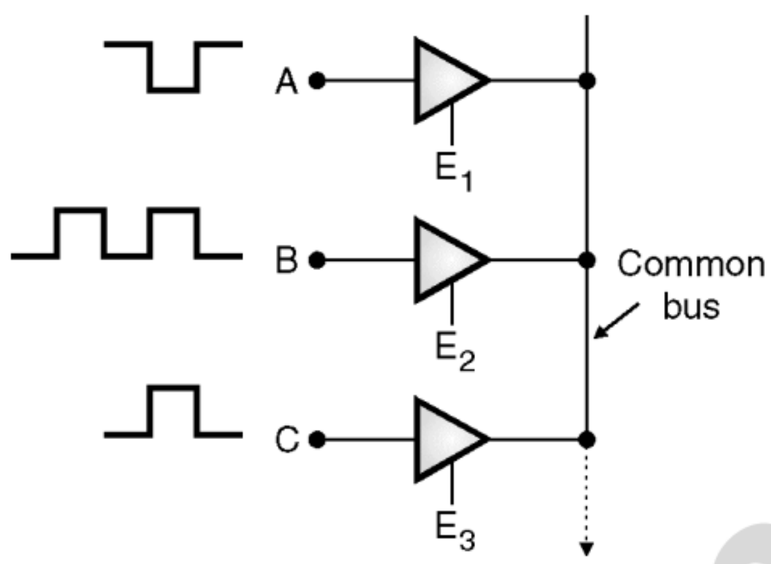
9.8.3 Applications of Tristate Buffers (Bus Organization) : SPPU : Dec. 04

University Questions

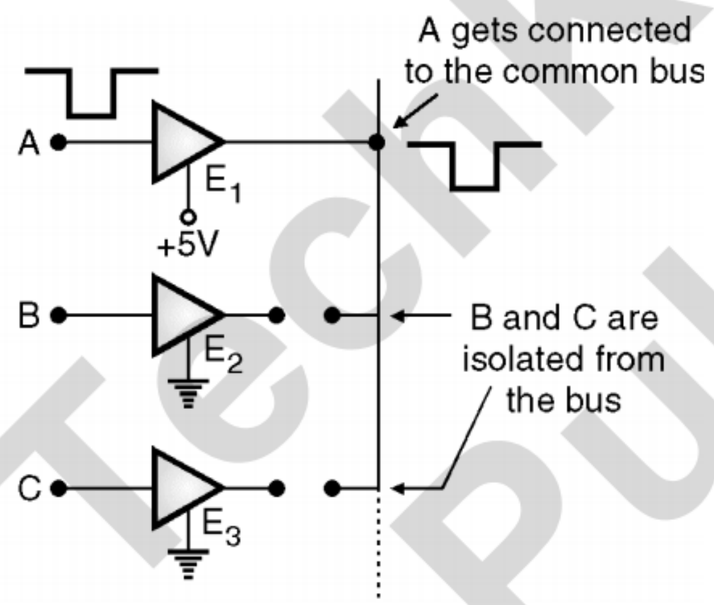
Q. 1 How tristate gates are useful in a bus connection ? **(Dec. 04, 6 Marks)**

– Tristate buffers are used in those applications in which several signals are simultaneously connected to a common bus.

– The basic idea about the common bus and the use of tristate buffers is illustrated in Fig. 9.8.3(a).



(a) A, B, C are connected to a common bus via tristate buffers



(b) A is getting transmitted on the bus

(C-1034) Fig. 9.8.3

– Three tristate buffers are used for connecting the inputs A, B and C to the common bus.

– As shown in Fig. 9.8.3(b), the enable signal E_1 is connected to +5 V. Therefore A gets connected to the bus.

– Since enable terminals E_2 and E_3 are connected to ground, the second and third buffer of Fig. 9.8.3(b) will go into the high impedance state. So B and C will be isolated from the common bus.

9.8.4 A TRI-STATE Inverter : SPPU : Dec. 05, Dec. 07, May 19

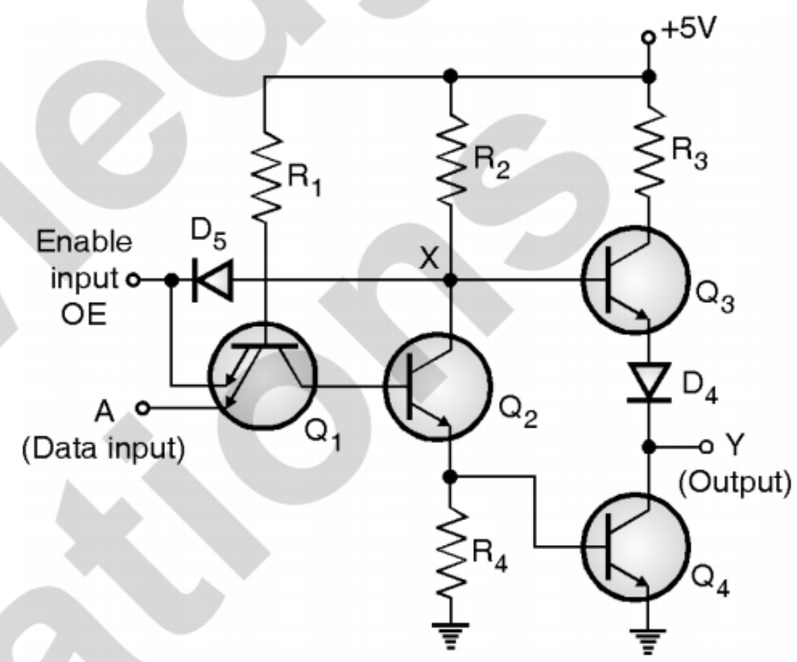
University Questions

Q. 1 Explain with the help of circuit diagram the operation of TTL inverter with totempole output. **(Dec. 05, Dec. 07, 8 Marks)**

Q. 2 Explain Tristate logic and Tristate TTL inverter. **(May 19, 6 Marks)**

Circuit diagram :

– Fig. 9.8.4 shows the circuit diagram of a Tri-State Logic (TSL) inverter.



(C-1035) Fig. 9.8.4 : A TSL inverter

– It has an Enable Input (EI) and a data input A and one output. The output stage is totempole type which has been discussed earlier.

Operation :

1. When OE = LOW :

– When OE = 0, diode D_5 is forward biased and is turned ON. Therefore the voltage at point X is equal to 0.7 Volts, irrespective of the state of data input A.

– V_X = 0.7 V is not sufficient to turn ON Q_3 , so it remains OFF.

– "OE" is also connected to one emitter of transistor Q_1. As OE = "0", transistor Q_1 is ON, hence Q_2 is OFF (as discussed for the NAND gate).

– As Q_2 is OFF, current through R_4 is zero. So Q_4 is in the OFF state.

– Thus with OE = LOW (0), both the output transistors Q_3 and Q_4 will be OFF and the inverter will be in the **High-Impedance** state.

2. When OE = HIGH :

– When OE = 1 (HIGH), the diode D_5 will be reverse biased. So it will be ineffective.

– Also the corresponding emitter base junction of Q_1 will be reverse biased. So the circuit of TSL inverter now acts as a TTL NAND gate with only one input A.

– If A = 0, then Q_1 is ON, so Q_2 is OFF, so $V_X = V_{CC}$, hence Q_3 turns ON and output Y = 1.

– If A = 1, then Q_1 is OFF, so Q_2 is ON, so V_X is low, but sufficient voltage appears across R_4, so Q_4 goes into saturation and Y = 0.

– Thus with OE = 1, the given circuit operates as a normal inverter.

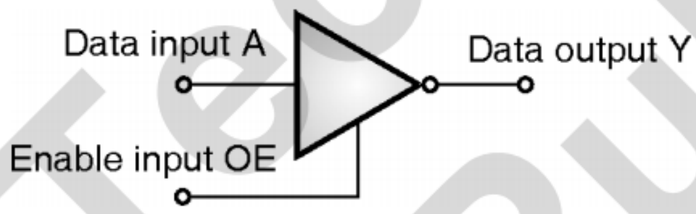
– Table 9.8.1 summarizes the operation of the TSL inverter.

Table 9.8.1 : Operation of TSL inverter

Enable input	Data input	Output
OE = 1	A = 0	Y = 1
OE = 1	A = 1	Y = 0
OE = 0	A = 0 or 1	Hi-Impedance

Logic symbol :

– The logic symbol of a TSL inverter is shown in Fig. 9.8.5.



(C-1036) **Fig. 9.8.5 : Logic symbol of TSL inverter**

9.9 Standard TTL Characteristics :

SPPU : Dec. 13, May 15, Dec. 17, May 18, May 19, Dec. 19

University Questions

Q. 1 Define the following terms and mention its standard values for TTL family :

1. Voltage and current parameter.
2. Power dissipation.
3. Noise margin. **(Dec. 13, May 19, 6 Marks)**

Q. 2 Explain the standard TTL characteristics in detail. **(May 15, 8 Marks)**

Q. 3 Define the following terms and mention the standard values for TTL logic family :

1. Noise margin. 2. Fan out.
3. Power dissipation.
4. Propagation delay. **(Dec. 17, 8 Marks)**

Q. 4 State the following characteristics of digital IC logic family TTL and CMOS :

1. FAN out.
2. Noise margin **(May 18, 4 Marks)**

Q. 5 Explain the following characteristics of TTL logic families :

1. Power dissipation 2. Noise margin
3. Propagation delay. **(Dec. 19, 6 Marks)**

– The Texas instrument first introduced two TTL series namely 54 series and 74 series.

Supply voltage and temperature ranges :

– Table 9.9.1 gives the supply voltage and temperature ranges for two TTL families.

Table 9.9.1

TTL series	Supply voltage range	Temperature range
74 series	4.75 V to 5.25 V	0° C to 70° C
54 series	4.5 V to 5.5 V	– 55° C to 125° C

Voltage levels :

– Table 9.9.2 shows the input and output logic voltage levels for the TTL 74 series.

Table 9.9.2 : Voltage levels for TTL 74 series

Voltages	Minimum	Typical	Maximum
V_{OL}	–	0.2 V	0.4 V
V_{OH}	2.4 V	3.4 V	–
V_{IL}	–	–	0.8 V
V_{IH}	2.0 V	–	–

Noise margin :

– We have already defined the noise margins as,

High level noise margin,

$$V_{NH} = V_{OH\,(min)} - V_{IH\,(min)}$$

and Low level noise margin,

$$V_{NL} = V_{IL\,(max)} - V_{OL\,(max)}$$

TechKnowledge Publications

– Substituting the values from Table 9.9.2, the noise margin for the TTL logic families can be calculated.

$$\therefore V_{NH} = 2.4 - 2 = 0.4 \text{ V.}$$

$$V_{NL} = 0.8 - 0.4 = 0.4 \text{ V.}$$

– Thus noise margin for the TTL family is 0.4 V. This means that as long as the induced noise voltage is less than 0.4 V, the operation of the TTL ICs will be unaffected. The noise margins have been shown in section 9.3.3.

Power dissipation :

– The average power dissipation for the standard TTL 74 series is approximately 10 mW.

– It is dependent on the parameters such as tolerances, signal level etc.

Propagation delay :

– The propagation delay of a standard TTL gate is approximately 10 nanoseconds (1 nanosecond = 10^{-9} seconds)

Fan out :

– A standard TTL gate is capable of driving at the most 10 other TTL gates simultaneously, maintaining its output voltage within the specified limits. Hence fan out of a TTL gate is 10.

– All the important characteristics of TTL logic family are listed in Table 9.9.3.

Table 9.9.3 : Important parameters of TTL logic family

Sr. No.	Parameter	Values
1.	Supply voltage	74 series : (4.75 to 5.25 V) 54 series : (4.5 to 5.5 V)
2.	Temperature range	74 series : 0 to 70° C 54 series : – 55 to 125° C
3.	Voltage levels	$V_{IL\ (max)} = 0.8$ V $V_{OL\ (max)} = 0.4$ V $V_{IH\ (min)} = 2$ V $V_{OH\ (min)} = 2.4$ V
4.	Noise margin	0.4 V
5.	Power dissipation	10 mW
6.	Propagation delay	10 nanosec.
7.	Fan out	10
8.	Figure of merit	100

9.10 Advantages and Disadvantages of TTL :

9.10.1 Advantages of TTL :

1. TTL circuits are fast.
2. Low propagation delay.
3. Power dissipation is not dependent on frequency.
4. Compatible to all the other families.
5. Latch ups do not take place.
6. These are not susceptible to the damage due to static charges.
7. Higher current sourcing and sinking capabilities.

9.10.2 Disadvantages of TTL :

1. Large power dissipation.
2. Fan out is lower than that of CMOS.
3. Less component density.
4. Can operate only on + 5 V power supply.
5. Poor noise immunity.

9.11 MOS - Logic Family :

– The MOS - logic family uses MOSFET as the basic device the way TTL uses BJT.

– The MOSFETs are of two types namely the depletion MOSFETs and the enhancement type MOSFETs.

– In logic circuits, the enhancement MOSFETs are used. We use the E-MOSFET as a switch.

9.12 CMOS Logic :

Definition :

– CMOS stands for complementary MOSFET. It is obtained by using a p-channel MOSFET and n-channel MOSFET simultaneously.

– The p and n channel MOSFETs are connected in series, with their drains connected together and output taken from common drain point.

– Input is applied at the common gate terminal formed by connecting two gates together.

TechKnowledge Publications

– The 74C00 CMOS series is a group of CMOS circuits which are pin-for-pin and function for function compatible with the TTL 7400 devices.

– For example 74C32 is a quad 2-input OR gate in the CMOS family whereas 7402 is a quad 2 input TTL gate in the 7400 TTL family.

9.12.1 CMOS Inverter :

SPPU : May 12, May 17, Dec. 17, May 18, Dec. 19

University Questions

Q. 1 Draw the structure of CMOS inverter gate. Explain its working. **(May 12, 4 Marks)**

Q. 2 Explain CMOS inverter. **(May 17, 4 Marks)**

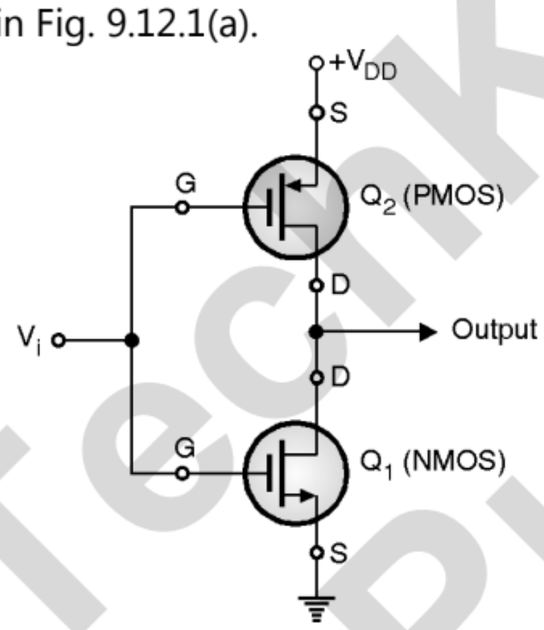
Q. 3 Draw and explain the circuit diagram of CMOS inverter. **(Dec. 17, 5 Marks)**

Q. 4 Explain with neat diagram CMOS inverter. **(May 18, 4 Marks)**

Q. 5 Draw and explain the circuit diagram of CMOS inverter. **(Dec. 19, 7 Marks)**

Circuit diagram :

– CMOS inverter with positive input and output voltages is shown in Fig. 9.12.1(a).



(C-1053) **Fig. 9.12.1(a) : CMOS inverter with positive voltages**

Operation :

1. With V_i = 0 Volt (logic 0) :

– With V_i = 0 Volt, the gate to source voltage V_{GS} of Q_1 (NMOS) will be 0 Volt, hence it will be OFF.

– But V_{GS} of Q_2 (PMOS) will be equal to – V_{DD} . So Q_2 will be ON.

– Hence the output voltage will be equal to + V_{DD} i.e. logic 1.

2. With $V_i = V_{DD}$ (logic 1) :

– With V_i = + V_{DD} , the gate to source voltages of the two MOSFETs are as follows :

Q_1 (NMOS) : V_{GS1} = V_{DD}

Q_2 (PMOS) : V_{GS2} = 0 Volt

– Hence Q_1 will be turned ON and Q_2 will be OFF. So output gets connected to ground i.e. it will be "0". Thus the inversion will take place.

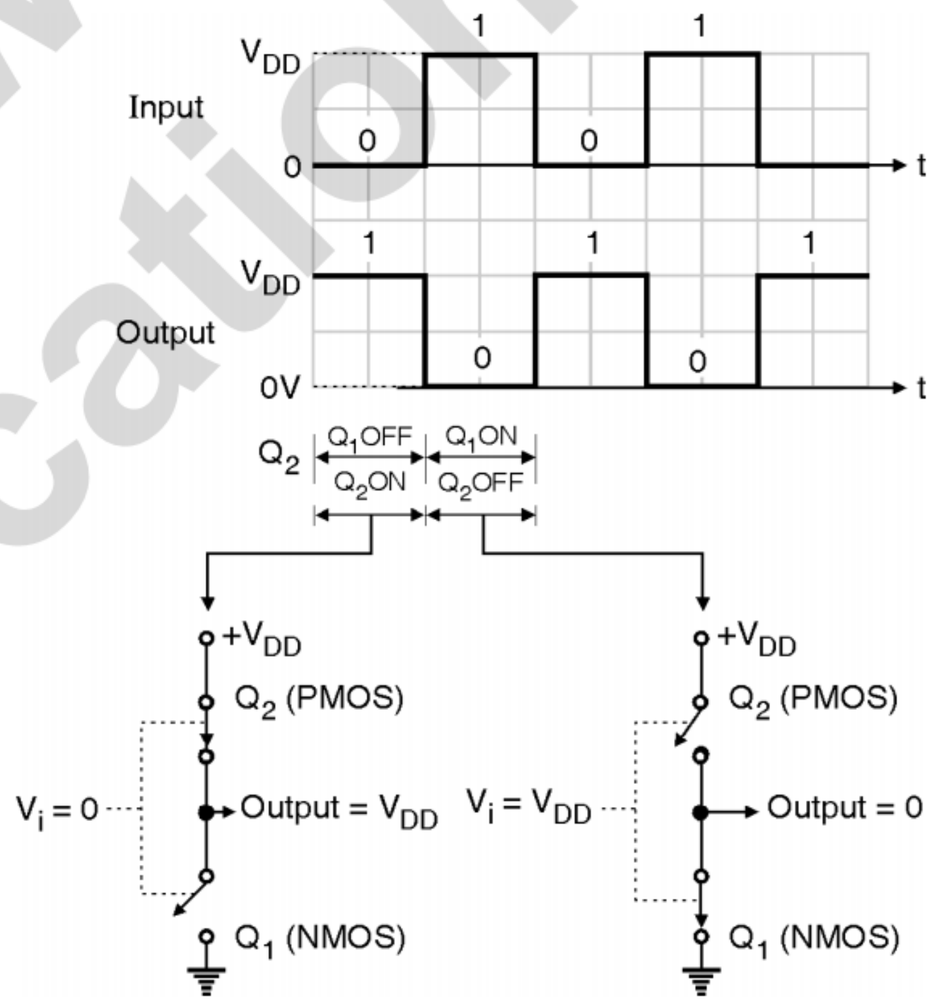
– The operation of this inverter has been summarized in Fig. 9.12.1(b).

V_i	Q_1	Q_2	Output
0 V (logic 0)	OFF	ON	V_{DD} (logic 1)
V_{DD} (logic 1)	ON	OFF	0 (logic 0)

(C-1053) **Fig. 9.12.1(b) : Summary of operation**

Waveforms and equivalent circuits :

– The input output voltage waveforms for the inverter are shown in Fig. 9.12.2.



(C-1054) **Fig. 9.12.2 : Input output voltage waveforms and equivalent circuits for a CMOS inverter**

9.12.2 CMOS NOR Gate :

SPPU : Dec. 10, Dec. 16, Dec. 18

University Questions

Q. 1 Explain working of 2-input CMOS-NOR gate. **(Dec. 10, 4 Marks)**

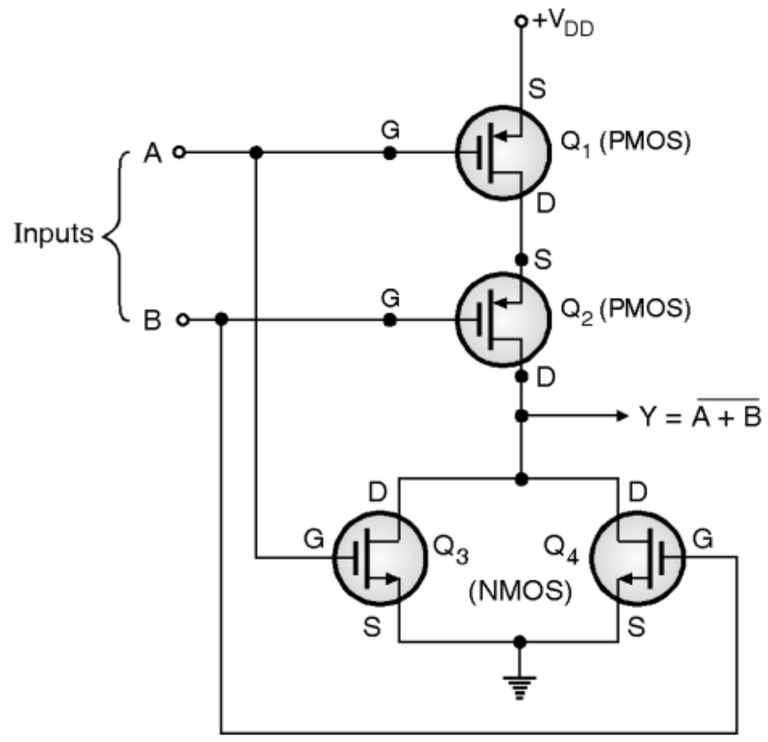
Q. 2 Compare TTL and CMOS logic family and also draw CMOS-NOR gate. **(Dec. 16, 7 Marks)**

Q. 3 Explain with a neat diagram CMOS NOR gate. **(Dec. 18, 4 Marks)**

TechKnowledge Publications

Circuit diagram :

– Fig. 9.12.3 shows the CMOS 2-input NOR gate. It is obtained by modifying the CMOS inverter circuit.



(C-1055) **Fig. 9.12.3 : CMOS NOR gate**

– Q_1 and Q_2 are p-channel MOSFETs connected in series and Q_3 Q_4 are n-channel MOSFETs connected in parallel with each other.

– Input A is connected to the gates of Q_1 and Q_3 while input B is connected to the gates of Q_2 and Q_4 .

– Output Y = V_{DD} = 1 if Q_1 and Q_2 both ON and Q_3 Q_4 both OFF. And output Y = 0 if Q_1 or Q_2 or both are OFF and Q_3 or Q_4 both are ON.

Operation :

1. With A = B = 0 :

– $V_{GS1} = -V_{DD}$, $V_{GS2} = -V_{DD}$. Hence Q_1 and Q_2 will be ON.

– $V_{GS3} = 0$, $V_{GS4} = 0$. So Q_3 and Q_4 will be OFF.

– The equivalent circuit for this mode is shown in Fig. 9.12.4(a) which shows that output Y = V_{DD} (logical 1).

∴ For A = 0, B = 0, output Y = 1.

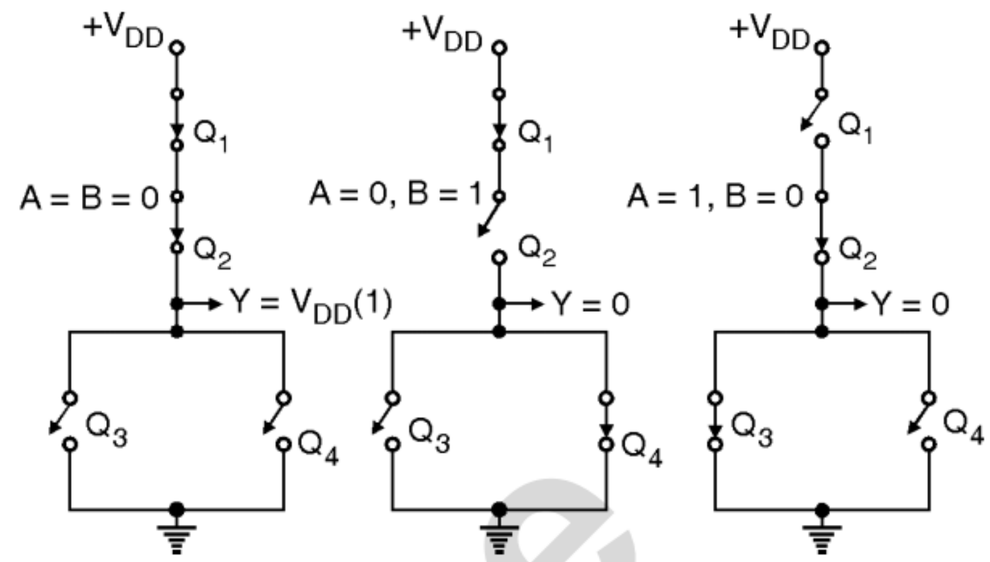
2. With A = 0, B = 1 :

– If A = 0 and B = 1, then Q_1 will remain ON, but Q_2 will turn OFF.

– Q_3 will remain OFF but Q_4 will be ON.

– The equivalent circuit for this mode is shown in Fig. 9.12.4(b) which shows that output Y = 0 Volt (logical 0).

– Thus Y = 0 for A = 0 and B = 1.



(a) For A = B = 0 (b) For A = 0, B = 1 (c) For A = 1, B = 0

(C-1056) **Fig. 9.12.4 : Equivalent circuits**

3. With A = 1, B = 0 :

– If A = 1 and B = 0 then Q_1 will be OFF and Q_2 will turn ON.

– Q_3 will be turned ON but Q_4 will turn OFF.

– Equivalent circuit of this mode is shown in Fig. 9.12.4(c) which shows that Y = 0 Volt (logical 0).

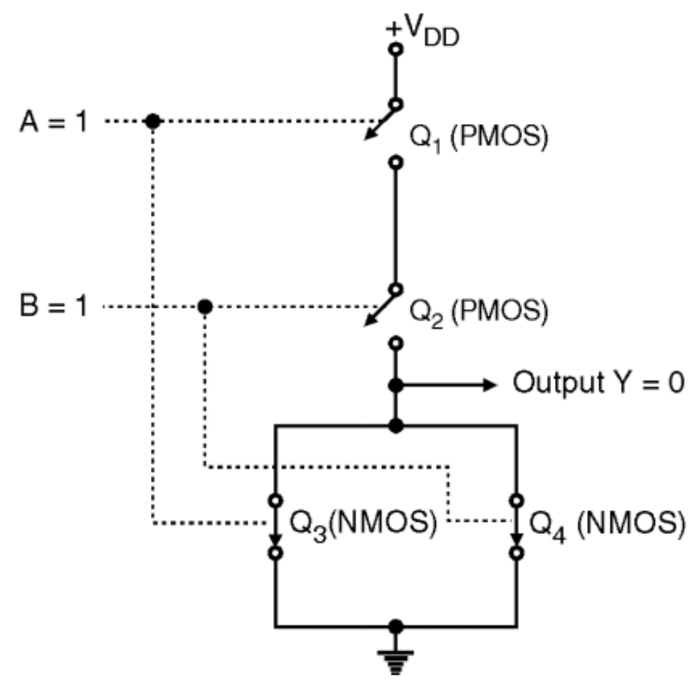
– Thus Y = 0 for A = 1, B = 0.

4. For A = B = 1 :

– If A and B both are high (1), then Q_1 and Q_2 both will be OFF.

– Q_3 and Q_4 both will be ON. Hence output Y = 0.

– The equivalent circuit for this mode is shown in Fig. 9.12.4(d).



(C-1057) **Fig. 9.12.4(d) : Equivalent circuit for A = B = 1**

Truth Table :

– Table 9.12.1 shows the truth table of the two input NOR gate.

TechKnowledge Publications

Table 9.12.1 : Truth Table

Inputs		Transistors				Output
A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	ON	ON	OFF	OFF	V_{DD} (1)
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	0
1	1	OFF	OFF	ON	ON	0

9.12.3 CMOS NAND Gate :

SPPU : Dec. 11, Dec. 12, May 14, Dec. 14

University Questions

Q. 1 Draw the structure of two input CMOS NAND gate. Explain its working. **(Dec. 11, 4 Marks)**

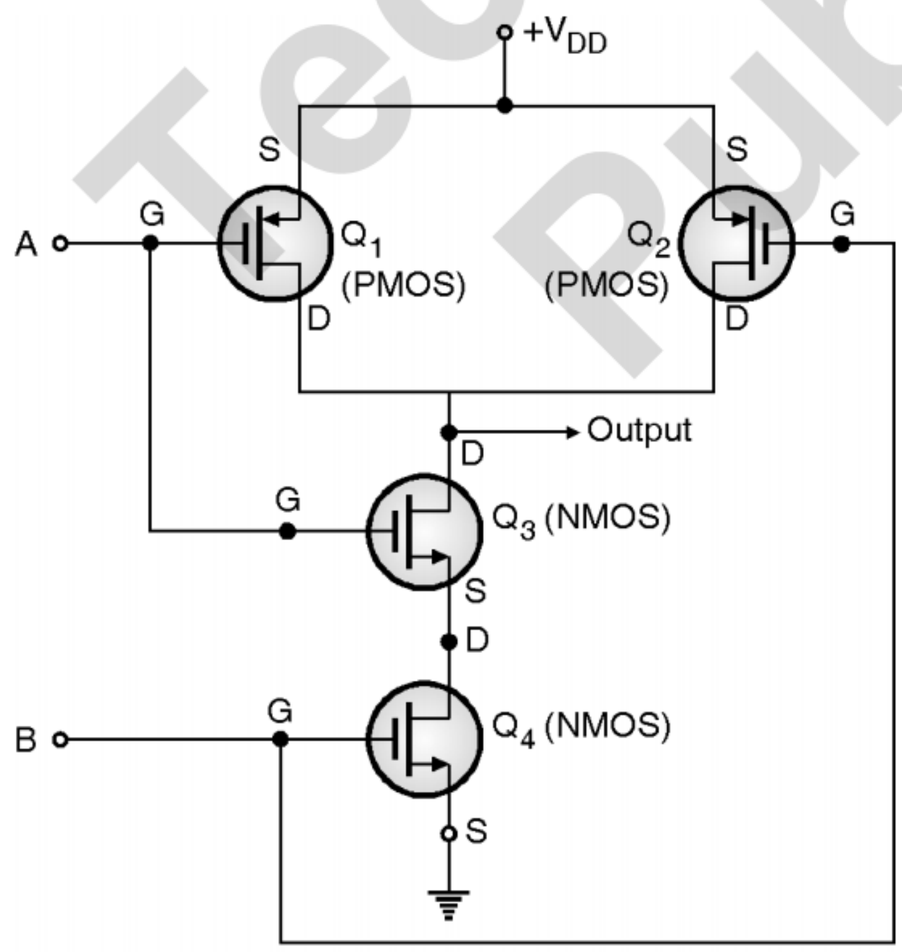
Q. 2 Draw and explain the working of a 2-input CMOS NAND gate. **(Dec. 12, 8 Marks)**

Q. 3 Explain with diagram 2 input CMOS NAND gate. **(May 14, 6 Marks)**

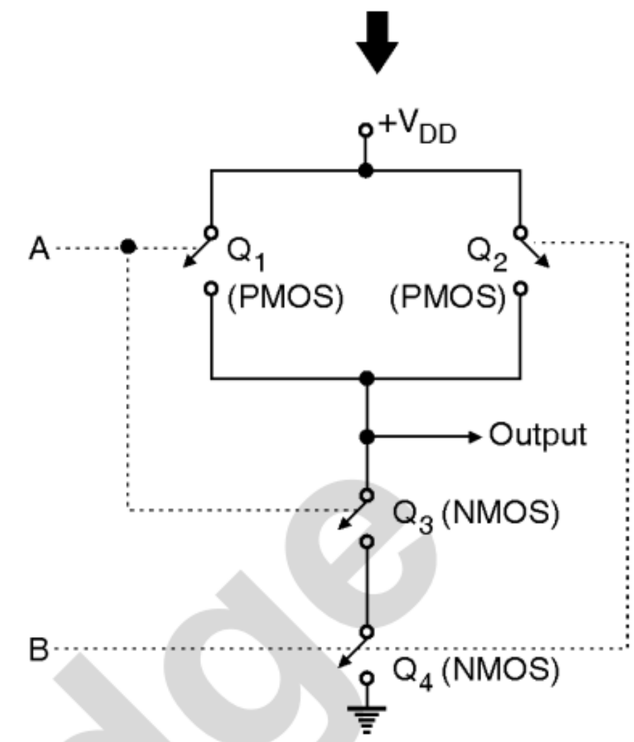
Q. 4 Explain with neat diagram two input CMOS NAND gate. **(Dec. 14, 6 Marks)**

Circuit diagram :

– The two input CMOS NAND gate is shown in Fig. 9.12.5(a) and its equivalent circuit by replacing each MOSFET by a switch is shown in Fig. 9.12.5(b).



(a) 2-input CMOS NAND gate



(b) Equivalent circuit

(C-1058) **Fig. 9.12.5**

– Q_1 and Q_2 are p-channel MOSFETs. They are connected in parallel with each other.

– Q_3 and Q_4 are n-channel MOSFETs. They are connected in series with each other.

– Input A is connected to gates of Q_1 and Q_3. So A controls the status of MOSFETs Q_1 and Q_3.

– Input B is connected to gates of Q_2 and Q_4. So B controls the status of MOSFETs Q_2 and Q_4.

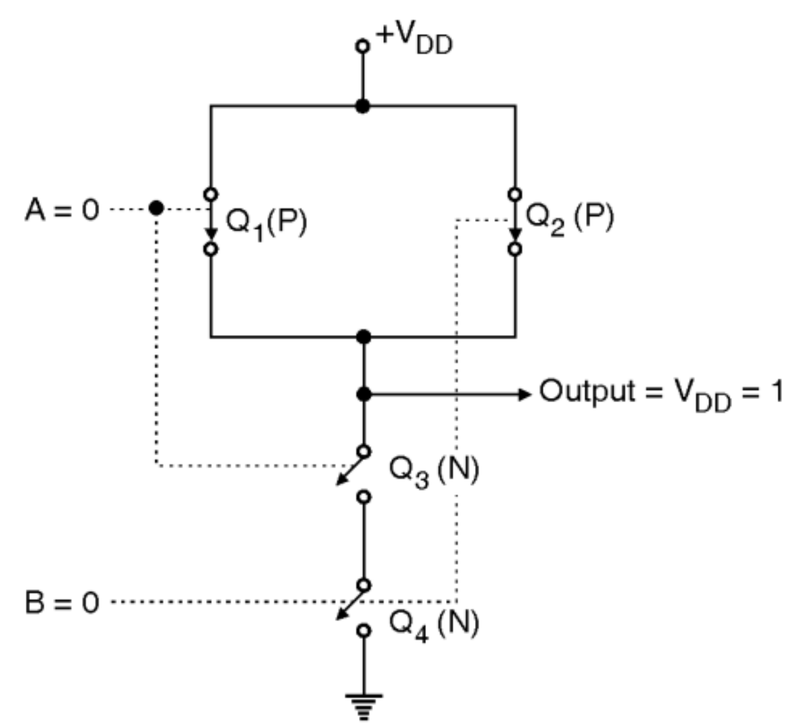
Operation :

1. A = B = 0 :

– With A = 0 and B = 0, both the PMOSFETs i.e. Q_1 and Q_2 will be ON. But both the N-MOSFETs i.e. Q_3 and Q_4 will be OFF.

– As seen from the equivalent circuit of Fig. 9.12.6(a), the output Y = + V_{DD} (logic 1).

– So Y = 1 if A = B = 0.



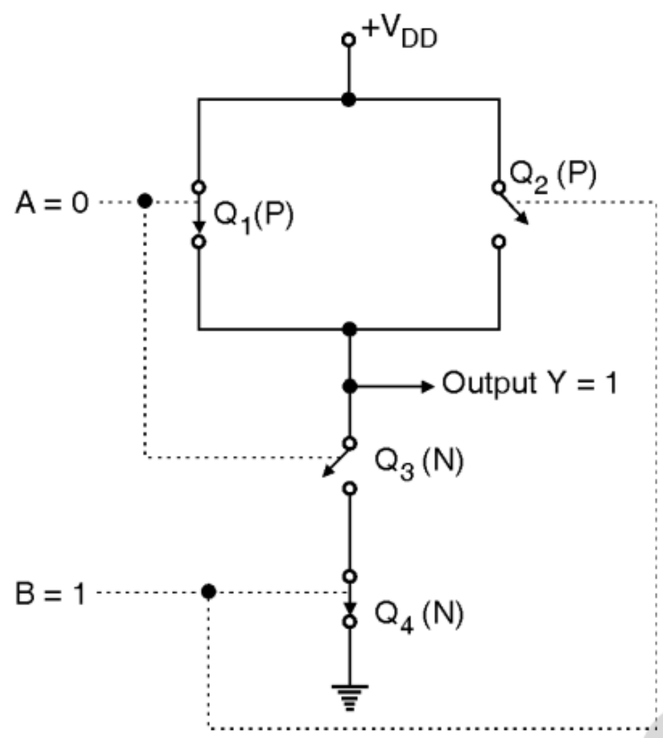
(C-1059) **Fig. 9.12.6(a) : Equivalent circuit for A = B = 0**

TechKnowledge Publications

2. With A = 0 and B = 1 :

– With A = 0 and B = 1, Q_1 will continue to be ON and Q_3 continues to be OFF. But Q_2 will now turn OFF and Q_4 will be turned ON.

– The equivalent circuit of this mode is shown in Fig. 9.12.6(b) which shows that output Y = + V_{DD} i.e. logic 1. So Y = 1 if A = 0 and B = 1.



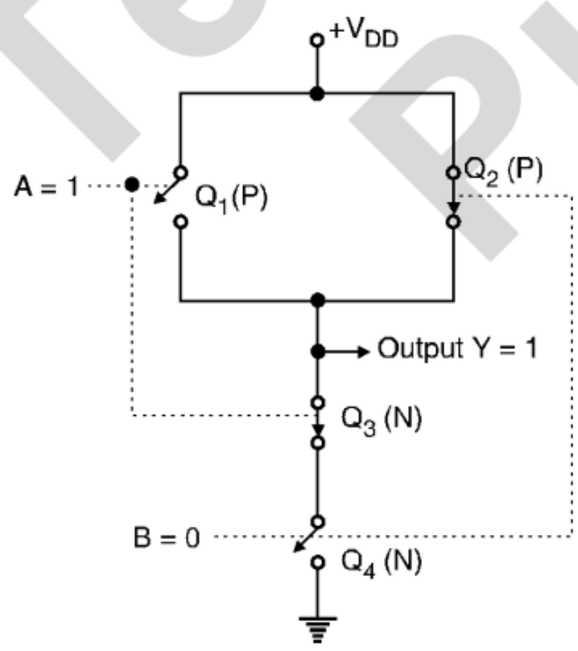
(C-1059) **Fig. 9.12.6(b) : Equivalent circuit for A = 0, B = 1**

3. With A = 1 and B = 0 :

– With A = 1, Q_1 will be turned OFF and Q_3 will turn ON.

– And with B = 0, Q_2 will be turned ON and Q_4 will be turned OFF.

– As seen from the equivalent circuit of Fig. 9.12.6(c), the output Y = + V_{DD} (logic 1).



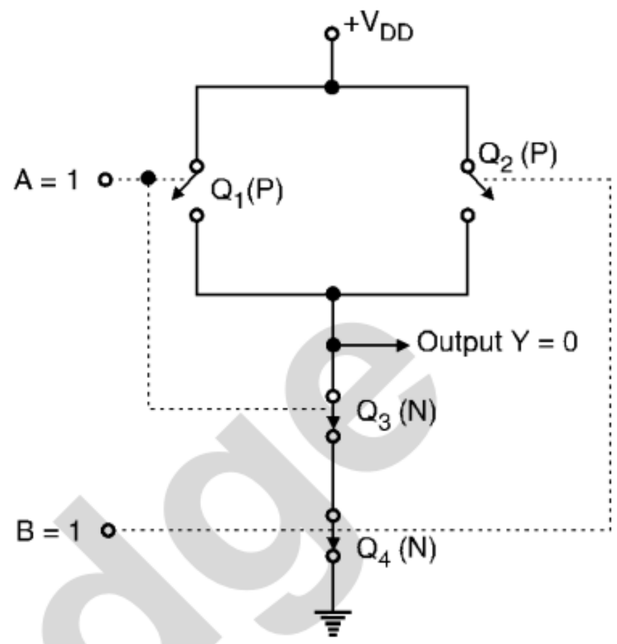
(C-1059) **Fig. 9.12.6(c) : Equivalent circuit for A = 1, B = 0**

– So Y = 1 if A = 1 and B = 0.

4. With A = 1, B = 1 :

– With A = B = 1 both P-MOSFETs i.e. Q_1 and Q_2 will be OFF and both the N-MOSFETs i.e. Q_3 and Q_4 will be ON.

– The equivalent circuit of this mode is shown in Fig. 9.12.6(d).



(C-1060) **Fig. 9.12.6(d) : Equivalent circuit for A = 1, B = 1**

– It shows that output Y = 0 (LOW).

So Y = 0 if A = B = 1.

Truth Table :

– Table 9.12.2 shows the truth table of the two input NAND gate.

Table 9.12.2 : Truth Table of a CMOS NAND gate

Inputs		Transistors				Output
A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

9.12.4 CMOS Series :

– The popular CMOS series are 4000/14000 series, 74C series, 74 HC/HCT (High speed CMOS), 74 AC/ACT (Advanced CMOS).

9.13 CMOS Characteristics :

Some of the important CMOS characteristics are as follows :

9.13.1 Power Supply Voltage :

– The 4000/14000 series and 74C series CMOS ICs are capable of operating over a wide range of power supply voltage (typically 3 V to 15 V).

– That means the CMOS ICs from these series are extremely versatile. They can operate on 3 V batteries as well as 5 V TTL compatible power supplies.

TechKnowledge Publications

– These ICs can operate on higher power supply voltages, when higher noise margin is required.

– The other CMOS families such as 74 HC/HCT, 74 AC/ACT and 74 AHC/AHCT operate over a voltage range of 2 to 6 V.

9.13.2 Logic Voltage Levels :

– The logic voltage levels for different CMOS series have different values. Table 9.13.1 lists various logic voltage levels for different CMOS series.

Table 9.13.1 : Logic voltage levels (in volts) with $V_{DD} = V_{CC} = +5$ V

Para-meter	CMOS							TTL			
	4000B	74 HC	74 HCT	74 AC	74 ACT	74 AHC	74 AHCT	74	74 LS	74 AS	74 ALS
$V_{IH(min)}$	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

Important points from Table 9.13.1 :

1. All the voltage levels shown in Table 9.13.1 correspond to a supply voltage of + 5 Volts.
2. Note that $V_{OH} \approx 5$ V and $V_{OL} \approx 0$ V.

9.13.3 Noise Margins :

SPPU : May 10, Dec. 16, May 18

University Questions

Q. 1 Describe what happens to the following CMOS characteristic as V_{DD} is increased : Noise margin and in which applications is CMOS ideally suited ? **(May 10, 3 Marks)**

Q. 2 State the following characteristics of digital TTL and CMOS ICs : Noise immunity. **(Dec. 16, 2 Marks)**

Q. 3 State the following characteristics of digital IC logic family TTL and CMOS : Noise margin. **(May 18, 2 Marks)**

– Table 9.13.1 contains the high level and low level noise margins V_{NH} and V_{NL} for each CMOS and TTL series.

– These are calculated as :

$$V_{NH} = V_{OH(min)} - V_{IH(min)},$$

$$V_{NL} = V_{IL(max)} - V_{OL(max)}.$$

– It can be observed that the CMOS devices will have higher noise margins as compared to those of TTL. So CMOS ICs should be preferred to TTL for operation in the noisy environment.

– The noise margins in CMOS will increase further if the supply voltages are increased further.

9.13.4 Power Dissipation :

SPPU : May 06

University Questions

Q. 1 Define power dissipation and give typical values of this parameters with respect to CMOS logic family. **(May 06, 2 Marks)**

– The power dissipation of CMOS devices is extremely low when they are in the static (stable) state.

– Typically the dc power dissipation is 2.5 nW per gate when $V_{DD} = 5$ V and 10 nW for $V_{DD} = 10$ V. This is too small as compared to TTL gates ($P_D = 10$ mW).

– Hence CMOS devices are preferred for the battery operated systems.

– But power dissipation is low under the dc operating conditions and at low frequencies.

– But power dissipation increases as the circuit switching frequency is increased.

– The relation between power dissipation and frequency is demonstrated in Table 9.13.2.

Table 9.13.2 : Relation between P_D and frequency

Frequency	0 (dc)	100 kHz	1 MHz
Power dissipation P_D	10 nW	0.1 mW	1 mW

9.13.5 Fan Out :

SPPU : May 06, May 18

University Questions

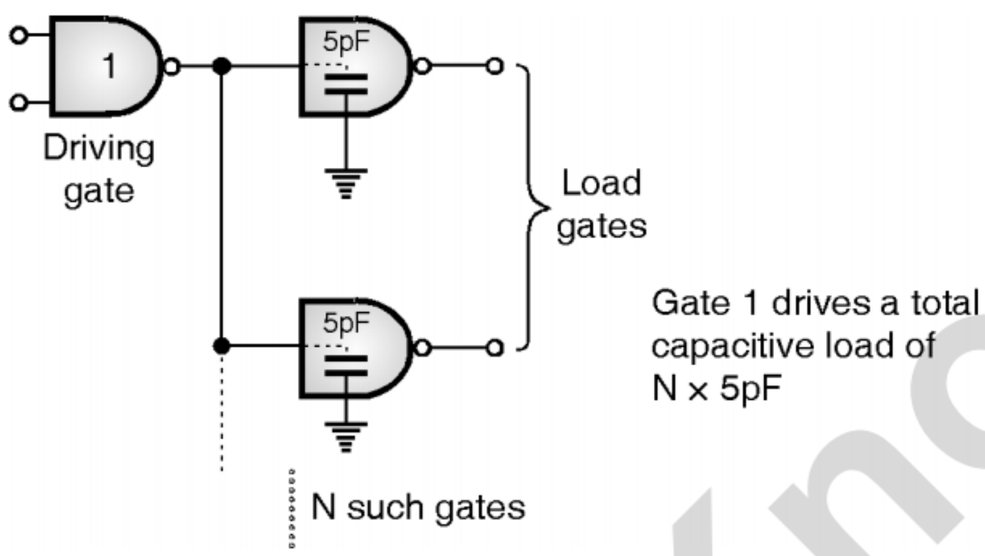
Q. 1 Define fan out and give typical values of this parameters with respect to CMOS logic family. **(May 06, 2 Marks)**

Q. 2 State the following characteristics of digital IC logic family TTL and CMOS : FAN out. **(May 18, 2 Marks)**

– The input resistance of CMOS devices is very high (10^{12} Ω). So their input current is very very small, almost zero.

Tech Knowledge Publications

- Therefore one CMOS gate can drive a large number of other CMOS gates. Hence fan out of CMOS devices will be large as compared to fan out of TTL.
- Typically the fan out is restricted to 50 for operation below 1 MHz.
- Why is fan out restricted to only 50 ? This is because the input capacitance of each CMOS gate is about 5 pF. These capacitors act as load on the driving gate as shown in Fig. 9.13.1.
- The charging current for these capacitor has to be supplied by the driving gate. This current should not be too large. This will limit the fan-out to 50.



(C-1061) Fig. 9.13.1 : Fan out and switching speed are dependent on the input capacitance of load gates

9.13.6 Switching Speed :

SPPU : May 06, Dec. 16

University Questions

Q. 1 Define speed of operation and give typical values of this parameters with respect to CMOS logic family. **(May 06, 2 Marks)**

Q. 2 State the following characteristics of digital TTL and CMOS ICs : Speed of operation. **(Dec. 16, 2 Marks)**

- The output resistance of CMOS is low in both the states (0 or 1) of output.
- So eventhough it has to drive large capacitive loads, the switching speed can still be faster than the NMOS or PMOS devices.
- The NAND gate of 4000 series has following values of propagation delays :

  Average t_{pd} = 50 nS ..... at V_{DD} = 5 V

  Average t_{pd} = 25 nS ..... at V_{DD} = 10 V
- The average time delay for various CMOS ICs are given in Table 9.13.3.

(C-6352) Table 9.13.3 : Switching speeds for various CMOS ICs at V_{DD} = 5 V

Family	4000	74 HC/HCT	74 AC/ACT	74 AHC
Propagation delay t_{pd}	50nS	8nS	4.7nS	4.3nS

9.13.7 Unconnected Inputs :

- Note that the CMOS inputs should never be left floating.
- All the CMOS inputs should be either connected to 0 V (ground) or V_{DD}, or to another inputs.
- This is necessary to avoid permanent damage of CMOS ICs.
- Sometimes there are some unused gates on a chip. The inputs of such gates also should be connected to ground or + V_{DD}.
- The CMOS ICs are damaged due to induced voltages at the floating inputs due to noise or static charges. Such voltages can bias the P-MOS or N-MOS in their conduction state that may cause overheating and damage.

9.14 Open Drain Outputs (CMOS) :

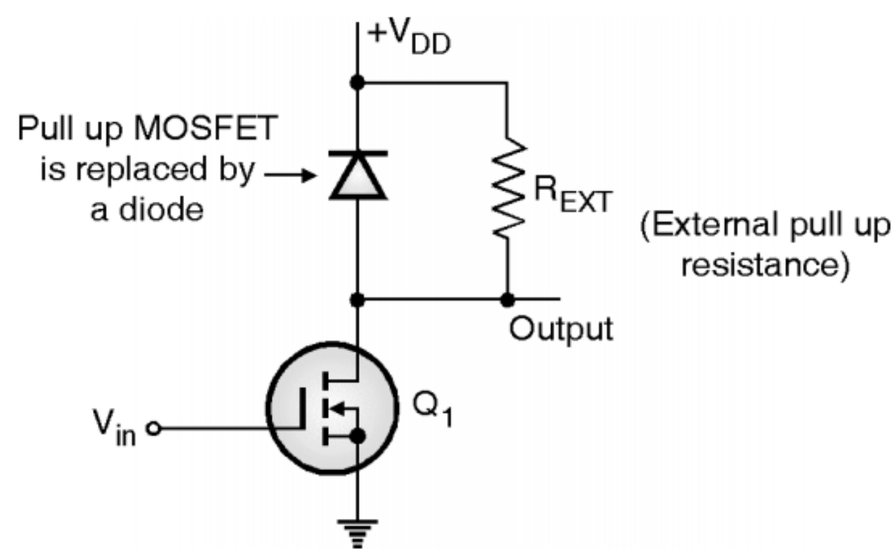
SPPU : May 12

University Questions

Q. 1 List differences between open drain and wired logic CMOS. **(May 12, 4 Marks)**

Circuit diagram :

- We have already seen the significance of open collector output for the TTL family.
- The same principle is extended to the CMOS family, as well. Such CMOS gates are known as the open drain outputs.
- Fig. 9.14.1 shows the circuit diagram of an open drain CMOS gate.



(C-1062) Fig. 9.14.1 : Open drain output

TechKnowledge Publications

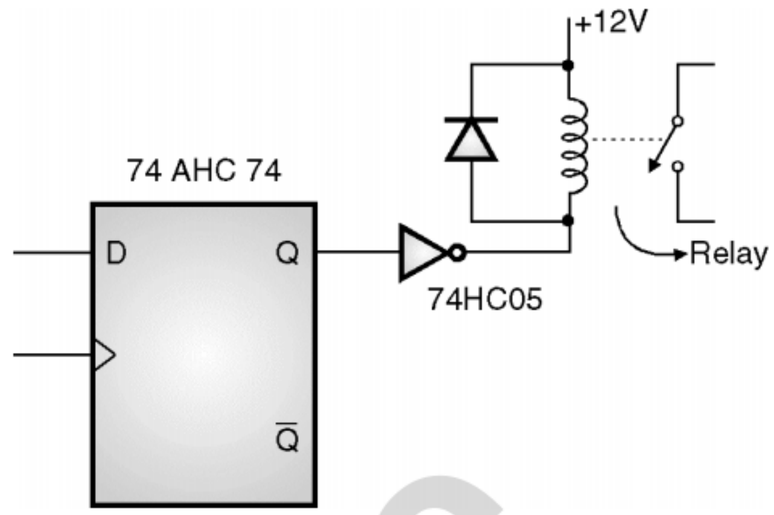
- The outputs of conventional CMOS outputs should never be connected together.
- In the common drain output type CMOS IC, the output is taken at the drain of the N-channel pull-down MOSFET. The drain terminal is left open (not connected to any other circuitry).
- The P-channel pull-up MOSFET is replaced by a diode as shown in Fig. 9.14.1.
- It is possible to wire AND the CMOS ICs having open drain outputs. An external pull up resistance is required to be connected between the open drain and supply voltage + V_{DD}, of all the CMOS ICs that are being wire ANDed.
- Note that the open collector or open drain circuit cannot actively force their output HIGH (1). They are capable of forcing the output LOW (0).
- The pull up resistance R_{EXT} will decide the charging current and the charging time constant for the capacitive load.
- As the value of R_{EXT} is in kΩ, the charging time constant will be large. This makes the open drain circuits slow.
- So the switching speed of open drain CMOS devices is always less than the conventional CMOS devices.

Applications of Open Drain Output :

- The open drain output CMOS are used as buffer/driver.
- A buffer or driver is a logic circuit which can supply a larger voltage or current than the ordinary logic gate.
- So if we connect a buffer after an ordinary logic gate, then the gate becomes capable of sourcing higher currents.

9.14.1 Applications of Open Drain Inverter :

- The 74 HC 05 is an open-drain hex inverter. It has a current sinking capability of 25 mA.
- Fig. 9.14.2 shows how to drive a control relay using 74 AHC 74 D-FF. This D-FF is interfaced with 74 HC 05 NOR inverter.



(C-1063) **Fig. 9.14.2 : Application of open drain inverter**

9.15 Wired Logic (CMOS) :

SPPU : Dec. 04, May 17, May 18

University Questions

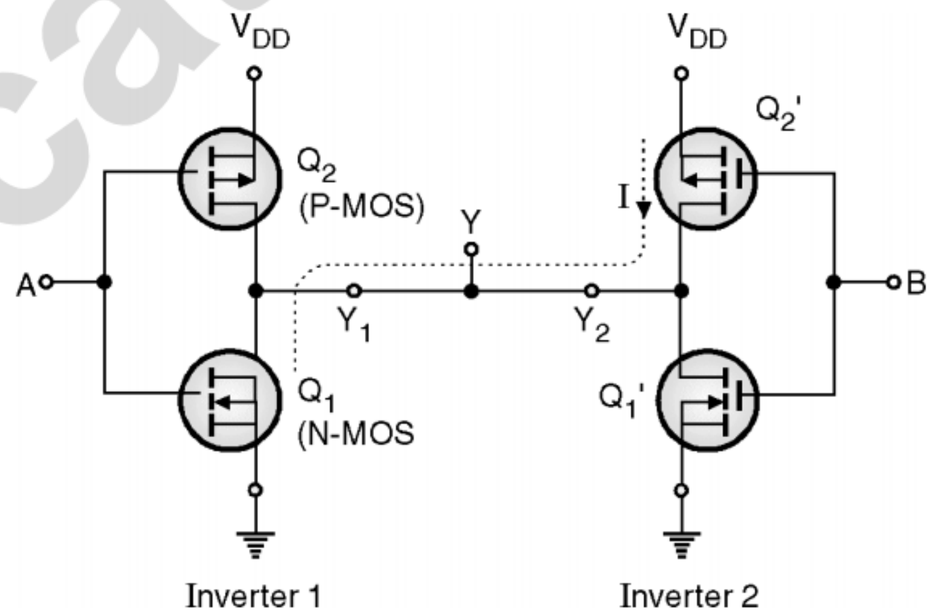
Q. 1 Comment about the wired logic capability of the CMOS gate. **(Dec. 04, 4 Marks)**

Q. 2 Why wired logic is not possible in CMOS operation. **(May 17, 4 Marks)**

Q. 3 Explain wired logic in CMOS. **(May 18, 4 Marks)**

Circuit diagram :

- Fig. 9.15.1 shows the two CMOS inverters with their outputs connected together.



(C-1064) **Fig. 9.15.1 : CMOS inverters with outputs connected**

Operation :

- When A = B = 0, Q_1 and Q_1' are turned off. Hence Y = 1.
- When A = B = 1, Q_1 and Q_1' are on and Y = 0.
- But the problem is when A = 1 and B = 0 as shown in Fig. 9.15.1, Q_1 and Q_2' conduct simultaneously and a large current I flows through them. This large current will damage Q_2' and Q_1.
- In addition to this the output voltage $Y = \frac{V_{DD}}{2}$ which is neither 0 nor 1. So the circuit will not operate properly.
- Similarly for A = 0 and B = 1, Q_2 and Q_1' will conduct a large current. The circuit will not operate properly.

Conclusion :

– The wired logic should not be used for CMOS logic circuits.

9.15.1 Advantages of CMOS : SPPU : Dec. 07

University Questions

Q. 1 What are the advantages of CMOS devices over TTL devices ? Explain in short. **(Dec. 07, 4 Marks)**

1. Low power dissipation.
2. High fan out (typically 50).
3. High noise margin for higher values of V_{DD}.
4. Capable of working over a wide range of supply voltage.
5. Switching speeds comparable to those of TTL.
6. High packaging density since (more devices can be accommodated in the same space) MOS devices need less space.

9.15.2 Disadvantages of CMOS :

1. Propagation delays longer than those of TTL (25 to 100 nS).
2. Slower than TTL.
3. CMOS ICs can get damaged due to static charge.
4. Latch ups can take place which can damage the device.
5. Need protection circuitry.

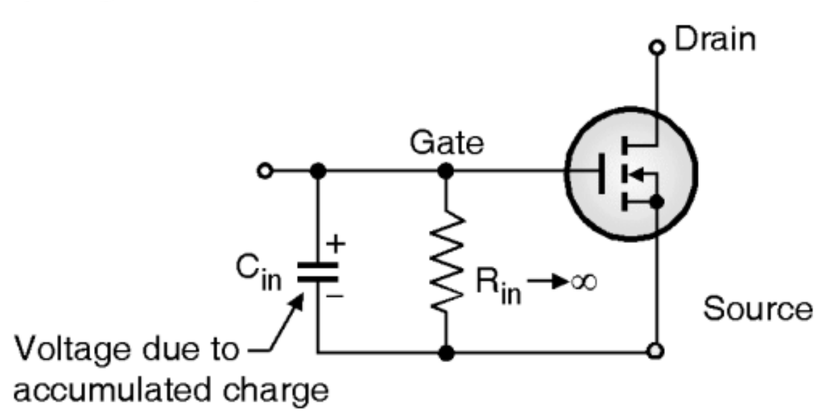
9.15.3 Handling Precautions for CMOS Devices : SPPU : Dec. 04

University Questions

Q. 1 What precautions should be taken while handling CMOS ICS ? Why ? **(Dec. 04, 4 Marks)**

– All the MOS devices are likely to get damaged due to Electrostatic Discharge (ESD).

– We know that the gate terminal of a MOSFET is insulated from the channel due to inclusion of SiO_2 layer. This will make the input resistance of the device very large (ideally infinite).



(C-1066) Fig. 9.15.2 : Effect of input capacitance

– The gate leakage current of the MOSFET (I_{GSS}) is of the order of few pico amperes whereas the gate reverse current for a JFET is of the order of few nA.

– Input capacitance of a MOSFET is large due to its isolated gate structure. This capacitance can accumulate excess static charge.

– If the voltage across the input capacitance, due to accumulated static charge is higher than the breakdown voltage of gate then the MOSFET will get damaged.

– To avoid the damage due to ESD, the user has to take certain precautions while handling the MOS devices.

– They are as follows :

1. Store the MOS devices in a conductive foam.
2. All the instruments, metal benches used in assembly or testing should be connected to earth.
3. The assembler's or handler's wrist should be connected to earth via a long wire and a high value series resistor.
4. Do not remove a MOS device from the circuit when the power is ON.
5. Do not apply signal to a MOS device when the dc voltage applied to it is zero.

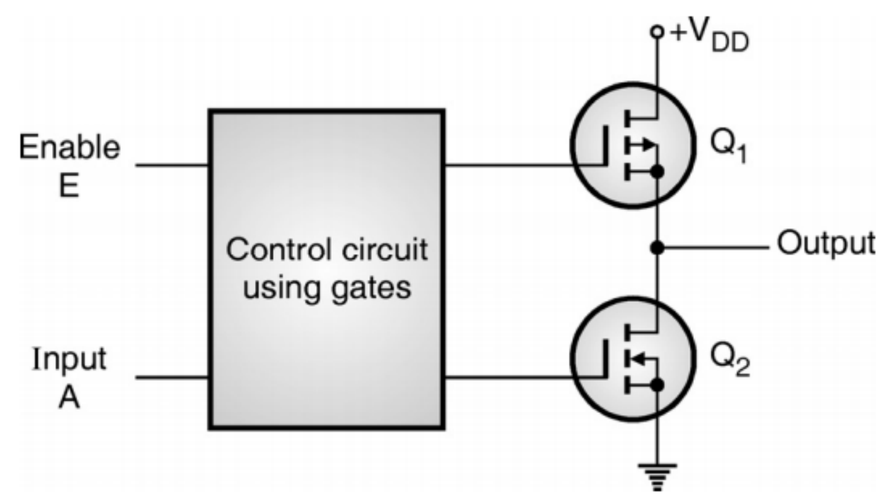
9.15.4 Tristate Logic Output : SPPU : May 17

University Questions

Q. 1 Explain tristate logic. **(May 17, 5 Marks)**

Circuit diagram :

– Fig. 9.15.3 shows an inverting driver alongwith logic gates.



(C-1065) Fig. 9.15.3 : A tristate CMOS inverter

– When both the MOSFETs are in the OFF state, the output is in its high impedance state.

– When the enable input E = 1, the inverter operates in a normal way. Either Q_1 is ON or Q_2 is ON, depending on state of the input A.

– But if E = 0, then neither Q_1 nor Q_2 is ON. Then irrespective of input A, both the MOSFETs will be OFF and the inverter will go into the high impedance state.

9.16 Interfacing :

– The meaning of word interfacing is to connect one or more outputs of one system to the one or more inputs of another system or circuits which have different electrical characteristics.

– If the electrical characteristics of the two circuits or systems are different then we cannot have a direct connection between them.

– So an "interface" circuit is required to be inserted between the "driver" circuit and the "load" circuit as shown in Fig. 9.16.1.

(C-1067) Fig. 9.16.1 : Interfacing between two different systems

– The task of the interface circuit is to accept the output of the driver circuit and "condition" it so that it becomes compatible with the load circuit.

– ICs from the same family are designed in such a way that they can be connected together without any special considerations.

– But when we connect the output of one type of IC to the input of another type of IC, we have to be concerned about the differences related to the voltage and current levels of both the ICs.

9.16.1 TTL to CMOS Interfacing :

SPPU : Dec. 08, May 12, Dec. 18

University Questions

Q. 1 Draw and explain the TTL to CMOS and CMOS to TTL interfacing. **(Dec. 08, 8 Marks)**

Q. 2 Which parameters are significant while interfacing TTL and CMOS ? **(May 12, 4 Marks)**

Q. 3 Explain the interfacing of TTL and CMOS : TTL driving CMOS. **(Dec. 18, 4 Marks)**

– When we interface different types of ICs, it is necessary to check whether the driving IC is capable of meeting the current and voltage requirements of the load IC or not.

– For example Table 9.16.1 reveals that the output current capability of TTL ICs is much higher than the input current values of CMOS ICs.

– Therefore there is no problem for a TTL IC to drive CMOS as far as current is concerned.

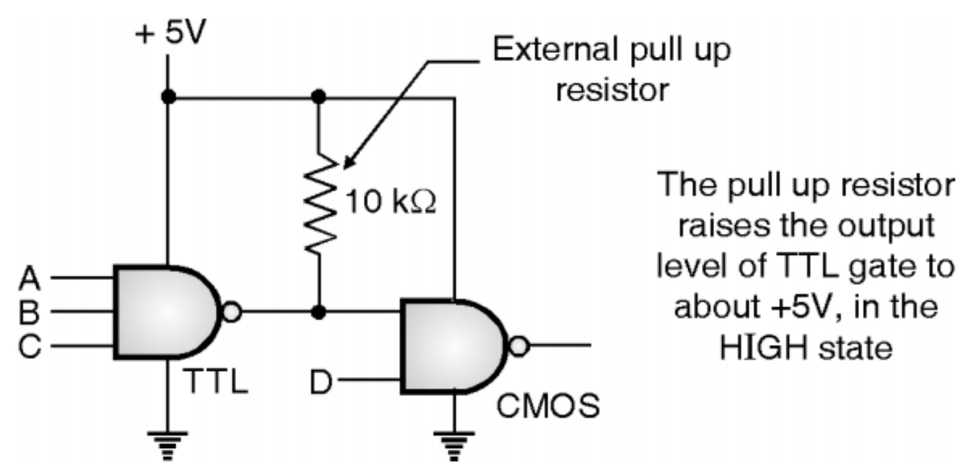
– But there is a problem when we compare the voltage levels of TTL and CMOS. Because $V_{OH\ (min)}$ of a TTL series is very low as compared with $V_{IH\ (min)}$ required for the CMOS series like 400B, 74HC or 74AC.

– In such situations, something has to done to increase the level of TTL output voltage to an acceptable voltage level for CMOS.

(C-6353) Table 9.16.1 : Input output currents for standard devices with a supply voltage of 5 V

Parameter	CMOS		TTL		
	4000 B	74HC/HCT	74	74 LS	74 AS
$I_{IH(max)}$	1 μA	1 μA	40 μA	20 μA	20 μA
$I_{IL(max)}$	1 μA	1 μA	1.6 mA	0.4 mA	0.5 mA
$I_{OH(max)}$	0.4 mA	4 mA	0.4 mA	0.4 mA	2 mA
$I_{OL(max)}$	0.4 mA	4 mA	16 mA	8 mA	20 mA

– The situation in which a TTL IC is driving CMOS is shown in Fig. 9.16.2.



(C-1068) Fig. 9.16.2 : TTL driving CMOS

– Note the presence of pull up resistor at the output of the TTL gate. Due to this resistor the TTL output will rise to approximately + 5 V in its HIGH state.

– This will provide the sufficient voltage level at the input of the CMOS gate.

– Such a pull up resistance is not required if the CMOS gate belongs to 74 HCT or 74 ACT family, because these families can accept the TTL outputs directly. In other words they are TTL compatible CMOS families.

Tech Knowledge Publications

9.16.2 TTL Driving High Voltage CMOS :

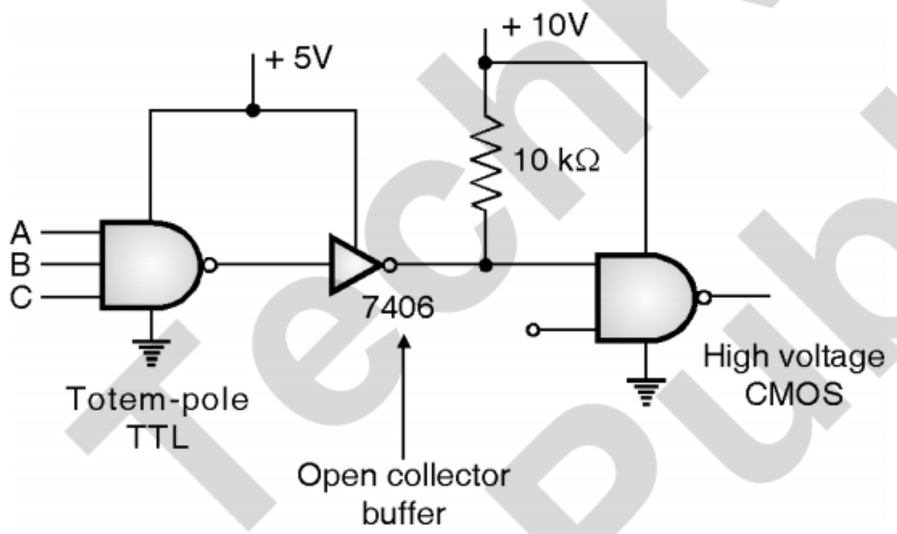
SPPU : May 05, May 12

University Questions

Q. 1 A TTL gate is supposed to drive CMOS gates, what arrangement has to be made to work this interface satisfactorily ? Justify your answer with suitable circuit diagram. **(May 05, 8 Marks)**

Q. 2 Draw and explain TTL driving CMOS gate. **(May 12, 4 Marks)**

- But if the CMOS IC is operated with a supply voltage V_{DD} greater than 5 V then the interfacing becomes a little bit difficult.
- For example with V_{DD} = 10 V, then CMOS input requires $V_{IH\,(min)}$ = 7 V.
- The output stages of many TTL devices cannot operate at a voltage higher than + 5 V. Therefore the technique of using the pull up resistance can no more be employed.
- The interfacing of TTL to a high voltage CMOS is shown in Fig. 9.16.3.

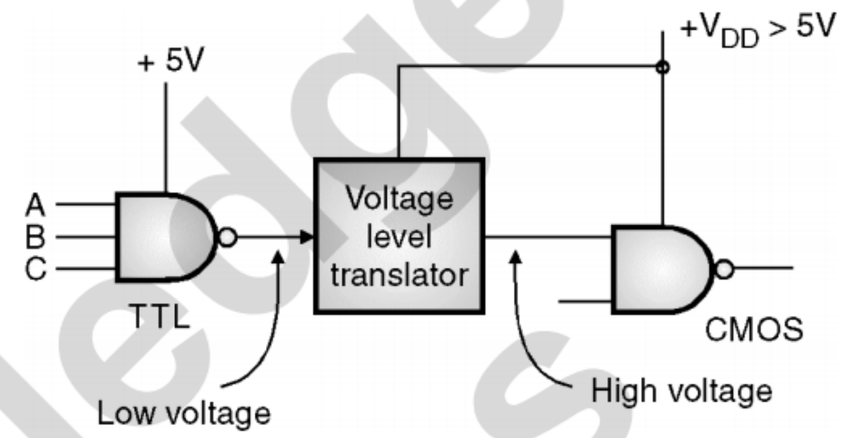


(C-1069) Fig. 9.16.3 : TTL to high voltage CMOS interfacing, using open collector buffer

- When it is not possible to pull up the totem-pole output to + V_{DD}, an open collector buffer is used as interface between TTL totem-pole and high voltage CMOS gate.
- IC 7406 is an inverting CMOS buffer.
- It has an output voltage rating of 30 V. So we can connect it as an interfacing buffer along with a pull up resistance of 10 kΩ is shown in Fig. 9.16.3.
- Note that the pull up resistance has been connected to the 10 Volts supply.
- Therefore the buffer output and therefore the CMOS input could reach 10 Volts level in its HIGH state. Thus the TTL can be successfully interfaced with a high voltage CMOS.

9.16.3 Interfacing using Level-Shifter (TTL to High Voltage CMOS) :

- Another way to interface TTL with high voltage CMOS is to use a voltage level translator (a level shifter) for interfacing as shown in Fig. 9.16.4.



(C-1070) Fig. 9.16.4 : Interfacing using level-shifter

- The voltage translator circuit accepts a low voltage from TTL and translates it into a high voltage output for CMOS IC inputs.
- Note that the supply voltage to the level shifter is the supply voltage of the CMOS IC.
- The example of level-shifter is IC 4504 B.

9.16.4 CMOS to TTL Interface :

SPPU : Dec. 08, Dec. 11, Dec. 18

University Questions

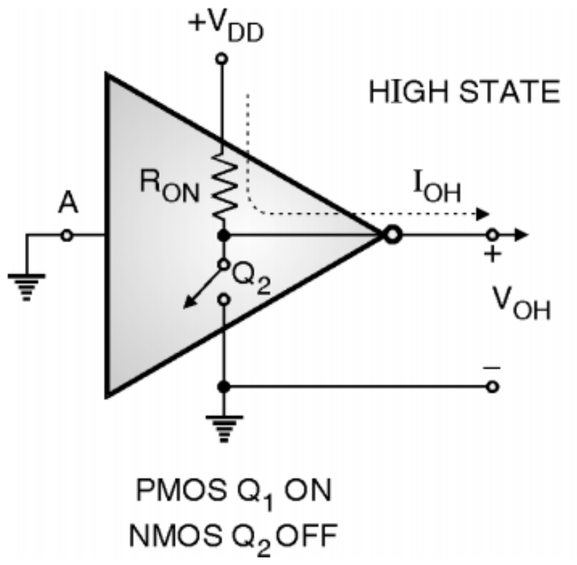
Q. 1 Draw and explain the TTL to CMOS and CMOS to TTL interfacing. **(Dec. 08, 8 Marks)**

Q. 2 Which specifications of logic families are significant in CMOS-TTL interfacing ? Explain the same when CMOS drives TTL. **(Dec. 11, 8 Marks)**

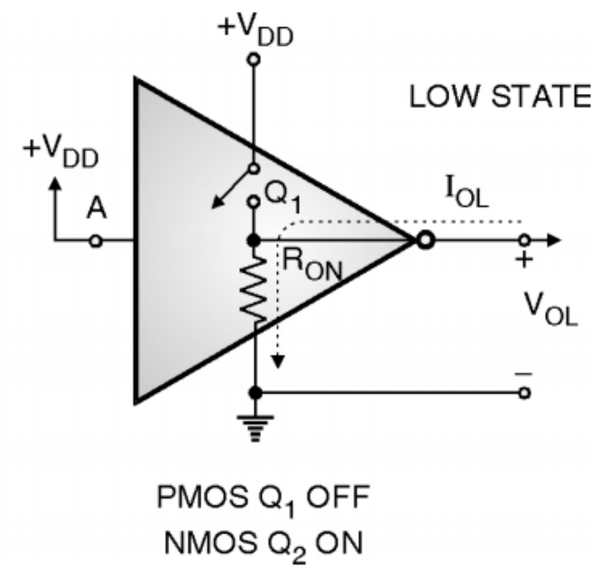
Q. 3 Explain the interfacing of TTL and CMOS : CMOS driving TTL. **(Dec. 18, 4 Marks)**

- Before we consider the problem of CMOS interfacing with TTL, let us review the output characteristics of CMOS in two different states.
- Fig. 9.16.5(a) shows the equivalent output circuit of a CMOS inverter in the HIGH state output, whereas Fig. 9.16.5(b) shows its equivalent in the LOW state output.

Tech Knowledge Publications



(a) Equivalent circuit of a CMOS inverter for HIGH state



(b) Equivalent circuit of a CMOS inverter for low state

(C-1071) Fig. 9.16.5

Equivalent circuit in HIGH state :

- In the HIGH state of output, the P-channel MOSFET Q_1 will be ON, so it has been replaced by the ON state resistance R_{ON}. Whereas the N-channel MOSFET Q_2 will be OFF. So it has been replaced by an open switch as shown in Fig. 9.16.5(a).
- Therefore the CMOS output is equivalent to a V_{DD} source with a source resistance of R_{ON}.

Equivalent circuit in LOW state :

- The equivalent circuit of a CMOS inverter in its low state of output is shown in Fig. 9.16.5(b). In this state, the P-MOSFET Q_1 is OFF and represented by an open switch.
- Whereas the N-channel MOSFET is ON. So it has been replaced by resistance R_{ON}. So the CMOS inverter now acts as a low resistance connected to ground and sinks current.
- Table 9.16.2 presents the various voltage and current levels for the standard CMOS (4000B) and TTL (74) series for comparison.

(C-6259) Table 9.16.2

Sr. No.	Parameters for driving gate (CMOS) of 4000 B series	Parameters for the load gate (TTL) of 74 series
1.	$V_{OH(min)} = 4.95V$	$V_{IH(min)} = 2V$
2.	$V_{OL(max)} = 0.05V$	$V_{IL(max)} = 0.8V$
3.	$I_{OH(max)} = 0.4mA$	$V_{IH(max)} = 40\mu A$
4.	$I_{OL(max)} = 0.4mA$	$I_{IL(max)} = 1.6mA$

For low state output

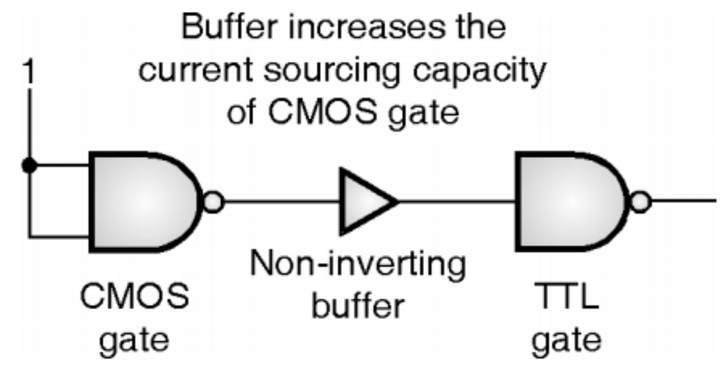
For high state output

9.16.5 CMOS Driving TTL in the HIGH State :

- Now let us understand what happens when a CMOS IC drives TTL. Let the CMOS output voltage be high (1). So consider the first and third parameters in Table 9.16.2 i.e. V_{OH} and I_{OH} for CMOS and V_{IH} and I_{IH} for TTL.
- Looking at these parameters we understand that the CMOS output can supply sufficient voltage as well as current to the TTL inputs.
- So there is no need of using any special interface for high state. We can directly connect CMOS output to TTL input.

9.16.6 CMOS Driving TTL in the LOW State :

- Now let the CMOS output be low (0). So consider the second and fourth parameters in Table 9.16.2 i.e. V_{OL}, I_{OL} for CMOS gate and V_{IL} and I_{IL} for TTL gate.
- Looking at these parameters we understand that the low state output voltage V_{OL} of CMOS can satisfy the V_{IL} requirement of TTL.
- But the low level output current of a CMOS gate (I_{OL}) is not sufficient to drive even a single TTL gate with $I_{IL\,(max)} = 1.6$ mA.
- Therefore to improve the current sourcing capability of CMOS we should use a non-inverting buffer between the CMOS and TTL gates as shown in Fig. 9.16.6.

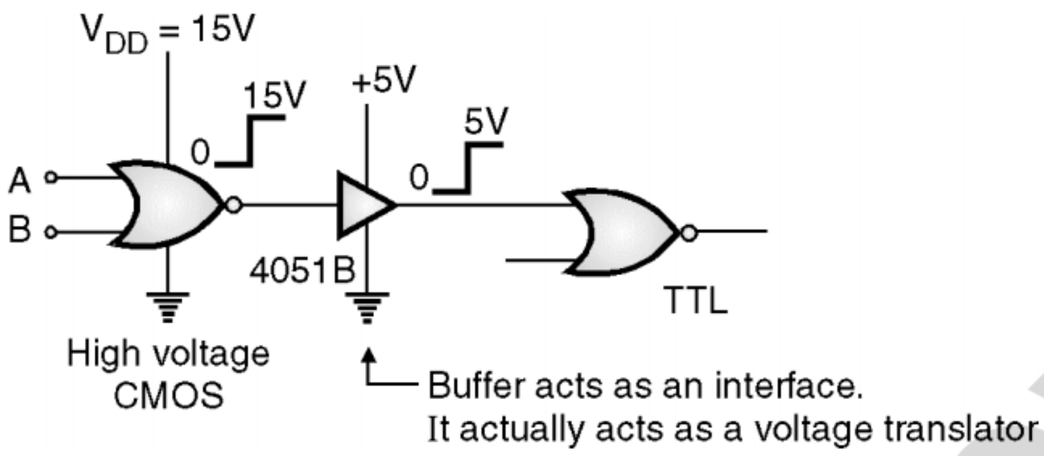


(C-1072) Fig. 9.16.6 : CMOS driving TTL in low state

TechKnowledge Publications

9.16.7 High Voltage CMOS Driving TTL :

- The high voltage CMOS gates operate at high supply voltages (typically 15 V). So their output voltage in the HIGH state will be $V_{OH\ (max)} \approx 15$ V.
- Almost all the TTL ICs have a maximum input voltage $V_{IH\ (max)} = 7$ V.
- Therefore an interface as shown in Fig. 9.16.7 will be required between high voltage CMOS gates and TTL. A level shifting buffer is used as an interface as shown.



(C-1073) Fig. 9.16.7 : High voltage CMOS driving TTL

- The buffer 4051B is connected to a + 5 V supply. It is a non-inverting buffer. So it converts a 15 V HIGH output of the CMOS gate into a 5 V input for the TTL gate.
- Thus the 4051B buffer acts like a level translator.

9.17 Comparison of CMOS and TTL :

SPPU : Dec. 09, Dec. 11, Dec. 12, Dec. 13, May 15, Dec. 16, May 19, Dec. 19

University Questions

Q. 1 Compare TTL and CMOS logic family. Also draw CMOS-NOR gate.

(Dec. 09, 6 Marks, Dec. 16, 7 Marks)

Q. 2 List difference between CMOS and TTL.

(Dec. 11, 4 Marks)

Q. 3 Differentiate between standard TTL and CMOS logic circuit with respect to :

1. Propagation delay 2. Fanout
3. Figure of merit **(Dec. 13, 6 Marks)**

Q. 4 List the differences between CMOS and TTL.

(May 15, 6 Marks)

Q. 5 Compare TTL and CMOS logic family.

(Dec. 19, 6 Marks)

Table 9.17.1 : Comparison of CMOS and TTL

Sr. No.	Parameter	CMOS	TTL
1.	Device used	N-channel MOSFET and P-channel MOSFET	Bipolar junction transistor
2.	$V_{IH\ (min)}$	3.5 V (V_{DD} = 5 V)	2 V
3.	$V_{IL\ (max)}$	1.5 V	0.8 V
4.	$V_{OH\ (min)}$	4.95 V	2.7 V
5.	$V_{OL\ (max)}$	0.05 V	0.4 V
6.	High level noise margin	V_{NH} = 1.45 V	0.4 V
7.	Low level noise margin	V_{NL} = 1.45 V	0.4 V
8.	Noise immunity	Better than TTL	Less than CMOS
9.	Propagation delay	105 nS (Metal gate CMOS)	10 nS. (Standard TTL)
10.	Switching speed	Less than TTL.	Faster than CMOS
11.	Power dissipation per gate.	P_D = 0.1 mW. Hence used for battery backup applications	10 mW
12.	Speed power product.	10.5 pJ	100 pJ
13.	Dependence of P_D on frequency	P_D increases with increase in frequency.	P_D does not depend on frequency.
14.	Fan out	Typically 50.	10
15.	Unconnected inputs	Unused inputs should be returned to GND or V_{DD}. They should never be left floating.	Inputs can remain floating. The floating inputs are treated as logic 1s.

TechKnowledge Publications

Sr. No.	Parameter	CMOS	TTL
16.	Component density	More than TTL since MOSFETs need smaller space while fabricating an IC.	Less than CMOS since BJT needs more space.
17.	Operating areas	MOSFETs are operated as switches. i.e. in the ohmic region or cut off region.	Transistors are operated in saturation or cut off regions.
18.	Power supply voltage	Flexible from 3 V to 15 V.	Fixed equal to 5 V.

Review Questions

Short Answer Questions :

Q. 1 Which are the different logic families ? Write their characteristics.

Q. 2 What is passive pull-up load and active pull-up load ?

Q. 3 Give the advantages of active pull-up over passive pull-up.

Q. 4 Explain the use of multi-emitter inputs.

Q. 5 Define the following terms regarding a logic family :

1. Noise margin 2. Propagation delay

Q. 6 Compare the performance of TTL, CMOS and ECL logic.

Q. 7 Explain the features of complementary symmetry logic (CMOS).

Q. 8 State the application of CMOS logic inverter.

Q. 9 What are the precautions taken while using CMOS ICs ?

Q. 10 Mention the advantages and disadvantages of TTL, CMOS and ECL IC families.

Q. 11 List any four characteristics of logic gates.

Q. 12 Compare between TTL and CMOS logic.

Q. 13 Classify the IC according to their scale of integration.

Q. 14 Explain the terms SSI, MSI, LSI, VLSI.

Q. 15 Explain what is meant by TTL ?

Q. 16 Draw the circuit diagram of two input TTL NAND gate and explain the function of each component in it.

Q. 17 Explain with diagram the working of High-Threshold Logic (HTL). State the advantage of HTL over DTL ?

Q. 18 Explain briefly the operation of TTL NAND gate in tri-state output configurations.

Q. 19 Give the important characteristics of logic families and explain their importance.

Q. 20 Explain 7400 TTL series. Sketch pin-outs of different ICs of 7400 series and state their functions.

Q. 21 State specifications of standard TTL family.

❑❑❑

TechKnowledge Publications

Unit VI

Chapter 10

Introduction to Computer Architecture

Technical Knowledge Publications

Syllabus

Introduction to ideal microprocessor – Data bus, Address bus, Control bus, Microprocessor based systems – Basic operation, Microprocessor operation, Block diagram of microprocessor, Functional units of microprocessor – ALU using IC 74181, Basic arithmetic operations using ALU IC 74181, 4-bit multiplier circuit using ALU and shift registers, Memory organization and operations, Digital circuit using decoder and registers for memory operations.

Exemplar / Case Study : Microprocessor based system in Communication / Instrumentation Control.

Chapter Contents

10.1	Introduction to Ideal Microprocessor – Data Bus, Address Bus, Control Bus	10.6	Basic Arithmetic operations using ALU IC 74181
10.2	Microprocessor Based Systems – Basic Operation	10.7	Memory Organization and Operations
10.3	Block Diagram of Microprocessor	10.8	Digital Circuit Using Decoder and Registers for Memory Operations
10.4	Functional Units of Microprocessor – ALU	10.9	Exemplar / Case Studies: Microprocessor Based System in Communication / Instrumentation Control
10.5	ALU using IC 74181		

10.1 Introduction to Ideal Microprocessor – Data Bus, Address Bus, Control Bus

– When we hear the word microprocessor, what comes to our mind is a small (i.e. micro) IC (integrated circuit) that can process data i.e. perform arithmetic and logical operations.

10.1.1 Some Basic Terms used in Microprocessors

1. Opcode :

– A binary code, that indicates the operation to be performed is called as an Opcode.

2. Operands :

– The data on which the operation is to be performed (as well as the result of an operation) are termed as operands.

3. Instruction :

– The combination of opcode and operands, that can be used to instruct a system, is called as an instruction.

4. Instruction set :

– A list of all the instructions that can be issued to a system, is called as instruction set of that system.

5. Program/subroutine/routine :

– A set of instructions written in a particular sequence, so as to implement a given task is called as a program. A subroutine in assembly language refers to function as of High level languages like C/C++.

– The term routine is used in a special case called as Interrupt Service Routine (ISR).

6. Bus :

– A group of lines, pins or signals having common function is termed as bus.

– The functional grouping of signals results in

(a) Data bus i.e. signals used to carry data.

(b) Address bus i.e. signals to address or select a memory or I/O location.

(c) Control bus are signals to issue and receive control operations.

7. Register :

– It is a flip-flop based circuit used to store the data or as the name says, register the data into it. It is as good as a PIPO (Parallel InParallel Out) register.

– These are present inside the processor to store data temporarily. These are few in count.

10.1.2 Microprocessor Characteristics and Functions

The computing power of the microprocessor can be determined by certain characteristics listed below :

1. The processor size or number of bits :

For example, processors are 8-bit, 16-bit, 32-bit, 64-bit etc. A processor is n-bit processor implies that,

(a) Its ALU is n-bit, i.e. the ALU can perform n-bit operation simultaneously.

(b) Its internal data bus and register size is n-bit.

(c) Its external data bus is also n-bit (in most of the cases).

2. Processing capability :

– It depends on the number of instructions supported by a processor.

– It also depends on the different types of data supported for execution by an instruction like binary, BCD, ASCII etc.

3. Clock frequency :

– The different operations that a microprocessor does on the external bus are memory read, memory write, input read and output write.

– The internal operations involve transfer of data between storage, arithmetic operations, logical operations etc.

– All these operations are synchronized with the clock. The frequency of this clock decides the speed of the processor.

4. Width of the address bus :

– The address bus width decide the number memory locations and I/O locations it can access.

– For a microprocessor, access refers to reading and writing a location.

5. Interrupt capability :

– The number of hardware and software interrupts for a processor depends on the number of interrupt pins and number of interrupt instructions respectively.

– Different types of interrupts like vectored/non-vectored, maskable/non-maskablehave their importance, we will study the same later in this chapter.

TechKnowledge Publications

10.1.2.1 Functions of a Microprocessor

- **Microprocessor** is an IC that can fetch, decode and execute an instruction.
- Thus microprocessor is a semiconductor chip, which can fetch or get the instruction from the memory.
- The instruction fetched is in binary form and hence, it decodes the instruction.
- After decoding, the operation as specified in the instruction is executed by the processor.
- The basic functions of a microprocessor are :
  1. Fetching instruction from the memory location pointed by the program counter. (Fetching means to bring)
  2. Decoding the instruction fetched. Since the instruction is coded in binary form, it is to be decoded to find out the operation to be performed.
  3. Executing the instruction. Once the instruction is decoded, the required operation is to be performed. The various operations performed by the processor are :

(a) Transferring a data from one location to another in the system. This could be transferring of data from memory to the processor, memory to the Input /Output device, from processor to memory etc.

(b) Performing different arithmetic operations like addition, subtraction, increment, decrement etc.

(c) Performing various logical operations like AND, OR, EXOR etc.

(d) Performing branching operations.

10.1.3 Buses and Memory Accessing

10.1.3.1 Address Bus

- A set of lines that are used to select a memory or I/O location forms a bus (group of lines) called as the **address bus**.
- The address lines are used to select the location of memory or input/output device to be read or to be written on.
- The number of locations accessible by a processor depends on the number of address lines. Thus if there are 'n' lines, the processor can access 2^n memory locations.
- As processor selects the location it wants to access, the address bus is a unidirectional bus (indicated by the arrow).

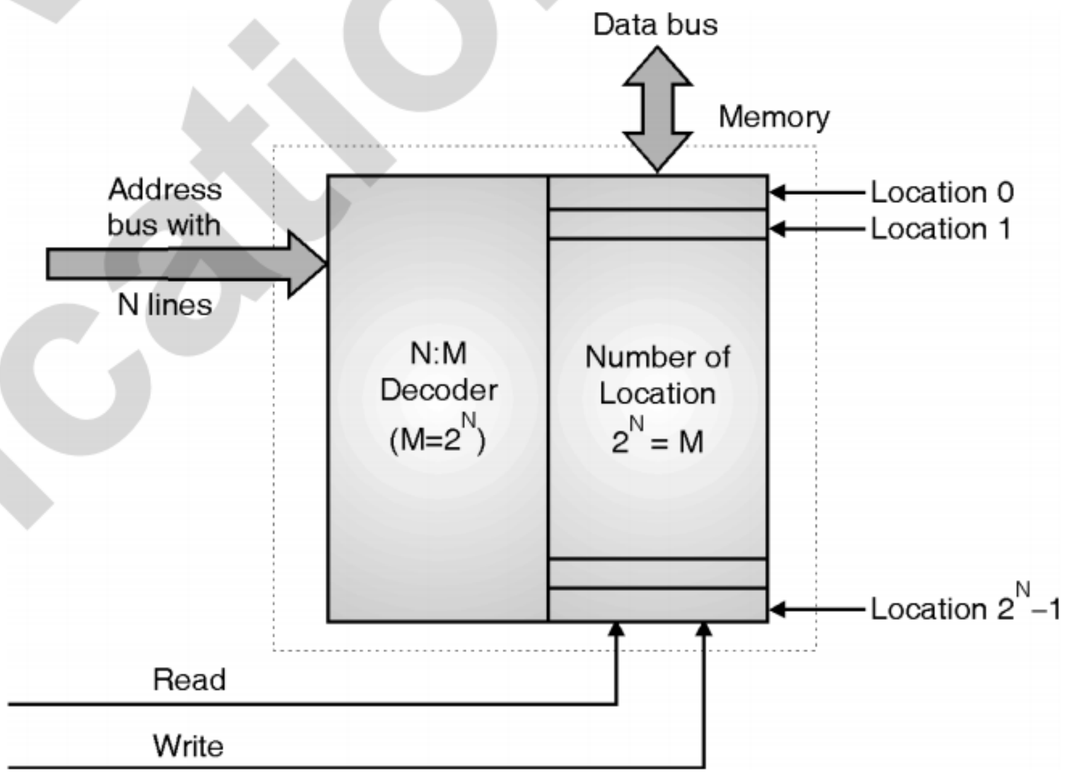


Fig. 10.1.1 : Structure of Memory

- If the number of address lines, N = 16 then it can address 2^{16} = 65,536 or 64K memory locations.

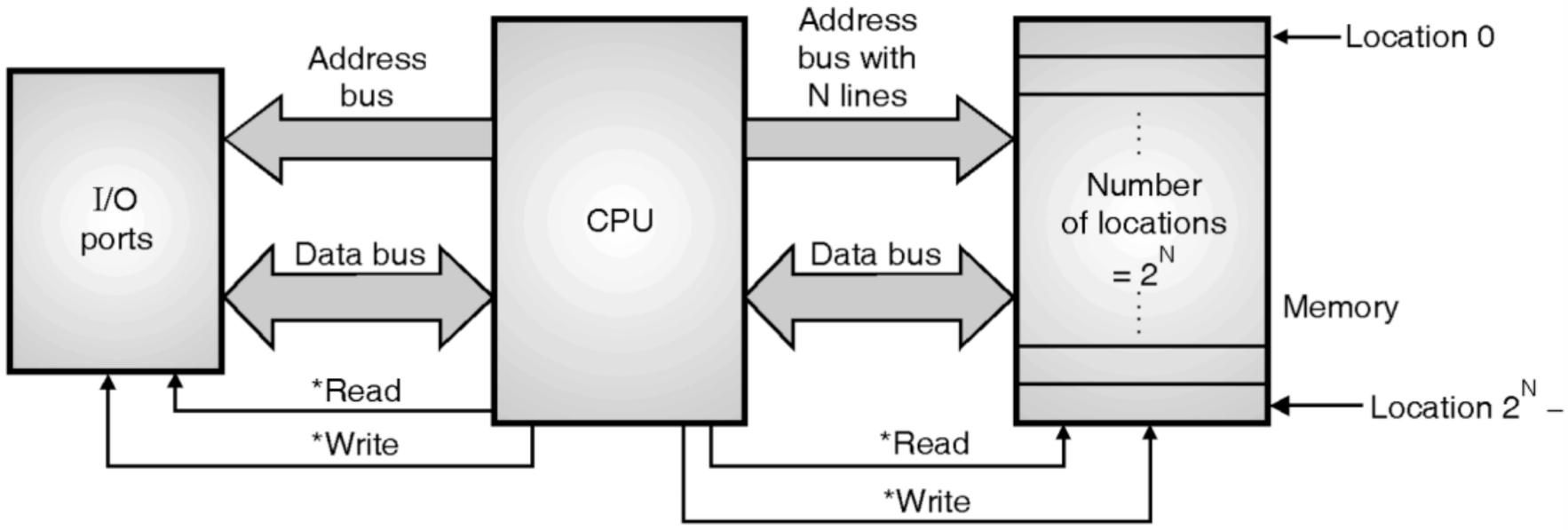


Fig. 10.1.2 : The three types of buses and their utility

$2^0 = 1$	$2^1 = 1$
$2^2 = 4$	$2^3 = 8$
$2^4 = 16$	$2^5 = 32$
$2^6 = 64$	$2^7 = 128$
$2^8 = 256$	$2^9 = 512$
2^{10} = (1024) 1 K (1 Kilo)	2^{20} = 1 K * 1 K = 1 M (1 Mega)
2^{30} = 1 G (1 Giga)	2^{40} = 1 T (1 Tera)

Ex. 10.1.1 : A processor has 24 address lines, how many memory locations it can access.

Soln. :

$$2^{24} = 2^4 \times 2^{20} = 16 \times 1\text{ M} = 16\text{ MB}$$

Ex. 10.1.2 : A processor has 4 GB memory, how many address lines are required to access this memory.

Soln. :

$$4\text{ GB} = 4 \times 1\text{ G} = 2^2 \times 2^{30} = 2^{32}$$

$\therefore$ 32 address lines are required.

10.1.3.2 Data Bus

- These lines are used to carry data between memory, processor and I/O devices.
- The size of data bus decides the number of bits that the processor can access simultaneously from memory or I/O device.
- The address lines select a location to be accessed (read/write), the data read or to be written is available on data bus.

10.1.3.3 Control Bus

- Since the same address lines are used for memory as well as I/O, there has to be some signalling that indicates the address is to access memory or I/O.
- Similarly, since same data bus is used to read or write data, some control signalling is required to indicate the operation to be carried out is read or write.
- Such signals are called as control signals. Some of the control bus signals are as follows :

  1. Memory read 2. Memory write
  3. I/O read 4. I/O write

Combined utilization of the three buses :

- In practice the CPU needs to make use of all the buses in order to perform a desired operation.
- For example if a byte (8 bits) is to be read from the memory location 18F8H, then the sequence of operations followed by the CPU is as follows :
- Task : Read data from memory location 18F8H

Step 1 : CPU sends out "18F8H" i.e. the address of the desired memory location, on the 16-bit address bus.

Step 2 : CPU sends the memory read signal on the control bus.

Step 3 : The memory read signal enables the memory device to put the data stored in the location 18F8H onto the data bus. This data travels on the data bus from memory to CPU.

10.1.4 ALU (Arithmetic and Logic Unit)

- This unit is used to perform arithmetic operations (like add, subtract, multiply, divide etc) and logical operations like (AND, OR, EX-OR etc)
- It consists of two inputs operands(4 bits in case of the ALU in Fig. 10.1.3) 'A' and 'B' and one output operand i.e. 'F'. 'S' are the select lines to select the operation (OPCODE).
- Besides these, there is also an input called as mode select for selecting Arithmetic or Logical operations.

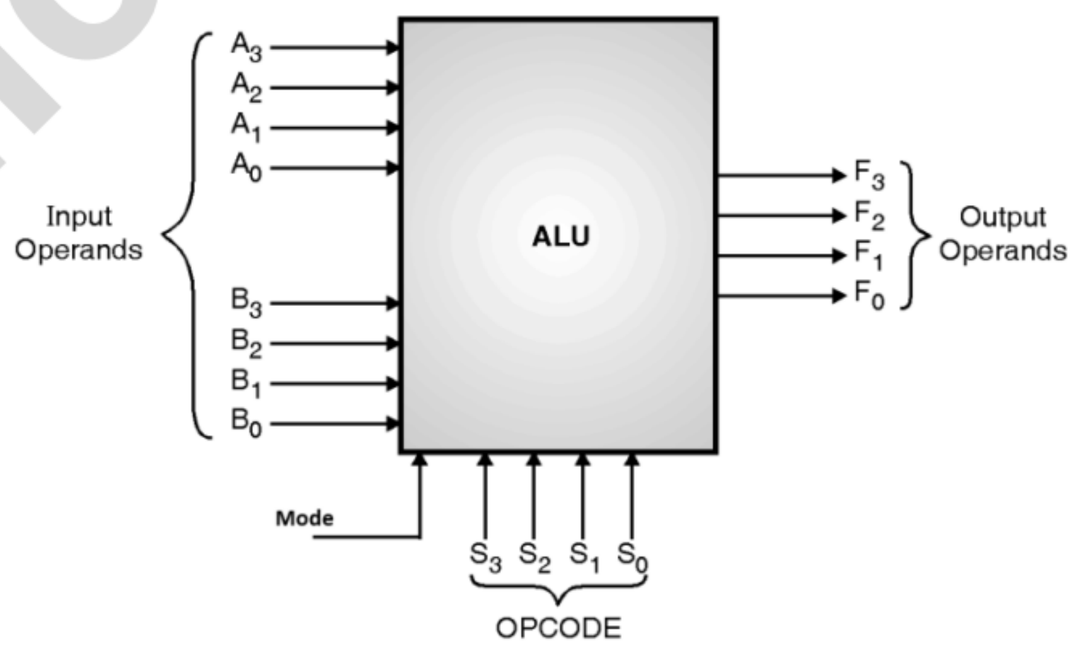


Fig. 10.1.3 : An ALU (Arithmetic logic unit)

- Now in the Fig. 10.1.3, to give an instruction to the system (ALU), the programmer has to give OPCODE and OPERAND i.e. the operation to be performed is to be given on the select lines while the data to be operated on is to be given on input operand lines.
- For e.g. if the operation is 5 + 7 i.e. $(0\ 1\ 0\ 1)_2 + (0\ 1\ 1\ 1)_2$ the data lines of A are to be $(0\ 1\ 0\ 1)_2$, while that of B are to be $(0\ 1\ 1\ 1)_2$.
- A binary code for addition is to be given on select lines. Suppose $(0\ 0\ 1\ 1)_2$ refers to add operation for this ALU.

– So the connections are to be made by the programmer as shown in Fig. 10.1.4.

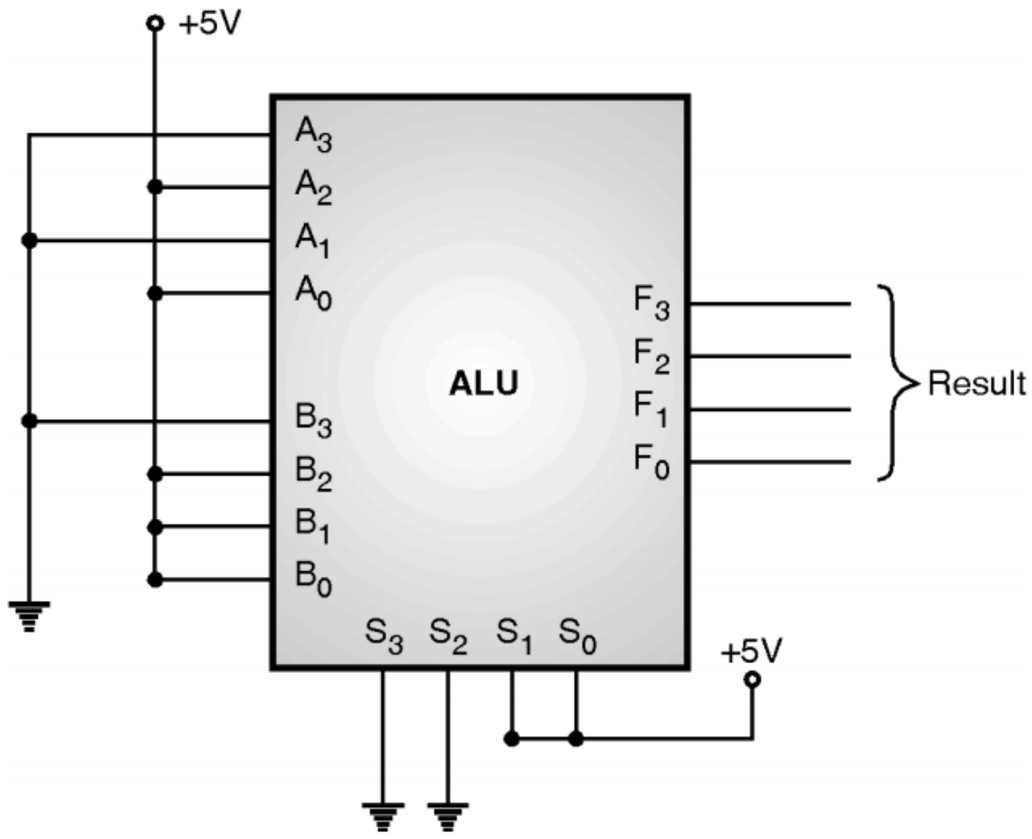


Fig. 10.1.4 : ALU connections to implement 5 + 7

– To do these connections, the programmer may take few seconds, while the IC can perform the operation in microseconds.

– Hence, the time in which the ALU could have performed few million operations, it can perform only one operation.

– The reason for this delay is the slow speed of the programmer to enter the instruction.

– Hence if the instructions were stored in a memory and the processor accesses them, it would be much faster.

– Thus one major additional aspect of a microprocessor over and ALU is that it can take instructions stored in memory, called as stored program concept.

10.1.5 Some More Terms

1. Interrupt :

– It is a mechanism by which an Input/Output device (Hardware interrupt) or an instruction (Software interrupt) can suspend the normal execution of the processor and get itself serviced.

2. Interrupt Service Routine (ISR) :

– A small program or a routine that when executed services the corresponding interrupting source is called as an ISR.

3. Vectored/Non-vectored interrupt :

– If the ISR address or some information related to the ISR address, of an interrupt is to be taken from the interrupting source itself, it is called as a non-vectored interrupt; else it is a vectored interrupt.

4. Maskable / Non-maskableinterrupt :

– Interrupt that can be masked (disabled) and unmasked (enabled) by the programmer is called as maskable interrupt else it is a non maskable interrupt.

10.2 Microprocessor Based Systems – Basic Operation

– A computer or a microcomputer mainly consists of three components, namely

1. Microprocessor or the CPU
2. Memory
3. Input/Output Devices

– Besides the above three, it also consists of buses that connect the three components of the microcomputer.

– Fig. 10.2.1 shows the block diagram of a microcomputer.

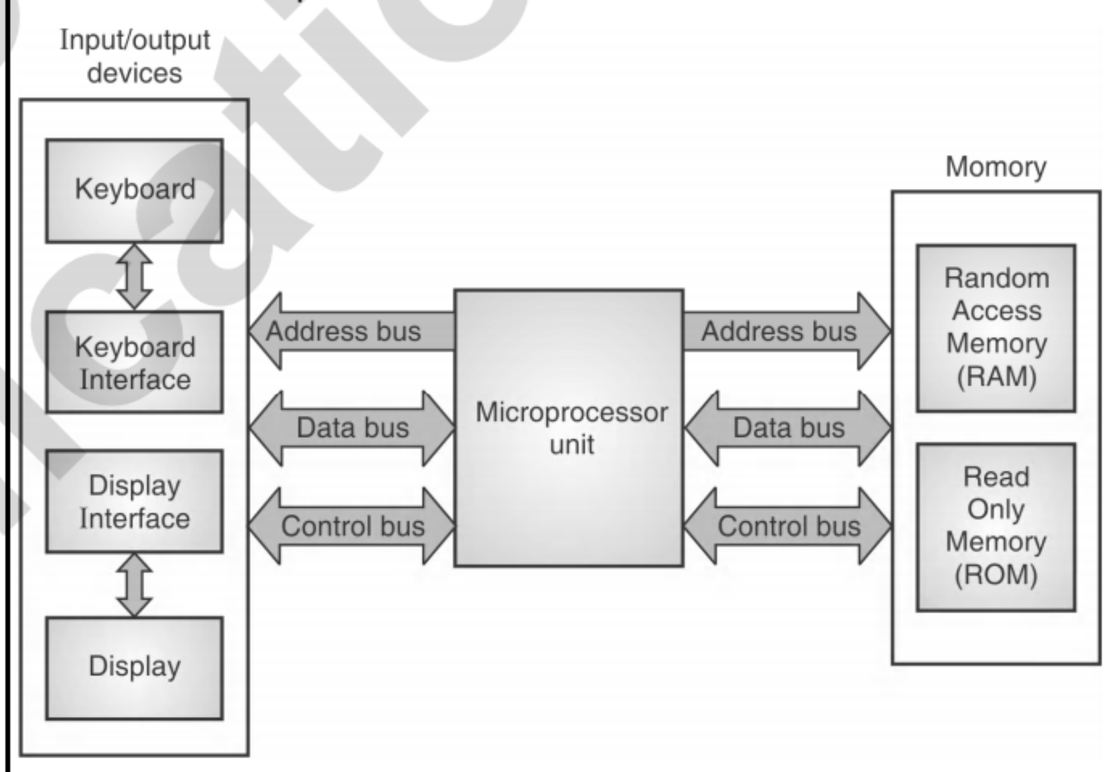


Fig. 10.2.1 : Block Diagram of a Microcomputer

1. Microprocessor Unit :

– The microprocessor unit or the microprocessor contains Arithmetic and Logical Unit (ALU) as well as a Control Unit (CU).

– The ALU as the name says is responsible for all the arithmetic and logical operations done by the processor.

– The CU controls not only the components inside the microprocessor unit, but also outside the microprocessor in the system.

– It produces control signals to be issued to the memory as well as the Input/Output (I/O) devices.

TechKnowledge Publications

2. Memory :

- It mainly contains Random Access Memory (RAM) and Read Only Memory (ROM) as shown in the Fig. 10.2.1.
- RAM is normally used to store temporary data as RAM is volatile i.e. the data is lost as soon as the supply goes off.
- ROM is used to store permanent data as it retains data even after the system is switched off. Hard disk, CD are various examples of ROM

3. I/O devices :

- Each input device or output device normally works at a very slower speed than the microprocessor.
- Also each input or output device follows a different method of data transfer.
- Hence, for each input or output device a separate interface is required as shown in the Fig. 10.2.1. Keyboard interface to interface the keyboard to microprocessor unit and similarly display interface to interface display to the microprocessor unit.

4. Buses :

- To connect the different components of a microcomputer, we need various connections.
- These connections having common functions when combined together form a bus. Thus we have three busses namely data bus, address bus and the control bus.

10.3 Block Diagram of Microprocessor

- Fig. 10.3.1 below shows the block diagram of a generic microprocessor (which is similar to the architecture of 8080 microprocessor).
- The various units in this block diagram are :

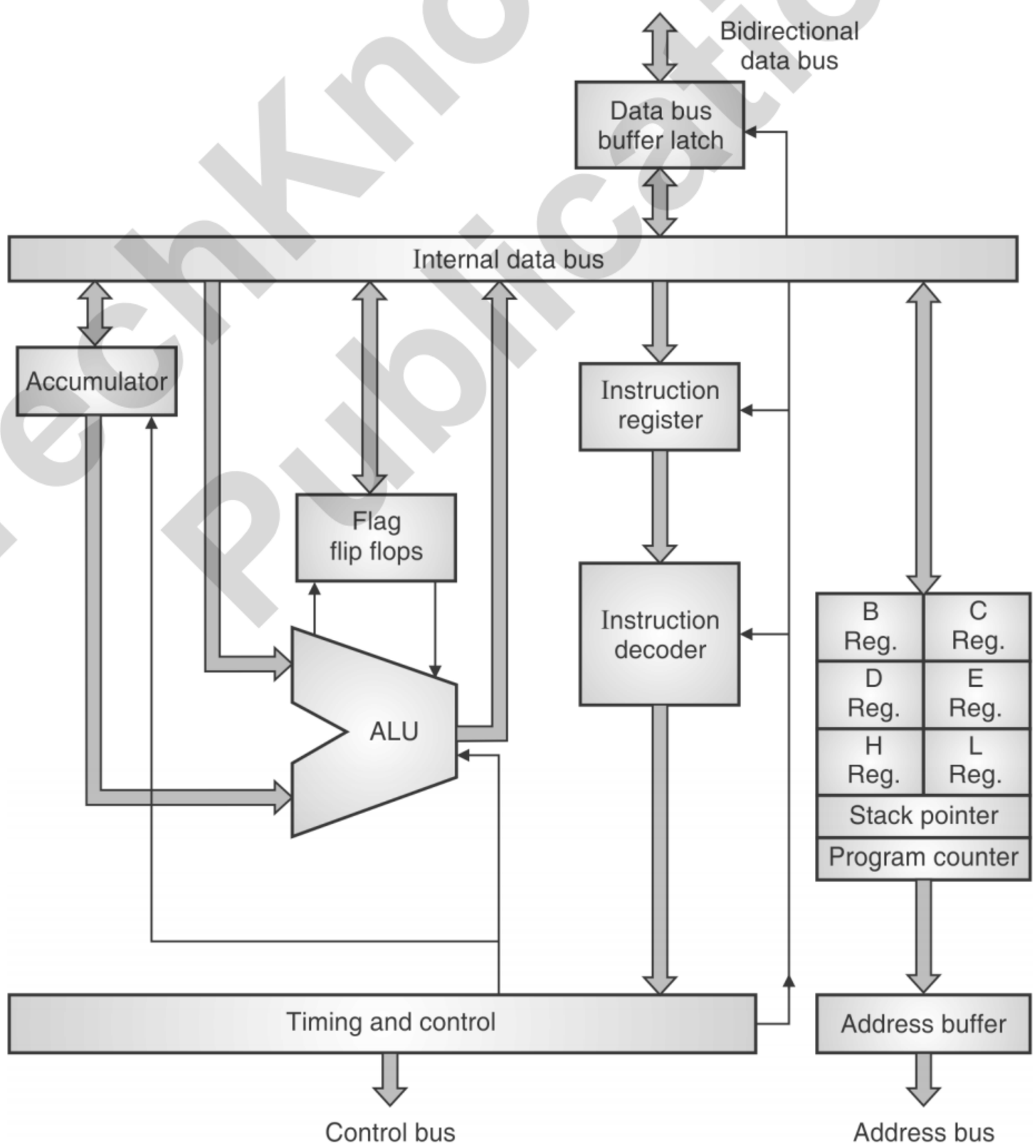


Fig. 10.3.1 : Block diagram of Generic Microprocessor

1. Internal bus :

– The internal bus as shown in the Fig. 10.3.1 above connects all the components inside the processor for data communication between them.

2. Arithmetic and Logical Unit (ALU) :

– ALU is responsible for performing all Arithmetic and Logical operations like addition, subtraction, AND, OR etc.

– It gets two inputs from the left side (as shown in the fig) and provides the output from right.

– It also gives output to flag flip-flops. The inputs are taken from the register called accumulator and from the internal bus.

3. Flag Flip flops :

– There are special flip-flops that give the status of the result of ALU operation.

– Many times it is not required to know the exact result, but only its status is required, in such cases these flip-flops are useful.

– For example, when we compare two numbers, the processor performs subtraction operation.

– Now, to know which number is greater, we need not know the exact difference, instead only the sign of the result (positive or negative) is enough for us.

– Similarly whether the result is zero or non-zero will tell if the two numbers are equal or not.

4. Accumulator :

– It is a general purpose register to store temporary data.

– It provides one input to the ALU for its operation and also stores (or accumulates) the result given by ALU.

5. Timing and Control Unit :

– It provides timing signal for synchronization of all the components inside and outside the processor and also provides the control signals.

– As seen in the Fig. 10.3.1, it provides control signals to all units inside the processor as well as on external control bus.

6. Instruction register and Instruction decoder :

– The Instruction register holds the opcode of the instruction in execution.

– The opcode is decoded by the instruction decoder, for the processor to understand the operation to be performed.

– The decoder then indicates the timing and control unit, as to what operation is to be done, which in turn performs the said operation by issuing control signals in that accordance.

7. Program Counter :

– This is a register that points to the next instruction to be executed.

– Hence, whenever the processor executes a particular instruction, the address of next instruction to be executed is given by program counter

8. Stack pointer :

– It is a pointer that points to the top of stack. Stack is used to store the data of incomplete operations to be completed in near future.

– Hence, if the processor is performing a given task, and something more important is to be done in between; the processor will put the current task data on the stack memory and start the new important task.

– Once the important task is completed, the processor will be able to resume the incomplete task by taking the data kept on stack.

– We will understand more about stack by reading the instructions related to stack in next chapter.

9. Register pair HL :

– Memory contains three things namely, program, stack and data.

– Program counter is used to point to the program, stack pointer for stack and similarly register pair HL for data.

– Thus, HL pair is used to point to the data to be accessed by an instruction.

10. General purpose registers (B,C,D and E) : These registers as the name says are general purpose registers. Hence the programmer can use them for whatever purpose required in their programs.

11. Data bus buffer latch :

– This is used to connect the internal bus of the processor to its external bus.

– If the data is to be communicated within the processor, internal data bus is used.

– But if the data is to be communicated between memory and processor or memory and I/O device the data has to be on external data bus communicated with processor through this buffer.

12. Address buffer :

– Whenever the processor has to access an external location (memory or I/O), it gives the address on the address bus through this buffer.

Tech Knowledge Publications

10.4 Functional Units of Microprocessor – ALU

- This unit is used to perform arithmetic operations (like add, subtract, multiply, divideetc). and logical operations like (AND, OR, EX-OR etc).
- It consists of two inputs operands(4 bits in case of the ALU in Fig. 10.4.1) 'A' and 'B' and one output operand i.e. 'F'. 'S' are the select lines to select the operation (OPCODE).
- Besides these, there is also an input called as mode select for selecting Arithmetic or Logical operations.

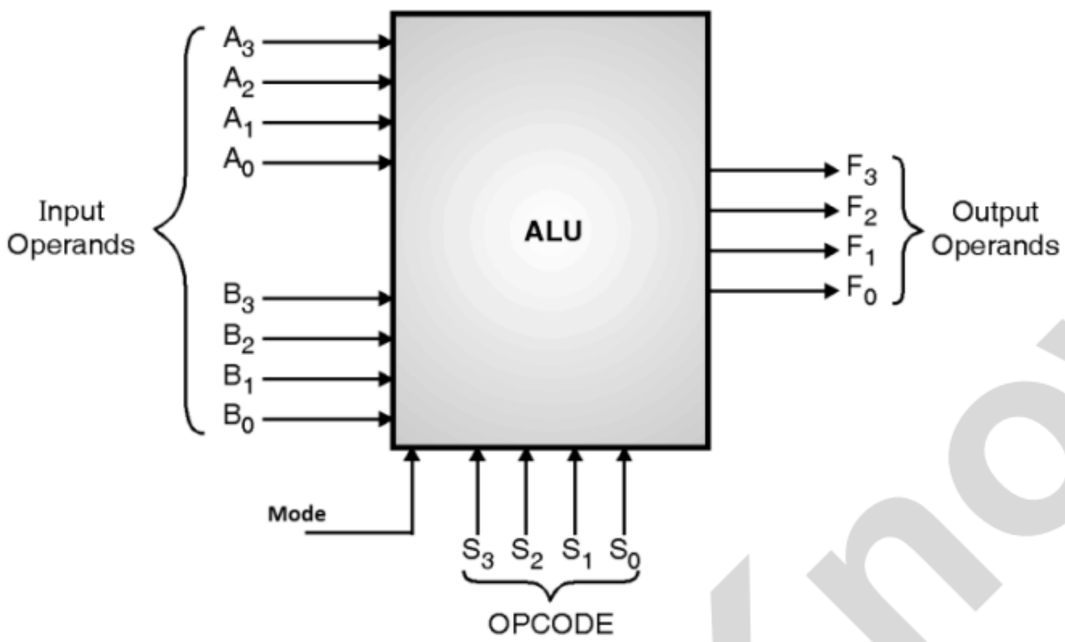


Fig. 10.4.1 : An ALU (Arithmetic logic unit)

- An ALU comprises of various blocks as shown in the Fig. 10.4.2below :

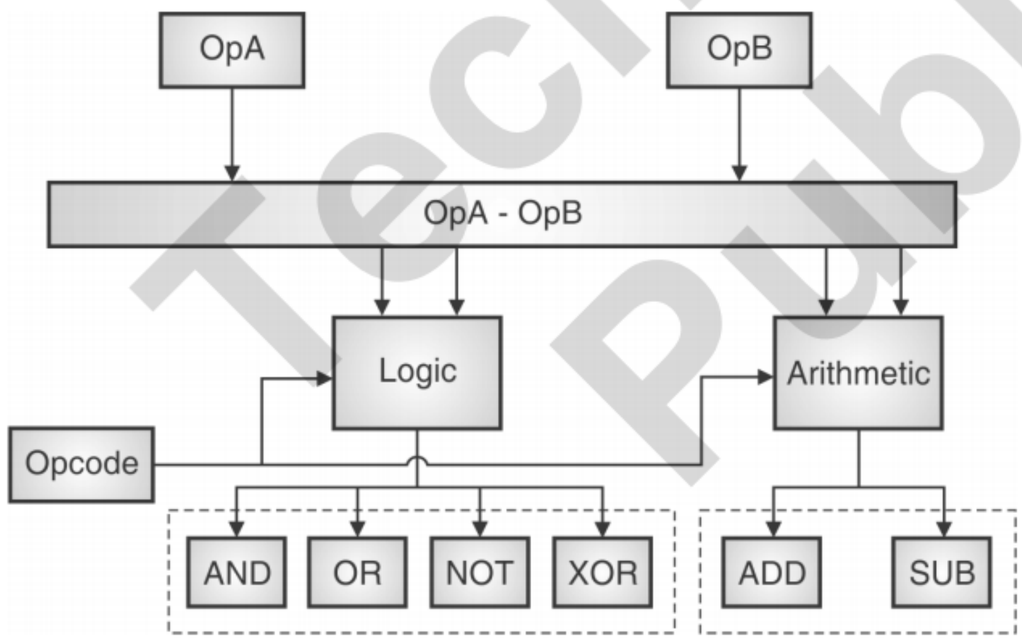


Fig. 10.4.2 : Block diagram of Generic ALU

- As shown in the figure, it comprises two registers OpA and OpB, to store the two operands to operate on. It has an input called as Opcode, which indicates the operation to be performed. There are two units
  1. Logic to perform various logical operations like AND, OR, NOT and EXOR
  2. Arithmetic to perform various operations like addition, subtraction etc.
- Now in the Fig. 10.4.2, to give an instruction to the system (ALU), the programmer has to give OPCODE and OPERAND i.e. the operation to be performed is to be given on the select lines while the data to be operated on is to be given on input operand lines.
- For e.g. if the operation is 5 + 7 i.e. $(0\ 1\ 0\ 1)_2 + (0\ 1\ 1\ 1)_2$ the data lines of A are to be $(0\ 1\ 0\ 1)_2$, while that of B are to be $(0\ 1\ 1\ 1)_2$.
- A binary code for addition is to be given on select lines. Suppose $(0\ 0\ 1\ 1)_2$ refers to add operation for this ALU. So the connections are to be made by the programmer as shown in Fig. 10.4.3.

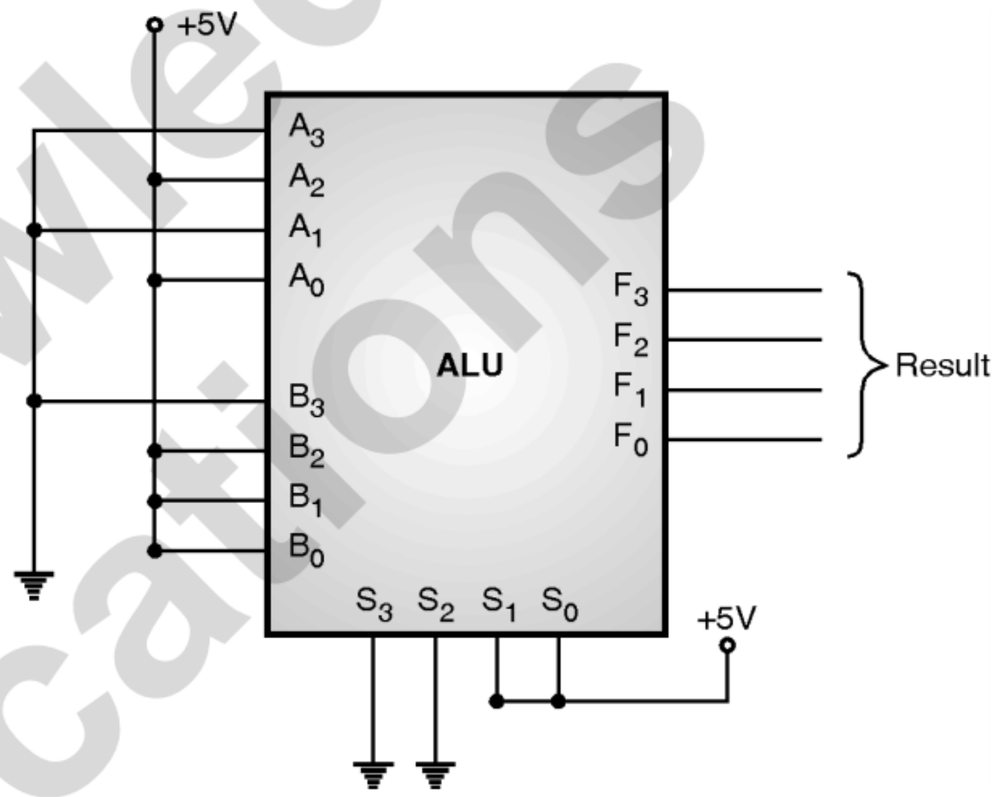


Fig. 10.4.3 : ALU connections to implement 5 + 7

- To do these connections, the programmer may take few seconds, while the IC can perform the operation in microseconds.
- Hence, the time in which the ALU could have performed few million operations, it can perform only one operation.
- The reason for this delay is the slow speed of the programmer to enter the instruction.
- Hence if the instructions were stored in a memory and the processor accesses them, it would be much faster.
- Thus one major additional aspect of a microprocessor over and ALU is that it can take instructions stored in memory, called as stored program concept.

10.5 ALU using IC 74181

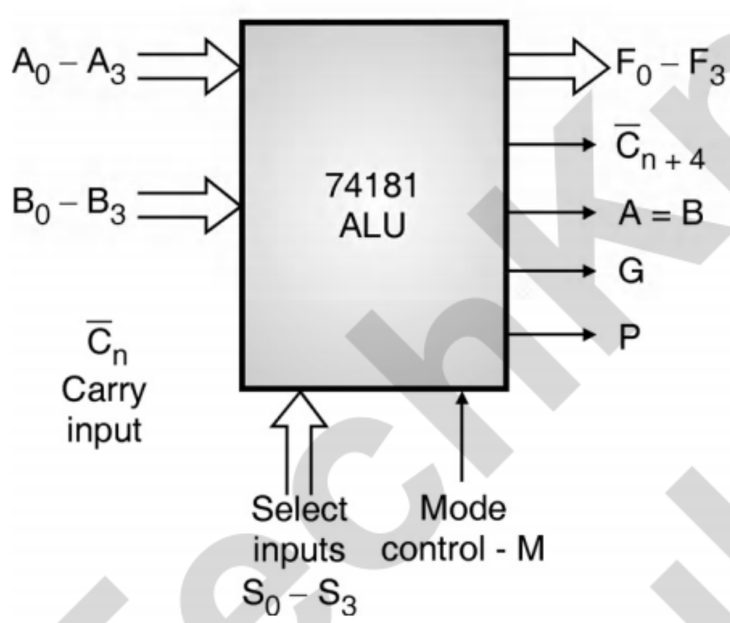
- ALU is a very widely used and popular combinational circuit.
- It is capable of performing the arithmetic as well as the logic operations.
- ALU is the heart of any microprocessor.

Tech Knowledge Publications

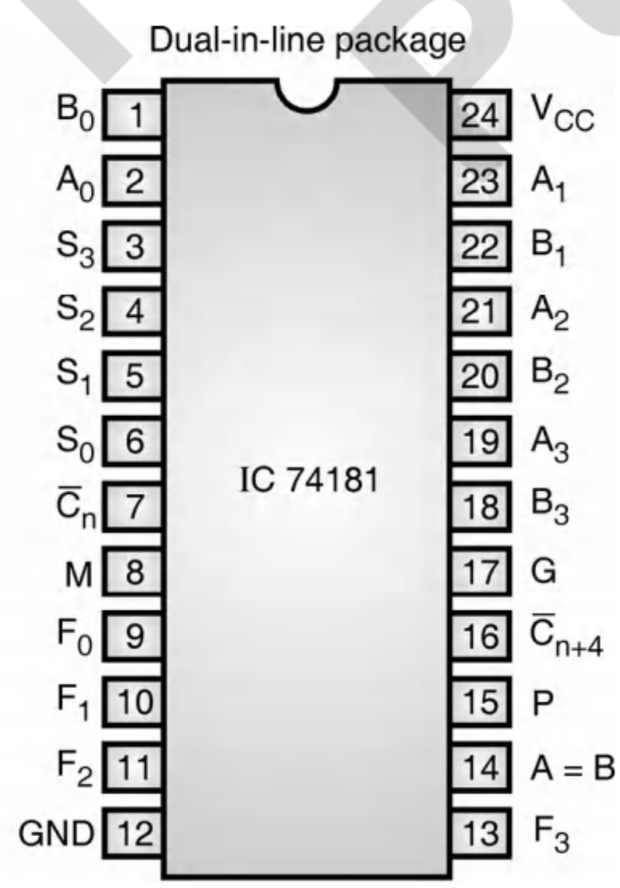
– Fig. 10.5.1 shows the block diagram of ALU IC 74181, Table 10.5.1 gives the pin description and Fig. 10.5.2 gives its pin configuration.

Table 10.5.1 : Pin description of 74181

Pin name	Description
$A_0 – A_3$	Operand inputs
$B_0 – B_3$	Operand inputs
$S_0 – S_3$	Function select inputs
M	Mode control input
$\overline{C}_n$	Carry input (active low)
$F_0 – F_3$	Function output
A = B	Comparator output (equality output)
G	Carry generate output
P	Carry propagate output
$\overline{C}_{n+4}$	Carry output (active low)



(C-1079)**Fig. 10.5.1 : Block diagram of ALU IC 74181**



(C-1079)**Fig. 10.5.2 : Pin configuration of the ALU IC 74181**

– 74181 is a 24-pin IC and it uses the Dual InPackage (DIP) line.

– A ($A_0 – A_3$) and B($B_0 – B_3$) are the two 4 bit words applied at its input.

– It can perform a total of 16 different arithmetic operations which includes addition, subtraction, comparison and doubling operations.

– It provides many logic operations such as AND, OR, NOR, NAND, EX-OR, compare as well on the two four bit variables.

– 74181 is a high speed 4 bit parallel ALU.

– The function to be performed is controlled by the four function select inputs $S_0 – S_3$. These lines can select 16 different operations for one mode (arithmetic) and 16 another operations for the other mode (logic).

– M is the mode control input. Its value decides the mode of operation to be either arithmetic or logic as follows :

M = 0 For arithmetic operations,

M = 1 For logic operations.

– G and P outputs are used when more than one IC 74181 circuits are to be used in cascade along with 74182, which is the look ahead carry generator circuit to make the arithmetic operations faster.

– When mode control input is high (M = 1), then the logic operations are performed on the individual bits and all the internal carries are inhibited.

– When mode control input is low (M = 0), then the arithmetic operations are performed on the two 4-bit words and all the internal carries are enabled.

– IC 74181 incorporates full internal carry lookahead. Due to this its speed of operation gets increased to a great extent.

– It provides a ripple carry between the devices using the C_{n+4}output. (see cascading of two 74181 s).

– Or for exploiting the option of carry look ahead between the packages, we have to use the P (carry propagate) and G (carry generate) outputs. This option should be used only when the speed requirements are stringent.

– If low speed of operation is acceptable, the ripple carry operation using C_{n+4} and C_n should be exercised.

A = B output :

1. A = B output is used to indicate that both the operands are equal. This output goes HIGH when the unit is in the subtract mode and A = B.

TechKnowledge Publications

2. This output also goes high when all the four "Function outputs" are HIGH.
3. It is possible to wire AND the A = B outputs when more than one 74181s are being used. The wire ANDing becomes possible because A = B is an open collector output. This enables us to compare words which are longer than 4-bits.

10.6 Basic Arithmetic Operations using ALU IC 74181

Function tables :

– Table 10.6.1(a) shows the function table for IC 74181. It is valid for the active high operands and active high outputs, and with $\overline{C}_n$ = 1 i.e. no carry.

Table 10.6.1(a) : Function table for IC 74181 with active high data and $\overline{C}_n$= 1 (no carry)

Function Select Inputs				Active high data and $\overline{C}_n$ = 1	
S_3	S_2	S_1	S_0	Logic operations (M = 1)	Arithmetic operations (M = 0)
0	0	0	0	F = $\overline{A}$ (Inversion)	F = A
0	0	0	1	F = $\overline{A + B}$ (NOR)	F = A + B
0	0	1	0	F = $\overline{A} \cdot B$	F = A + $\overline{B}$
0	0	1	1	F = 0	F = minus 1(2's comp)
0	1	0	0	F = $\overline{AB}$ (NAND)	F = A plus A $\overline{B}$
0	1	0	1	F = $\overline{B}$ (Invert)	F = (A + B) plus A$\overline{B}$
0	1	1	0	F = A ⊕ B (EXOR)	F = A minus B minus 1
0	1	1	1	F = A$\overline{B}$	F = A$\overline{B}$ minus 1
1	0	0	0	F = $\overline{A}$ + B	F = A plus AB
1	0	0	1	F = $\overline{A \oplus B}$ (EXNOR)	F = A plus B
1	0	1	0	F = B	F = (A + $\overline{B}$)plus AB
1	0	1	1	F = AB (AND)	F = AB minus 1
1	1	0	0	F = logic 1	F = A plus A *
1	1	0	1	F = A + $\overline{B}$	F = (A + B) plus A
1	1	1	0	F = A + B (OR)	F = (A + $\overline{B}$) plus A
1	1	1	1	F = A	F = A Minus 1

– Each bit is shifted to the next more significant position.
– The arithmetic operations mentioned in function Table 10.6.1(a), correspond to no carry input.
– They will get modified if the carry input is present as shown in Table 10.6.1(b).
– It is possible to use IC 74181 with either active high inputs or with active low inputs.
– With active low inputs the device produces active low outputs and with active high inputs it produces active high outputs.
– The function table for active low inputs and outputs has been given in Table 10.6.1(b).

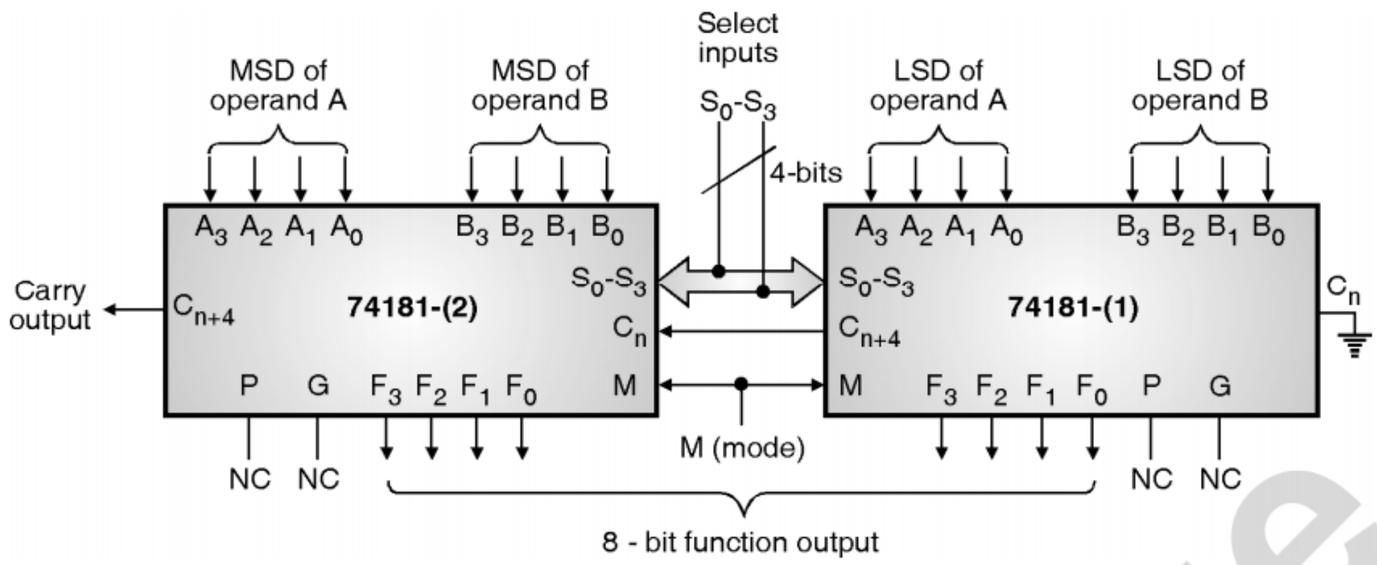
Table 10.6.1(b) : Function table for IC 74181 with active low data and $\overline{C}_n$ = 0 (with carry)

Function Select Inputs				Active low data and $\overline{C}_n$ = 0	
S_3	S_2	S_1	S_0	Logic operations M = 0	Arithmetic operations M = 1
0	0	0	0	F = $\overline{A}$ (inversion)	F = A minus 1
0	0	0	1	F = $\overline{AB}$ (NAND)	F = AB minus 1
0	0	1	0	F = $\overline{A}$ + $\overline{B}$	F = A $\overline{B}$ minus 1
0	0	1	1	F = 1	F = minus 1
0	1	0	0	F = $\overline{A}$ + $\overline{B}$	F = A plus (A + $\overline{B}$)
0	1	0	1	F = $\overline{B}$ (inversion)	F = AB plus (A + $\overline{B}$)

10.6.1 Cascading of Two 74181 ALUs

– When an 8-bit ALU is to be designed, two 74181 ICs are to be cascaded as shown in Fig. 10.6.1.
– The select input lines (S_3 – S_0) and the mode select (M) inputs of both the IC are connected to each other parallelly and brought out.
– This ensures that both the IC's operate in the same mode and select the same function.
– C_n (Carry input) of 74181-(1) is connected to ground whereas C_{n+4} (Carry output) of 74181-(1) is connected to carry input C_n of 74181–(2).
– This connection will propagate the carry of one stage propagated to the next stage.
– The final carry output $C_{n+4}$is produced by 74181–(2).
– The LSDs of the two input operands are connected to the four bit inputs of 74181-(1) and the MSDs are connected to the four bit inputs of 74181-(2).

TechKnowledge Publications



(C-1253(c)) Fig. 10.6.1 : 8 bit ALU using 74181

10.6.2 2-bit Multiplier Circuit using ALU and Shift Registers

– Let us implement 4-bit multiplication using a logic called as Booth's algorithm. The multiplication of Booth's algorithm is done using the following flowchart.

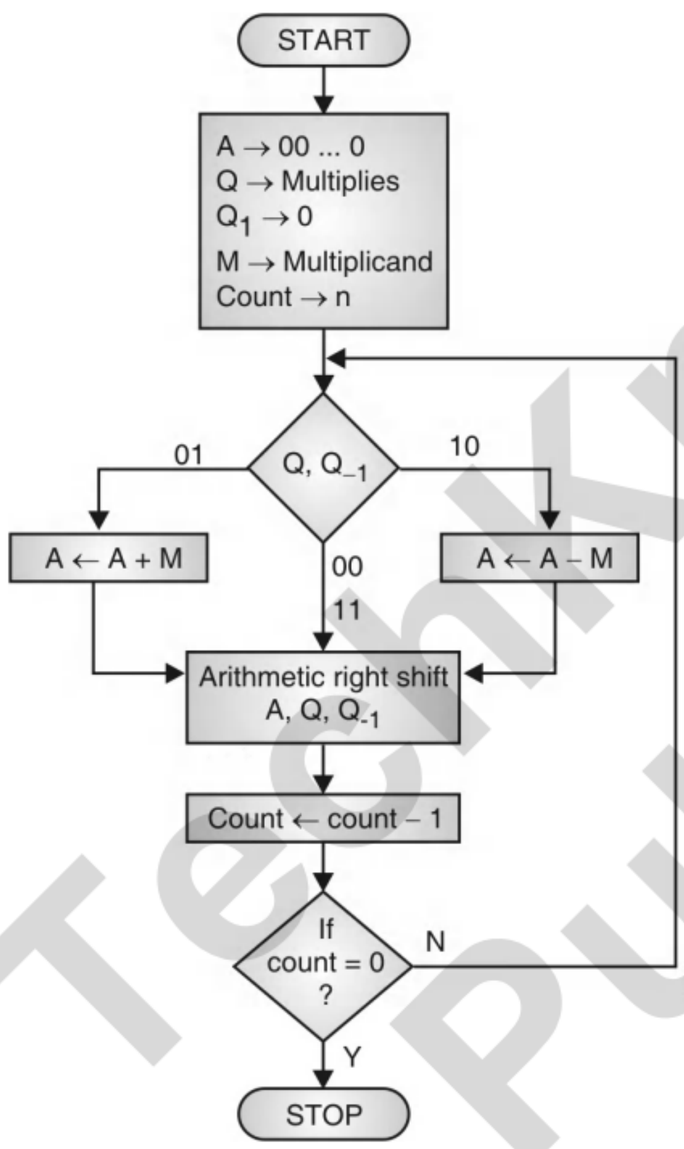


Fig. 10.6.2

Ex. 10.6.1 : Multiply:7 × – 3, using Booth's Algorithm.

A	Q	Q_{-1}	m	Count
0	1101	0	011	4
+101				
101	1101	0		
110	110	1		3
+011				
011	110	1		
01	111	0		2
101				
1010	111	0		
1101	011	1		1
110	1011	1		0

Soln. :

$$1101011 = -(010101)_2 = -(21)_{10}$$

– Fig. 10.6.3 shows the flowchart of Booth's algorithm. According to the booth's principle the bit preceding the series of 1s can be known if a zero followed by (i.e. 01), hence we add when this is the case.

– Similarly the last bit in the series is when the one is followed by zero (i.e. 10), hence we subtract in this case. While for consecutive zeros or ones we have not to do any arithmetic operation.

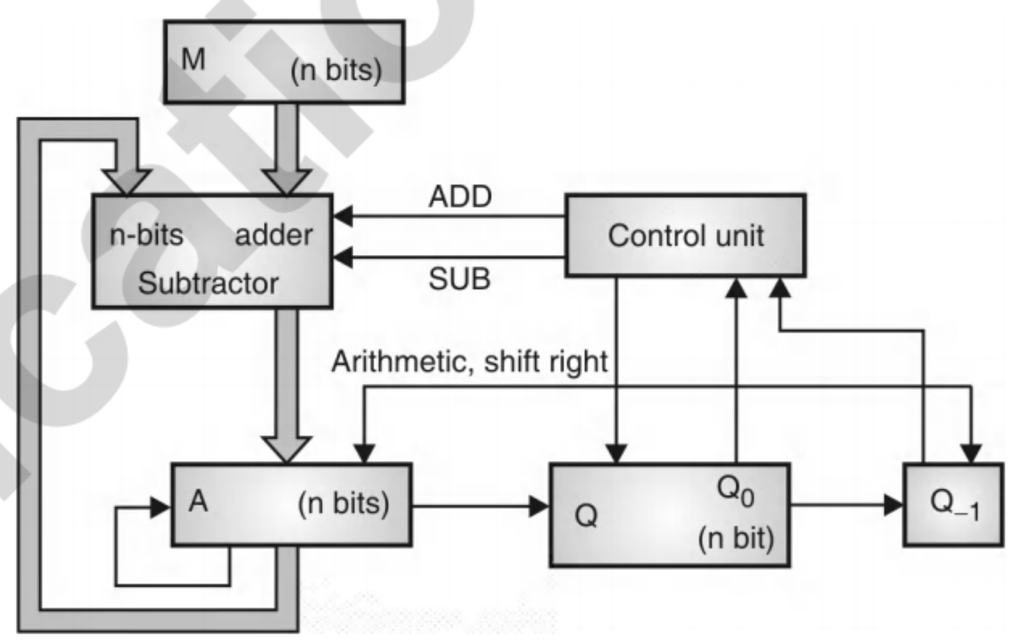


Fig. 10.6.3 : Block diagram of implementation of n-bit multiplication

– Fig 10.6.3 shows the hardware implementation of the n-bit multiplier.

– The accumulator or the A register and the Q register along with the Q_{-1} bit are combined together as a right shift register.

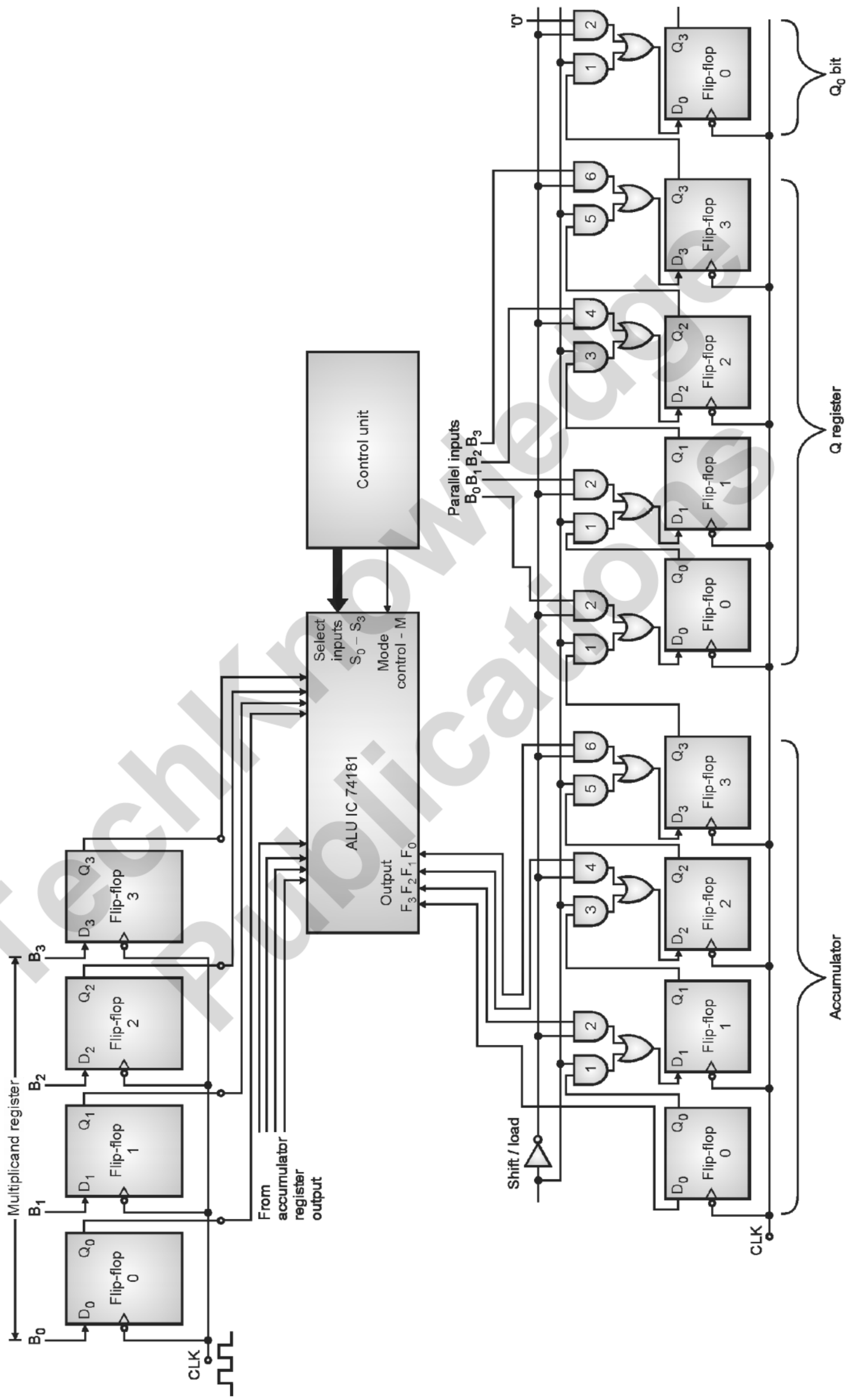
– The control unit checks the Q_0 and Q_{-1}bits to decide to add, subtract or directly shift. Arithmetic shift is implemented to maintain the sign.

– The register M, holds the multiplicand while the register Q holds the multiplier. Register A and Q_{-1} are initialized to zero.

– Fig 10.6.3 is a generalised n-bit multiplication. Let us see the implementation of a 4-bit multiplication using ALU IC74181

– Fig. 10.6.4 shows the implementation of multiplication using ALU IC 74181 and the shift registers using D Flipflops.

TechKnowledge Publications



(C-8261) **Fig. 10.6.4-bit Multiplier**

Tech Knowledge Publications

10.7 Memory Organization and Operations

10.7.1 Characteristics of Memory System

– When a memory is taken then there are various characteristics of this memory that are considered.

– The characteristics of memory are based on the following :

1. **Location :** The memory can be located in one of the following :

   CPU : This includes CPU registers and on-chip cache memory.

   Internal : This includes the memory that the processor can directly access.

   External : This is normally removable or virtual memory and hence access is slower.

2. **Capacity :** It is measured in terms of the word size and the number of words. Word size is the size of each location. Number of words is the number of locations.

3. **Unit of transfer :** This refers to the size of the data that is transferred in one clock cycle. It mainly depends on the data bus size. The data as discussed earlier may be internal or external and accordingly will be the data to be transferred in one clock pulse :

Internal : It is related to the communication of data with the memory directly accessible. It is usually governed by data bus width.

External : This is the data communication with the external removable memory or virtual memory. It is usually a block which is much larger than a word.

4. **Access method :** There are various methods of accessing the memory based on the memory organization. These methods are listed below with examples:

Sequential access :

– The sequential access means start from the beginning and read through in order until the byte to be read is reached.

– Hence the access time depends on location of data and previous location accessed. For e.g. magnetic tape.

– If one wants to listen to the third stanza of the fourth song stored in an audio cassette, he has to go through the entire first song second and the third song, and then the first stanza, second stanza of the third song and then reaches to the third stanza of that song.

Direct access :

– Here individual blocks have unique address and the access is done by jumping to vicinity plus sequential search.

– Hence access time depends on location and previous location.

– For e.g. magnetic or optical disk. Let take the same example that a person wants to listen to the third stanza of the fourth song on a CD, then he can directly reach to the fourth song, but thereafter he has to access the stanzas of the song sequentially to reach to the third stanza.

Random access :

– In case of random access individual addresses identify locations exactly.

– Hence the access time is independent of location or previous access. For e.g. RAM.

– In case of a RAM, any location to be accessed can be directly reached to without going through the locations sequentially.

Associative access :

– Here the data is located by a comparison with contents of a portion of the stored data(address).

– Hence the access time is independent of location or previous access. For e.g. cache.

– In case of cache memory, each location has a tag associated with it, and to reach to the required location the tags are to be compared with the location to be accessed.

– There are techniques used to reach to the required tagged location at a faster speed.

5. **Performance :**

– The performance of the memory depends on its speed of operation or the data transfer rate.

– The data transfer rate is the rate at which the data is transferred.

– The speed of operation depends on two things:

Access time : The time between providing the address and getting the valid data from memory is called as its access time i.e. the address to data time.

Memory cycle time : The time that is required for the memory to "recover" before next access i.e. the time between two addresses is called as memory cycle time.

Tech Knowledge Publications

6. Physical type :

– The physical material using which the memory is made can be different like:

Semiconductor :

– Memory can be made using semiconductor material i.e. ICs, for e.g. RAM.

Magnetic :

– Memory can also be made using magnetic read and write mechanism, for e.g. Magnetic disk and Magnetic tape.

Optical :

– Optical memories i.e. memories that use optical methods to read and write have become famous these days, for e.g. CD and DVD.

– There are some other methods using which data was stored in early days like Bubble and Hologram.

7. Physical characteristics :

– The physical characteristic of memory is also an important aspect to be considered.

– This includes the volatility, power consumption, erasable / not erasable, etc.

8. Organisation :

– It is not that always the memory will be organized sequentially. T

– here are some other types of memory organization like interleaved memory, etc.

10.7.2 Memory Hierarchy

– Memory Hierarchy explains that the nearer the memory to the processor, faster is its access. But costlier the memory becomes as it goes closer to the processor.

– The following sequence is in faster to slower or costlier to cheaper memory:

1. Registers i.e. inside the CPU.
2. Internal memory that includes one or more levels of cache and the main memory. Internal memory is always RAM, SRAM for cache and DRAM for main memory. The differences between the SRAM and DRAM will be seen in a later section in this chapter.
3. External memory or removable memory includes the hard disk, CDs, DVDs etc.

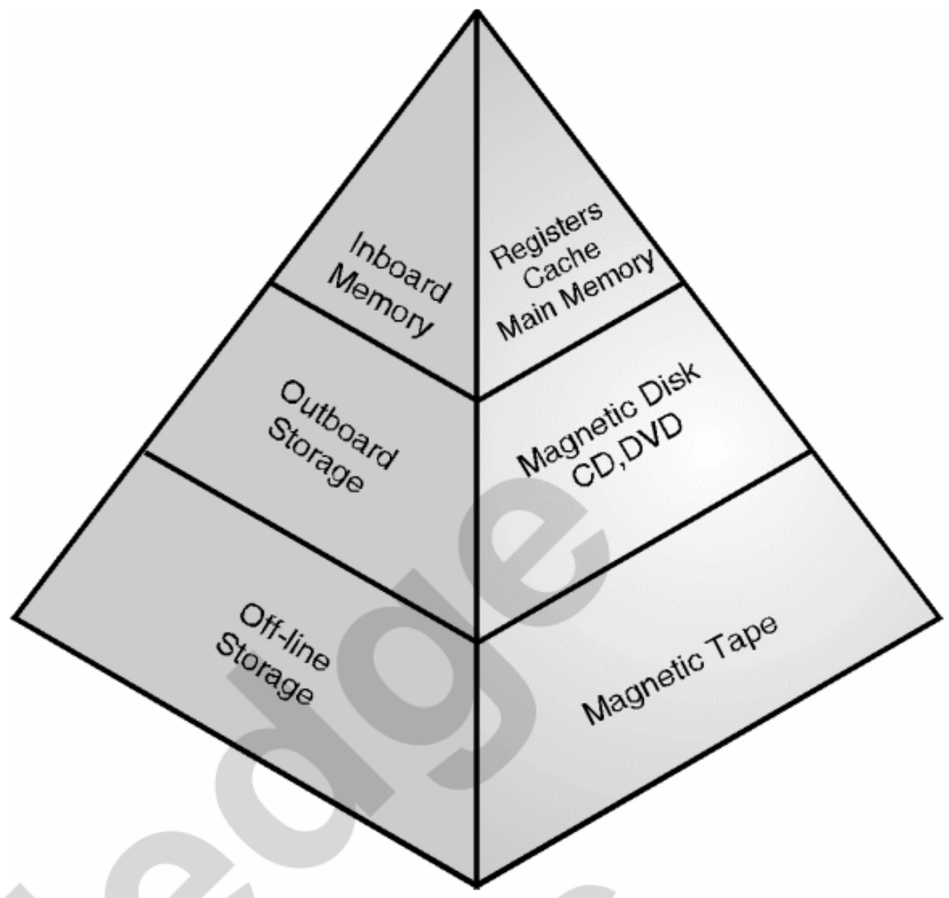


Fig. 10.7.1 : Memory Hierarchy

– Fig. 10.7.1 shows the memory hierarchy based on the closeness to the processor.

– The registers as discussed are the closest to the processor and hence are the fastest while off-line storage like magnetic tape are the farthest and also the slowest.

– The list of memories from closest to the processor to the farthest is given as below:

1. Registers
2. L1 Cache
3. L2 Cache
4. Main memory
5. Magnetic Disk
6. Optical
7. Tape

– To have a large faster memory is very costly and hence the different memories at different levels gives the memory hierarchy.

10.7.3 Basic Memory Cell

10.7.3.1 SRAM and DRAM

– RAM (Random Access Memory) is called so because any memory location in this IC can be accessed randomly.

– There are two types of RAM, namely, SRAM (Static RAM) and DRAM (Dynamic RAM).

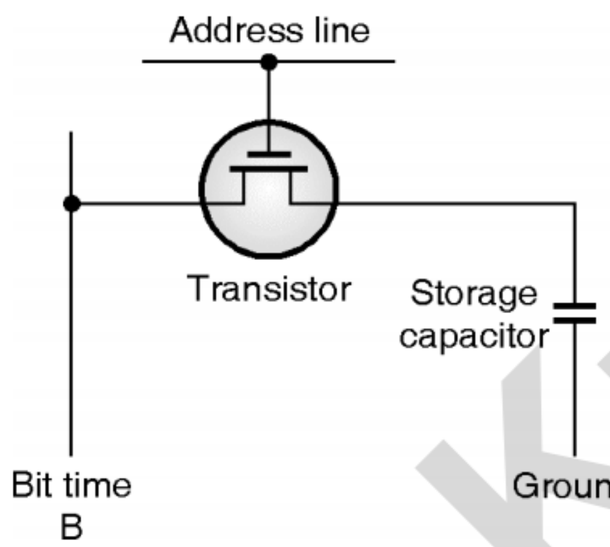
– SRAM is made up of flip flops while the DRAM is made up of capacitors.

– Since DRAM is made using capacitors, it requires less number of components to make a one bit cell, hence

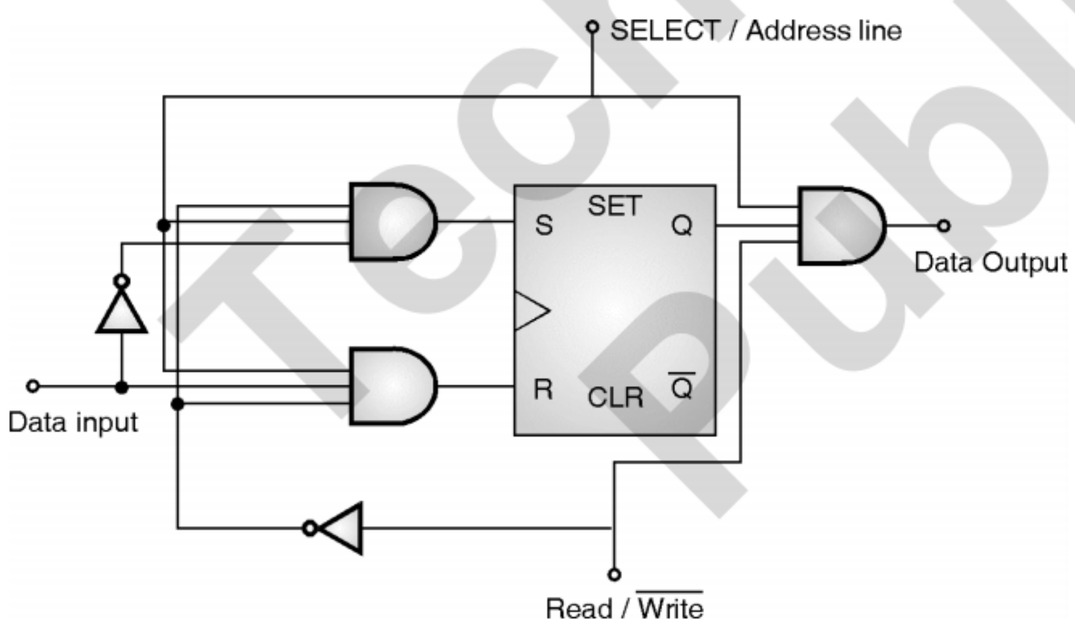
TechKnowledge Publications

also requires less space on the silicon wafer. Thus it is also comparatively cheaper.

- But it is slower than SRAM, because capacitors require time for charging and discharging.
- Also the capacitors loose charge in some time and hence need to be recharged according to the data, this is called as refreshing the DRAM. Fig.10.7.2(a) shows the structure of a DRAM cell.
- The address line selects the particular location, it enables the MOSFET that connects the capacitor to the data bus and hence if the data is to be read, simply the data line gets the data to be read; while if the data is to be written the data is to be given on the data line and will be written on the capacitor.



(a) DRAM cell structure



(b) SRAM cell structure

Fig. 10.7.2

- On the other hand, the SRAM has each cell made of a flip-flop, thus requires more components as compared to the DRAM cell.
- Hence it occupies more space on the silicon wafer, and is costlier.
- Thus it is also costlier. But the advantage is that it doesn't require any refreshing and is also very fast compare to DRAM. Fig. 10.7.2 (b) shows the structure of the DRAM cell.
- The flip-flop is used to store the data.
- There is a AND gate at the output of the flip-flop, which will be enabled by the select line (that works as an address line) and the Read / $\overline{\text{Write}}$ operation (when data has to be read) when it is logic '1'.
- Thus when both i.e. the select line and the Read / $\overline{\text{Write}}$ lines are '1', the output of the flip-flop will be available on the data line.
- Similarly for the write operation, the select line must be enabled by making it logic '1' and the Read / $\overline{\text{Write}}$ line must be logic '0'.
- Thus the data from the input line will be stored in the flip-flop.
- Table 10.7.1 shows the differences between the SRAM and DRAM.

Table 10.7.1

Sr. No.	SRAM	DRAM
1.	No refreshing required.	Continuous refreshing required (disadvantage).
2.	It is faster for accessing data.	It is slower in accessing data.
3.	It takes more space on chip as more number of components are required per bit.	It takes less space on chip as less number of components are required per bit.
4.	Hence is also costly.	Hence is cheaper.
5.	Bit density is lesser.	Bit density is more.
6.	The bit is stored in a flip-flop.	The bit is stored as a charged or discharged capacitor.
7.	SRAM is mainly used or selected forcache memory.	DRAM is mainly used or selected for semiconductor main memory.

10.7.4 ROM (Read Only Memory)

- ROM or the read only memory is quite cheaper compared to RAM and is mainly used for implementation of the secondary or the virtual memory.
- Thus the application of ROM is for virtual or secondary memories like hard disks, external storage like CD / DVD and floppy disks etc.
- We will see different types of ROM in the subsequent sub-section and thereafter some ROM memories used in computers like magnetic disk, CD, DVD etc.

Tech Knowledge Publications

10.7.4.1 Types of ROM

- There are various types of ROM available based on whether or not it can be re-written; they are called as ROM, PROM, EPROM and EEPROM.
- These types of memories will be studied in this section.
- There are some more ROMs available these days like flash memory, OTP etc.
- The ROM is a memory wherein, the user cannot write anything.
- The data to be stored in the ROM is to be given to the ROM manufacturer, who writes this data on the ROM and provides the same.
- The PROM (Programmable Read Only Memory) or also sometimes referred to as OTP (One Time Programmable) memory, as it can be written onto only once.
- When manufactured, it is blank, once written on it, it cannot be re-written.
- There are diodes that are used to store data, and they are fused or kept as it is to store the data in them.
- The internal diagram of the PROM is shown in Fig. 10.7.3.
- The AND array is used as address lines and the OR array as data lines.
- The AND array (on the left in Fig. 10.7.3) comes as predefined connections as shown in the Fig. 10.7.3 in sequence of binary, in this case from "000" to "111", as there are three bit address.
- The OR array (on the right hand in Fig. 10.7.3) comes with programmable link, the ones to be retained can be retained while the remaining fused or opened.
- Hence whenever a memory address is given to the address lines (a, b and c in Fig. 10.7.3) the specified location will be selected and according to the fused links, the data will be available on the OR gates output lines.
- The EPROM (Erasable Programmable Read Only Memory) although extinct today and is replaced by EEPROM or E^2PROM, but it used to be the only erasable memory available earlier.
- In case of EPROM the data written can be erased by keeping the EPROM IC in the UV box, as the UV rays erase the previously written data on the EPROM.
- The EPROM as discussed earlier are these days replace with EEPROM (Electrically Erasable Programmable Read Only Memory).
- The EEPROMs are erased by giving an extra supply voltage.

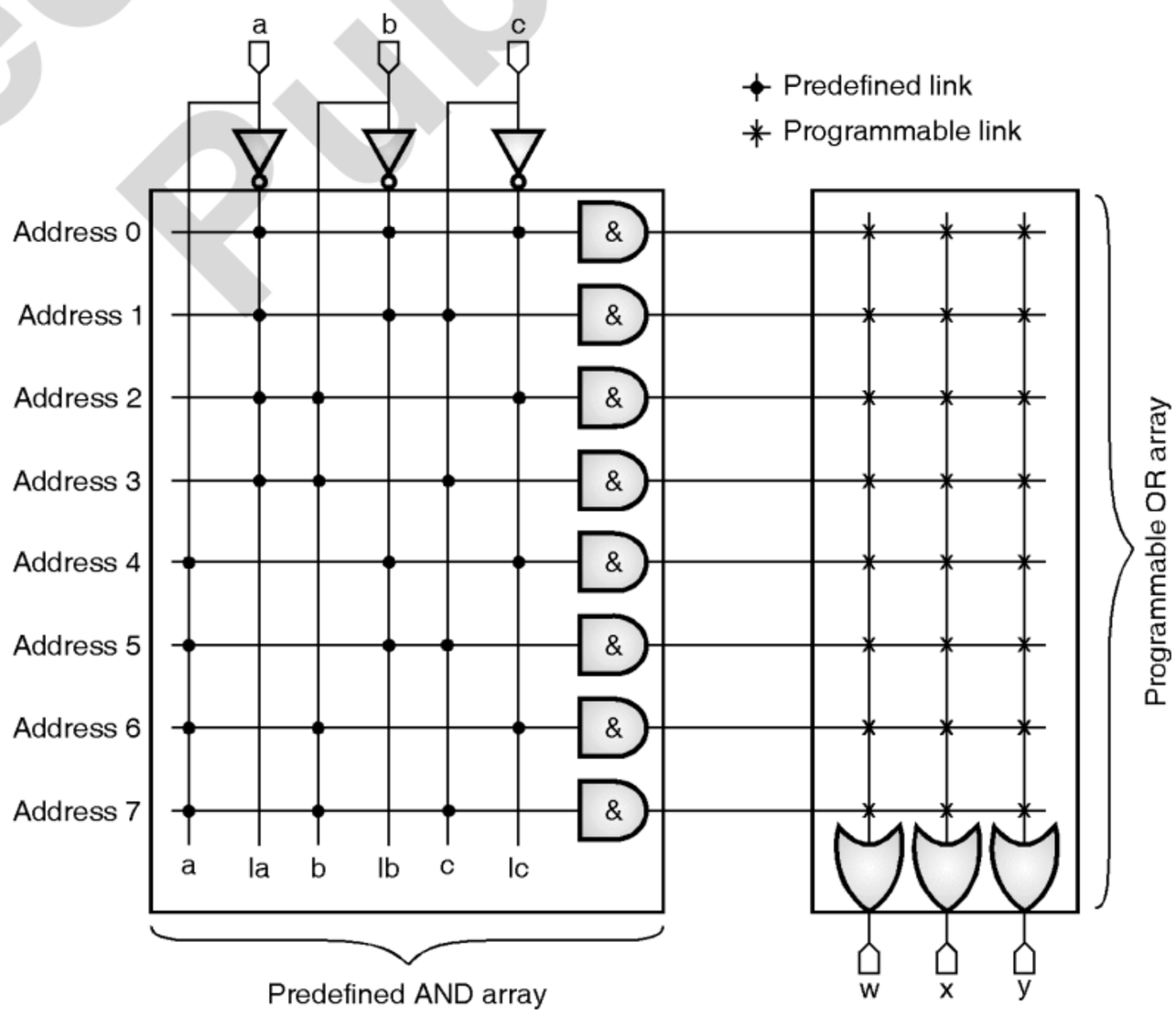


Fig. 10.7.3 : PROM

10.8 Digital Circuit Using Decoder and Registers for Memory Operations

- In this section we will study how to interface memory to the processor.
- We will also see how to place the memory in different locations in the system.
- To begin with we will revise the different types of memory that can be interfaced with the processor.

10.8.1 Types of Memory

- The different types of memories used with processors are EPROM (Erasable Programmable Read Only Memory) and RAM (Random Access Memory).

10.8.1.1 Memory Map, Structure and its Requirements

- The read / write memories consist of an array of registers wherein each register has a unique address.
- Fig. 10.8.1 shows the block diagram of a memory device.

  N : number of registers

  M : word length.
- e.g. : If a memory is having 13 address lines and 8 data lines, then number of registers / memory locations $= 2^N = 2^{13} = 8192$ word length = M bit = 8 bit.
- The number of address lines of a microprocessor depends on the size of the memory.

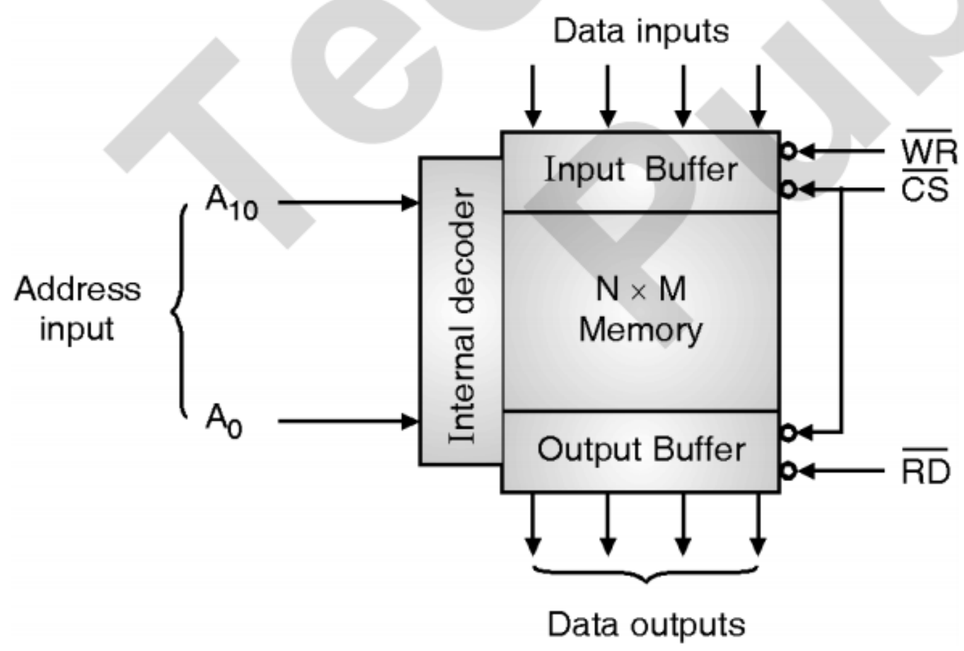


Fig. 10.8.1 : Block diagram of a memory device

Table 10.8.1

Number of address lines required	Size of memory in bytes
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024 ≈ 1 k
11	2048 ≈ 2 k
12	4096 ≈ 4 k
13	8192 ≈ 8 k
14	16384 ≈ 16 k
15	32768 ≈ 32 k
16	65536 ≈ 64 k

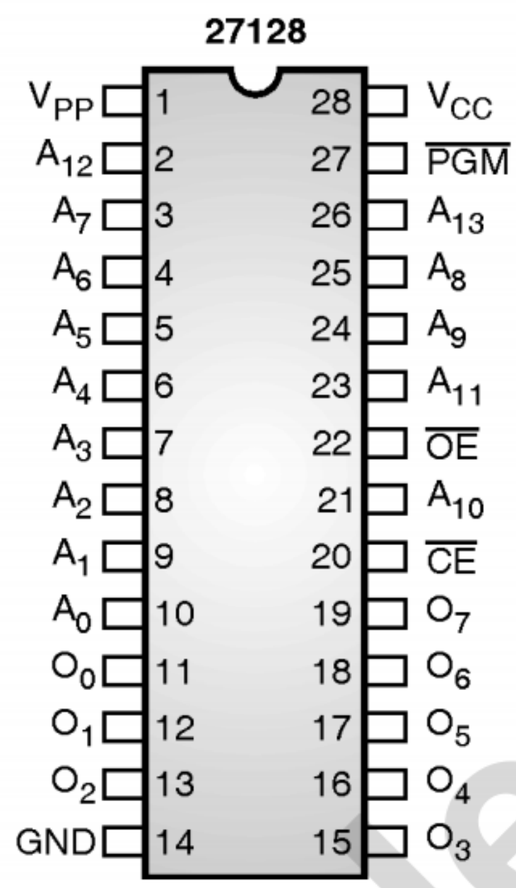
10.8.1.2 Memory Chip Size and Numbers

Table 10.8.1 : EPROM ICs available in the market

IC number	Memory size Address data	Number of pins
2716	2 k × 8	24
2732	4 k × 8	24
2764	8 k × 8	28
27128	16 k × 8	28
27256	32 k × 8	28
27512	64 k × 8	28

TechKnowledge Publications

27512	27256	2764	2732A	2716
A_{15}	V_{PP}	V_{PP}		
A_{12}	A_{12}	A_{12}		
A_7	A_7	A_7	A_7	A_7
A_6	A_6	A_6	A_6	A_6
A_5	A_5	A_5	A_5	A_5
A_4	A_4	A_4	A_4	A_4
A_3	A_3	A_3	A_3	A_3
A_2	A_2	A_2	A_2	A_2
A_1	A_1	A_1	A_1	A_1
A_0	A_0	A_0	A_0	A_0
O_0	O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2	O_2
GND	GND	GND	GND	GND



2716	2732A	2764	27256	27512
		V_{CC}	V_{CC}	V_{CC}
		$\overline{PGM}$	A_{14}	A_{14}
V_{CC}	V_{CC}	N.C.	A_{13}	A_{13}
A_8	A_8	A_8	A_8	A_8
A_9	A_9	A_9	A_9	A_9
V_{PP}	A_{11}	A_{11}	A_{11}	A_{11}
$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}$
A_{10}	A_{10}	A_{10}	A_{10}	A_{10}
$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
O_7	O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3	O_3

(a) Pin configuration

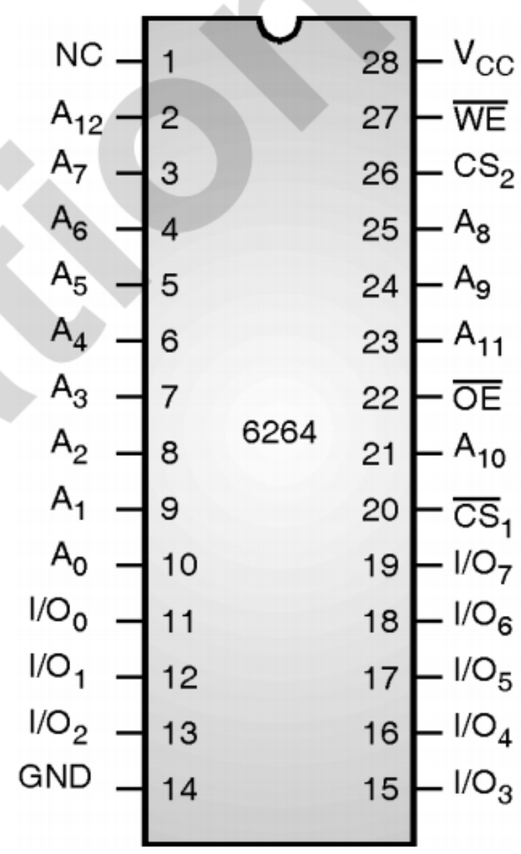
Pin Names	
$A_0 - A_{13}$	Addresses
$\overline{CE}$	Chip enable
$\overline{OE}$	Output enable
$O_0 - O_7$	Outputs
$\overline{PGM}$	Program
N.C.	No connect

(b)

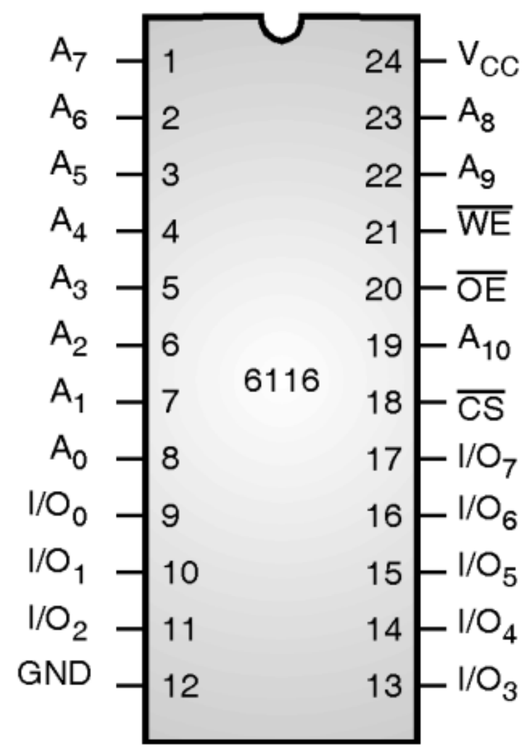
Fig. 10.8.2

Table 10.8.3 : RAM ICs

IC no.	Type	Memory size
6116	SRAM	2k × 8
6264	SRAM	8k × 8
2114	SRAM	1k × 4



(a)



(b)

Fig. 10.8.3 : Pin configuration

TechKnowledge Publications

Truth Table for 6264				
$\overline{WE}$	$\overline{CS_1}$	$\overline{CS_2}$	$\overline{OE}$	Mode
X	1	X	X	Not selected
X	X	0	X	(Power down)
1	0	1	1	Output disable
1	0	1	0	Read
0	0	1	1	Write
0	0	1	0	Write

Truth Table for 6116			
$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Mode
1	X	X	Not Selected
0	0	1	Read
0	1	0	Write
0	0	0	Write

Ex. 10.8.1 : Interface 4 KB of EPROM with starting address from 0000H and 2 KB of RAM with starting address followed by EPROM.

Soln. :

Note : Let us assume the processor has 16 address lines

Step 1 : Total EPROM required = 4 kB

Chip size available = 4 kB (assume) (IC 2732)

$\therefore$ No. of chips required $= \frac{4\text{ kB}}{4\text{ kB}} = 1$

Chip 1 : Starting address = 0000H

Chip size = 4kB = 0FFFH

(Actually 4 kB = 1000H, but here we have taken 0000H to 0FFFH)

$\therefore$ Ending Address = 0FFFH

Step 2 : Total RAM required = 2 kB

Chip size available = 2 kB (assume) (IC 6216)

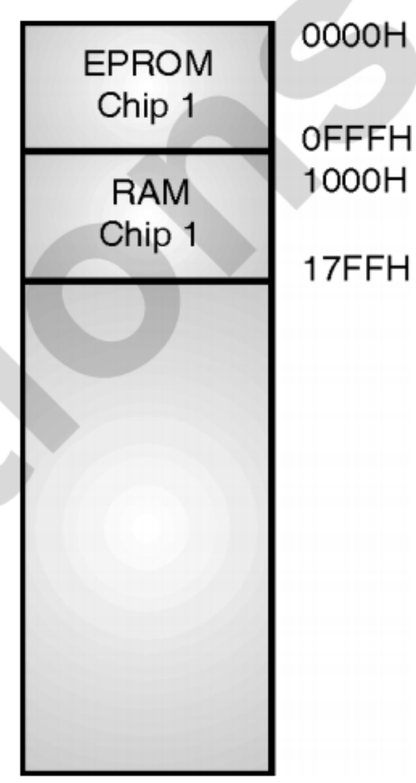
$\therefore$ No of chips required $= \frac{2\text{ kB}}{2\text{ kB}} = 1$

Chip 1 : Starting address = ending address of EPROM + 1

= 0FFFH + 1 = 1000H

Chip size = 2 kB = 07FFH

$\therefore$ Ending address = 17FFH



Step 3 : Memory Map :

		A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
EPROM chip 1 $\overline{y_0}.\overline{y_1}$	SA = 0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM chip 1 $\overline{y_2}$	SA = 1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 17FFH	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1

TechKnowledge Publications

Full decoding logic : EPROM chip size = 4kB and RAM chip size = 2kB.

∴ Smaller chip size ⇒ RAM = 2 kB = 2^{11}

∴ Neglect lower 11 address lines i.e. A_0 to A_{10} and consider remaining address lines A_{11} to A_{15} for decoding.

– Hence 5:32 decoder is required. Hence circle the five bits as shown in memory map. EPROM has two possibilities 00000b and 00001b, thus it requires $\overline{y_0}$ and $\overline{y_1}$ outputs of the 5:32 decoder. RAM has 00010b and hence it requires $\overline{y_2}$ output of the decoder.

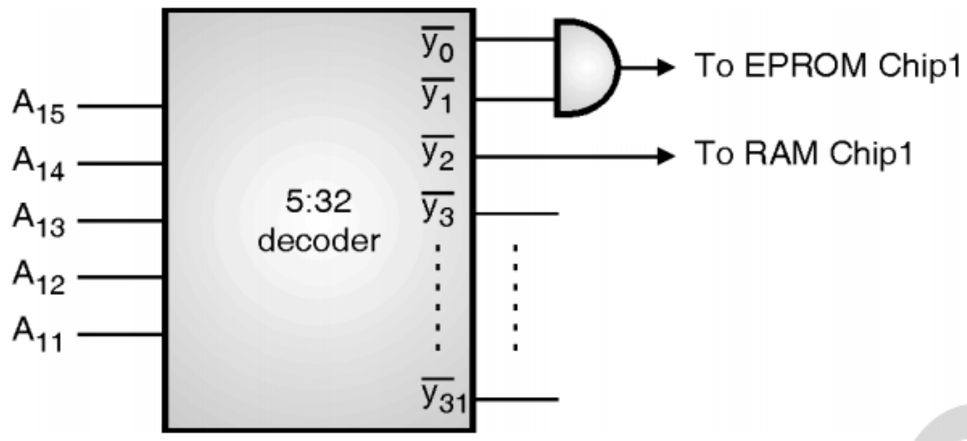


Fig. P. 10.8.1

Step 4 : Final implementation.

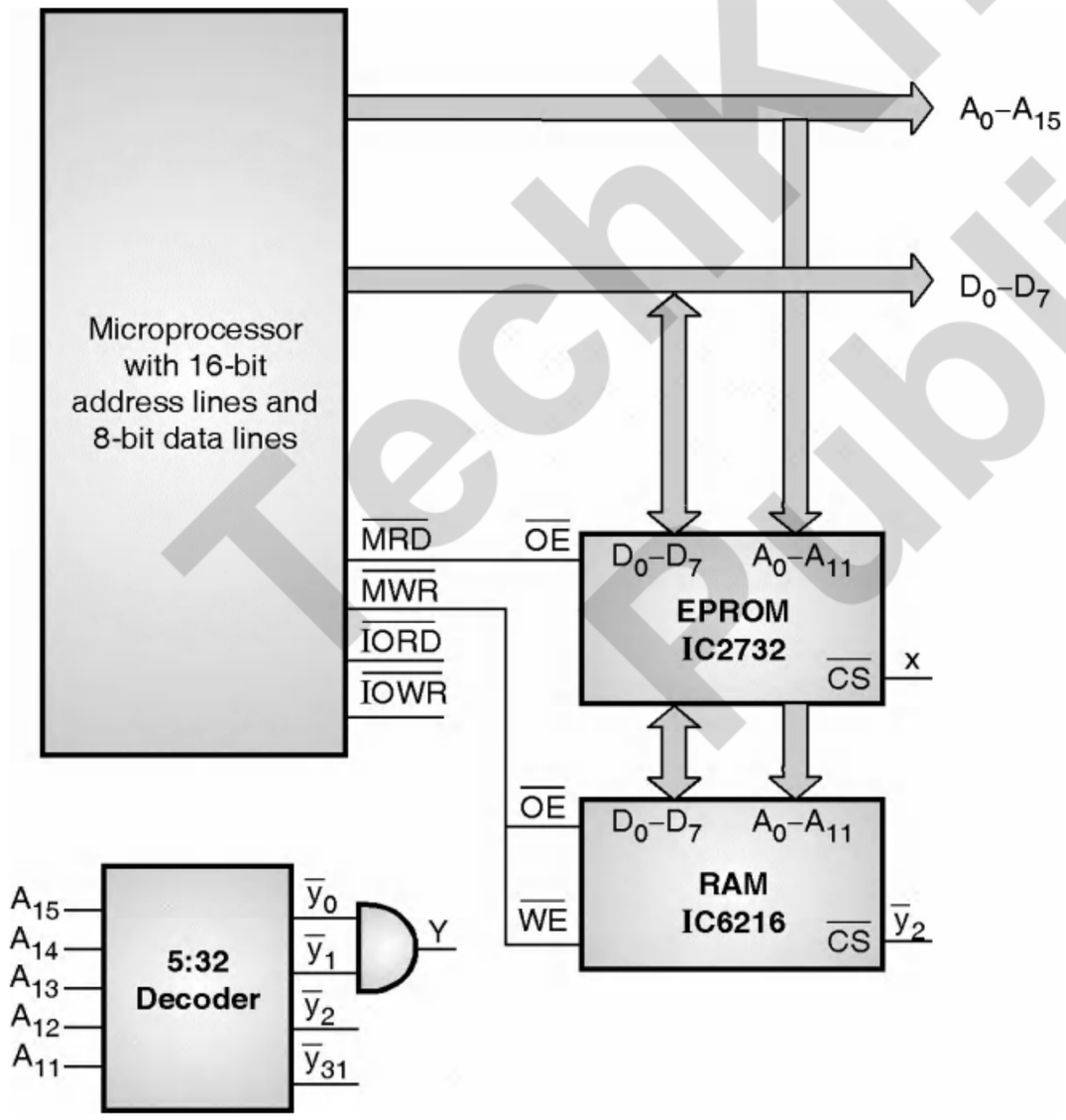


Fig. P. 10.8.1(a)

Ex. 10.8.2 : Interface 8K of EPROM and 8 KB of RAM using 4KB devices.

Soln. :

Note : Let us assume the processor has 16 address lines

Step 1 : Total EPROM required = 8 kB

Chip size available = 4 kB (IC 2732)

∴ No of chips required = $\frac{8\text{ kB}}{4\text{ kB}} = 2$

Chip 1 : Starting address = 0000H

Chip size = 4 kB = 0FFFH

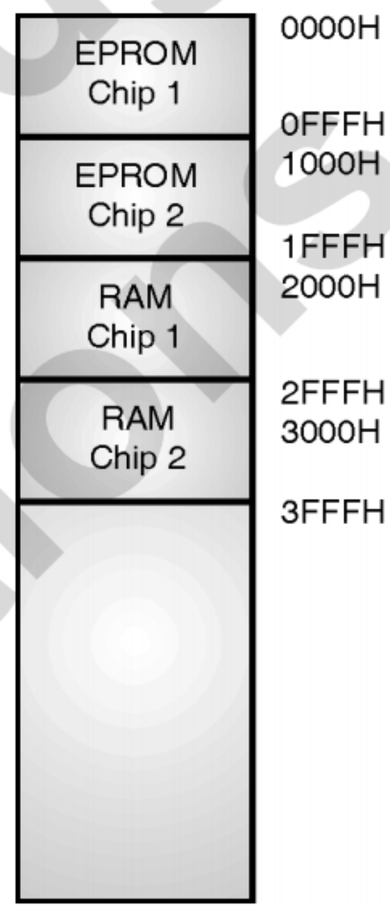
∴ Ending address = 0FFFH

Chip 2 : Starting address = Previous ending + 1

= 0FFFH + 1 = 1000H

Chip size = 0FFFH

∴ Ending address = 1FFFH



Step 2 : Total RAM required = 8 kB

Chip size available = 4 kB(IC 6232)

∴ No of chips required = $\frac{8\text{ kB}}{4\text{ kB}} = 2$

Chip 1 : Starting address = EPROM ending address + 1 = 1FFFH + 1 = 2000H

Chip size = 4 kB = 0FFFH

∴ Ending address = 2FFFH

Chip 2 : Starting address = previous ending + 1

= 2FFFH + 1 = 3000H

Chip size = 4 kB = 0FFFH

∴ Ending address = 3FFFH

TechKnowledge Publications

Step 3 : Memory Map :

		A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
EPROM chip 1 $\overline{y}_0$	SA = 0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 0FFFH	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
EPROM chip 2 $\overline{y}_1$	SA = 1000H	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 1FFFH	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM chip 1 $\overline{y}_2$	SA = 2000H	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 2FFFH	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM chip 2 $\overline{y}_3$	SA = 3000H	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	EA = 3FFFH	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Full decoding logic :

EPROM chip size = 4kB and RAM chip size = 4 kB.

$\therefore$ Smaller chip size (Both are same in this case) = 4 kB = 2^{12}

- Neglect lower 12 address lines i.e. A_0 to A_{11} and consider remaining address lines i.e. A_{12} to A_{15} for decoding.
- Hence 4:16 decoder is required. Hence circle the four bits as shown in the memory map.
- EPROM chip one has 0000b, thus it requires $\overline{y}_0$; while EPROM chip two has 0001b, thus it requires $\overline{y}_1$ and so on.

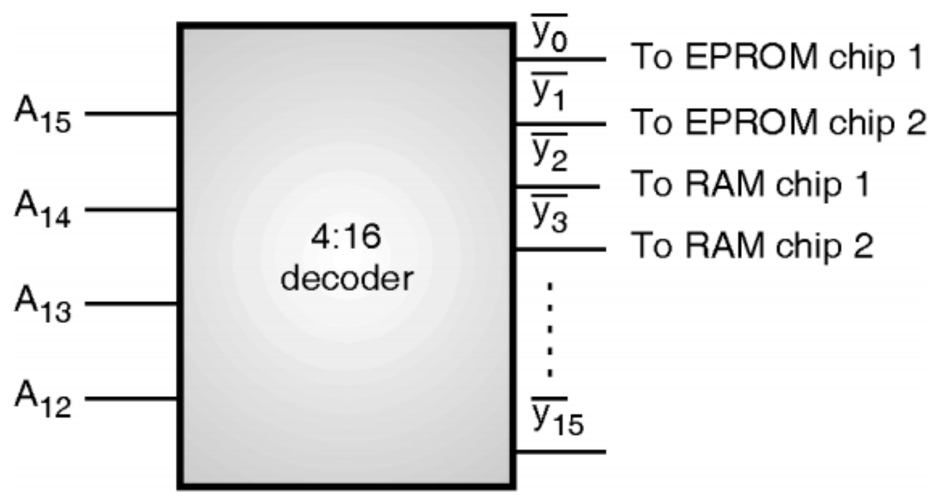


Fig. P. 10.8.2

Step 4 : Final implementation :

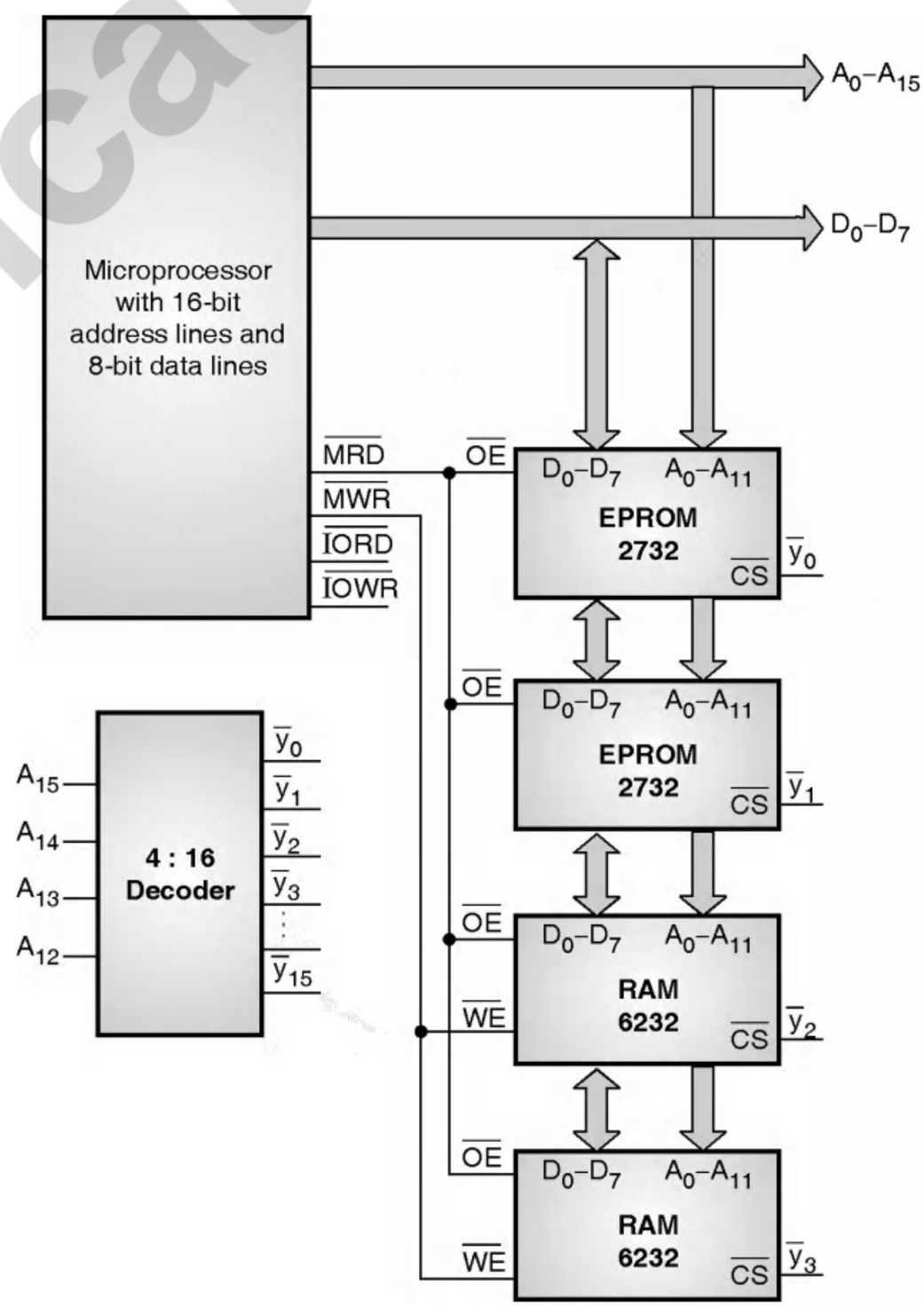


Fig. P. 10.8.2(a)

Ex. 10.8.3 : Interface 8 KB EPROM and 4 KB RAM to a processor with 16-bit address and 8-bit data bus.

Soln. :

Step 1 : Total EPROM required = 8 KB

Chip size = 8 KB (Assume)

∴ No of chips = 1

Chip 1 : Starting Address = 0000H

Chip size = 8 KB ⇒1FFF H

∴ Ending Address = 1FFF H

Step 2 : Total RAM required = 4 KB

Chip size = 4 KB (Assume)

∴ No. of chips = 1

Chip 1 : Starting Address = 2000H

Chip size = 4 KB ⇒ 0FFFH

∴ Ending Address = 2FFFH

Step 3 : Memory Map

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
EPROM Chip1 SA = 0000H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EA = 1FFFH	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{y_0}\cdot\overline{y_1}$																
RAM Chip1 SA = 2000H	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
EA = 2FFFH	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{y_2}$																

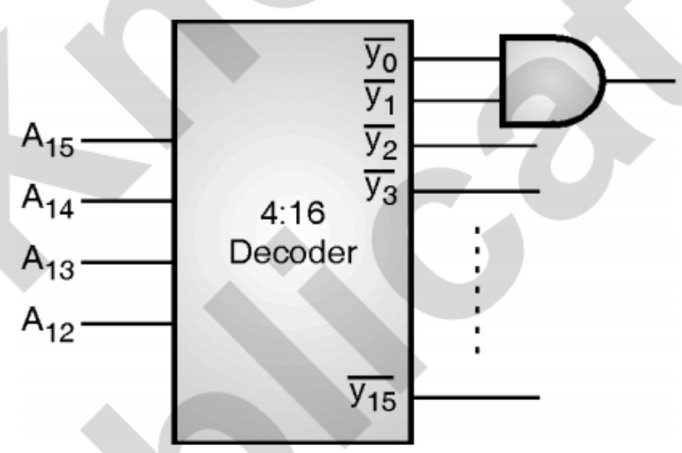


Fig. P. 10.8.3

Step 4 : Final Implementation :

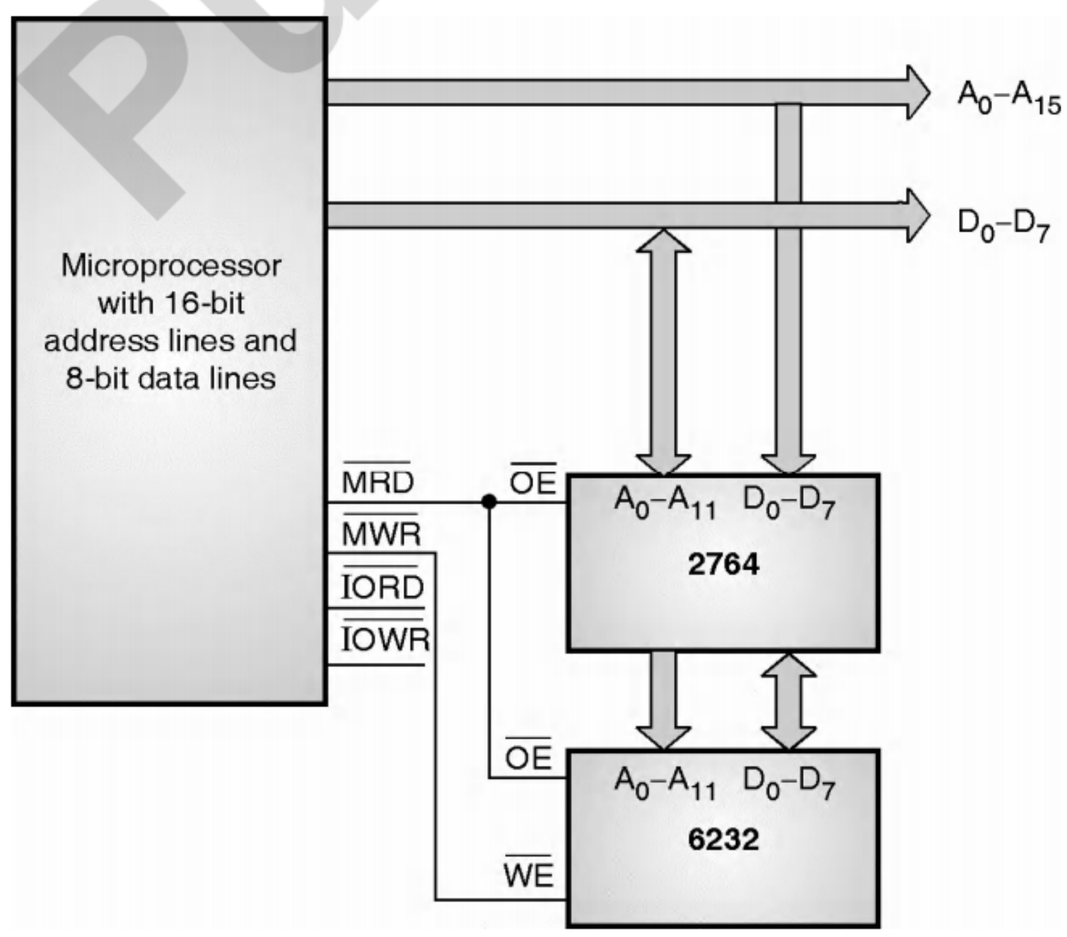


Fig. 10.8.3(a)

Ex. 10.8.4 : A computer system needs 512 bytes of RAM and 512 bytes of ROM. RAM chip available is of capacity 128 bytes and ROM chip available is of 512 bytes. Draw memory address map for a computer system and also draw a connection structure.

Soln. :

Assuming that the processor has 10 address lines to interface 1KB memory (512B + 512B)

Step 1 : Total EPROM required = 512 B
Chip size = 512B
∴ Number of chips required = 1
Chip 1 : Starting address = 0000H
Chip size = 512B ⇒ 01FFH
∴ Ending address = 01FFH

Step 2 : Total RAM required = 512B
Chip size available = 128B
∴ Total number of chips required = 4
Chip 1 : Starting address = Previous ending + 1
= 0200H
Chip size = 128B ⇒ 007FH
∴ Ending address = 027FH
Chip 2 : Starting address = Previous ending + 1
= 027FH + 1 = 0280H
Chip size = 128B ⇒ 007FH
∴ Ending address = 02FFH
Chip 3 : Starting address = Previous ending + 1
= 0300H
Chip size = 128B ⇒ 007FH
∴ Ending address = 037FH
Chip 4 : Starting address = Previous ending + 1
= 0380H
Chip size = 128B ⇒ 007FH
∴ Ending address = 03FFH

Step 3 : Memory map :

		A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
EPROM chip 1	SA = 0000H	0	0	0	0	0	0	0	0	0	0
	EA = 01FFH	0	1	1	1	1	1	1	1	1	1
$\overline{y_0}.\overline{y_1}.\overline{y_2}.\overline{y_3}$											
RAM chip 2	SA = 0200H	1	0	0	0	0	0	0	0	0	0
	EA = 027FH	1	0	0	1	1	1	1	1	1	1
$\overline{y_4}$											
RAM chip 1	SA = 0280H	1	0	1	0	0	0	0	0	0	0
	EA = 02FFH	1	0	1	1	1	1	1	1	1	1
$\overline{y_5}$											
RAM chip 2	SA = 0300H	1	1	0	0	0	0	0	0	0	0
	EA = 037FH	1	1	0	1	1	1	1	1	1	1
$\overline{y_6}$											
RAM chip 3	SA = 0380H	1	1	1	0	0	0	0	0	0	0
	EA = 03FFH	1	1	1	1	1	1	1	1	1	1
$\overline{y_7}$											

Absolute (full) decoding logic :

- EPROM chip size = 512B while RAM chipsize = 128B. Thus smaller chip size = 128B = 2^7.
- Therefore neglect lower 7 address lines i.e. A_0 to A_6. Now since three address lines are remaining, we need a 3 : 8 decoder.
- The remaining lines i.e. A_7 to A_9 will be inputs to the decoder, as shown by circles in memory map.

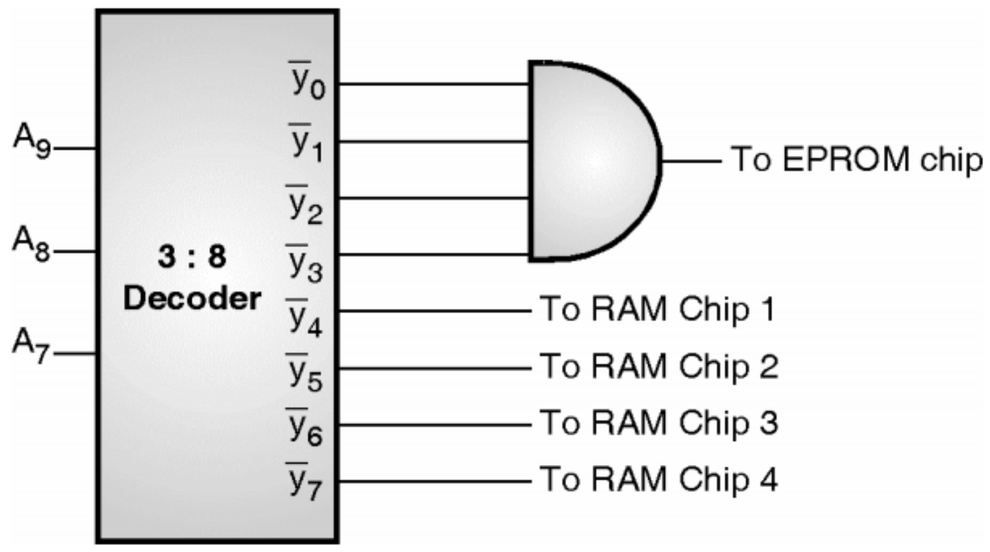


Fig. P. 10.8.4

Step 4 : Final Implementation :

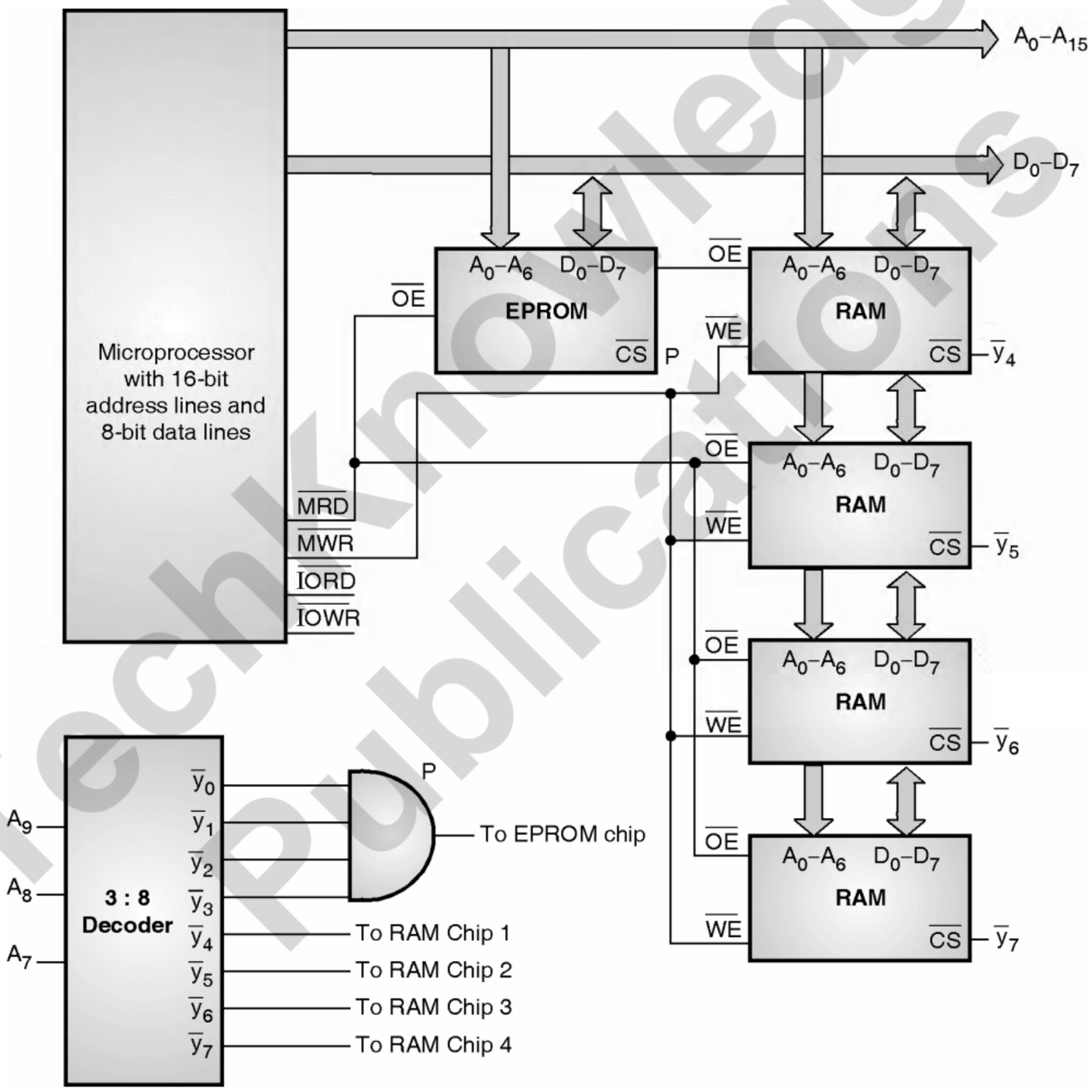


Fig. P. 10.8.4(a)

10.9 Exemplar / Case Studies : Microprocessor Based System in Communication / Instrumentation Control

- We will take an example of the car cruise control.
- It is a mechanism that automatically controls the constant speed of the car, when in cruise mode.
- It also takes care of reducing the speed when there is an obstruction nearing in front of the car.
- It will be example of Instrumentation control system.

10.9.1 Requirement Analysis

- We will need a sensor to measure the actual speed of the car.
- There has to be an actuator that increases or decreases the acceleration and hence the speed of the car.

- Another sensor based assembly to find obstructions in front of the car.
- Similarly another actuator to apply and release the brake of the car.

10.9.2 Hardware Block Diagram

- There are separate sensors to measure the speed and the objects in front of the car as discussed earlier.
- The speed sensor could be a simple photo sensor to count the number of revolutions of the car wheel and accordingly measure the speed.
- The obstruction sensor has to be a photo sensor like ultrasonic, Infra Red (IR) etc. The input from the speed sensor is to be converted in digital form to be fed to the microcontroller.
- The obstruction sensor input is to be given to the microcontroller through the detector or the optocoupler circuit.
- The microcontroller compares the actual speed and the constant speed for cruise control.
- In case if the actual speed is less, it gives an acceleration through the actuator and deceleration and brake in case of higher speed.
- Also in case if the obstruction is found in front of the car, brakes are applied slowly and the car is decelerated to reduce the speed.
- The Fig. 10.9.2 shows the Finite State Model (FSM) of the cruise control.
- The sensors measure various parameters i.e. speed and obstruction and accordingly feed the input to the microcontroller and finally the actuators are controlled according to the control required for the car.

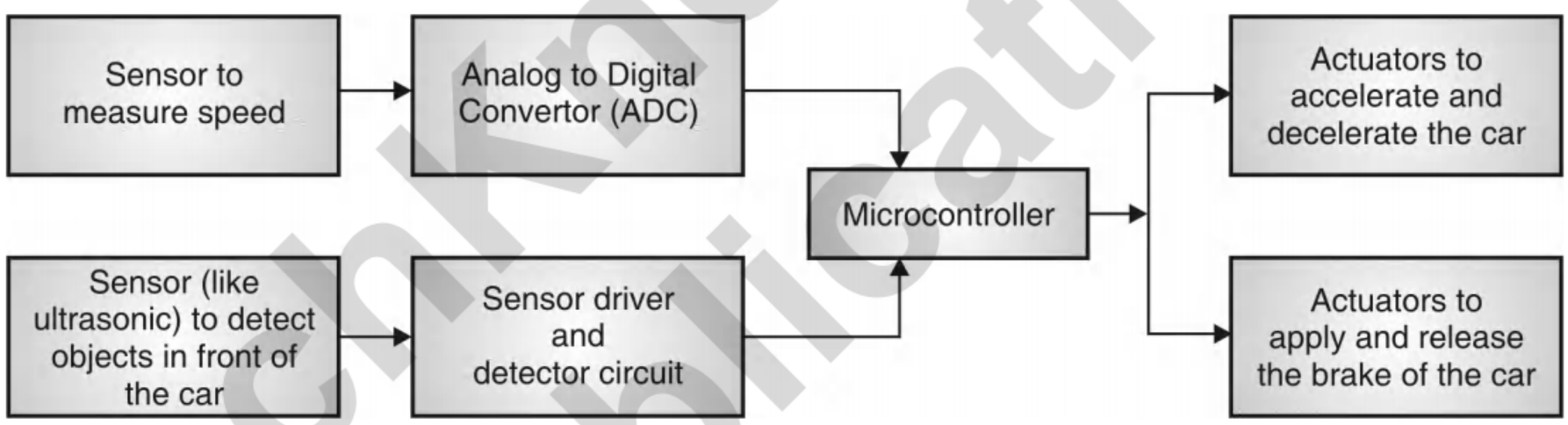


Fig. 10.9.1 : Hardware block diagram for automatic cruise control for a car

10.9.3 Software Model for Car Cruise Control

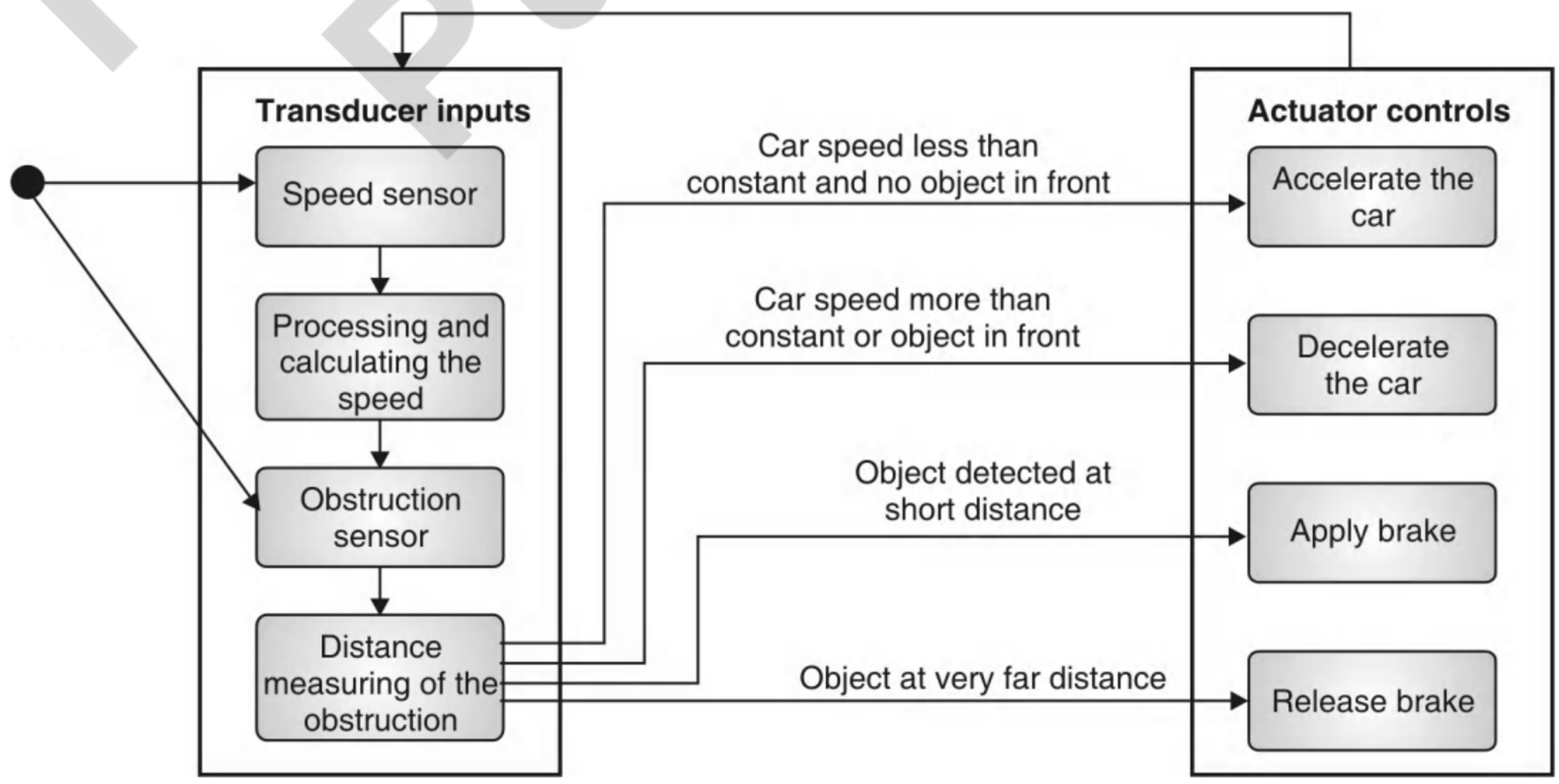


Fig. 10.9.2 : Cruise control software model

- There are various software modules required in a cruise control system.
- There has to be two modules for the input from sensors.
- The speed measure module has to continuously count the pulses from the speed sensor and accordingly calculate the actual speed of the car.
- Another module to detect objects at distance less safe, which should be managed using interrupt.
- Separate modules will be required to control the actuators.
- These modules will use the data shared by the sensor modules and accordingly control the actuators.
- The shared data will be used by the accelerate module to accelerate and decelerate the car.
- Similarly, the brake module will access the shared data from the sensor modules and apply or release brake by controlling the actuators.

10.9.4 Component / Hardware Selection

- The components required are the sensors, microcontroller and actuators.
- As discussed earlier, the speed sensor which is a photo sensor is used to indicate the completion of one revolution of the wheel, and hence measure the speed of the car.
- The count of pulses in one minute will indicate the number of revolutions in one minute and hence the speed of the car.
- To measure the speed in distance per second, the number of revolutions is multiplied with the circumference of the wheel. Another sensor as discussed can be a simple ultrasound distance sensor.
- This sensor has a ultrasound transmitter and receiver.
- The transmitter transmits ultrasound signal which is received by the ultrasound receiver and the time between transmitted signal and received signal is used to measure the distance to the obstruction in front.
- This sensor is mounted in front of the car to measure object distance.
- The information of the distance is given to the microcontroller to decide the application of brake or not.
- The actuators can be simple electronic circuit to control the acceleration and break control.

Review Questions

Q. 1 What are the different characteristics of microprocessor ?

Q. 2 Give the functions of microprocessor.

Q. 3 What are the different types of buses used in microprocessor ?

Q. 4 With neat block diagram explain microcomputer.

Q. 5 With neat block diagram explain generic microprocessor.

Q. 6 What are the different characteristics of memory system ?

Q. 7 Write short note on memory hierarchy.

Q. 8 Compare SRAM and DRAM.

Q. 9 What are the different types of ROM ?

Q. 10 With neat hardware and software block diagram explain automatic cruise control for a cars

TechKnowledge Publications