Department of Electrical & Electronics Engineering

Course File

DIGITAL ELECTRONICS

(Course Code: GR20A2026)

II B.Tech II Semester

2023-24

Mr.M.Srinu

Assist Professor

ANURAG

Anurag

ENGINEERING COLLEGE

(An Autonomous Institution)

Ananthagiri, Kodad, Telangana 508 206, India.

Department of Electrical & Electronics Engineering

DIGITAL ELECTRONICS

Check List

Department of Electrical & Electronics Engineering

Int. Marks:30 **Ext. Marks:70** **Total Marks:100**

ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

II Year B.Tech. EEE - II Sem

L	T	P	C
2	0	0	2

(EE404PC) DIGITAL ELECTRONICS

UNIT-I: **Fundamentals of Digital Systems and Logic Families:** Digital signals, Digital circuits, AND, OR, NOT, NAND, NOR and Exclusive-OR operations.

Boolean algebra, Examples of IC gates, Number systems-binary, Signed binary, Octal hexadecimal number, Binary arithmetic, One's and Two's complements arithmetic.

UNIT-II: Combinational Circuits-I: Standard representation for logic functions, K-map representation and simplification of logic functions using K- map, Minimization of logical functions, Don't care conditions, Multiplexer, De-Multiplexer

UNIT-III: Combinational Circuits-II: Adders, Subtractors, Carry look ahead adder, Digital comparator, Parity checker/generator, Code converters, Priority encoders, Decoders/Drivers for display devices, Q-M method of function realization.

UNIT-IV: Sequential Circuits: Introduction to flip-flops, SR, JK, T and D type's flip-flops, Shift registers, Conversion of flip-flops, Ring counter, Ripple (Asynchronous) counters, Synchronous counters.

UNIT-V:
Semiconductor Memories and Programmable Logic Devices: Memory organization and operation, expanding memory size, classification and characteristics of memories, sequential memory, read-only memory (ROM), ROM types, Read and write memory (RAM) types, Programmable logic array, Programmable array logic, Field Programmable Gate Array (FPGA).

TEXT BOOKS:

1. A. Kumar, "Fundamentals of Digital Circuits", Prentice Hall India, 2016.
2. M. M. Mano, "Digital logic and Computer design", Pearson Education India, 2016.

REFERENCE BOOKS:

1. R.S. Sedha, "A Textbook of Digital Electronics", S.Chand, 2005
2. R. P. Jain, "Modern Digital Electronics", McGraw Hill Education,2009.

Course Outcomes: After learning the contents of this paper the student must be able to

Department of Electrical & Electronics Engineering

Timetable

II B.Tech. I Semester – EMF (A Sec)

Day/Hour	9.40-10.30	10.30-11.20	11.20-12.00	12.00-12.55	12.55-1.50	1.50-2.45	2.45-3.50
Monday	DE						
Tuesday			DE				
Wednesday	DE						
Thursday				DE			
Friday					DE		
Saturday					DE		

Department of Electrical & Electronics Engineering

Vision of the Institute

To be a premier Institute in the country and region for the study of Engineering, Technology and Management by maintaining high academic standards which promotes the analytical thinking and independent judgment among the prime stakeholders, enabling them to function responsibly in the globalized society.

Mission of the Institute

To be a world-class Institute, achieving excellence in teaching, research and consultancy in cutting-edge Technologies and be in the service of society in promoting continued education in Engineering, Technology and Management.

Quality Policy

To ensure high standards in imparting professional education by providing world-class infrastructure, top-quality-faculty and decent work culture to sculpt the students into Socially Responsible Professionals through creative team-work, innovation and research

Vision of the Department

To impart technical knowledge and skills required to succeed in life, career and help society to achieve self sufficiency.

Mission of the Department

- To become an internationally leading department for higher learning.
- To build upon the culture and values of universal science and contemporary education.
- To be a center of research and education generating knowledge and technologies which lay groundwork in shaping the future in the fields of electrical and electronics engineering.
- To develop partnership with industrial, R&D and government agencies and actively participate in conferences, technical and community activities.

Department of Electrical & Electronics Engineering

Program Educational Objectives (B.Tech. – EEE)
Graduates will be able to

PEO 1: Have a successful technical or professional career, including supportive and leadership roles on multidisciplinary teams.

PEO 2: Acquire, use and develop skills as required for effective professional practices.

PEO 3: Able to attain holistic education that is an essential prerequisite for being a responsible member of society.

Program Outcomes (B.Tech. – EEE)
At the end of the Program, a graduate will have the ability to

PO 1: Apply knowledge of mathematics, science, and engineering.

PO 2: Design and conduct experiments, as well as to analyze and interpret data.

PO 3: Design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.

PO 4: Function on multi-disciplinary teams.

PO 5: Identify, formulates, and solves engineering problems.

PO 6: Understanding of professional and ethical responsibility.

PO 7: Communicate effectively.

PO 8: Broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context.

PO 9: Recognition of the need for, and an ability to engage in life-long learning.

PO 10: Knowledge of contemporary issues.

PO 11: Utilize experimental, statistical and computational methods and tools necessary for engineering practice.

PO 12: Demonstrate an ability to design electrical and electronic circuits, power electronics, power systems; electrical machines analyze and interpret data and also an ability to design digital and analog systems and programming them.

Department of Electrical & Electronics Engineering

COURSE OBJECTIVES

On completion of this Subject/Course the student shall be able to:

S.No	Objectives
1	To learn fundamental concepts of digital system design and common forms of numberrepresentations and their conversions.
2	To implement and design logical operations using combinational logic circuits and sequentiallogic circuits.
3	To be able to convert between different code converters.
4	To understand the operations of different types of flip-flops and counters.
5	To learn about Semiconductor Memories and Programmable Logic Devices

COURSE OUTCOMES

The expected outcomes of the Course/Subject are:

S.No	Outcomes
1.	Understand the working of logic families and logic gates.
2.	Design and implement Combinational and Sequential logic circuits.
3.	Implement the conversion between converters.
4.	Design logical circuits using different flip flops.
5.	To learn about Semiconductor Memories and Programmable Logic Devices.

Signature of faculty

Note: Please refer to Bloom’s Taxonomy, to know the illustrative verbs that can be used to state the outcomes.

Department of Electrical & Electronics Engineering

GUIDELINES TO STUDY THE COURSE / SUBJECT

Course Design and Delivery System (CDD):

- The Course syllabus is written into number of learning objectives and outcomes.
- Every student will be given an assessment plan, criteria for assessment, scheme of evaluation and grading method.
- The Learning Process will be carried out through assessments of Knowledge, Skills and Attitude by various methods and the students will be given guidance to refer to the text books, reference books, journals, etc.

The faculty be able to –

- Understand the principles of Learning
- Understand the psychology of students
- Develop instructional objectives for a given topic
- Prepare course, unit and lesson plans
- Understand different methods of teaching and learning
- Use appropriate teaching and learning aids
- Plan and deliver lectures effectively
- Provide feedback to students using various methods of Assessments and tools of Evaluation
- Act as a guide, advisor, counselor, facilitator, motivator and not just as a teacher alone

Signature of HOD　　　　　　　　　　　　Signature of faculty

Date:　　　　　　　　　　　　　　　　　Date:

Department of Electrical & Electronics Engineering

COURSE SCHEDULE

The Schedule for the whole Course / Subject is:

S. No.	Description	Duration (Date)		Total No. of Periods
		From	To	
1.	**UNIT-I: Fundamentals of Digital Systems and Logic Families:** Digital signals, Digital circuits, AND, OR, NOT, NAND, NOR and Exclusive-OR operations. Boolean algebra, Examples of IC gates, Number systems-binary, Signed binary, Octal hexadecimal number, Binary arithmetic, One's and Two's complements arithmetic.	05.023.2024	22.02.2024	18
2.	**UNIT-II: Combinational Circuits-I:** Standard representation for logic functions, K-map representation and simplification of logic functions using K- map, Minimization of logical functions, Don't care conditions, Multiplexer, De-Multiplexer	23.02.2024	05.03.2024	12
3.	**UNIT-III: Combinational Circuits-II:** Adders, Subtractors, Carry look ahead adder, Digital comparator, Parity checker/generator, Code converters, Priority encoders, Decoders/Drivers for display devices, Q-M method of function realization.	06.03.2024	24.03.2024	19
4.	**UNIT-IV: Sequential Circuits:** Introduction to flip-flops, SR, JK, T and D type's flip-flops, Shift registers, Conversion of flip-flops, Ring counter, Ripple (Asynchronous) counters, Synchronous counte	19.03.2024	12.04.2024	19
5.	**UNIT-V: Semiconductor Memories and Programmable Logic Devices**: Memory organization and operation, expanding memory size, classification and characteristics of memories, sequential memory, read-only memory (ROM), ROM types, Read and write memory (RAM) types, Programmable logic array, Programmable array logic, Field Programmable Gate Array (FPGA).	13.04.2024	26.04.2024	14

Total No. of Instructional periods available for the course: 73 Hours

Department of Electrical & Electronics Engineering

SCHEDULE OF INSTRUCTIONS - COURSE PLAN

Unit No.	Lesson No.	Date	No. of	Topics / Sub-Topics	Objectives &	References
			Periods		Outcomes	(Textbook, Journal)
					Nos.	
	1	02-05-24	1	UNIT-I: Fundamentals of Digital Systems and Logic Families introduction	1 1	"Digital logic and Computer design”- M. Mano,
	2	02-06-24	1	Digital signals	1 1	"Digital logic and Computer design”- M. Mano,
	3	02-07-24	1	Digital circuits	1 1	"Digital logic and Computer design”- M. Mano,
	4	02-08-24	1	AND, OR, NOT gate operations	1 1	"Digital logic and Computer design”- M. Mano,
	5	02-09-24	1	NAND, NOR operations	1 1	"Digital logic and Computer design”- M. Mano,
	6	02-10-24	1	Exclusive-OR operations	1 1	"Digital logic and Computer design”- M. Mano,
	7	02-11-24	1	Numarical problems	1 1	"Digital logic and Computer design”- M. Mano,
	8	02-12-24	1	Numarical problems	1 1	"Digital logic and Computer design”- M. Mano,
	9	02-13-24	1	Boolean algebra,	1 1	"Digital logic and Computer design”- M. Mano,
	10	02-14-24	1	Examples of IC gates	1 1	"Digital logic and Computer design”- M. Mano,
	11	02-15-24	1	Examples of IC gates	1 1	"Digital logic and Computer design”- M. Mano,
	12	02-16-24	1	Signed binary	1 1	"Digital logic and Computer design”- M. Mano,
	13	02-17-24	1	Octal hexadecimal number	1 1	"Digital logic and Computer design”- M. Mano,
	14	02-18-24	1	Binary arithmetic	1	"Digital logic and

					1	Computer design”- M. Mano,
	15	02-19-24	1	One’s and Two’s complements arithmetic	1 1	"Digital logic and Computer design”- M. Mano,
	16	02-20-24	1	One’s and Two’s complements arithmetic	1 1	"Digital logic and Computer design”- M. Mano,
	17	02-21-24	1	Numarical problems	1 1	"Digital logic and Computer design”- M. Mano,
	18	02-22-24	1	Numarical problems	1 1	"Digital logic and Computer design”- M. Mano,
	19	02-23-24	1	UNIT-II: Combinational Circuits-I:	2 2	"Digital logic and Computer design”- M. Mano,
	20	02-24-24	1	Standard representation for logic functions	2 2	"Digital logic and Computer design”- M. Mano,
	21	02-25-24	1	K-map representation	2 2	"Digital logic and Computer design”- M. Mano,
	22	02-26-24	1	simplification of logic functions using K- map	2 2	"Digital logic and Computer design”- M. Mano,
	23	02-27-24	1	Numarical problems	2 2	"Digital logic and Computer design”- M. Mano,
	24	02-28-24	1	Numarical problems	2 2	"Digital logic and Computer design”- M. Mano,
	25	02-29-24	1	Minimization of logical functions	2 2	"Digital logic and Computer design”- M. Mano,
	26	03-01-24	1	Numarical problems	2 2	"Digital logic and Computer design”- M. Mano,
	27	03-02-24	1	Don’t care conditions	2 2	"Digital logic and Computer design”- M. Mano,
	28	03-03-24	1	Numarical problems	2 2	"Digital logic and Computer design”- M. Mano,
	29	03-04-24	1	Multiplexer	2 2	"Digital logic and Computer design”- M. Mano,
	30	03-05-24	1	De-Multiplexer	2	"Digital logic and

					2	Computer design”- M. Mano,
	31	03-06-24	1	UNIT-III: Combinational Circuits-II:	3 3	"Digital logic and Computer design”- M. Mano,
	32	03-07-24	1	Adders	3 3	"Digital logic and Computer design”- M. Mano,
	33	03-08-24	1	Subtractors	3 3	"Digital logic and Computer design”- M. Mano,
	34	03-09-24	1	Carry look ahead adder	3 3	"Digital logic and Computer design”- M. Mano,
	35	03-10-24	1	Digital comparator	3 3	"Digital logic and Computer design”- M. Mano,
	36	03-11-24	1	Parity checker/generator	3 3	"Digital logic and Computer design”- M. Mano,
	37	03-12-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	38	03-13-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	39	03-14-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	40	03-15-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	41	03-16-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	42	03-17-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	43	03-18-24	1	Numarical problems	3 3	"Digital logic and Computer design”- M. Mano,
	44	03-19-24	1	Code converters	3 3	"Digital logic and Computer design”- M. Mano,
	45	03-20-24	1	Priority encoders	3 3	"Digital logic and Computer design”- M. Mano,
	46	03-21-24	1	Numarical problems	3	"Digital logic and

					3	Computer design"- M. Mano,
	47	03-22-24	1	Decoders/Drivers for display devices	3 3	"Digital logic and Computer design"- M. Mano,
	48	03-23-24	1	Q-M method of function realization.	3 3	"Digital logic and Computer design"- M. Mano,
	49	03-24-24	1	Numarical problems	3 3	"Digital logic and Computer design"- M. Mano,
	50	03-25-24	1	UNIT-IV: Sequential Circuits:	4 4	"Digital logic and Computer design"- M. Mano,
	51	03-26-24	1	Introduction to flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	52	03-27-24	1	JK, type flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	53	03-28-24	1	T type flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	54	03-29-24	1	D type flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	55	03-30-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	56	03-31-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	57	04-01-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	58	04-02-24	1	Shift registers	4 4	"Digital logic and Computer design"- M. Mano,
	59	04-03-24	1	Conversion of flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	60	04-04-24	1	Conversion of flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	61	04-05-24	1	Conversion of flip-flops	4 4	"Digital logic and Computer design"- M. Mano,
	62	04-06-24	1	Conversion of flip-flops	4	"Digital logic and

					4	Computer design"- M. Mano,
	63	04-07-24	1	Ring counter	4 4	"Digital logic and Computer design"- M. Mano,
	64	04-08-24	1	Ripple (Asynchronous) counters	4 4	"Digital logic and Computer design"- M. Mano,
	65	04-09-24	1	Synchronous counters	4 4	"Digital logic and Computer design"- M. Mano,
	66	04-10-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	67	04-11-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	68	04-12-24	1	Numarical problems	4 4	"Digital logic and Computer design"- M. Mano,
	69	04-13-24	1	UNIT-V: Semiconductor Memories and Programmable Logic Devices:	5 5	"Digital logic and Computer design"- M. Mano,
	70	04-14-24	1	Memory organization operation,	5 5	"Digital logic and Computer design"- M. Mano,
	71	04-15-24	1	expanding memory size	5 5	"Digital logic and Computer design"- M. Mano,
	72	04-16-24	1	classification and characteristics of memories	5 5	"Digital logic and Computer design"- M. Mano,
	73	04-17-24	1	sequential memory,	5 5	"Digital logic and Computer design"- M. Mano,
	74	04-18-24	1	read-only memory (ROM)	5 5	"Digital logic and Computer design"- M. Mano,
	75	04-19-24	1	ROM types	5 5	"Digital logic and Computer design"- M. Mano,
	76	04-20-24	1	Read and write memory (RAM) types	5 5	"Digital logic and Computer design"- M. Mano,
	77	04-21-24	1	Programmable logic array	5 5	"Digital logic and Computer design"- M. Mano,
	78	04-22-24	1	Programmable logic array	5	"Digital logic and

					5	Computer design”- M. Mano,
	79	04-23-24	1	Numarical problems	5 5	"Digital logic and Computer design”- M. Mano,
	80	04-24-24	1	Numarical problems	5 5	"Digital logic and Computer design”- M. Mano,
	81	04-25-24	1	Field Programmable Gate Array (FPGA).	5 5	"Digital logic and Computer design”- M. Mano,
	82	04-26-24	1	Numarical problems	5 5	"Digital logic and Computer design”- M. Mano,

Signature of HOD Signature of faculty

Date: Date:

Note:

1. Ensure that all topics specified in the course are mentioned.
2. Additional topics covered, if any, may also be specified in bold.
3. Mention the corresponding course objective and outcome numbers against each topic.

Department of Electrical & Electronics Engineering

Unit No.	Lesson No.	Date	DAY	Topics / Sub-Topics
	1	02-05-24	MON	UNIT-I: Fundamentals of Digital Systems and Logic Families introduction
	2	02-06-24	TUE	Digital signals
	3	02-07-24	WED	Digital circuits
	4	02-08-24	THU	AND, OR, NOT gate operations
	5	02-09-24	FRI	NAND, NOR operations
	6	02-10-24	MON	Exclusive-OR operations
	7	02-11-24	TUE	Numarical problems
	8	02-12-24	WED	Numarical problems
	9	02-13-24	THU	Boolean algebra,
	10	02-14-24	FRI	Examples of IC gates
	11	02-15-24	SAT	Examples of IC gates
	12	02-16-24	MON	Signed binary
	13	02-17-24	TUE	Octal hexadecimal number
	14	02-18-24	WED	Binary arithmetic
	15	02-19-24	THU	One’s and Two’s complements arithmetic
	16	02-20-24	FRI	One’s and Two’s complements arithmetic
	17	02-21-24	SAT	Numarical problems
	18	02-22-24	MON	Numarical problems
	19	02-23-24	TUE	UNIT-II: Combinational Circuits-I:
	20	02-24-24	WED	Standard representation for logic functions
	21	02-25-24	THU	K-map representation
	22	02-26-24	FRI	simplification of logic functions using K- map
	23	02-27-24	SAT	Numarical problems
	24	02-28-24	MON	Numarical problems
	25	02-29-24	TUE	Minimization of logical functions

	26	03-01-24	WED	Numarical problems
	27	03-02-24	THU	Don’t care conditions
	28	03-03-24	MON	Numarical problems
	29	03-04-24	TUE	Multiplexer
	30	03-05-24	WED	De-Multiplexer
	31	03-06-24	THU	UNIT-III: Combinational Circuits-II:
	32	03-07-24	FRI	Adders
	33	03-08-24	SAT	Subtractors
	34	03-09-24	MON	Carry look ahead adder
	35	03-10-24	TUE	Digital comparator
	36	03-11-24	WED	Parity checker/generator
	37	03-12-24	THU	Numarical problems
	38	03-13-24	FRI	Numarical problems
	39	03-14-24	SAT	Numarical problems
	40	03-15-24	TUE	Numarical problems
	41	03-16-24	WED	Numarical problems
	42	03-17-24	THU	Numarical problems
	43	03-18-24	SAT	Numarical problems
	44	03-19-24	THU	Code converters
	45	03-20-24	SAT	Priority encoders
	46	03-21-24	MON	Numarical problems
	47	03-22-24	WED	Decoders/Drivers for display devices
	48	03-23-24	MON	Q-M method of function realization.
	49	03-24-24	TUE	Numarical problems
	50	03-25-24	THU	UNIT-IV: Sequential Circuits:
	51	03-26-24	FRI	Introduction to flip-flops
	52	03-27-24	MON	JK, type flip-flops
	53	03-28-24	TUE	T type flip-flops
	54	03-29-24	WED	D type flip-flops
	55	03-30-24	THU	Numarical problems
	56	03-31-24	FRI	Numarical problems
	57	04-01-24	SAT	Numarical problems
	58	04-02-24	MON	Shift registers
	59	04-03-24	TUE	Conversion of flip-flops
	60	04-04-24	WED	Conversion of flip-flops
	61	04-05-24	THU	Conversion of flip-flops
	62	04-06-24	FRI	Conversion of flip-flops

	63	04-07-24	SAT	Ring counter
	64	04-08-24	MON	Ripple (Asynchronous) counters
	65	04-09-24	TUE	Synchronous counters
	66	04-10-24	WED	Numarical problems
	67	04-11-24	THU	Numarical problems
	68	04-12-24	FRI	Numarical problems
	69	04-13-24	MON	UNIT-V: Semiconductor Memories and Programmable Logic Devices:
	70	04-14-24	TUE	Memory organization operation,
	71	04-15-24	WED	expanding memory size
	72	04-16-24	THU	classification and characteristics of memories
	73	04-17-24	FRI	sequential memory,
	74	04-18-24	SAT	read-only memory (ROM)
	75	04-19-24	MON	ROM types
	76	04-20-24	TUE	Read and write memory (RAM) types
	77	04-21-24	WED	Programmable logic array
	78	04-22-24	THU	Programmable logic array
	79	04-23-24	FRI	Numarical problems
	80	04-24-24	MON	Numarical problems
	81	04-25-24	TUE	Field Programmable Gate Array (FPGA).
	82	04-26-24	WED	Numarical problems

Signature of HOD

Date:

Signature of faculty

Date:

Note:

4. Ensure that all topics specified in the course are mentioned.
5. Additional topics covered, if any, may also be specified in bold.
6. Mention the corresponding course objective and outcome numbers against each topic.

Department of Electrical & Electronics Engineering

ASSIGNMENT – 1

This Assignment corresponds to Unit No. 1

Question No.	Question	Objective No.	Outcome No.
1	(i)Convert $(A0F90EBA)_{16}$ to equivalent octal, decimal and binary values. (ii) Perform Excess-3 addition of $(8)_{10}$ and $(6)_{10}$	1	1
2	Discuss about logic gates.	1	1

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

ASSIGNMENT – 2

This Assignment corresponds to Unit No. 2

Question No.	Question	Objective No.	Outcome No.
1	Simplify the fallowing expression using K-Map F(A,B,C,D)=∑m(0,8,6,13,14)+ ∑d(2,4,10).	2	2
2	Minimize the expression using K-Map F(A,B,C,D)=πM(0,2,3,8,9,12,13,15) F(A,B,C,D))=∑m(1,3,7,11,15)+ ∑d(0,2,5).	2	2

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

ASSIGNMENT – 3

This Assignment corresponds to Unit No. 3

Question No.	Question	Objective No.	Outcome No.
1	Explain about binary adder - subtractor circuit with example.	3	3
2	Define circuit and combinational circuit and sequential circuit and compare combinational sequential circuits.	3	3

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

ASSIGNMENT – 4

This Assignment corresponds to Unit No. 4

Question No.	Question	Objective No.	Outcome No.
1	Explain the operation of JK master slave flip-Flop. Explain its truth table.	4	4
2	Discuss registers and counters	4	4

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

ASSIGNMENT – 5

This Assignment corresponds to Unit No. 5

Question No.	Question	Objective No.	Outcome No.
1	Define main memory, RAM, ROM, and types of ROM.	4	4
2	Discuss about Auxiliary memory & Associative memory.	4	4

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

TUTORIAL – 1

This tutorial corresponds to Unit No. 1 (Objective Nos.: 1, Outcome Nos.: 1)

Q1. Any signed negative binary number is recognised by its ________
a) MSB
b) LSB
c) Byte
d) Nibble

Q2.The parameter through which 16 distinct values can be represented is known as ________
a) Bit
b) Byte
c) Word
d) Nibble

Q3.If the decimal number is a fraction then its binary equivalent is obtained by ________ the number continuously by 2.
a) Dividing
b) Multiplying
c) Adding
d) Subtracting

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

TUTORIAL – 2

This tutorial corresponds to Unit No. 2 (Objective Nos.: 2, Outcome Nos.: 2)

Q1The terms in SOP are called ___________
a) max terms
b) min terms
c) mid terms
d) sum terms
Q2. Which operation is shown in the following expression: (X+Y).(X+Z).(Z+Y)
a) NOR
b) ExOR
c) SOP
d) POS
Q3. The expression Y=AB+BC+AC shows the _________ operation.
a) EX-OR
b) SOP
c) POS
d) NOR

Signature of HOD Signature of faculty

Date: Date:

Department of Electrical & Electronics Engineering

TUTORIAL SHEET – 3

This tutorial corresponds to Unit No. 3 (Objective Nos.: 3, Outcome Nos.: 3)

Q1. The basic building blocks of the arithmetic unit in a digital computers are ____________
a) Subtractors
b) Adders
c) Multiplexer
d) Comparator

Q2.A digital system consists of _____ types of circuits.
a) 2
b) 3
c) 4
d) 5

Q3.In a combinational circuit, the output at any time depends only on the _______ at that time.
a) Voltage
b) Intermediate values
c) Input values
d) Clock pulses

Signature of HOD | Signature of faculty

Date: | Date:

Department of Electrical & Electronics Engineering

TUTORIAL – 4

This tutorial corresponds to Unit No. 4 (Objective Nos.: 3, Outcome Nos.: 3)

Q1. A latch is an example of a

a) Monostable multivibrator

b) Astable multivibrator

c) Bistable multivibrator

d) None of the Mentione

Q 2.The NAND latch works when both inputs are
a) 1
b) 0
c) Inverted
d) Don't cares

Q 3.The truth table for an S-R flip-flop has how many VALID entries?

a) 1

b) 2

c) 3

d) 4

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

TUTORIAL SHEET – 5

This tutorial corresponds to Unit No. 5 (Objective Nos.: 5, Outcome Nos.: 5)

Q1. Which of the following memories are non volatile memories

A. PROM
B. Ferrite core memory
C. None of the above
D. ROM

2. The access time of bipolar RAM is of the order of

A. 20 micro sec
B. 20 milli sec
C. 20 sec
D. 20 nSec

3. The access time of MOS RAM is of the order of

A. 1 micro sec
B. 1 milli sec
C. 1 sec1
D. 1 nano sec

Signature of HOD	Signature of faculty
Date:	Date:

Department of Electrical & Electronics Engineering

EVALUATION STRATEGY

Target (s)

a. Percentage of Pass : 95%

Assessment Method (s) (Maximum Marks for evaluation are defined in the Academic Regulations)

a. Assignments
b. Online Quiz (or) Seminars
c. Continuous Internal Assessment
d. Semester / End Examination

List out any new topic(s) or any innovation you would like to introduce in teaching the subjects in this semester

Case Study of any one existing application

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

COURSE COMPLETION STATUS

Actual Date of Completion & Remarks if any

Units	Remarks	Objective No. Achieved	Outcome No. Achieved
Unit 1	completed on 28.10.2022	1	1
Unit 2	completed on 21.11.2022	2	2
Unit 3	completed on 16.12.2022	3	3
Unit 4	completed on 12.01.2023	4	4
Unit 5	completed on 06.02.2023	5	5

Signature of HOD

Date:

Signature of faculty

Date:

Department of Electrical & Electronics Engineering

Mappings

1. **Course Objectives-Course Outcomes Relationship Matrix**
   (Indicate the relationships by mark “X”)

Course-Outcomes Course-Objectives	1	2	3	4	5
1	H		M		
2		H			
3			H		
4				H	
5					H

2. **Course Outcomes-Program Outcomes (POs) & PSOs Relationship Matrix**
3.

PO’S CO’S	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO 1	M	M	H	H	L	M	H	H	L	L	L	L
CO 2	M	M	L	M	H	L	M	M	M	L	L	L
CO 3	H	M	M	M	M	M	M	L	L	L	L	L
CO 4	H	M	M	H	H	M	H	M	M	L	M	L
CO 5	M	H	M	M	M	L	M	L	M	L	H	L

H-High; M-Moderate; L-Low

Department of Electrical & Electronics Engineering
Rubric for Evaluation

Performance Criteria	**Unsatisfactory**	**Developing**	**Satisfactory**	**Exemplary**
	1	**2**	**3**	**4**
Research & Gather Information	Does not collect any information that relates to the topic	Collects very little information some relates to the topic	Collects some basic Information most relates to the topic	Collects a great deal of Information all relates to the topic
Fulfill team role's duty	Does not perform any duties of assigned team role.	Performs very little duties.	Performs nearly all duties.	Performs all duties of assigned team role.
Share Equally	Always relies on others to do the work.	Rarely does the assigned work - often needs reminding.	Usually does the assigned work - rarely needs reminding.	Always does the assigned work without having to be reminded
Listen to other team mates	Is always talking—never allows anyone else to speak.	Usually doing most of the talking-- rarely allows others to speak.	Listens, but sometimes talks too much.	Listens and speaks a fair amount.

Anurag
ENGINEERING COLLEGE
An Autonomous Institution
(Approved by AICTE, New Delhi & Affiliated to JNTUH)

Accredited with NAAC A+ Grade

Ananthagiri (V&M), Kodad, Suryapet (Dt.), Telangana – 508 206
www.anurag.ac.in +91 8683-22270

II B.TECH IV SEMESTER I MID EXAMINATIONS - APRIL 2024

Branch : B.Tech. (EEE) **Max. Marks: 30**
Date : 02 - Apr - 2024 AN **Time: 120 Minutes**
Subject : Digital Electronics,EE404PC

PART - A

ANSWER ALL QUESTIONS **10 X 1 M = 10 M**

Q.No	Question		CO	BTL
1.	The 1's complement of a binary number is obtained by changing (A). Each 1 to a 0 (B). Each 0 to 1 (C). Each 1 to 0 and each 0 to 1 (D). None of the Above	( )	CO1	L1
2.	Which gate is known as the universal gate? (A). NAND (B). OR (C). AND (D). None of the Above	( )	CO1	L1
3.	______ is an example of identity law (A). a+0=0+a=a (B). 1+a=a+1=1 (C). ab=ba (D). a+(b+c)=(a+b)+c	( )	CO1	L2
4.	The Base of the hexadecimal number systems is (A). 6 (B). 8 (C). 16 (D). 10	( )	CO1	L1
5.	When designing a circuit to emulate a truth table, both Product-of-Sums (POS) expressions and Sum-of-Products (SOP) expressions can be derived from? (A). k-map (B). NAND gate (C). NOR gate (D). X-NOR gate	( )	CO2	L1
6.	Which of the following gates is equivalent to a NOT gate followed by an OR gate? (A). NAND (B). NOR (C). AND (D). XOR	( )	CO2	L2
7.	Which of the following expressions is NOT equivalent to the other three? (A). A(B + C) (B). (AB) + (AC) (C). A + (BC) (D). (A + B)(A + C)	( )	CO2	L3
8.	What is a minterm? (A). A product of literals where each variable appears exactly once (B). A sum of literals where each variable appears exactly once (C). A term with minimum literals in a Boolean expression (D). A term with maximum literals in a Boolean expression	( )	CO2	L2
9.	How many two-input AND and OR gates are required to realize Y=CD+EF+G (A). 2,2. (B). 2,3. (C). 3,3. (D). none of these.	( )	CO3	L1
10.	The simplified expression of full adder carry is (A). c=xy+xz+yz (B). c=xy+xz (C). c=xy+yz (D). c=x+y+z	( )	CO3	L2

PART - B

ANSWER ANY FOUR **4 X 5 M = 20 M**

Q.No	Question	CO	BTL
11.	Draw the logic symbols and construct the truth table for all the Gates?	CO1	L2
12.	Find i) (A6)16 to ( )10 , ii) (35.45)10 to ( )8 iii) (3250-7253) using 10's complement iv) (110101)2 to ( )gray	CO1	L3
13.	Design a 4-bit binary ti BCD converter	CO2	L3
14.	Simplify the following boolean function by using a Tabular Method F(A,B,C,D) = m(0,2,3,6,7,8,10,12,13)	CO2	L3

15.	Explain half Subtractor and full Subtracto and design them using NAND gates	CO3	L3
16.	Explain half adder and full adder and design them using NAND gates	CO3	L2

Anurag
ENGINEERING COLLEGE
An Autonomous Institution
(Approved by AICTE, New Delhi & Affiliated to JNTUH)

Accredited with NAAC A+ Grade
Ananthagiri (V&M), Kodad, Suryapet (Dt.), Telangana - 508 206
www.anurag.ac.in [illegible]

II B.TECH IV SEMESTER II MID EXAMINATIONS - JUNE 2024

Branch : B.Tech. (EEE) **Max. Marks : 30M**
Date : 19-Jun-2024 **Session : Afternoon** **Time : 120 Min**
Subject : Digital Electronics,EE404PC

PART - A

ANSWER ALL THE QUESTIONS **10 X 1M = 10M**

Q.No	Question		CO	BTL
1.	An encoder generates ___ type code on each input? (A). Hexa (B). Binary (C). Octal (D). ASCII	( )	CO3	L1
2.	In which one of the following logic circuits the feedback loop is not present? (A). Sequential logic circuit (B). Combinational logic circuit (C). Both a and b (D). None of the abov	( )	CO3	L1
3.	When both set and reset are disabled in S-R flip flop then the output will be ________ (A). Set (B). Reset (C). No change (D). Indeterminate	( )	CO4	L2
4.	The no-change conditions occur when ________ in JK flip flop (A). J=1, K=1 (B). J=0, K=0 (C). J=1, K=0 (D). J=0, K=1	( )	CO4	L2
5.	The flip flop requires ________ (A). More number of gates (B). More power (C). Both a and b (D). None of the above	( )	CO4	L2
6.	When toggle condition occurs in JK flip flop? (A). J=1, K=1 (B). J=0, K=0 (C). J=1, K=0 (D). J=0, K=1	( )	CO4	L1
7.	Periodic re-charging of the memory cells at regular interval of 3 to 8 millisec is required in a (A). ROM (B). Static RAM (C). Dynamic RAM (D). PLA	( )	CO5	L2
8.	A RAM is (A). Non-volatile memory (B). Only static memory (C). Only dynamic memory (D). Volatile and either static or dynamic memor	( )	CO5	L1
9.	The memory in which the stored data is lost, when power is switched off is (A). ROM (B). Ferrite core memory (C). RAM (D). PROM	( )	CO5	L1
10.	The power consumption of the dynamic RAM is (A). More than that of the static RAM (B). Equal to that the static RAM (C). Less than that of the static RAM (D). Zero	( )	CO5	L1

PART - B

ANSWER ANY FOUR **4 X 5M = 20M**

Q.No	Question	CO	BTL
11.	Simplify the folloing boolean function by usinga Quine Mc-Cluskey method. F(A,B,C,D)=m(0,2,3,6,7,8,10,12,13)	CO3	L2
12.	Explain decoder and Explain operation of 3 X 8 decoder with necessary diagrams.	CO3	L3
13.	Explain the JK flip flop with logic diagram and truth table.	CO4	L3
14.	Explain Shift registers and classification of type of Shift registers	CO4	L2
15.	Implement the given function by using Programable Read Only Memory x=m(0,3,4,7) y=m(1,2,5,7)	CO5	L3
16.	Explain the classification of ROM.	CO5	L3

Continuous Internal Assessment (R-22)

Programme**: BTech** Year: **II** Course: **Theory** **A.Y: 2023-24**

Course: Digital electronic Section: Faculty Name: M.Srinu

S. No	**Roll No**	**MID-I (35M)**	**MID-II (35M)**	**Avg. of MID I & II**	**Viva-Voce/Poster Presentation (5M)**	**Total Marks (40)**
1	22C11A0201	28	27	28	5	33
2	22C11A0202	**33**	30	32	5	37
3	22C11A0203	**26**	25	26	5	31
4	22C11A0204	**25**	24	25	5	30
5	22C11A0205	**27**	27	27	5	32
6	22C11A0206	**25**	25	25	5	30
7	22C11A0207	**31**	34	33	5	38
8	22C11A0208	**28**	31	30	5	35
9	23C15A0201	**27**	24	26	5	31
10	23C15A0202	**25**	24	25	5	30
11	23C15A0203	**31**	27	29	5	34
12	23C15A0204	**34**	31	33	5	38
13	23C15A0205	**28**	30	29	5	34
14	23C15A0206	**30**	32	31	5	36
15	23C15A0207	**30**	30	30	5	35
16	23C15A0208	**25**	25	25	5	30
17	23C15A0209	**25**	32	29	5	34
18	23C15A0210	**31**	30	31	5	36
19	23C15A0211	**25**	26	26	5	31
20	23C15A0212	**32**	31	32	5	37
21	23C15A0213	**33**	25	29	5	34

No. of Absentees: **00**

Total Strength: **21**

Signature of Faculty

: **Signature of HoD**

ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad, Accredited by NAAC with A+ Grade)

Ananthagiri (V & M), Kodad, Suryapet (Dist), Telangana.

Program		
B.Tech. ✓	M.Tech.	M.B.A.

YEAR	SEMESTER	MID EXAMINATION
II^nd	II^nd	II^nd

HALL TICKET NO.									
2	3	C	1	5	A	0	2	0	5

Regulation : R22 Branch or Specialization: EEE

Course: Digital Electronics.

Signature of Student: S. Jaiyanth

Signature of invigilator with date: 19/06/24

Signature of the Evaluator:

Q.No. and Marks Awarded										
1	2	3	4	5	6	7	8	9	10	11

Maximum Marks	30	Marks Obtained	25

(Start Writing From Here)

PART-A

1 (B)

2 (B)

3 (C)

4 (B)

5 (B)

6 (A)

7 (D)

8 (B)

9 (C)

10 (A)

PART-B

11. Given function is F(A,B,C,D) = M(0,2,3,6,7,8,10,12,13)

step1: list all the arrangement order.

minterms.	Binary
M0	0000
M2	0010
M3	0011
M6	0110
M7	0111
M8	1000
M10	1010
M12	1100
M13	1101

step2: arrange all in one's position.

minterms	Binary
M0	0000
M2	0010
M8	1000
M3	0011
M6	0110
M10	1010
M12	1100
M7	0111
M13	1101

steps : Difference the number with minterms

minterms	Binary.
m0, m2	00–0 ✓
m0, m8	–000 ✓
m2, m3	001– ✓
m2, m6	0–10 ✓

m_8, m_{10}	10-0 ✓
m_8, m_{12}	1-00
m_3, m_7	0-11 ✓
m_6, m_7	011- ✓
m_{12}, m_{13}	110-

<u>step 4</u>:- Difference the - in the no. of place cancel on 1 order.

mintesms	Binary
$m_0 m_2, m_8 m_{10}$	-0-0
~~$m_0 m_8, m_2 m_{10}$~~	~~-0-0~~
~~$m_2, m_3, m_6 m_7$~~	~~0-1-~~
$m_2 m_6, m_3 m_7$	0-1-

<u>step5</u>: write the remaining numbers and calculate and write logic numbers.

mintesms	Binary	
m_8, m_{12}	1-00	$\rightarrow A\bar{C}\bar{D}$
m_{12}, m_{13}	110-	$\rightarrow AB\bar{C}$
$m_0 m_2 m_8 m_{10}$	-0-0	$\rightarrow \bar{B}\bar{D}$
$m_2 m_3 m_6 m_7$	0-1-	$\rightarrow \bar{A}C$

Input	m_0	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{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)		⊙	⊙			⊙	⊙						

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

12. 3×8 decoder

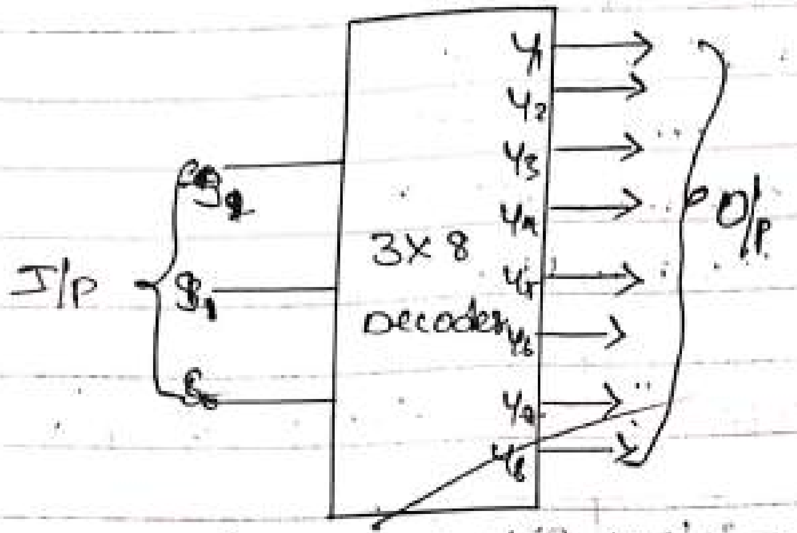


Fig: 3×8 decoder.

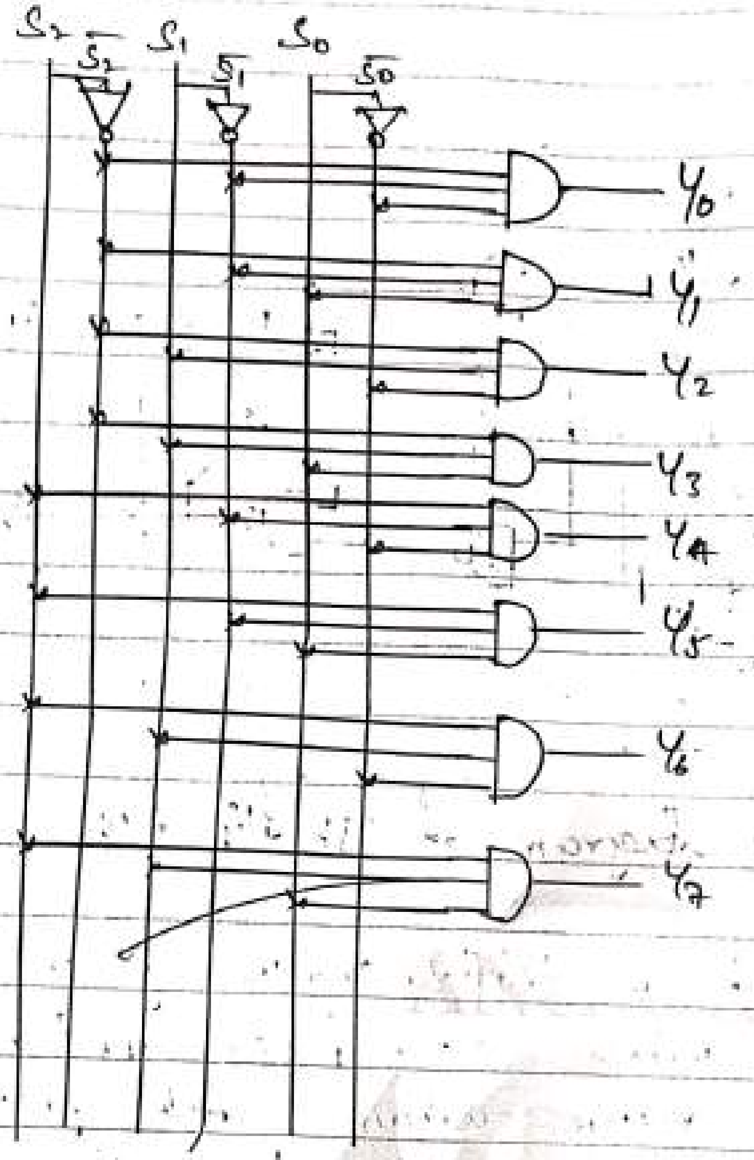
when we give the 3 inputs output should be "1" Ex: 000 and the Y_7 should get "1"
The operation shown in truth table.

Truth Table:

$S_2 S_1 S_0$	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
000	0	0	0	0	0	0	0	1
001	0	0	0	0	0	0	1	0
010	0	0	0	0	0	1	0	0
011	0	0	0	0	1	0	0	0
100	0	0	0	1	0	0	0	0
101	0	0	1	0	0	0	0	0
110	0	1	0	0	0	0	0	0
111	1	0	0	0	0	0	0	0

This the logic truth table for the decoder. And the using the truth table following logic diagram constructed.

The Inputs 000 = $\overline{S_2}\,\overline{S_1}\,\overline{S_0}$ 101 = $S_2\overline{S_1}S_0$
001 = $\overline{S_2}\,\overline{S_1}\,S_0$ 110 = $\overline{S_2}\,S_1\overline{S_0}$
010 = $\overline{S_2}\,S_1\overline{S_0}$ 111 = $S_2 S_1 S_0$
011 = $\overline{S_2}\,S_1 S_0$
100 = $S_2\overline{S_1}S_0$


S_2
$\bar{S}_2$
S_1
$\bar{S}_1$
S_0
$\bar{S}_0$
Y_0
Y_1
Y_2
Y_3
Y_4
Y_5
Y_6
Y_7

[illegible]

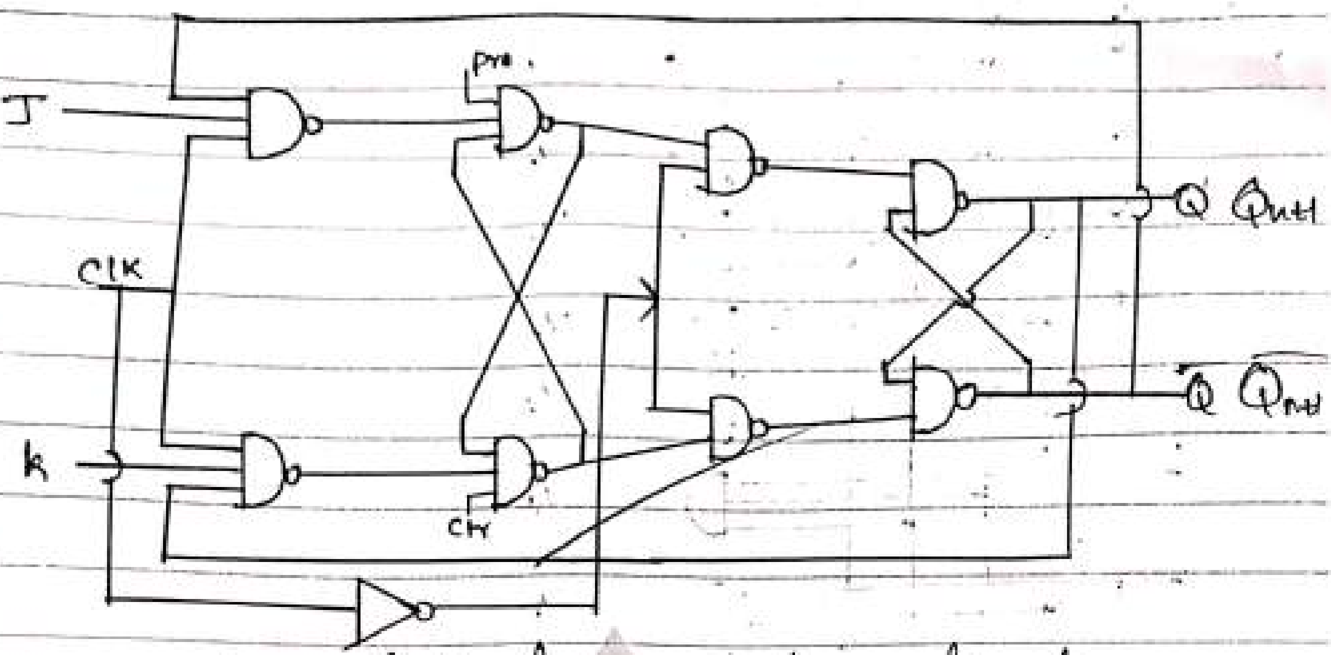


fig:- logic diagram of Jk flip flop

When the clock 0 initial value gates are 0 No change. and when clock pulse applied 1 No change will occure. when clock pulse applied and J=1 also applied J/p 1 there will be change Q=Qn+1 and o/p is set. and when clock pulse applied k=1 J=0 there is change Q= Qn+1 and o/p is Reset.

Truth Table

S.No	C/k	J	K	Qn+1	Status.
1	0	x	x	Qn	Nc
2	1	x	x	Qn	Nc
3	1	1	0	Qn+1	set
4	1	0	1	$\overline{Q_{n+1}}$	Reset
5	1	1	1	$\overline{Q_n}$	Toggle.

when the clock pulse applied and also J=1 k=1 there is action Toggle output is $\overline{Q_n}$.

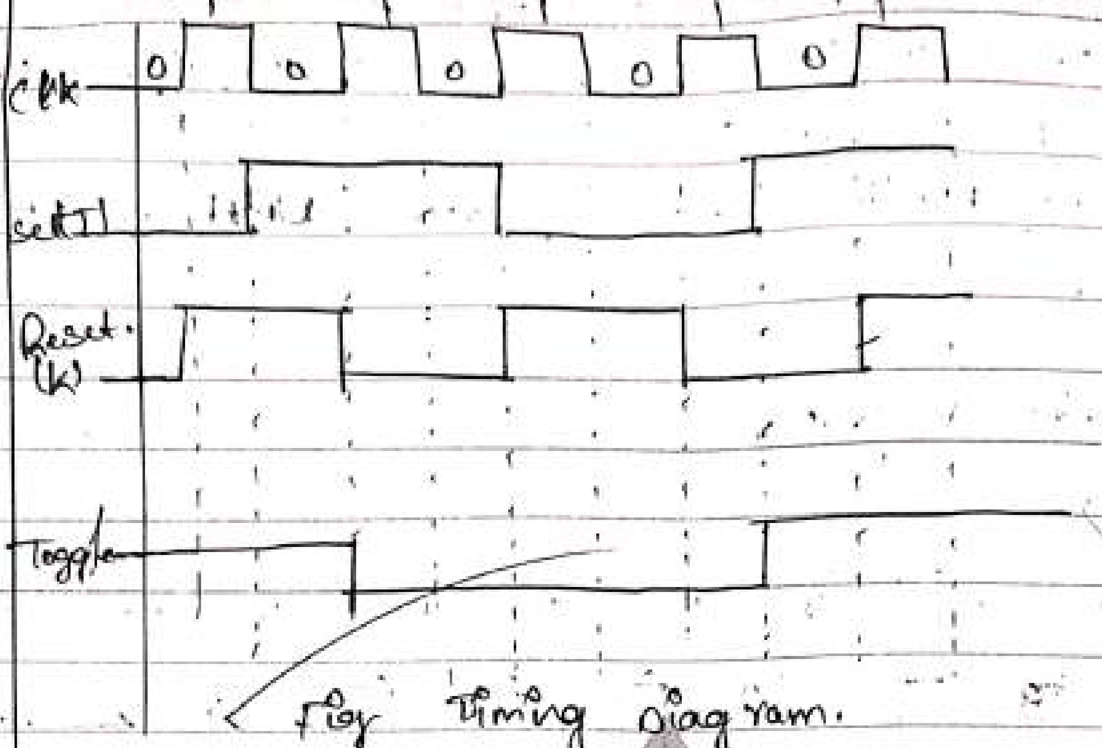


Fig: Timing diagram.

14) shift registers.

There are two types of shift registers.

1) Right shift register

2) Left shift register

1) Right shift register.

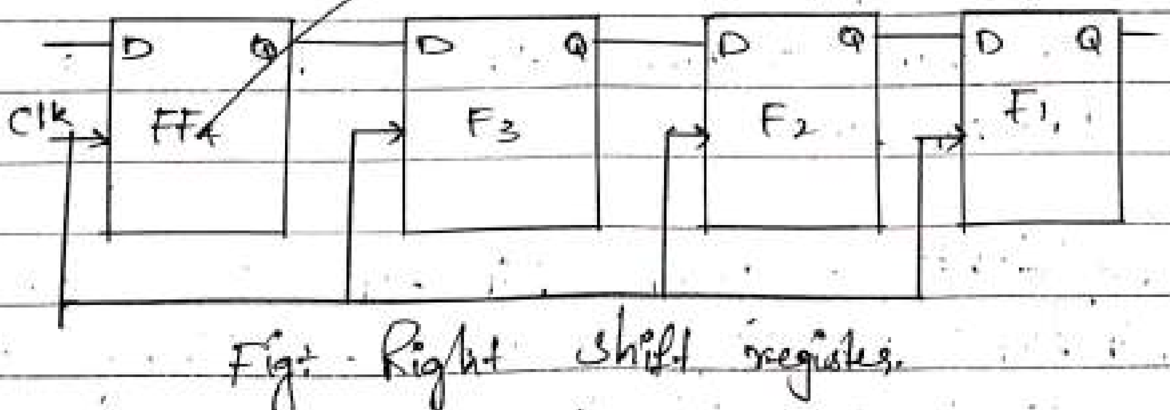


Fig: Right shift register.

When we Binary numbes to the first flipflop is (1111)

→ The Binary numbes 1 is first enters the for F4 is 1 and the output is (1000)

→ And F1 will transmit to the next flipflop. and the output is (0100)

It will continues the operation until to last flip flop.

Left shift register:

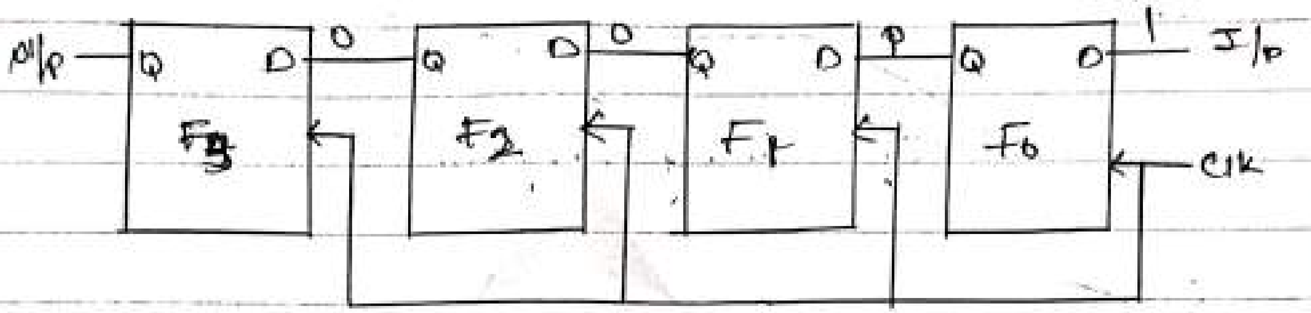


Fig: Left shift register.

When we apply the the binary numbes from the F0 and the binary number is (1111)

→ The binary number 1 is enter in F0 and the output (0001).

→ And F0 will transmit to the next flip flop and the output is (0010)

It will continues the operation until to last flip flop

Register: A group flip flops is called Register.

flip flop: which can store the binary number is called flip flop.

ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad, Accredited by NAAC with A+ Grade)

Ananthagiri (V & M), Kodad, Suryapet (Dist), Telangana.

Program		
B.Tech. ✓	M.Tech.	M.B.A.

YEAR	SEMESTER	MID EXAMINATION
II	II	II

HALL TICKET NO.

2	3	c	1	5	A	0	2	0	8

Regulation : R22 Branch or Specialization: EEE

Course: Digital Electronics

Signature of Student: Nikhil. N

Signature of invigilator with date:

Signature of the Evaluator:

Q.No. and Marks Awarded

1	2	3	4	5	6	7	8	9	10	11

Maximum Marks	30	Marks Obtained	20

(Start Writing From Here)

11. simplify the folloing boolean function by using Quine mc-cluskey $F(A,B,C,D) = m.(0,2,3,6,7,8,10,12,13)$.

0000-0   step-1

0010-2   grope 1: 0000-0   step 2:

0011-3   g 2: 0010-2   000-0 =1,2

0110-6   1000-8   001- = 2,3

0111-7   g3: 0011-3   -010 = 2,10

1000-8   0110-6   10-0 = 8,10

1010-10   1010-10   1-00 = 8,12

1100-12   1100-12   0-11 = 3,7

1101-13   g4: 1101-13   011- = 6,7

0111-7   110- = 12,13

let the p.I   step3: -0-0 = 1,2,8,10

$\bar{B}\bar{D} = 1,2,8,10$ 0-1- = 2,3,6,7

$\bar{A}C = 2,3,6,7$ 2,6,3,7 = 0-1-

$\bar{B}C\bar{D} = 2,10$

$A\bar{B}\bar{D} = 8,10$

$AB\cdot\bar{C} = 12,13$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$\bar{B}\bar{D} = 1,2,8,10$		Y	x						x		y			
$\bar{A}C = 2,3,6,7$			x	X			x	Y						
$\bar{B}C\bar{D} = 2,10$			x								x			
$A\bar{B}\bar{D} = 8,10$									x		y			
$AB\bar{C} = 12,13$													X	Y

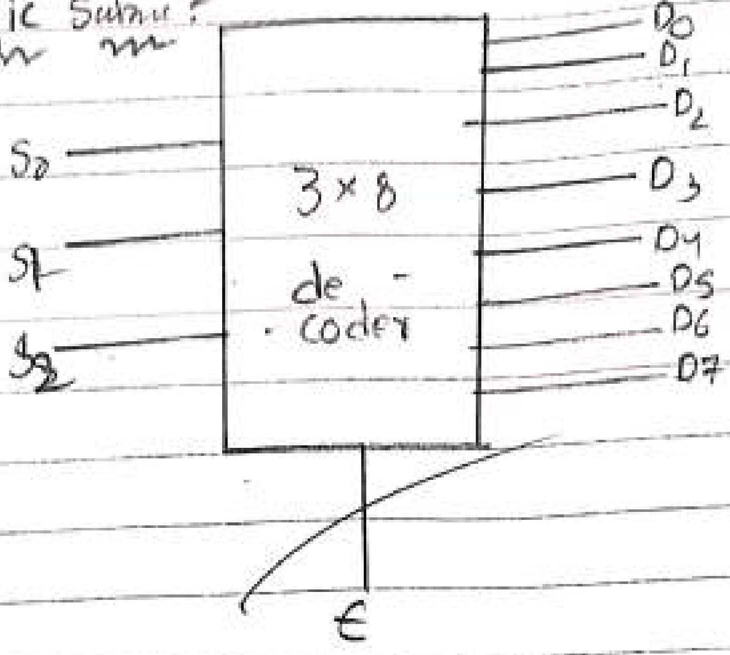
$$F = \bar{B}\bar{D} + \bar{A}C + AB\bar{C}$$

the folloing boolean function by usinga mc-cluskey is

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

12. Ex plain decoder and Explain operation of 3 x 8 decoder with necessary.

- the operation of 3 x8 decoder is input are S_0, S_1, S_2 and output are D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7. in the 3x8 decoder using gate is AND gate. the AND gate input are 3 in and logic sabm:



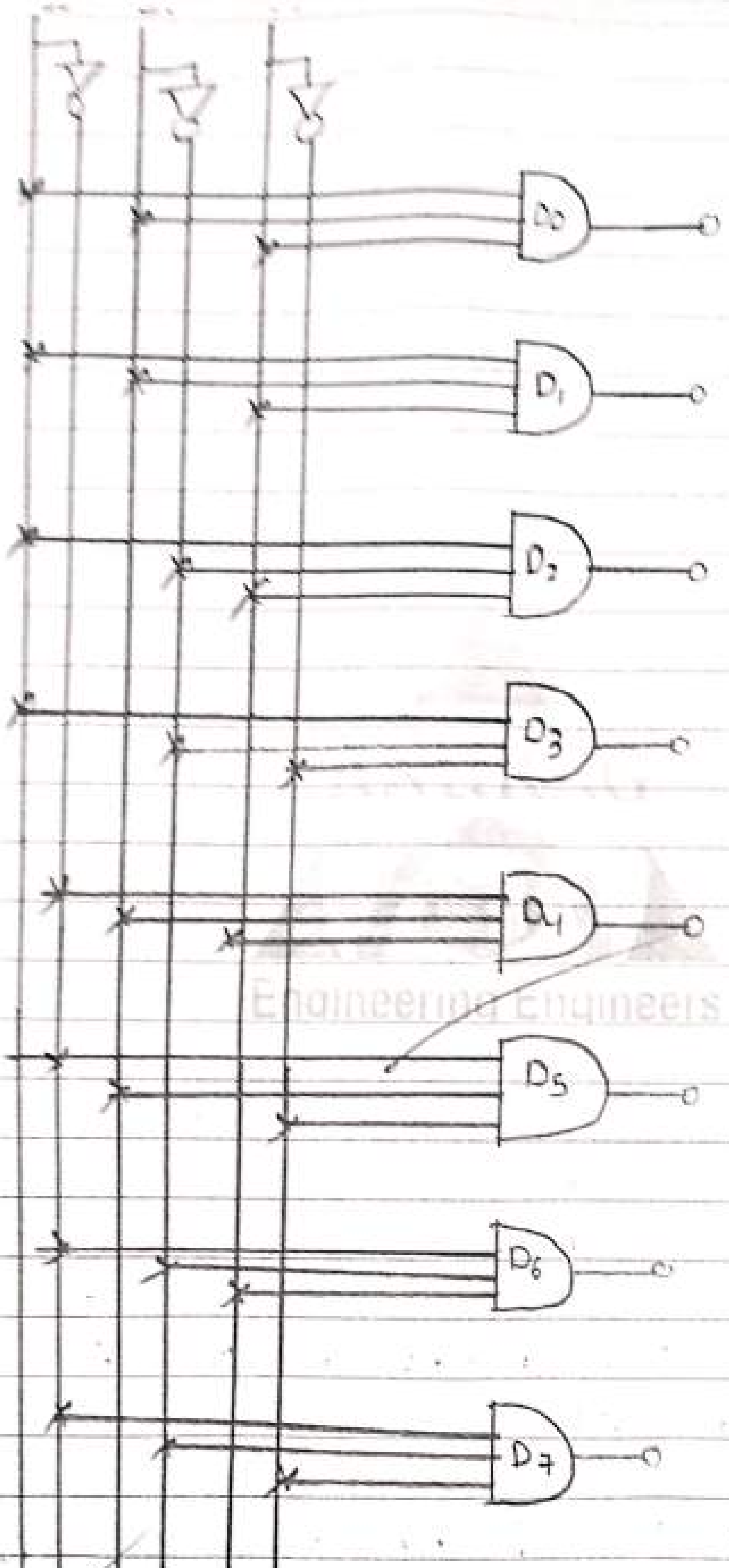
truth table:

input				output							
S0	S1	S2	E	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	1	0
0	1	0	1	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

-> In the 3x8 decoder E = 0 the input are OFF

-> In the 3x8 decoder E = 1 the input are on

-> the input $s_0 = 0$, $S_1 = 0$, $S_2 = 0$ the output is 0


D0
D1
D2
D3
D4
D5
D6
D7

→ If the $S_1:1$ $S_0=1$ $S_2=1$ the output of 3x8 decoder is D7.

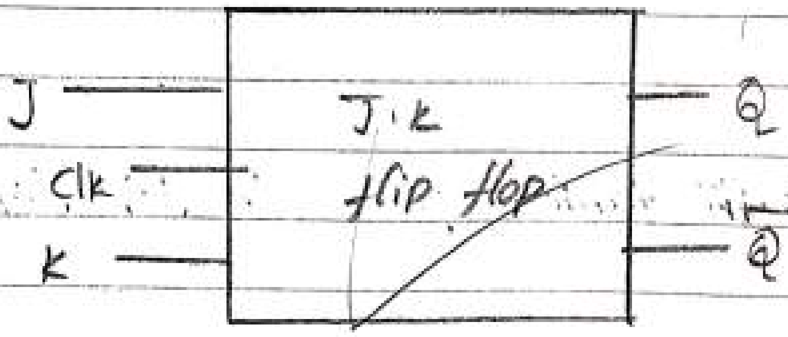
→ the input are more in 3x8 decoder.

13. Explain the JK flip flop with logic diagram and truth table.

- the JK flip flop is input are 3 and output 2

- the JK flip flop is input are J, K clk, and output are Q, $\bar{Q}$

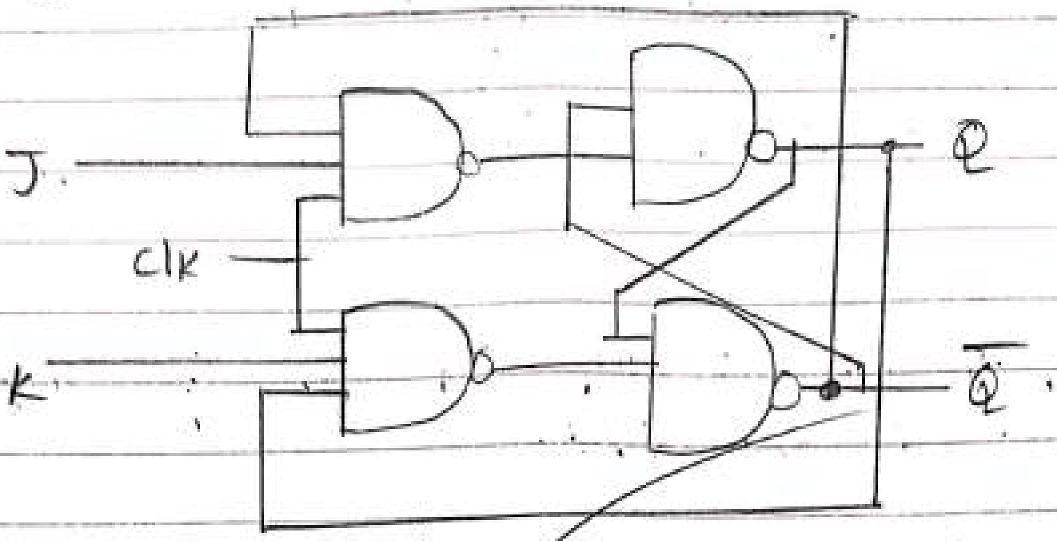
the output of logic diagram of JK flip flop is Binary number 0, 1



→ In the Logic circuits $Q=1$ and the $\bar{Q}=0$.

the J.K flip flop output are Q and $\bar{Q}$

logic diagram:



the logic diagram of JK flip flop.

truth table!

input			output		
J	K	clk	Q	Q^{n+1}	comment
0	0	⎍	0	1	NC
0	1	⎍	1	0	NC
1	0	⎍	$\bar{Q}_n$	Q	set
1	1	⎍	Q	$\bar{Q}_n$	Reset

the truth table of J'K flip flop.

1. (B)
2. (B)
3. (C)
4. (A)
5. (B)
6. (B)
7. (C)
8. (D)

ANURAG ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Affiliated to JNTUH, Hyderabad, Accredited by NAAC with A+ Grade)

Ananthagiri (V & M), Kodad, Suryapet (Dist), Telangana.

Program		
B.Tech. ✓	M.Tech.	M.B.A.

YEAR	SEMESTER	MID EXAMINATION
II	II	II

HALL TICKET NO.									
2	3	C	1	5	A	0	2	0	6

Regulation : R-22 Branch or Specialization: EEE

Course: Digital Electronics

Signature of Student: P.Karthik

Signature of invigilator with date:

Signature of the Evaluator:

Q.No. and Marks Awarded										
1	2	3	4	5	6	7	8	9	10	11

Maximum Marks 30 Marks Obtained 27

(Start Writing From Here)

Part-A

1] B

2] B

3] C

4] B

5] B (more power)

6] A

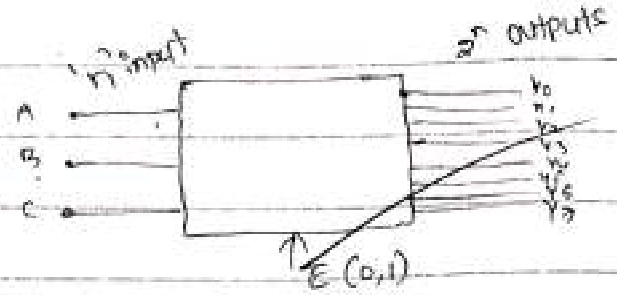
7] C

8] D

9] A

Part-B

12] the operator of 3×8 decoder



E	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

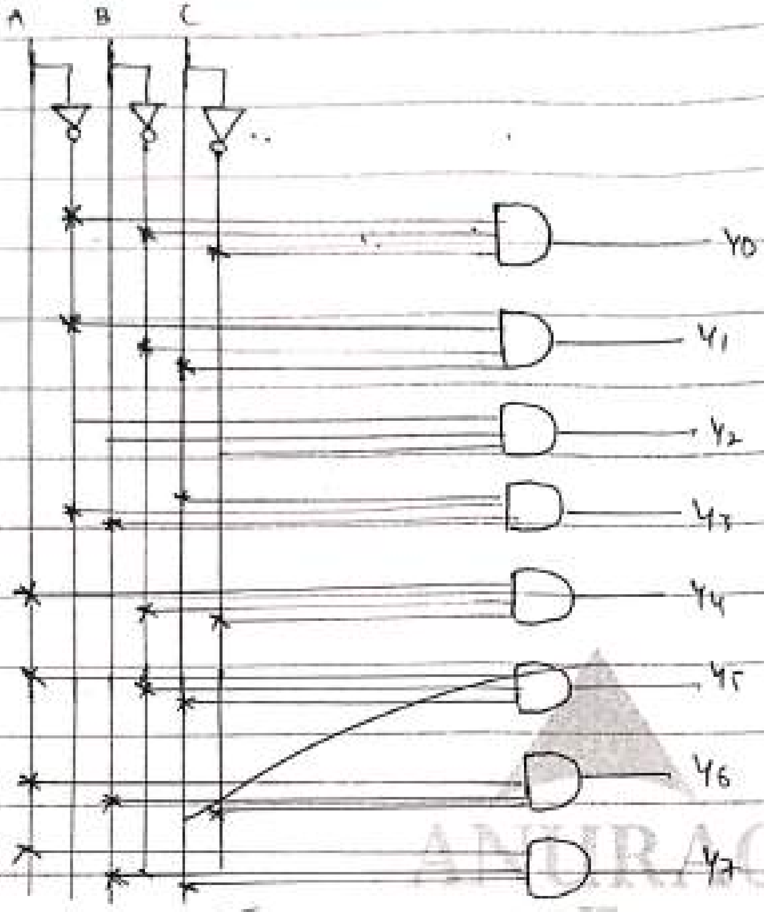
The decoder is have 'n' inputs to get 2^n outputs.

Here $n=3$

$2^n = 2^3 = 8$ outputs.

Here we E Either 1, 0 when we apply 0 all the outputs are 0.

logic diagram :-



15) RRogrammable 91ead only memory

$x = m(0,3,4,7)$ $y = d(1,2,5,7)$

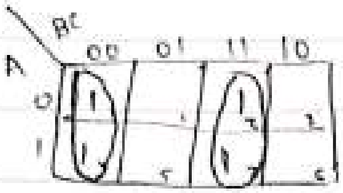
step-1 :-

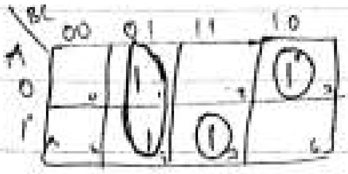
inputs	Binary
~~M0~~	~~000~~
~~B1~~	~~011~~
~~M4~~	~~100~~
~~M7~~	~~111~~
M2	

input			outputs	
A	B	C	X	Y
0	0	0	1	0
0	0	1	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Boolean logic expression

for x





$$\frac{\begin{matrix}000\\100\end{matrix}}{00} \quad \overline{B}\,\overline{C}$$

$$\frac{\begin{matrix}011\\111\end{matrix}}{11} \quad BC$$

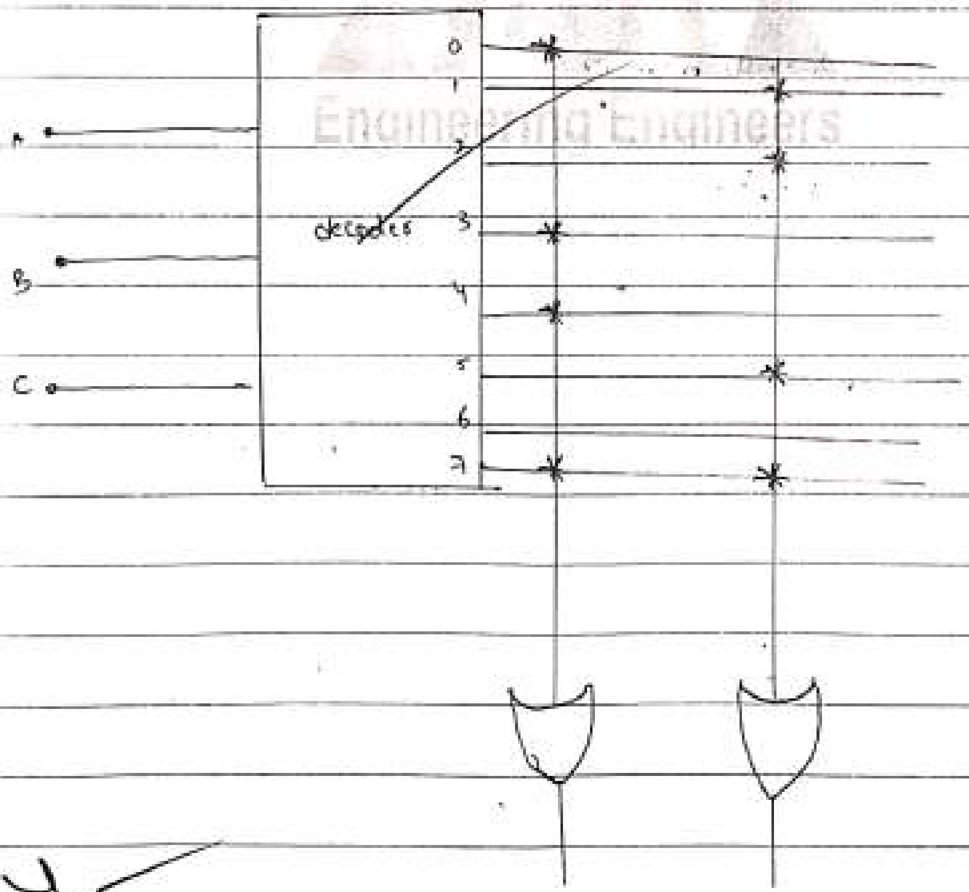
$x = \overline{B}\,\overline{C} + BC$

$$\frac{\begin{matrix}001\\101\end{matrix}}{01} \quad \overline{B}C$$

$$\frac{111}{ABC} \qquad \frac{010}{\overline{A}B\overline{C}}$$

$y = \overline{B}C + ABC + \overline{A}B\overline{C}$.

logic diagram :-



16] The classification of Rom

The Rom stands for Sread out memory.

the Rom is non-volatile becouse when the power is off the stored data is cleared.

Structure is = fixed ANDgates + programmable ORgates

There are different types of Rom's are there

1) PRom

2) EPRom

3) EEPRom

4) Maskable Rom

5) flash

1) prom ÷

-> it stands for Programmable Sread out memory.

-> it is intiary cleared storage then we need to programe the storage.

2] Eprom ÷

-> Eprom standy for Erasable Electrically ~~&~~ programmable Sread out memory

-> it is Erasable for only one time.

-> with the help of " uv " rays to Erase.

3) EEprom ÷

-> EEprom standy for Electrically Erasable programming Sread out memory.

-> it is Erasable for so many times.

-> it is Erasable by bit by bit.

4] Mastable ÷

-> it is also So many times Erasable.

-> But it is Erasable faster than EEprom.

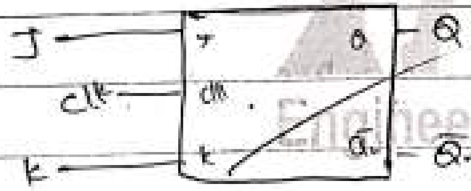
-> it is Erased by block by block

5] flash rom ÷

-> it is instal storage or Erase the data at the time of manufacture only.

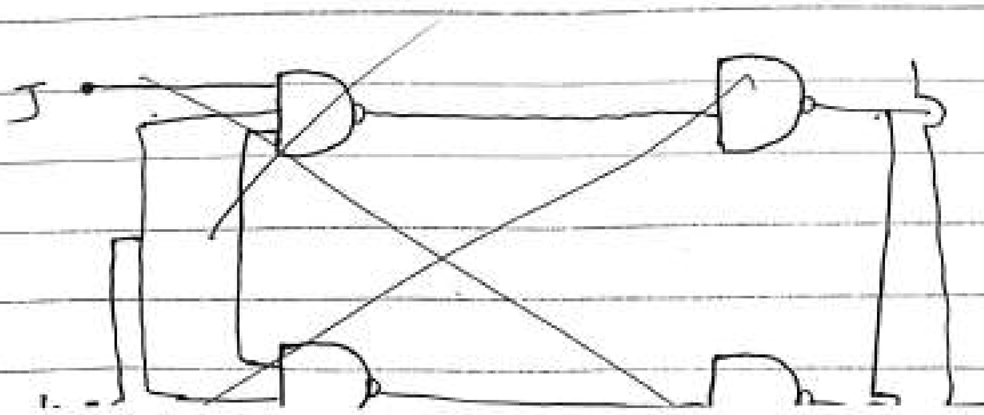
13] JK flip flop

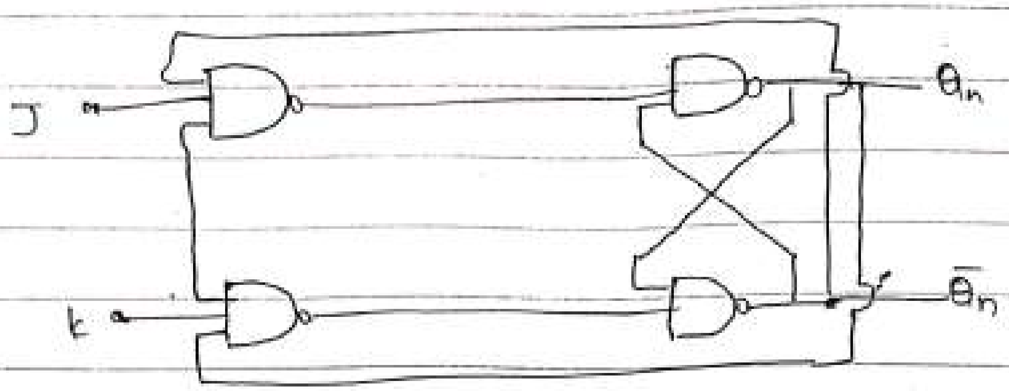
logic diagram Symbol.



Symbol.

logic diagram ÷





truth table :-

CLK	J	K	Q	$\bar{Q}_n$	State
0	0	0	X	X	NC
1	0	0	X	X	NC
1	0	1	0	1	Set
1	1	0	1	0	Reset
1	1	1	0	1	toggle

Flip flop is a sequential circuit & memory storage bit.

→ when the clk is Enable or '0' then Q_n & $\bar{Q}_n$ is disable so the Comment is No change.

→ when the clk is ON '1' the $J=0$ & $K=0$ then the Comment is Nochange

→ when clk is '1' then $J=0$ & $K=1$ then the $Q=0$, $\bar{Q}_n=1$ the comment is set.

→ when clk is '1' then $J=1$ & $K=0$ then the $Q_n=1$ & $\bar{Q}_n=0$

→ when the clk is '1' then J=1 & k=1 then the Comment is "toggle".

UNIT-I

Boolean Algebra & Logic Gates

* Digital systems:

Digital System is a combination of different Devices Design to manipulate physical quantity or information that can be represented in digital form (0, 1)

→ Some of the more of familiar Digital system Examples:

1) Calculators
2) digital computers
3) Digital watches
4) Telephone systems
5) Digital Audio & Video System etc - - -

⇒ The worlds largest system is Telephone system.

Advantages of Digital System:

* Digital systems are easier to design comapared to analog system.
* Digital System are less effectived by noise
* Memories are stable
* Re programming is possible

* Size is less and cost is less

* power discipation is less etc.....

Limitations of Digital system / Drawbacks

* The main world is anolog

* Human doesnot understand digital data

The General representation of number System is

$$(N)_b = -\underbrace{d_l \ldots\ldots d_3\, d_2\, d_1\, d_0}_{\text{Integer Portion}}\ \underset{\substack{\downarrow \\ \text{Base} \\ \text{-Point} \\ \text{(or)} \\ \text{Radix} \\ \text{-Point}}}{.}\ \underbrace{d_{-1}\, d_{-2}\, d_{-3} \ldots\ldots d_{-f}}_{\text{Fractional Portion}}$$

Examples

$$\Longrightarrow \quad (\underbrace{738}_{\substack{\swarrow \\ \text{Integer} \\ \text{Portion}}}\ \underset{\substack{\downarrow \\ \text{Radix} \\ \text{point}}}{.}\ 6 \xrightarrow{\quad} \text{Fractional Portion})_{10}$$

1

Number Systems

Non-positional number systems

Ex:-

⇒ Roman number System
(I, II, III..)

⇒ Un weighted number system

⇒ Excess-3code

⇒ Gray code

⇒ Cyclic code

Drawback:-

Number system we can't represented of symbol is "zero"

Positional number system

Ex:-

⇒ weighted number system

⇒ Decimal number System

⇒ Binary number system

⇒ Octal number system

⇒ Hexa decimal number system

* Digital number systems:- The most widely used in:-

1) Decimal number systems

$()_{10}$ (or) $()_{D}$

Eg:- $(456)_{10}$

2) Binary number systems

$()_{2}$ (or) $()_{B}$

Eg:- $(1011)_{2}$

3) Octal number systems

$( )_8$ (or) $( )_0$

Eg:- $(126)_8$

4) Hexa decimal number system

(or)

Alpha numerical number system

$( )_{16}$ to $( )_H$

Eg:- $(A91)_{16}$

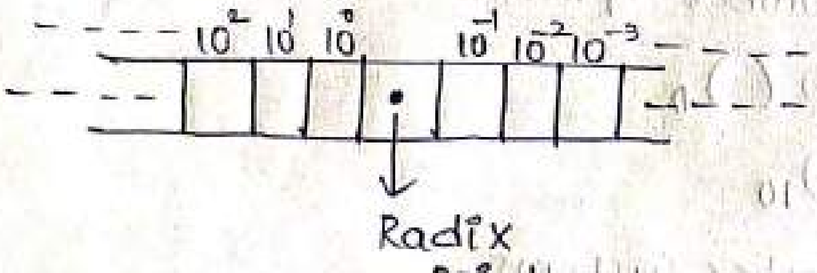
⇒ Decimal number system:-

⇒ It has base $( )_{10}$ (or) $( )_D$

⇒ It has 10 different symbols, the symbols are from 0 to 9 (0, 1, 2, 3, - - - - .9)

⇒ It is a positional number system

⇒ In Decimal number system each number indicates of a 10^i



⇒ Decimal positional values as a power of 10.

Example :-

$(\overset{10^2}{4}\,\overset{10^1}{5}\,\overset{10^0}{3})_{10}$

MSB

LSB

$10^0 \times 3 = 1 \times 3 = 3$

$10^1 \times 5 = 50$

$10^2 \times 4 = 400$

⇒ Represent $(98.72)_{10}$ in power of 10 and identify MSB & LSB

$(\overset{10^1}{9}\,\overset{10^0}{8}\,.\,\overset{10^{-1}}{7}\,\overset{10^{-2}}{2})_{10}$

MSB

LSB

$= 10^1 \times 9 + 10^0 \times 8 + 7 \times 10^{-1} + 10^{-2} \times 2$

$= (98.72)_{10}$

* Binary number System:

⇒ It has a base $(\,)_2$ (or) $(\,)_B$

⇒ It is a positional number system

⇒ It has 2 base number that is 0 & 1

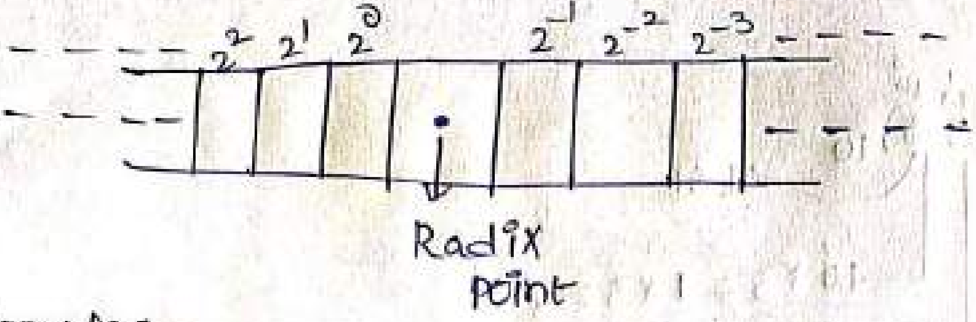
⇒ These 1's & 0's we called it as Bits

Note:-

(i) Group of 4 bits is "Nibble"

1011

ii) Group of 8 bits is "Byte"



Example:-

(i) Convert Binary number into Decimal number.

$\Rightarrow$ $(1100.11)_2$

$$(\overset{2^3}{1}\overset{2^2}{1}\overset{2^1}{0}\overset{2^0}{0}.\overset{2^{-1}}{1}\overset{2^{-2}}{1})_2$$

MSB LSB

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

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

$= 12 + 0.5 + 0.25$

$= (12.75)_{10}$

ii) $(1011.100)_2$ convert into Decimal number

$$(\overset{2^3}{1}\overset{2^2}{0}\overset{2^1}{1}\overset{2^0}{1}.\overset{2^{-1}}{1}\overset{2^{-2}}{0}\overset{2^{-3}}{0})_2$$

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

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

$= (11.5)_{10}$

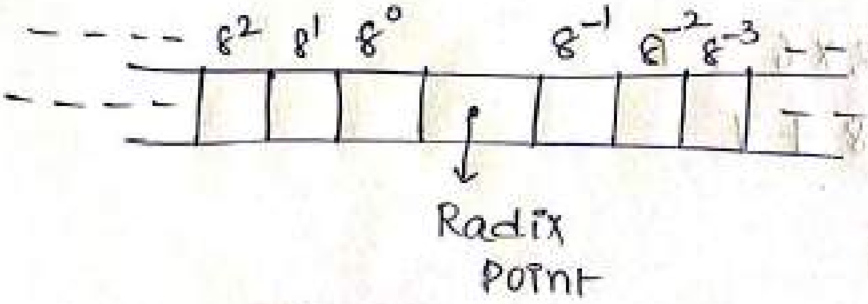
Octal number system :-

⇒ It has base $( )_8$ (or) $( )_0$

⇒ It is a positional number system

⇒ It has 8 different symbols (0 to 7)
(0, 1, 2, 3, - - - - - - 7)

→ Octal positional values as a power of 8.



→ It is a grouping of 3 binary bits

→ It is used to digital computer that will saves the time when we entering (or) Reading instructions

Octal digits	Binary digits 4 - 2 - 1
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

For example :-

(i) $(5\ 6\ 7)_8$ in powers of 8 identify the LSB & MSB

$$(\underset{\text{MSB}}{\overset{8^2}{5}}\ \overset{8^1}{6}\ \underset{\text{LSB}}{\overset{8^0}{7}})_8$$

$$= 8^2 \times 5 + 8^1 \times 6 + 8^0 \times 7$$

$$= 5 \times 64 + 8 \times 6 + 7$$

$$= 320 + 48 + 7$$

$$= (375)_{10}$$

(ii) Convert $(1\ 6\ 5)_8$ into Binary.

$$(\underset{\text{MSB}}{1}\ 6\ \underset{\text{LSB}}{5})_8$$

$$\begin{array}{ccc} 1 & 6 & 5 \\ \downarrow & \downarrow & \downarrow \\ 001 & 110 & 101 \end{array}$$

$$= (001110101)_2$$

Hexa decimal number System

(or)

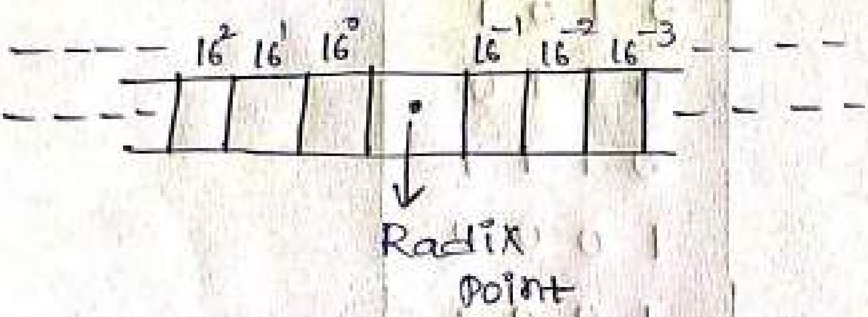
Alpha numeric number system

⇒ It has a base $(\ )_{16}$ (or) $(\ )_H$

⇒ It is a positional number system

⇒ It has 16 different Symbols (0 to 9) to (A to F)

(0, 1, 2, - - - - 9) (A, B, - - - F)



⇒ Hexa Decimal positional values as a power of 16.

⇒ It is used in microprocessor with programming

⇒ It has both numerical and Alphabets

⇒ Positional number System means each digit indicates Hexa decimal positional values as a power of 16.

For Example :-

Convert $(7\ A)_H$ to Decimal number

$(\overset{16^1}{7}\ \overset{16^0}{A})_H$

$= 16^1 \times 7 + 16^0 \times A$

$= 112 + 10$

$= (122)_{10}$

⇒ It is grouping of 4 digits

Hexa decimal digits	Binary digits
	8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10 (A)	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

Example :-

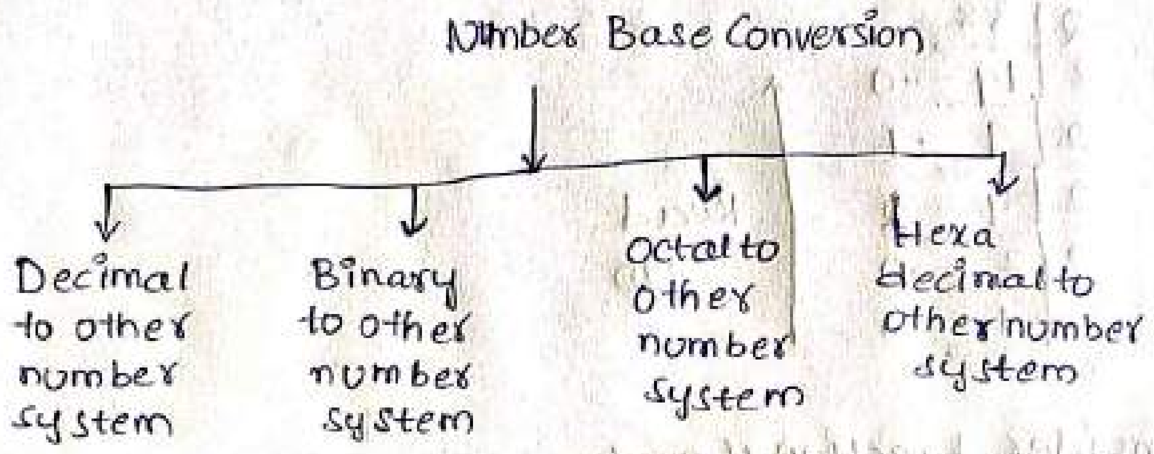
Convert $(7\ A)_{16}$ into Binary

$$\begin{tikzcd}[column sep=small] 7 \arrow[d] & A \arrow[d] \\ 0111 & 1010 \end{tikzcd}$$

$(7\ A)_{16}$

$= 0111\ 1010 \rightarrow (01111010)_2$

* Base conversion methods :-



Number Base Conversion

Decimal to other number system

Binary to other number system

Octal to other number system

Hexa decimal to other number system

1) D → B	1) B → D	1) O → D	1) H → D
2) D → O	2) B → O	2) O → B	2) H → B
3) D → H	3) B → H	3) O → H	3) H → O

⇒ 1) Decimal to Binary conversion :-

$(\ )_{10} \rightarrow (\ )_{2}$

(i) The integer decimal number is repeatedly divided by '2' and writting the remainder after each divider until Quotient is '0' obtained.

(ii) To convert fractional decimal into Binary multiply the number by '2', the Integral part of the product is the MSB of the Binary

For Example:-

1) ⇒ Convert $(38.15)_{10}$ into Binary.

First consider the Integer part

$$\begin{array}{r|l} 2 & 38 \\ 2 & 19 - 0 \\ 2 & 9 - 1 \\ 2 & 4 - 1 \\ 2 & 2 - 0 \\ & 1 - 0 \end{array} \quad \uparrow \text{ Read Up}$$

Consider Fractional part

$0.15 \times 2 = 0.3 \rightarrow 0$
$0.3 \times 2 = 0.6 \rightarrow 0$
$0.6 \times 2 = 1.2 \rightarrow 1$
$0.2 \times 2 = 0.4 \rightarrow 0$
$0.4 \times 2 = 0.8 \rightarrow 0$
$0.8 \times 2 = 1.6 \rightarrow 1$

↓ Read top to Bottom

$\therefore (38.15)_{10} = (100110.001001)_2$

2) Decimal to Octal Conversion: $(\ )_{10}$ to $(\ )_8$

⇒ Decimal to octal conversion same as to Decimal to Binary

⇒ For Integer part is repeatedly divided by 8 and Fractional part is multiplied by 8.

Example :-

1) convert $(974.35)_{10}$ to octal.

Consider Integer part

$$\begin{array}{r|l} 8 & 974 \\ \hline 8 & 121 - 6 \\ \hline 8 & 15 - 1 \\ \hline 8 & 1 - 7 \\ \hline & 0 - 1 \end{array}$$

Consider fractional part

$0.35 \times 8 = 2.8 \rightarrow 2$

$0.8 \times 8 = 6.4 \rightarrow 6$

$0.4 \times 8 = 3.2 \rightarrow 3$

$0.2 \times 8 = 1.6 \rightarrow 1$

$0.6 \times 8 = 4.8 \rightarrow 4$

$0.4 \times 8 = \underbrace{3.2 \rightarrow 3}_{\text{Repeated}}$

$(974.35)_{10} = (1716.26314)_8$

⇒3) Decimal to Hexadecimal conversion:

$(\ )_{10}$ to $(\ )_{16}$

→ To convert decimal number to Hexadecimal number system:-

→ For Integer part repeatedly divided by 16 and For Fractional part is multiplied by 16.

Example :-

⇒ Convert $(675.625)_{10}$ into Hexa Decimal number.

The Given number is decimal number system

consider Integral part

$$\begin{array}{r|l} 16 & 675 \\ 16 & 42 - 3 \\ 16 & 2 - 10\,(A) \\ & 0 - 2 \end{array} \quad \uparrow \text{Read up}$$

consider Fractional part

$0.625 \times 16 = 10.0 \longrightarrow 10(A)$

$\therefore (675.625)_{10} = (2A3.A)_{16}$

⟹ 4) Binary to Decimal conversion

If a Binary number has to be converted into decimal number, then we should multiplied the positional values of each bit and the bit value is add

Example :-

Convert $(11011)_2$ into Decimal number system.

$$(\overset{2^4}{1}\overset{2^3}{1}\overset{2^2}{0}\overset{2^1}{1}\overset{2^0}{1})_2$$

$$= 2^4 \times 1 + 2^3 \times 1 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$= 16 + 8 + 0 + 2 + 1$$

$$= (27)_{10}$$

$\therefore (11011)_2 = (27)_{10}$

5) Octal to Decimal conversion

$( )_8 \longrightarrow ( )_{10}$

To convert Octal to decimal, each octal digit is multiplied by its positional value and add.

Example:-

convert $(6327.4051)_8$ into Decimal

$$\begin{array}{ccccccccc} 8^3 & 8^2 & 8^1 & 8^0 & & 8^{-1} & 8^{-2} & 8^{-3} & 8^{-4} \\ (6 & 3 & 2 & 7 & . & 4 & 0 & 5 & 1)_8 \end{array}$$

$$= 8^3 \times 6 + 8^2 \times 3 + 8^1 \times 2 + 8^0 \times 7 + 8^{-1} \times 4 + 8^{-2} \times 0 + 8^{-3} \times 5 + 8^{-4} \times 1$$

$$= 3072 + 192 + 16 + 7 + \frac{1}{2} + 0 + \frac{5}{512} + \frac{1}{4096}$$

$= (3287.51)_{10}$

$\therefore (6327.4051)_8 = (3287.51)_{10}$

6) HexaDecimal to Decimal conversion

$( )_{16}$ to $( )_{10}$

To convert Hexa Decimal number into Decimal number by multiplying each Hexa Decimal digit by its positional value and add.

Example:-

convert $(3A.2F)_{16}$ into Decimal number.

The Given number is Hexa Decimal number

$$(\overset{16^1}{3}\overset{16^0}{A}.\,\overset{16^{-1}}{2}\overset{16^{-2}}{F})_{16}$$

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

$$= 48 + 10 + \frac{2}{16} + \frac{15}{256}$$

$$= (58.1835)_{10}$$

$\therefore (3A.2F)_{16} = (58.1835)_{10}$.

7) Binary to octal conversion

$( )_2$ to $( )_8$

In this conversion, If a Binary Bit stream is grooped into group OF 3 bits starting at the LSB and then each group is converted into its octal equivalent.

Example:-

(i) convert $(110101101)_2$ into octal number

$$= (\underbrace{110}_{6}\underbrace{101}_{5}\underbrace{101}_{5})_2$$

$$= (655)_8$$

$\therefore (110101101)_2 = (655)_8$

NOTE :-

⇒ For left side of the Radix point we grouped the Bits from LSB bits

⇒ For right side of the Radixpoint we grouped the Bits from MSB bits.

(ii) Convert $(101010.111)_2$ to octal number

$(\underset{5}{\underline{101}}\,\underset{2}{\underline{010}}\,.\,\underset{7}{\underline{111}})_2$

$= (527)_8$

$\therefore (101010.111)_2 = (527)_8$

8) Octal to Binary conversion

$(\ )_8$ to $(\ )_2$

To convert octal to binary, each digit of the octal number is individually converted to its Binary equivalent.

(i) Example :-

Convert $(725.36)_8$ into Binary number.

$(725.36)_8$

111 010 101 011 110

$= (111010101011110)_2 \Rightarrow \therefore (725.36)_8 = (111010101011110)_2$

9) Binary to Hexa decimal conversion

$( )_2 \rightarrow ( )_{16}$

In this conversion the binary bits stream into group of 4 bits starting at the LSB and then each group is converted into its Hexadecimal equivalent.

Example :-

Convert $(11101.101101)_2$ into Hexa decimal Number.

$$\underbrace{0001}_{1}\,\underbrace{1101}_{D}\,.\,\underbrace{1011}_{B}\,\underbrace{0100}_{4}$$

Add 3 zero's ↗ ↑ Add 2 zero's

$= (1D.B4)_{16}$

$\therefore (11101.101101)_2 = (1D.B4)_{16}$

10) Hexa Decimal to Binary number conversion

$( )_{16} \rightarrow ( )_2$

To convert Hexa Decimal to Binary number replacing each hexa decimal digits by its four bit binary equivalent

Example :-

Convert $(3AB2.DE)_{16}$ into Binary Number.

3 A B 2 . D E

↙ ↓ ↓ ↘ ↘ → 1110

0011 1010 1011 0010 1101

$= (0011\,1010\,1011\,0010\,1101\,1110)_2$

$\therefore (3AB2.DE)_{16} = (001110101011001011011110)_2$

11) Octal to Hexa Decimal conversion :-

$(\ )_8$ to $(\ )_{16}$

The easiest way to convert Octal to hexa decimal number.

(i) Convert octal to binary equivalent

(ii) Convert Binary to hexa Decimal equivalent

Example :-

Convert $(26.2)_8$ into Hexa Decimal number

Sol :- (i) Octal to Binary

$(2\ \ 6\ .\ 2)_8$

↙ ↓ ↓

010 110 010

(ii) Binary to Hexa Decimal

(010110.010)

$= \underbrace{0001}_{1}\underbrace{0110}_{6}.\underbrace{0100}_{4}$

$= (16.4)_{16}$

$\therefore (26.2)_8 = (16.4)_{16}$

12) Hexa Decimal to octal conversion

$(\ )_{16}$ to $(\ )_8$

The easiest way to convert hexa decimal to Octal is

(i) convert HexaDecimal to Binary equivalent

(ii) convert Binary to octal equivalent

Example:-

Convert $(E8A.F)_{16}$ into octal number.

(i) Hexa decimal to Binary

E 8 A . F

↓ ↓ ↓ ↓

1110 1000 1010 . 1111

(ii) Binary to octal

$\underbrace{111}_{7}\underbrace{010}_{2}\underbrace{001}_{1}\underbrace{010}_{2}.\underbrace{111}_{7}\underbrace{100}_{4}$

$= (7212.74)_8$

$\therefore (E8A.F)_{16} = (7212.74)_8$

* Binary Arithmetic :-

⇒ computer (or) Digital circuits do not process numbers; they process binary numbers

⇒ Arithmetic operations such as addition, subtraction, multiplication and division can be carried out in any number system.

⇒ The method followed is similar to the method followed in the decimal number system.

① Binary Addition :-

→ since there are only two digits possible in the binary number system (i.e., 0 and 1). we would get only combinations of the binary addition they are

$$\text{(i)}\ \begin{array}{r} 0 \\ +0 \\ \hline 0 \\ \hline \end{array} \quad \text{(ii)}\ \begin{array}{r} 0 \\ +1 \\ \hline 1 \\ \hline \end{array} \quad \text{(iii)}\ \begin{array}{r} 1 \\ +0 \\ \hline 1 \\ \hline \end{array} \quad \text{(iv)}\ \begin{array}{r} 1 \\ +1 \\ \hline \underline{10} \\ \hline \end{array}\ \text{(carry 1)}$$

⇒ First three cases is similar to the addition of decimal number. In fourth case 1 and 1 add giving '2' which is $\underline{1}0\ (\overset{2^1}{1}\ \overset{2^0}{0} = 2+0=2)$ in binary. In this number the bit '0' is known as the sum and the bit '1' is known as the carry bit. The carry bit should be added to the next higher significant bits

Rules for Binary Addition

A B	sum (A+B)	carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Problem: Add $(1010)_2$ and $(0011)_2$

```
  1 <------- carry
 1010
 0011
 ----
 1101
```

Verification:

1010 → 10
0011 → 3
1101 → 13

Problem: Add 1011.101 and 110.1

```
  1111 <------ carry bits
 1011.101
   110.1
 ---------
 10010.001
```

Problem: Add 28 and 15 in binary

First we find the binary equivalent of 28 and 15

```
2 | 28           ↑ LSB
2 | 14 — 0       |
2 |  7 — 0       | Read
2 |  3 — 1       | Up
2 |  1 — 1       |
     0 — 1         MSB
```

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

```
2 | 15           ↑ LSB
2 |  7 — 1       |
2 |  3 — 1       | Read
2 |  1 — 1       | Up
     0 — 1       |
                   MSB
```

$(15)_{10} = (1111)_2$

Addition of 28 and 15

```
  1 1  <———— carry        Verification :
  1 1 1 0 0                  11100 → 28
    1 1 1 1                   1111 → 15
 ___________               ______________
 1 0 1 0 1 1               101011 → 43
```

② Binary subtraction :-

Considering the four combinations of subtracting one bit from the other, we have

```
(i)  0     (ii)  10     (iii)  1     (iv)  1
    -0            0           -0          -1
    ___          -1           ___         ___
     0           ___           1           0
                  1
```

→ Since the first, third and fourth cases is similar to the decimal subtraction

→ Let us concentrate on the second case, a higher value (1) has to be subtracted from smaller value (0), we borrow <u>one</u> from the next higher significant position. Then, the number become 10 (i.e., 2 in decimal) subtracting 1 from 10 would result in a difference of 1.

Rules for Binary subtraction :-

A (Minuend) B (subtrahend)	Difference (A — B)	Borrow
0 — 0	0	0
0 — 1	1	1
1 — 0	1	0
1 — 1	0	0

Problem:- subtract $(0101)_2$ from $(1011)_2$

Minuend → 1011

Subtrahend → 0101

Borrow →

$$\begin{array}{r} 1\overset{10}{\cancel{0}}11 \\ -\,0101 \\ \hline 0110 \\ \hline \end{array}$$

verification:

$$\begin{array}{rl} 1011 & \rightarrow 11 \\ 0101 & \rightarrow (-)5 \\ \hline 0110 & \rightarrow 6 \end{array}$$

Problem:- Subtrahen 101101 from 110110

Minuend → 110110

Subtrahend → 101101

$$\begin{array}{r} 11\overset{10}{\cancel{0}}11\overset{10}{\cancel{0}} \\ (-)\,101\,101 \\ \hline 001\,001 \\ \hline \end{array}$$

③ Binary Multiplication:-

⇒ Binary multiplication is performed in the same manner as with decimal numbers

⇒ Remember the following multiplication table in binary

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Note:- In case the numbers to be multiplied have the same sign; then the result is positive. If they are of opposite signs, then the result is negative.

Example:- Moltipy 1011 by 101

1011 → Multiplicand
101 → Multiplier

```
     1011 x 101
   ____________
carry (1) 1 0 1 1     → multiplying 1011 by LSB 1
          0 0 0 0 x   → multiplying 1011 by next higher bit 0
        1 0 1 1 x x   → multiplying 1011 by MSB 1
   ____________
        1 1 0 1 1 1   → Adding
   ____________
```

Verification:-

```
 11 x 5
 ______
   55
```

④ Binary Division:-

→ Binary division is carried out in the same manner as the decimal division

Problem:- Divide 1111 by 11

Divisor → 11
Dividend → 1111

```
11) 1111 ( 101
(-) 11
   ____
    01
    00
   ____
     11
     11
   ____
     (0)
```

Quotient is 101
Remainder is 0

Verification:-

```
3) 15 ( 5
   15
   __
   (0)
```

→ Since the divisor has two bits, we consider 2 significant bits of the dividend from the MSB. If this is greater (or) equal to the divisor, we put a 1 in the MSB of the quotient and the divisor, we put a 1 in the MSB of the quotient and the divisor below the MSB of the dividend and subtract.

→ If the number is less then we consider 3 bits of the dividend. Then next lower significant bit is brought down giving the result.

→ If this number is less than divisor then put 0 in the next lower significant bit of the quotient and 0 is subtracted from this number. Add next lower significant is brought down. Same procedure is continues till the end of division.

Problem:- Divide 1011 by 10 by the remainder 0.

```
10) 1011(101.1
(-)10
    01
(-) 00
     11
     10
     10
     10
    (0)
```

Dividend :- 1011

Divisor :- 10

Quotient :- 101.1

Remainder :- 0.

Note :

(i) Octal Addition :

→ when we are adding 8 digits then the sum of 2 octal digit is also a octal digit

→ If the octal sum is '8' or greater than '8', subtract '8' to obtain the octal digit. A carry of '1' is produced then the octal sum is corrected.

Example:-

Add $(342)_8$ and $(164)_8$

$$\begin{array}{r} ① \\ 3\ 4\ 2 \\ 1\ 6\ 4 \\ \hline 5\ 2\ 6 \\ \hline \end{array} \qquad \begin{array}{l} 10 > 8 \\ 10-8=2 \end{array}$$

∴ The sum is $(526)_8$

Hexadecimal Addition:

Examples:- Add $(3)_{16}$ and $(9)_{16}$

$$\begin{array}{r} 3 \\ +9 \\ \hline 12 \\ \hline \end{array}$$

∴ The sum is $(C)_{16}$

→ The sum of 2 hexadecimal digits should be a hexa decimal digit

→ If the hexa decimal sum is 16 (or) greater than 16 Subtract 16 to obtain hexa decimal digit.

-A carry of '1' is produced

Eg:- Add $(A)_{16}$ and $(8)_{16}$

$$\begin{array}{r} A \\ +8 \\ \hline 12 \end{array}$$

18>16

18-16 =2 with carry ①

∴ The sum is $(12)_{16}$

* Complement of Numbers :

γ's complement
radix complement

(γ-1)'s complement
Diminshed radix complement

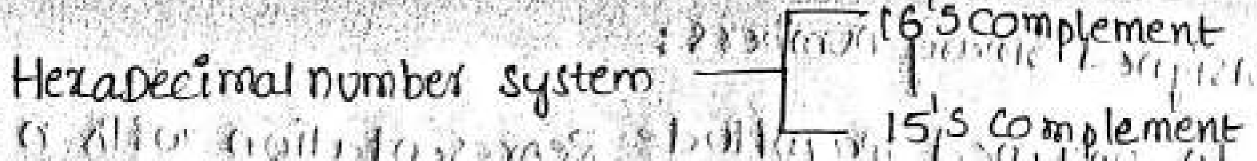
→ The digital systems and digital computers uses complements to simply subtraction operation and logic manipulation operation based on the 'γ' value

⇒ The 'γ' values are classified into 2-types

Binary number system (2) — (2's complement) / 1's complement

Decimal number system (10) — 10's complement / 9's complement

Octal number system (8) — 8's complement / 7's complement.



Binary Number System :-

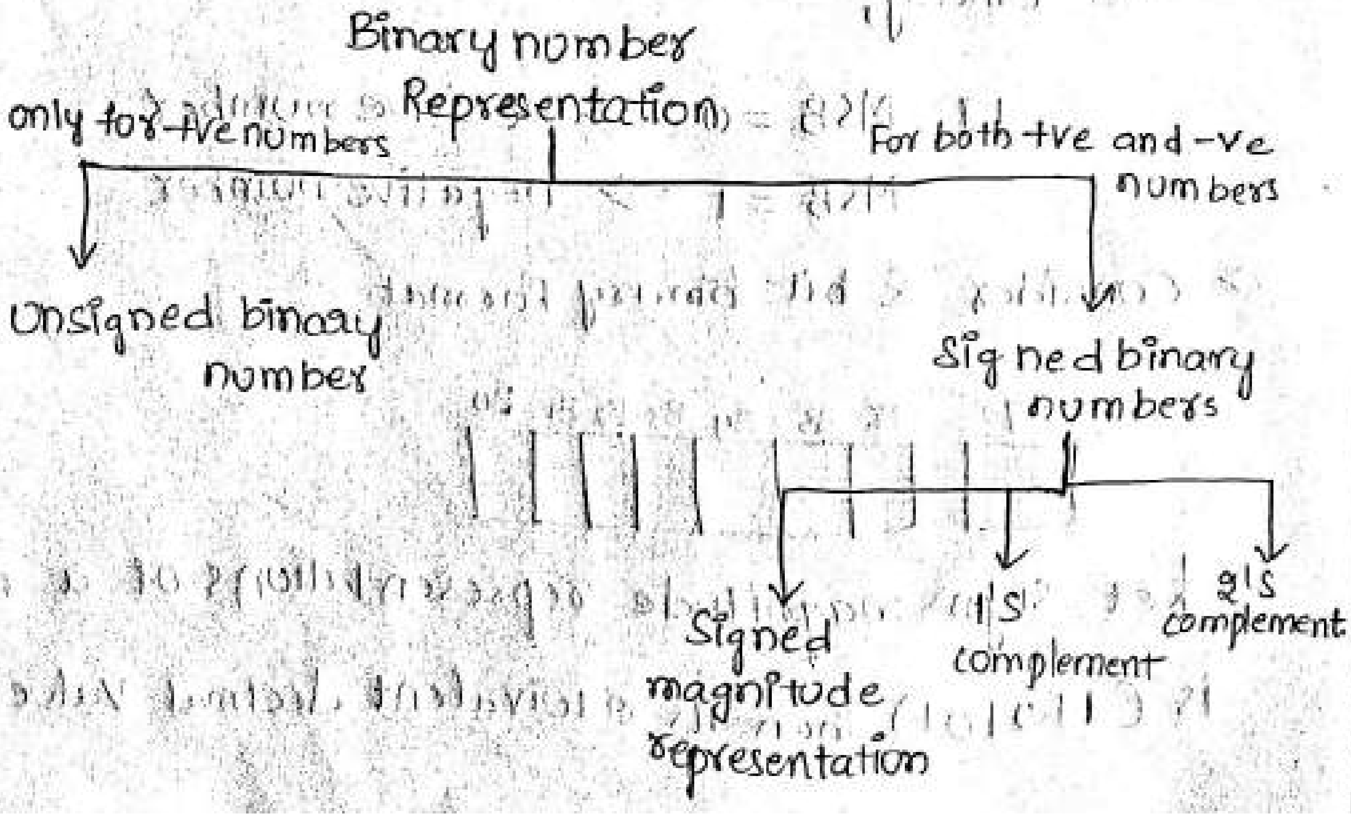
→ A Binary compliments are used to Represents the negative numbers.

→ The subtraction of Binary number is carried out By taking the complement of the number and adding. This is similar to adding a negative number to a positive number.

Binary Number Representation :-

For Example :-

In binary Number system we substitute Base value '2' in place of 'r' to refer complements are 1's and 2's complement.



Unsigned Binary numbers:

→ In unsigned magnitude Representation with n bits, the possible integer values are $\boxed{0 \text{ to } (2^n - 1)}$

→ Unsigned Magnitude Representation only for positive numbers

* Signed Binary numbers :-

Signed binary numbers Represents both positive and negative numbers

① Signed magnitude Representation

Generally.

+ ~~→~~ → positive number

- ~~→~~ → Negative number

In Binary,

If MSB = 0 ⟹ positive number

MSB = 1 → negative number

→ Consider 8 bit Binary Format

B7		B6	B5	B4	B3	B2	B1	B0

⟹ Let sign magnitude representations of a number is (110101) then its equivalent decimal value

(110101)

$$\boxed{1}\ \overset{2^4}{1}\ \overset{2^3}{0}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{1}$$

$(-21)_{10}$

In an sign Magnitude Representation the range of n bits

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$$

* 1's complement Representation :-

The 1's complement of a binary number is got by taking the complement of each bit

i.e., complement of 1 → 0

complement of 0 → 1

(or)

Simply say that,

Subtracting each digit from '1' to get a 1's complement of '1'.

- $1-0=1$
- $1-1=0$

⟹ Range of 1's complement of a number is

$$-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)$$

Example :-

1's complement of few numbers

1) 11011 ⟶ 00100

2) 00001 ⟶ 11110

3) 1111101 ⟶ 0000010

* **2's complement Representation :-**

The 2's complement of a binary number is got by adding '1' to the LSB of the 1's complement of a binary number.

i.e., 1's complement + 1 = 2's complement

Example :-

Find 2's complement of $(11011)_2$

Given number

11011

1's complement : 00100

2's complement : + 1

00101

The Range of 2's complement of a number is

$$-2^{n-1} \text{ to } +(2^{n-1}-1)$$

Few Examples:-

Number	1's complement	2's complement
11111	00000	00001
110110	001001	001010

⇒ 1's complement subtraction:-

Subtraction of binary numbers can be accomplished by the direct method by using the 1's complement method, which allows to perform subtraction using only Addition

For subtraction of 2 numbers we have 2 cases:-

1) Subtraction of a smaller number from a larger number
2) Subtraction of a larger number number from a smaller number.

Case I :- Subtraction of smaller number from larger number:. Follow the below steps.

Step 1:- Determine (or) find the 1's complement of a smaller number

Step 2:- Add the 1's complement of a smaller number to the larger number.

Step 3 :-

Remove (or) Replace the carry and add it to the Result, that carry is called

"End - around - carry".

Example :-

subtract $(101011)_2$ from $(111001)_2$

111001 ⟶ minuend

101011 ⟶ subtrahend

(i) 1's complement of the subtrahend (smaller number) :-

101011

1's complement : 010100

(ii) Add the 1's complement of subtrahend to the minuend (larger number)

```
  ①
   111001
  ①010100
  -------
  ①001101
```

(iii) Replace the carry and add it to the Result

```
  ①001101
       +1
  -------
   001110
  -------
```

Case II :- subtraction Of larger number from a smaller number

Step I :- Find the 1's complement of larger number

Step II : Add this 1's complement of larger number to smaller number

Step III : The Result is negative and in 1's complement Form. Note that there is no carry

Step IV :- To get the difference take 1's complement and put (-) sign (minus)

<u>Note :-</u> 1's complement subtraction Method.

1) Find the 1's complement of subtraction

2) Add 1's complement subtrahend to the minuend

3) If carry is present replace the carry and add it to the Result

4) If there is no carry find the 1's complement and put (-) sign to the Result.

Example:-

Subtract $(111001)_2$ from $(101011)_2$ using the 1's complement method

(111001) ⟶ subtrahend

(101011) ⟶ minuend

(i) 1's complement of subtrahend (larger number)

11001

1's complement : 00110

(ii) Add 1's complement subtrabend to the minuend (smaller number).

$$\begin{array}{r}
\textcircled{1}\textcircled{1}\ \textcircled{1} \\
101011 \\
000110 \\
\hline
110001 \\
\hline
\end{array}$$

∴ The difference is 110001

(iii) There is no carry again find the 1's complement and put (−) sign to the Result.

$$\begin{array}{rr}
 & 110001 \\
\text{1's comp :} & 001110 \\
\hline
 & (-)001110.
\end{array}$$

(ii) subtract $(10101)_2$ from $(11111)_2$

Minuend ⟶ $(11111)_2$

Subtrahend ⟶ $(10101)_2$

(i) 1's complement of subtrahend

$$\begin{array}{rr}
 & 10101 \\
\text{1's comp :} & 01010
\end{array}$$

(ii) Add this 1's complement to the minuend.

$$\begin{array}{r}
\textcircled{1}\textcircled{1}\textcircled{1} \\
11111 \\
\textcircled{1}\ 01010 \\
\hline
\textcircled{1}\ 01001
\end{array}$$

(iii) Carry is present, Replace the carry and add it to the Result.

$$\begin{array}{r} \textcircled{1}01001 \\ \hookrightarrow +1 \\ \hline 01010 \\ \hline \end{array}$$

∴ The difference is $(01010)_2$

* Advantages OF 1's complement Subtraction method :-

→ This method is used in Arithmatic and logic circuits

→ This method is easy.

→ The 1's complement OF a number is easily obtained by inverting each bit with given number.

* 2's Complement :-

→ Follow the below steps :-

Step I :- Find the 2's complement OF Subtrahend

Step II :- Add this 2's Complement of Subtrahend to the minuend

Step III :- If carry is present neglect (or) Discard the carry

Step IV :- If there is no carry find the 2's complement OF the Result and put (-) sign

Example:-

Subtract $(101011)_2$ from $(111001)_2$ by using 2's complement method.

Minuend = 111001

Subtrahend = 101011

(i) Find 2's complement of subtrahend = 1's + 1

$$\begin{array}{lr} & 101011 \\ \text{1's complement:} & 010100 \\ \text{2's complement:} & +1 \\ \hline & 010101 \\ \hline \end{array}$$

(ii) add 2's complement of subtrahend to the minuend

$$\begin{array}{r} \textcircled{1}\quad\;\;\;\textcircled{1}\; \\ 111001 \\ \textcircled{1}\;010101 \\ \hline \textcircled{1}\;001110 \\ \hline \end{array}$$

(iii) carry is present, neglect the carry.

$\cancel{\textcircled{1}}\,001110$

∴ The difference is 001110

(2) subtract $(111001)_2$ from $(101011)_2$

Minuend ⟶ 101011

Subtrahend ⟶ 111001

(i) 2's complement of subtrahend = 1's + 1

$$\begin{array}{lr} & 111001 \\ \text{1's complement:} & 000110 \\ & +1 \\ \hline \text{2's complement:} & 000111 \end{array}$$

(ii) add 2's complement of subtrahend to the minuend

$$\begin{array}{r} 1\,1\,1\,1\;\;\;\; \\ 1\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1 \\ \hline 1\,1\,0\,0\,1\,0 \\ \hline \end{array}$$

(iii) There is no carry, find the 2's complement of the Result

∴ The difference is 110010

$$\begin{array}{rr} \text{1's comp :-} & 001101 \\ & +1 \\ \text{2's comp :-} & \overline{} \\ & (-)\;001110 \\ \hline \end{array}$$

NOTE :-

⇒ The 7's complement, 9's complement, 15's complement Same as the 1's complement

⇒ And the 8's complement, 10's complement, 16's complement Same as the 2's complement.

* Binary codes (or) Digital codes :-

The combination of binary bits can be used to Represent numerical numbers, Alphabets, Any other characters (or) symbols is called Digital code (or) Binary codes. Binary codes are used in Digital system.

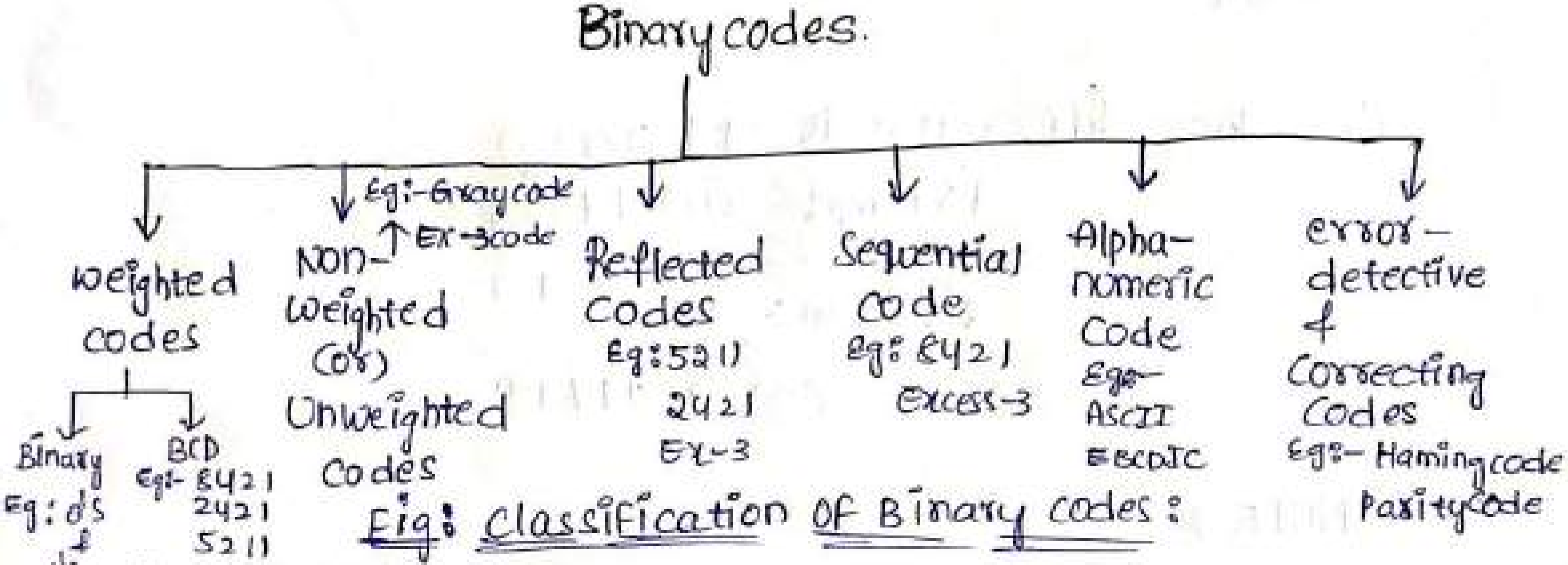


Fig: Classification Of Binary codes:

1) weighted codes:

⇒ In the weighted Code each bit is given a weighted and the decimal number is obtained by adding the weights of the bits where the '1' is present

Examples :-

1) Binary codes (1's and 0's)

2) BCD codes

BCD codes (Binary coded Decimal codes)

⇒ BCD is a weighted code

⇒ BCD is a numeric code

⇒ The most commly used is 8-4-2-1 code, in which each decimal digit represented by 4-bit binary number

→ It is very useful and convenient code for input and output operations in digital circuit

⇒ In multi digit coding, each decimal digit is individually coded with 8-4-2-1 BCD code

⇒ Examples :-

8-4-2-1 → widely used code
2-4-2-1
5-2-1-1
4-2-2-1

Decimal digit	BCD code			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

8-4-2-1 code.

For examples:-

1) convert $(943)_{Decimal}$ to BCD code

$$\begin{tikzcd} 9 \arrow[d] & 4 \arrow[d] & 3 \arrow[d] \\ 1001 & 0100 & 0011 \end{tikzcd}$$

$\therefore (943)_{10} (100101000011)_2$

Note:-

⇒ Other BCD codes are

2 - 4 - 2 - 1 ⟶ 1 ⟹ 0001
5 - 2 - 1 - 1 ⟶ 4 ⟹ 0111
4 - 2 - 2 - 1 ⟶ 5 ⟹ 1001

-Advantages of BCD code:-

⇒ The main advantage of BCD code is easy of converting two and from decimal

BCD to decimal ⟶ decimal to BCD.

⇒ Only the 4digit code groops for the decimal digits 0 to 9 need to be Remembered

Note:-

⇒ In 4 bit binary format, total number of possible representati are $2^4 = 16$

→ Here 10 are valid BCD codes & 6 are invalid BCD codes

In 8 bit binary Format,

Valid BCD codes are = 100

Invalid BCD codes are = 156

Disadvantages of BCD code :-

⇒ Arithmetic operations are more complex than they are in pure binary and it's low efficiency

* Let us see the arithmetic operations using 8-4-2-1 BCD

BCD Addition :-

⇒ when sum in any four-bit column does not exceed 9(1001) then the result is valid BCD number.

When sum in any four bit column exceed 9(1001), then add 6(0110) to the four bit column to get valid BCD number.

⇒ Perform each of the following decimal additions in 8-4-2-1 BCD

(a) 24 + 48

$$\begin{array}{rcl}
24 & \Rightarrow & 0010\ 0100 \\
+48 & \Rightarrow & 0100\ 1000 \\
\hline
72 & & 011\overset{1}{0}\ \overset{1}{1}100 \xrightarrow{} \text{Invalid BCD, so add } 6(0110) \\
 & & +0110 \\
\hline
 & & 0111\ 0010 \\
\hline
 & & \downarrow \quad\quad \downarrow \\
 & & 7 \quad\quad 2
\end{array}$$

BCD subtraction :-

A Negative BCD number can be expressed by taking the 9's (or) 10's complement

⇒ Perform decimal Subtractions in 8-4-2-1 BCD using 9's & 10's complement method : ⓐ 79 - 26.

```
9's :-  79 → 0111 1001
        26 → 0010 0110
   (i) -26 → 0111 0011

              1 11   11
   (ii) Add ⇒ 0111 1001 ──→ 79
              0111 0011 ──→ -26
             ─────────        ──
                               53
              1 1  1 1
        >9 ← 1110 1100 → >9
              0110 0110
             ─────────
           (1) 0101 0010 → 9's complement Add carry to
            └──────────→1                     LSB
             ─────────
              0101 0011 → BCD for 53
             ─────────

               111
10's : 79 → 0111 1001
      -26 → 0111 0100
             ─────────
             1 1  1 1
             1110 1101
             0110 0110
             ─────────
          (X) 0101 0011 ──→ BCD for 53
             ─────────
```

Note :- other 4-bit BCD codes are

	Decimal digit	equivalent 4-bit BCD code
2 4 2 1 →	1 →	0 0 0 1
5 2 1 1 →	4 →	0 1 1 1
4 2 2 1 →	5 →	0 1 1 1

etc..

* Unweighted codes:- The bit positions in the codegroups do not have anyspecific weight assign to them.

Eg: Ex-3code, Gray Code.

Excess-3 code:-

⇒ It is a 4-bit code

⇒ It can be derived from BCD code by adding "3" to each coded number

⇒ It is an "unweighted code".

⇒ It is a "self complementing code". i.e., the 1's complement of an excess-3 number is the excess-3 code for the 9's complement of the corresponding decimalnumber

⇒ This code is used in arithmetic circuits because of its property of self complementation

Eg:- convert $(48)_{10}$ into Excess-3 code.

Sol:-

$$\begin{array}{cc} 4 & 8 \\ +3 & +3 \\ \hline 7 & 11 \\ \downarrow & \downarrow \\ 0111 & 1011 \end{array} \quad \text{(or)} \quad 48 \rightarrow \begin{array}{r} 0100\ 1000 \\ 0011\ 0011 \\ \hline 0111\ 1011 \\ \hline \end{array}$$

$$\therefore (48)_{10} = (01111011)_{Ex-3}$$

Decimal Digit	Excess-3 Code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 0
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

Excess-3 Addition

→ To perform Ex-3 addition we have to

(I) Add two Ex-3 numbers

(II) If carry = 1 → add 3 to the sum of two digits
= 0 → subtract 3

Example :- perform 8 + 6 Ex-3 addition

```
BCD code for 8 ⇒ 1 0 0 0
BCD code for 6 ⇒ 0 1 1 0
                   ①①
Ex-3 for 8 →     1 0 1 1
Ex-3 for 6 →    +1 0 0 1
               ------------
            1    0 1 0 0
       ①①1
       0 0 1 1  0 0 1 1
   ---------------------
       0 1 0 0  0 1 1 1   → Ex-3 for 14
   ---------------------
```

Excess-3 Subtraction :-

→ To perform Excess-3 subtraction we have to

(i) Find complement of the subtrahend

(ii) Add complemented subtrahend to minuend

(iii) If carry = 1 ; Result is positive. Add 3 and end around carry

(iv) If carry = 0 ; Result is negative. subtract 3.

Example : perform 8 − 5 by using Ex-3 subtraction

BCD code for 8 → 1000

BCD code for 5 → 0101

Ex-3 for 8 → 1011

Ex-3 for 5 → 1000

$$\begin{array}{lrl}
\text{Ex-3 for } 8 \rightarrow & \overset{1}{1}\,\overset{1}{0}\,\overset{1}{1}\,1 & \\
\text{complement of } 5 \rightarrow \text{ in Ex-3} & +0\,1\,1\,1 & \\
\hline
\boxed{1} & 0\,\overset{1}{0}\,1\,0 & \\
 & 0\,0\,1\,1 & \\
\hline
 & 0\,1\,0\,1 & \\
+ & +\,1 & \\
\hline
 & 0\,1\,1\,0 & \longrightarrow \text{Excess-3 for 3}
\end{array}$$

* Gray code :-

→ The code which exhibits only a single bit change from one code number to the next is known as 'Gray code.'

→ The gray code is also called as reflected code (or) Unit – distance code (or) cyclic code.

→ Gray code is used to measures angular displacement and measures of linear displacement.

Binary to Gray Conversion:

→ 'MSB' in the gray code is same as corresponding digit in binary number

→ Starting from "Left to Right", add each adjacent pair of binary digits to get next and gray code digit. (Discard the carry if generated).

Example:- convert $(10010)_2$ to gray code.

Sol:-

MSB → 1 + 0 + 0 + 1 + 0 → Binary

↓ ↓ ↓ ↓ ↓

MSB → 1 1 0 1 1 → Gray

∴ $(10010)_2 = (11011)_{Gray}$.

Gray to Binary conversion:-

→ "MSB" of binary is same as that of gray code.

→ Add each binary digit to the generated gray digit in the next adjacent position (discard the carry if generated)

Example:- Convert $(11011)_{Gray}$ to Binary code

Sol:-

MSB → 1 1 0 1 1 → Gray

MSB → 1 + 0 + 0 + 1 + 0 → Binary

∴ $(11011)_{Gray} = (10010)_2$

* Five Bit codes:-

→ 5-bit BCD codes having special characteristics. These special characteristics of the code are usefol for error detection. Eg:- 6 3210, shift counter, 51111 etc.

Eg:- 51111 → 5 → 10000

* Reflective Codes :-

⇒ A code is said to be reflected when the code for 9 is the complement for the code for 0, 8 for 1, 7 for 2. Note that the 8421, 5211 and excess-3 codes are reflective.

* Sequential codes :

→ In sequential codes each succeding code in one binary nomber greater than its preceding code. The 8421 and Excess-3 are sequential

* Alphanumeric codes:-

→ The codes which consists of both numbers and alphabetic characters are called alphanumeric codes.

→ It is used in many computers to represent alphanumeric characters and symbols internally and so it is also called "Internal code".

→ The Most commonly used alphanumeric codes are:

(i) ASCII (American Standard Code for Information Interchange

(ii) EBCDIC (Extended Binary coded Decimal Interchange code)

(iii) Hollerith code.

(i) ASCII :-

→ This is most widely used alphanumeric code in computer and in main frames.

→ It is a 7 bit code

→ It can represent $2^7 = 128$ possible Characters

→ There fore the ASCII code for letter A is

A → 1000001 (41)

(ii) EBCDIC code :-

→ This is an 8 - bit code

→ It can represent $2^8 = 256$ possible Characters

This uses BCD Representation of alphanumeric characters

(iii) Hollerith code :-

→ This is a Code essentially used with the punched card In this each Character is represented as a sequence of 0's and 1's. Each code in 12 bit long.

* Error Detecting and correcting codes :-

→ when the digital information in the binary form is transmitted from one circuit (or) system to another circuit (or) System an error may occur.

This means a signal corresponding to 0 may Change to 1 (or) Vice- versa due to presence of noise

→ To maintain the data integrity between transmitter and receiver, extra bit (or) more than one bit are added in the data.

→ The data, along with the extra bit|bits forms the code

→ codes which allow only error detection are called error-detecting codes and codes which allows error detection and correction are called error detecting and correcting codes.

Ex :- 1) parity Bit

2) Hamming code

* Parity Bit :-

→ A parity bit is used for the purpose of detecting errors during transmission of binary information

→ A parity bit is an extra bit included with a binary message to make the number of 1's either odd (or) even

→ The circuit that generates the parity bit in the transmitter is called a paritygenerator and the circuit that checks the parity in the receiver is called parity checker.

→ In even parity the added parity bit will make the total number of 1's an even. Total number of ones in the code group is an even number.

→ In odd parity the added parity bit will make the total number of 1's an odd ammount. Total number of

Ones in the code group is odd number

For examples:-

(a) we have, (10000 11)

By even parity method (the no.of 1's including parity bit is always even), new code group is

$\boxed{1}\,1000011$

↑ added parity bit

(b) we have, (1000001)

By odd parity method (the no.of 1's including parity bit is always odd), new code group is

$\boxed{1}\,1.000001$

↑ added parity bit.

Example:- The received code is 1000 0001. Check whether code is correctly received (or) not if add parity is used.

Sol: The received code has even parity (no.of is $\underset{\substack{\downarrow \\ \text{even number}}}{2}$)

hence the code is not received correctly.

Hamming code:-

→ Hamming code not only provides the detection of a bit error, but also identifies which bit is in error so that it can be corrected.

Thus Hamming code is called error detecting and correcting code.

→ The Hamming code is one of the most use-ful error correcting code

→ Other error correcting codes available are Block code, cyclic code, convolution code etc---

→ "Hamming code" for a group of n-bit message (or) information is generated by adding 'K' parity (or) check bits to form an (n+K) bit code.

→ Step ① :- Number of parity Bits :

Number of parity bits depends on the number of information bits. If the number of information bits is designed n, then the number of parity bits, K is determined by the following relationship:

$$\boxed{2^K \geq n+K+1}$$

K → no. of parity bits

n → no. of information bits.

If $n=4$, $K=3$

then $2^3 \geq 4+3+1 \Rightarrow 8 \geq 8$ ✓

For example, If we have four information bit i.e., $n=4$ then K is found by trail and error using equation. Let $K=2$, then $2^K = 2^2 = 4$ and $n+K+1 = 4+2+1 = 7$. Since 2^K must be equal (or) greater than $n+K+1$, then relationship in aboveequation is not satisfied.

Hence we try to the next value of 'K'. Let $K=3$ then $2^3 = 2^K = 8$ and $n+K+1 = 4+3+1 = 8$.

The value of 'K' Satisfies the relationship given in above equation, and therefore we say that three parity bits are required to provide single error correction for four information bits.

→ Step②: Location of parity Bits in the code:

In the above Example we have four information bits and three parity bits. Therefore the code is of seven bits $(n+K = 4+3 = 7)$

The LSB is designated bit 1, the next bit is bit 2, ---

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
D_7	D_6	D_5	D_4	D_3	D_2	D_1

The parity bits are located in the positions that are numbered corresponding to ascending powers of two $(\underset{(2^0)}{1}, \underset{(2^1)}{2}, \underset{(2^2)}{4}, \underset{(2^3)}{8} \text{-----})$. Therefore, for 7-bit code, locations for parity bits and information bits are shown below:-

$D_7, D_6, D_5, P_4, D_3, P_2, P_1$.

→ Step③ :-

Assigning values of parity Bits (By using eitherevern (or) odd parity method)

Problem:- Encode the binary word 1011 into seven bit even parity Hamming code.

Given binary code is 1011

number of information bits $n=4$

Step① :- Find the number of parity bits required

Let us assume $K=3$, then

$2^K = 2^3 = 8$

$n+K+1 = 4+3+1 = 8$ i.e., $2^K \geq n+K+1$

$8 \geq 8$

Three parity bits are sufficient

∴ Total code bits $= 4+3 = 7$

Step② :- Location of parity bits in the code i.e., $2^0, 2^1, 2^2$ --

P_1, P_2, P_3 are 1, 2, 4 Locations

	D_4	D_3	D_2	P_3	D_1	P_2	P_1
information bits	1	0	1		1		
Parity bits				0		0	1
	D7	D6	D5	D4	D3	D2	D1

Step③ : Determine the parity bits

	P_3	P_2	P_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

For P_1 : Bit locations 1, 3, 5, 7

P_1 : P_1, 1, 1, 1.

even parity : $\boxed{P_1 = 1}$

For P_2 : Bit locations 2, 3, 6, 7

P_2 : P_2, 1, 0, 1.

even parity : $\boxed{P_2 = 0}$

For P_3 : bit locations 4, 5, 6, 7.

P_3 : P_3, 1, 0, 1

even parity $\boxed{P_3 = 0}$

∴ Seven bit Hamming code is 1010101

Problem:- Determine the single error- correcting code for the information code 10111 for odd parity.

Sol:- Given information code is 10111

$$\boxed{n = 5}$$

Condition for error correcting (or) Hamming code is

Let us assume K = 4 $\boxed{2^K \geq n+K+1}$

$$2^4 \geq 5+4+1$$

$$16 \geq 10$$

∴ Condition satisfied & total number of bits in the code is n+k i.e., 9 (D_9 -----D_1)

and parity bits are P_1, P_2, P_3 & P_4 located at (1, 2, 4, 8)
2^0 2^1 2^2 2^3

D_5	P_4	D_4	D_3	D_2	P_3	D_1	P_2	P_1
1	0	0	1	1	1	1	1	0.
9	8	7	6	5	4	3	2	1

Bit locations
(1's place of P_1)

$P_1 \rightarrow 1, 3, 5, 7, 9$

for odd parity : $P_1 \rightarrow P_1, 1, 1, 0, 1$

$\boxed{P_1 = 0}$

$P_2 \rightarrow 2, 3, 6, 7.$

for odd parity : $P_2 \rightarrow P_2, 1, 1, 0$

$\boxed{P_2 = 1}$

$P_3 \rightarrow 4, 5, 6, 7$

for odd parity : $P_3 \rightarrow P_3, 1, 1, 0$

$\boxed{P_3 = 1}$

$P_4 \rightarrow 8, 9$

for odd parity : $P_4 \rightarrow P_4, 1$

$\boxed{P_4 = 0}$

	P_4	P_3	P_2	P_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

∴ Total information bits of Hamming code is

$\boxed{100111110}$

* Detecting and correcting an error :-

→ In the last section we have seen how to construct Hamming code for given number of information bits Now we will see how to use it to locate and correct an error.

→ To do this, each parity bit, along with its corresponding group of bits must be checked for proper parity. The correct resolt marked by '0' otherwise '1'

→ After all parity checks, binary word is formed taking resulting bit for p_1 as LSB. This word gives bit location where error has occored.

→ If word has all bits '0' then there is no error in the Hamming code.

Problem :- Assume that the even parity Hamming code is in example (0110011) is transmitted and that 0100011 is received. The receiver does not know what was transmitted Determine bit location where error has occured using received code.

Sol :- Step ① : construct bit location table

	P_3	P_2	P_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

$n+K=7$

Received Code	D_4	D_3	D_2	P_3	D_1	P_2	P_1
	0	1	0	0	0	1	1
	7	6	5	4	3	2	1

Step ② : Check for parity bits

For p_1 : P_1 Checks locations 1,3,5,7

$$P_1 : \underset{1}{\underset{\downarrow}{\boxed{P_1}}},0,0,0$$

There is a '1' in the group

∴ parity checks for even parity is wrong → $\underset{(LSB)}{1}$

for p_2 : $P_2 \rightarrow 2,3,6,7$

$$\underset{1}{\underset{\downarrow}{\boxed{P_2}}},0,1,0$$

∴ parity check for even parity is correct → 0

For P_3 : $P_3 \rightarrow 4, 5, 6, 7$

$$\underset{\underset{0}{\downarrow}}{P_3}, 0, 1, 0$$

∴ parity check for even parity is wrong → 1

The resultant word is $\overset{2^2\ 2^1\ 2^0}{\boxed{1\ 0\ 1}}$

$P_3\ P_2\ P_1 \rightarrow$ LSB

This says that bit in the number 5 location is in error. It is 0 and should be a 1.

∴ The correct code is 0110011.

Problem :- The Hamming code 1011 0 11 01 is received correct it if any errors. There are four parity bits and odd parity is used.

$n+k = 9$

$k = 4$ $\quad 2^0, 2^1, 2^2, 2^3$

$n = 5$

$k = 4$

Sol :- (i) construct a bit location table

D5	P4	D4	D3	D2	P3	D1	P2	P1
1	0	1	1	0	1	1	0	1
9	8	7	6	5	4	3	2	1

(ii) check for parity bits

$P_1 \rightarrow 1, 3, 5, 7, 9 \rightarrow \underset{\underset{1}{\downarrow}}{P_1}, 1, 0, 1, 1.$

∴ parity check for odd parity is wrong → 1 (LSB)

For $P_2 \rightarrow 2, 3, 6, 7 \rightarrow \underset{\underset{0}{\downarrow}}{P_2}, 1, 1, 1$

∴ parity check for odd parity is correct → 0

For $P_3 \rightarrow 4, 5, 6, 7 \rightarrow \underset{\underset{1}{\downarrow}}{P_3}, 0, 1, 1$

∴ Parity check for odd parity is correct → 0

0	P4	P3	P2	P1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

for $p_4 \rightarrow 8, 9 \rightarrow p_4, 1 \rightarrow 0, 1$

∴ parity check for odd parity is correct → 0

The resultant word is $\boxed{\overset{2^3}{0}\ \overset{2^2}{0}\ \overset{2^1}{0}\ \overset{2^0}{1}}$ — 1

1st location is in error. It is 1 and should be a '0'

∴ The correct code is 101101100.

* Binary storage and Registers

In digital systems when we have to stores binary information we have to use binary cell.

Binary cell :- Binary cell stores only 1 bit of information.

⇒ It has two stable states

1) set state (1)

2) Reset state (0)

⇒ It is also called as flipflop (or) one bit storage device (or) 1 bit memory storage device.

→ when we have to store more than 1 bit of information we have to use Registers

Registers :- Group of binary cells / flipflops is called Registers.

* when we have to store 8 bits of information require 8 bit register.

8 – bit register.

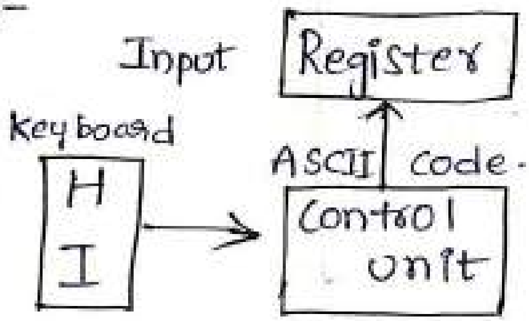
1	1	0	1	1	1	0	1
R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

Similarly, 16-bit register

1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	1
R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

For n bit of information storage we have to use n bit – Register

Example :-



* Binary logic :-

logic ⟶ If the condition is Satisfied ⟶ True

If the condition is not satisfied ⟶ False

True ⟹ Binary "1"

False ⟹ Binary "0"

Example :- Switch

Switch is closed ⇒ Binary '1'
Switch is open ⟶ Binary '0'
Switch is ON ⇒ '1'
Switch is OFF ⇒ '0'.

Binary logic levels

Binary 1	Binary 0.
True ON closed High 5V	False OFF open low 0V

Binary logic are 2 types :
(i) positive binary logic
(ii) Negative binary logic

(i) positive binary Logic
0v ⟶ Low Level ⇒ binary '0'
5V ⟶ High Level ⇒ binary '1'

(ii) Negative binary logic
0V ⇒ High Level ⇒ Binary '1'
5V ⟶ Low Level ⇒ Binary '0'

Boolean Algebra: Boolean Algebra is a set of rules, laws and theorems by which logical operations can be mathematically expressed. It is also known as switching algebra.

* It is one of the tool for designing of the digital system.

* In this, we use only 0's and 1's and $a, b, c, \bar{a}, \bar{b}, \bar{c} \ldots$

* Boolean algebra operations are

1) AND
2) OR
3) NOT

* These operations are performed by using logical operands, they are 0 & 1's.

* If TRUE it is represented by one (1).

* If FALSE it is represented by zero (0).

1) AND Operation:

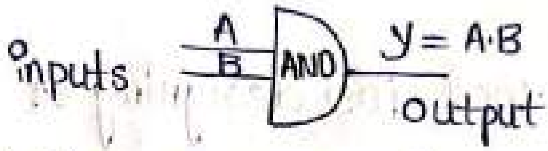
→ If two (or) more operands are anded together then the result will be a logic 1, if all the operands are one.

→ When we use (0&1's) 2 operands, then the possible number of combinations are '4'. $(2^2 = 4)$

→ It is represented by (·) sign

0 AND 0 = 0	0·0 = 0
0 AND 1 = 0	0·1 = 0
1 AND 0 = 0	1·0 = 0
1 AND 1 = 1	1·1 = 1

Symbol:



* If A & B are the logical operands then 'y' is the $\boxed{y = A \cdot B}$

* The 'y' is one only if A and B are one.

2) OR Operation:

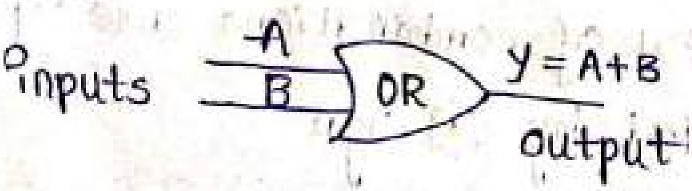
⟹ If two (or) more operands are ored, then the result will be a logic 1, if any one of the operand is a logic 1.

⟹ It has "four" possible combinations.

⟶ It is represented by positive sign.

0 OR 0 = 0	0+0 = 0
0 OR 1 = 1	0+1 = 1
1 OR 0 = 1	1+0 = 1
1 OR 1 = 1	1+1 = 1

Symbol:



* If A & B are the operands and 'y' is the result of the "OR" operation then,

$$\boxed{y = A+B}$$

* If y is zero both A & B are zero
(or)

* The y is one when any one of the operand is one.

3) NOT Operation:

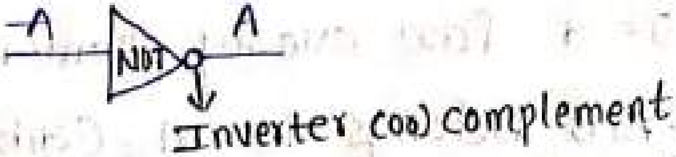
The NOT operation corresponds to the complementing the input variables. There could be only one input and one output and the output is NOT the input.

* Then the number of possible combination of the operands (0 & 1) are $2^1 = 2$.

* If the input $A = 1 \Rightarrow y = \bar{1} = 0$
* If the input $A = 0 \Rightarrow y = \bar{0} = 1$

$$\left.\begin{array}{l} \text{NOT } 0 = 1 \\ \text{NOT } 1 = 0 \end{array}\right\} \begin{array}{l} \bar{0} = 1 \\ \bar{1} = 0 \end{array}$$

Symbol:



* If A is the input variable then the result of the NOT operation is, $y = \bar{A}$.

Axiomatic Definitions of Boolean Algebra:

(or)

Fundamental Postulates of Boolean Algebra:

→ Anything which is not proved but assumed to be TRUE is known as postulate.

→ The theorems of Boolean algebra can be derived from these postulates.

* If 'M' is used as a set and a&b are the elements (or) two objects. Then, the notation

$\boxed{a,b \in M}$

1) An operation $(\cdot)$ is defined such that,

(i) If $y = A \cdot B$ then $y = a \cdot b$ then, y belongs to M for $\boxed{y \in M}$ every pair of elements $a,b \in M$.

(ii) An operation '+' is defined such that if $y = a+b$, then y belongs to M $(y \in M)$ for every pair of elements $a,b \in M$.

2) (i) There exist an element 'one' in M such that $a \cdot 1 = a$ for every element $a \in M$.

(ii) There exist an element 'zero' in M such that $a+0 = a$ for every element $a \in M$.

3) For A,B belongs to M $(a,b \in M)$ the commutative law follow as

(i) $a+b = b+a$

(ii) $a \cdot b = b \cdot a$

4) For a,b,c belongs M then, the distributive law holds as follows

(i) $a(b+c) = ab+ac$

(ii) $a+(b \cdot c) = (a+b) \cdot (a+c)$

5) For every element a belongs to M, there exist an element $\bar{a}$. Such that,

(i) $a \cdot \bar{a} = 0$

(ii) $a+\bar{a} = 1$

6) There are atleast two elements a, b belongs to M. such that, $a \neq b$.

Basic Theorems & Properties of Boolean -Algebra:-

Duality:- The principle of duality says that, starting with a boolean relation, you can derive another boolean relation by

(i) Changing each OR sign to AND sign

(ii) Changing each AND sign to OR sign

(iii) Complementing any 0 (or) 1 in the given expression by keeping literals as it is.

For eg:- What is Dual expression of Dual$(A+\overline{A})=1$

$A\cdot\overline{A}=0$

Note: Duality is very important property of Boolean Algebra (or) Switching Algebra.

⟹ For 'n' variables, maximum possible self dual functions are $2^{(2^n/2)}$

Basic Theorems:

1) AND Operation Theorem:

(i) $A\cdot 0 = 0$

(ii) $A\cdot 1 = A$

(iii) $A\cdot A = A$

(iv) $A\cdot\overline{A} = 0$

Proof:

(i) $A\cdot 0 = 0$

if $A=0 \Rightarrow 0\cdot 0 = 0$

if $A=1 \Rightarrow 1\cdot 0 = 0$

(ii) $A\cdot 1 = A$

if $A=0 \Rightarrow A\cdot 1 = 0\cdot 1 = 0$

if $A=1 \Rightarrow 1\cdot 1 = 1$

(iii) $A\cdot A = A$

if $A=0 \Rightarrow 0\cdot 0 = 0$

if $A=1 \Rightarrow 1\cdot 1 = 1$

(iv) $A \cdot \overline{A} = 0$

if $A = 0 \Longrightarrow 0 \cdot 1 = 0$

if $A = 1 \Longrightarrow 1 \cdot 0 = 0$

2. OR Operation Theorem:

(i) $A + 0 = A$

(ii) $A + 1 = 1$

(iii) $A + A = A$

(iv) $A + \overline{A} = 1$

Proof:

(i) $A + 0 = A$

if $A = 0 \Longrightarrow 0 + 0 = 0$

if $A = 1 \Longrightarrow 1 + 0 = A$

(ii) $A + 1 = 1$

if $A = 0 \longrightarrow 0 + 1 = 1$

if $A = 1 \Longrightarrow 1 + 1 = 1$

(iii) $A + A = A$

if $A = 0 \longrightarrow 0 + 0 = 0$

if $A = 1 \longrightarrow 1 + 1 = 1$

(iv) $A + \overline{A} = 1$

if $A = 0 \Longrightarrow 0 + 1 = 1$

if $A = 1 \Longrightarrow 1 + 01 = 1$

3. Involution Theorem (or) NOT Operation Theorem:

(i) $(A')'$ (or) $\overline{(\overline{A})} = A$

Proof:

if $A = 0 \Longrightarrow \overline{A} = 1$ and $\overline{\overline{A}} = 0$

if $A = 1 \Longrightarrow \overline{A} = 0$ and $\overline{\overline{A}} = 1$

Here, both A and $\overline{\overline{A}}$ are same.

4. Commutative Law:

(i) $A + B = B + A$

(ii) $A \cdot B = B \cdot A$

Proof:

(i) $A + B = B + A$

if $A = 0; B = 0$ then,

$LHS = 0 + 0 = 0$ | $RHS = 0 + 0 = 0$

$\boxed{LHS = RHS}$

if $A = 0; B = 1$ then

$LHS = 0 + 1 = 1$ | $RHS = 1 + 0 = 1$

$\boxed{LHS = RHS}$

if $A = 1; B = 0$ then,

(LHS) $1 + 0 = 0 + 1$ (RHS)

$1 = 1$

if $A=1$; $B=1$ then,

$$1+1 = 1+1$$

(LHS) $1 = 1$ (RHS)

(ii) $A \cdot B = B \cdot A$

if $A=0$; $B=0$ then,

$$0 \cdot 0 = 0 \cdot 0$$

(LHS) $0 = 0$ (RHS)

if $A=0$; $B=1$ then,

$$0 \cdot 1 = 1 \cdot 0$$

(LHS) $0 = 0$ (RHS)

if $A=1$; $B=0$ then,

$$1 \cdot 0 = 0 \cdot 1$$

(LHS) $0 = 0$ (RHS)

if $A=1$; $B=1$ then,

$$1 \cdot 1 = 1 \cdot 1$$

(LHS) $1 = 1$ (RHS)

5) Associative Law:

(i) $A+(B+C) = (A+B)+C = A+B+C$

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

proof:

(i) $A+(B+C) = A+B+C$

if $A=0$; $B=0$; $C=1$

$A+(B+C) = 0+(0+1) = 0+1 = 1$

$(A+B)+C = (0+0)+01 = 0+1 = 1$

$A+B+C = 0+0+1 = 1 \Rightarrow$ LHS = RHS

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

if $A=1;\ B=1;\ C=0$

$A\cdot(B\cdot C) = 1\cdot(1\cdot 0) = 0$

$(A\cdot B)\cdot C = (1\cdot 1)\cdot 0 = 0$

$A\cdot B\cdot C = 1\cdot 1\cdot 0 = 0$

$\therefore$ LHS = RHS

6) Distributive law:

(i) $A\cdot(B+C) = A\cdot B + A\cdot C$

(ii) $A+(B\cdot C) = (A+B)\cdot(A+C)$

proof:

(i) $A\cdot(B+C) = A\cdot B + A\cdot C$

LHS = $A\cdot(B+C)$ RHS = $A\cdot B + A\cdot C$

if $A=1;\ B=1;\ C=1$

LHS = $1\cdot(1+1)$
$= 1\cdot 1$
$= 1$

RHS = $1\cdot 1 + 1\cdot 1$
$= 1+1$
$= 1$

LHS = RHS

7) Auxillary Laws:

(i) $A + A \cdot B = A$

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

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

proof:

(i) $A + A \cdot B = A$

$LHS = A + A \cdot B$

$= A(1+B)$

$= A(1)$ $(\because 1+B = 1)$

$= A$

$= RHS$

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

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

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

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

$= AB + A\bar{B} + \bar{A}B$ $\left\{ \begin{array}{l} \therefore A \cdot 1 = A \\ A \cdot A = A \\ A + A = A \end{array} \right\}$

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

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

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

$= A \cdot 1 + B \cdot 1$

$= A + B$

$= RHS$

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

$$\begin{aligned} LHS &= AB + BC + \overline{B}C \\ &= AB + C(B+\overline{B}) \\ &= AB + C \\ &= RHS \end{aligned}$$

Note :-

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

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

8) Demorgan's Theorem:

Demorgan's theorem is used in simplifying expression in which product (or) sum variables inverted.

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

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

Proof: The complement of the product is equals to the sum of the individual complements.

Truth Table:

A	B	$A \cdot B$	$\overline{A \cdot B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

The complement of sum is equals to the product of the individual complements.

A	B	$A+B$	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

9) Transposition Theorem:

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

proof: Take LHS,

$(A+B)(A+C)$

$= A.A + A.C + B.A + B.C$

$= A + AC + AB + BC \quad (\because A.A = A)$

$= A(1+C) + AB + BC \quad (\because 1+C = 1)$

$= A.1 + AB + BC$

$= A + AB + BC$

$= A(1+B) + BC$

$= A.1 + BC \quad (\because A.1 = A)$

$= A + BC$

$= RHS$

10) Distribution Theorem:

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

Consider LHS,

$$A+BC = LHS$$

Proof: Take $RHS = (A+B)(A+C)$

$$= A \cdot A + A \cdot C + B \cdot A + B \cdot C$$

$$= A + AC + AB + BC$$

$$= A(1+C) + AB + BC$$

$$= A \cdot 1 + AB + BC$$

$$= A + AB + BC$$

$$= A(1+B) + BC$$

$$= A \cdot 1 + BC$$

$$= A + BC$$

$$= LHS$$

11) Consensus Theorem:

⟹ This theorem is used to eliminate redundant term.

⟹ It is applicable only when if a boolean function,

(i) Contains 3-variables

(ii) each variable used 2-times

(iii) Only one variable is in complemented (or) uncomplemented form.

(iv) Then the related terms to that complemented (or) uncomplemented variable is the answer.

for example:

(i) $AB + \bar{A}C + BC = ?$

So:- Here, number of variables = 3 i.e., A, B, C

each used 2 times

complemented variable is 'A'

So, $AB + \bar{A}C + BC = AB + \bar{A}C$

Here, BC is the redundant term.

(ii) Solve the given expression using consensus theorem

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

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

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

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

12) Complementary Theorem:

For obtaining complement expression we changes each OR sign by AND sign and vice-versa

Complement any '0' and '1' appearing in expression

Complement the individual literals

Eg:- Write complement function of

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

$\bar{f}$ = complement of f

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

Boolean Function (or) Switching Functions:-

⇒ Boolean expressions are constructed by connecting the Boolean constants and variables with the Boolean operations.

These Boolean expressions to describe Boolean functions.

* We used Boolean expressions are also known as Boolean formulas.

* for example, if the Boolean expression $(A+\bar{B})C$ is used to describe the function f, then Boolean function is written as,

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

* Let us consider the four-variable Boolean function.

$$f(A,B,C,D) = \overset{\downarrow}{\underset{\uparrow}{A}} + \overset{\downarrow}{\underset{\uparrow}{\bar{B}}\,\underset{\uparrow}{C}} + \overset{\downarrow}{\underset{\uparrow}{A}\,\underset{\uparrow}{C}\,\underset{\uparrow}{\bar{D}}} \quad \begin{matrix}\text{Product terms}\\ \\ \text{Literals}\end{matrix}$$

Note:- A binary variable in either a complemented (or) an uncomplemented form is called a literal.

Let us consider another variable Boolean function.

$$f(A,B,C,D) = \overset{\downarrow}{(\underset{\uparrow}{B}+\underset{\uparrow}{\bar{D}})} \cdot \overset{\downarrow}{(\underset{\uparrow}{A}+\underset{\uparrow}{\bar{B}}+\underset{\uparrow}{C})} \cdot \overset{\downarrow}{(\underset{\uparrow}{\bar{A}}+\underset{\uparrow}{C})} \quad \begin{matrix}\text{Sum terms}\\ \\ \text{Literals}\end{matrix}$$

Boolean Function Representation

Canonical form

- All the terms contains each literal

Standard form

- All the term do not have each literal

Eg:- $F(A,B,C) = \bar{A}BC + ABC + \bar{A}BC$ Eg:- $F(A,B,C) = A + BC + A\bar{B}C$

* All these literals and terms are arranged in one of the two forms:

1) Sum of product form (SOP) and
2) Product of sum form (POS)

1) Sum of product Form (SOP):

⟹ A sum of products (SOP) is a group of product terms ORed together.

⟹ The SOP expression usually takes the forms of two (or) more variables ANDed together.

Example : 1) $f(A,B,C) = \underbrace{ABC \overset{\text{sum}}{+} AB\bar{C}}_{\text{product terms}}$

2) $f(P,Q,R,S) = \underbrace{\bar{P}Q \overset{\text{sum}}{+} \bar{Q}R \overset{\text{sum}}{+} RS}_{\text{product terms}}$

⟶ SOP forms are used to write logical expression for the output becoming Logical '1'.

2) Product of Sum Form (POS):-

→ A product of sums (POS) is any groups of sum terms ANDed together.

→ Each of these product of sums expressions consists of two (or) more sum terms, (OR) that are ANDed together.

Example:- 1) $f(A,B,C) = \underbrace{(A+B)\overset{\text{product}}{\cdot}(\bar{B}+C)}_{\text{sum terms}}$

2) $f(P,Q,R,S) = \underbrace{(P+Q)\overset{\text{product}}{\cdot}(\bar{R}+\bar{S})\overset{}{\cdot}(P+S)}_{\text{sum terms}}$

→ POS forms are used to write logical expression for the output becoming logic '0'.

Canonical Form (Standard SOP and POS Forms):

→ The canonical forms are the special cases of SOP and POS forms. These are also known as standard SOP and POS forms.

1) Standard SOP Form (or) Minterm Canonical Form:

→ We can realize that in the SOP form, all the individual terms do not invove all literals For example, in expression $AB + AB\bar{C}$ the first product term do not contain literal c.

→ If each term in SOP form contains all the literals then the SOP form is known as standard (or) canonical SOP form

⇒ Each individual term in the standard SOP form is called minterm. Therefore, canonical form is also known as minterm canonical form.

For eg: $f(A,B,C) = AB\bar{C} + ABC + \bar{A}BC + A\bar{B}C$

Each product term consists of all literals in either complemented form (or) uncomplemented form

standard SOP form

2) Standard POS Form (or) Maxterm Canonical Form:

⇒ If each term in POS form contains all the literals then the POS form is known as standard (or) canonical POS form

→ Each individual term in the standard POS form is called maxterm. Therefore, canonical POS form is also known as maxterm canonical form.

for example: $f(A,B,C) = (A+B+C)\cdot(A+\bar{B}+C)$

Each sum term consists of all literals either complemented (or) uncomplemented form

Converting Expressions in Standard SOP (or) POS Form

Steps to convert SOP to standard SOP form:

(i) Find the missing literal in each product term if any.

(ii) AND each product term having missing literals with term form by ORing the literal and it's complement.

(iii) Expand the terms by applying distributive law and reorder the literals in the product terms

(iv) Reduce the expression by omitting repeated product terms if any. Because $A+A=A$

Example ①: Convert the given expression in standard SOP form $f(A,B,C) = AC+AB+BC$

Sol:- (i) Identify the missing variables in product terms

$$f(A,B,C) = AC + AB + BC$$

BC → Literal A is missing

AB → Literal C is missing

AC → Literal B is missing

(ii) Multiply (variable + it's complement) (or) AND product term with (missing literal + it's complement)

$$f(A,B,C) = AC\,(B+\bar{B}) + AB\,(C+\bar{C}) + BC\,(A+\bar{A})$$

Missing literals & their complements

(iii) Expand the terms and reorder literals

$$f(A,B,C) = ABC + A\bar{B}C + ABC + AB\bar{C} + ABC + \bar{A}BC$$

(iv) Omit repeated product terms (or) neglect them

$$f(A,B,C) = ABC + ABC + ABC + AB\bar{C} + ABC + \bar{A}BC$$

$$\therefore f(A,B,C) = ABC + A\bar{B}C + AB\bar{C} + \bar{A}BC$$

Example ②:- Convert the given expression in standard SOP form.

$$f(A,B,C) = A + ABC$$

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

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

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

$$\therefore f(A,B,C) = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} \quad [\because A + A = A]$$

Steps to convert POS to standard POS form:

(i) Find the missing literals in each sum term if any.

(ii) OR each sum team having missing literals with terms form by ANDing the literal and it's complement.

(iii) Expand the terms by applying distributive law and reorder the literals in the sum terms.

(iv) Reduce the expression by omitting repeated sum terms if any because $A.A = A$

Example ①:- Convert the given expression in standard POS form.

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

Sol:- (i) finding the missing literal in each term

$$f(A,B,C) = (A+B).(B+C).(A+C)$$

↳ Literal B is missing

↳ Literal A is missing

↳ Literal C is missing

(ii) OR sum term with (missing literal. its complement)

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

(iii) Expand the terms and reorder literals

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

(iv) Omit repeated sum-terms

$$\therefore f(A,B,C) = (A+B+C).(A+B+\bar{C}).(\bar{A}+B+C).(A+\bar{B}+\bar{C}) \quad [\because A.A = A]$$

Example ②:- Convert the given expression in standard POS form

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

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

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

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

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

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

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

* M - Notations: Minterms and Maxterms

⟶ Each individual term in standard SOP form is called minterm and each individual term in standard POS form is called maxterm.

⟹ The concept of minterms and maxterms allows us to introduce a very convenient shorthand notations to express logical functions.

⟹ n - binary variables have 2^n possible combinations and each of these possible combination is called "Minterm (or) standard product".

Representation: Minterm ⟹ $\Sigma m(\ldots)$

Maxterm ⟹ $\Pi M(\ldots)$

⟶ Maxterm is the complement of corresponding minterm i.e., $M = \bar{m}$

For 3 variable:

A B C	Minterms (mi)	Maxterms (Mi)
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	$\overline{A}\,B\,\overline{C} = m_2$	$A+B+C = M_2$
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	$A\,B\,C = m_7$	$\overline{A}+\overline{B}+\overline{C} = M_7$

Minterms and maxterms for three variables

→ From this above table we conclude that

→ Each minterm is represented by mi and each maxterm is represented by Mi, where the subscript 'i' is the decimal number equivalent of the natural binary number.

→ In minterm we assign '1' to each uncomplemented variable and '0' to each complemented variable.

→ In maxterms we assign '0' to each uncomplemented variable and '1' to each complemented variable.

For example: Let us consider, the following truth table

Input (3-variables)			Output (Y)
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

The standard sum of product form is as,

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

$$\therefore Y = m_3 + m_5 + m_6 + m_7$$

also, $\boxed{Y = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC}$

The standard product of sum form is as,

$$f(A,B,C) = \Pi M\ (0,1,2,4) \longrightarrow ②$$

$$\therefore Y = M_0 \times M_1 \times M_2 \times M_4$$

also,

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

Note: We also conclude that from equation ① and from equation ② if,

$$Y = \Sigma m(3,5,6,7)$$
$$Y = \Pi M(0,1,2,4)$$

Algebraic Simplification:

simplify the following expression:

(i) $A.\bar{A}C$

$\Rightarrow A.\bar{A}C$

$0.C$

$= 0$ [∵ from AND operation theorem $A.\bar{A} = 0$]

$\therefore A.\bar{A}C = 0$

(ii) $ABCD + ABD$

$= ABD(1+C)$

$= ABD.1$ [∵ from OR operation theorem $1+C = 1$ and, AND operation $A.1 = A$]

$= ABD$

$\therefore ABCD + ABD = ABD$

(iii) $ABCD + A\bar{B}CD$

$= ACD(B+\bar{B})$

$= ACD.1$ [∵ $B+\bar{B} = 1$

$= ACD$ $A.1 = A$]

$\therefore ABCD + A\bar{B}CD = ACD$

(iv) $A(A+B)$

$= A.A + A.B$

$= A + AB$ [∵ $A.A = A$,

$= A(1+B)$ $1+B = 1$,

$= A.1 = A$ $A.1 = A$]

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

(v) $AB + ABC + AB(D+E)$

$= AB(1+C+D+E)$ $[\because 1+A=1$, $A.1 = A]$

$= AB.1$

$= AB$

$\therefore AB + ABC + AB(D+E) = AB$

(vi) $xy + xyz + xy\bar{z} + \bar{x}yz$

$= xy(1+z) + xy\bar{z} + \bar{x}yz$

$= xy.1 + xy\bar{z} + \bar{x}yz$ $[\because 1+A=1$, $1.A = A]$

$= xy + xy\bar{z} + \bar{x}yz$

$= xy(1+\bar{z}) + \bar{x}yz$ $[\because 1+A=1]$

$= xy.1 + \bar{x}yz$ [∵ from Auxilary laws $A+\bar{A}B = A+B$]

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

$= y(x+z)$

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

(vii) $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC$

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

$= \bar{A}.\bar{C}.1 + \bar{A}BC$ $[\because A.1 = A]$

$= \bar{A}.\bar{C} + \bar{A}BC$

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

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

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

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

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

$= AC.1 + AB\bar{C}$ $\quad [\because A+\bar{A}=1$ & $A.1 = A]$

$= AC + AB\bar{C}.$

$= A(C+\bar{C}.B)$ $\quad [\because A+\bar{A}B = A+B]$

$= A(C+B)$

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

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

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

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

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

$= B\bar{D}.1 + B\bar{C}D$ $\quad [\because A+\bar{A}=1$ & $A.1 = A]$

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

$= B(\bar{D} + \bar{\bar{D}}\bar{C})$ $\quad [\because A+\bar{A}B = A+B$, $\bar{A}+AB = \bar{A}+B]$

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

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

(x) $AC + C(A+\bar{A}B)$

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

$= AC + AC + \bar{A}BC$ $\quad [\because A+A = A]$

$= AC + \bar{A}BC$

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

$= C(A+B)$

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

(xi) $\overline{A}B\overline{C}D + \overline{A}BCD + ABD$

$= \overline{A}BD(C+\overline{C}) + ABD$

$= \overline{A}BD.1 + ABD$ [∵ $A+\overline{A}=1$ & $A.1=A$]

$= \overline{A}BD + ABD$

$= BD(A+\overline{A})$ [∵ $A+\overline{A}=1$]

$= BD.1$ [∵ $A.1=A$]

$= BD$

∴ $\overline{A}B\overline{C}D + \overline{A}BCD + ABD = BD$

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

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

$= A+B+A\overline{B}$ [∵ $A+\overline{A}B = A+B$

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

$A+A = A$]

$= A+A+B$

$= A+B$

(xiii) $\overline{\overline{AB}+\overline{A}+AB}$

$= \overline{\overline{A}+\overline{B}+\overline{A}+AB}$ [∵ $\overline{AB} = \overline{A}+\overline{B}$,

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

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

$= \overline{\overline{A}+1}$ $B+\overline{B} = 1$,

$= \overline{1} = 0$ $\overline{A}+\overline{A} = \overline{A}$,

∴ $\overline{\overline{A}B+\overline{A}+AB} = 0$ & $A+1 = 1$]

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

$= AB+\overline{AC}+AAB.\overline{B}C + A\overline{B}C.C$

$= AB+\overline{AC}+0+A\overline{B}C$ [∵ $B.\overline{B}=0$

$C.C = C$]

$= AB + \bar{A} + \bar{C} + A\bar{B}C$ [∵ $\overline{AB} = \bar{A} + \bar{B}$ Demorgan's theorem]

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

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

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

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

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

$= \bar{A} + \bar{C} + 1$

$= 1$

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

(xv) Simplify the expression $Z = AB + A\bar{B}.(\overline{\bar{A}.\bar{C}})$

Step 1: Apply the Demorgan's theorem and multiply out all terms to get expression in sum of product form.

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

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

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

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

Step 2: Search for common terms for factorization and apply boolean rules

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

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

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

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

$Z = A.1$

$Z = A$

Digital Logic Gates:

Logic Operators:- We know that, to represent and solve arithmetic expression we use arithmetic operators such as +, -, x and ÷ .

- * Similarly, we can use logical operators to represent and solve logical expressions.
- * There are three basic logical operators: NOT / INVERT, AND and OR.

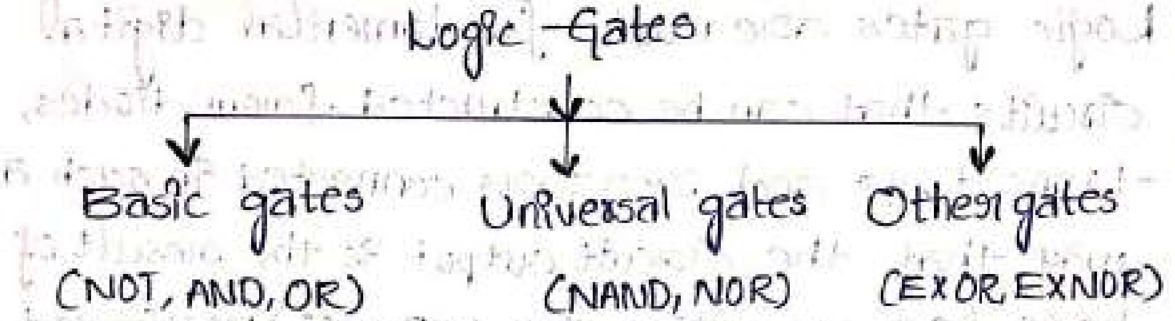
Logic Gates:

- * Logic gates are most fundamental digital circuits that can be constructed from diodes, transistors and resistors connected in such a way that the circuit output is the result of a basic logic operation (OR, AND, NOT) performed on the inputs
- * Logic Gate is a digital circuit that has one (or) more inputs and one output.
- * The function of each logic gate will be represented by boolean expression
- * The boolean '0' & '1' represents the "Logic Level".

Logic '0'	Logic '1'
False	True
OFF	ON
Low	High
NO	Yes
Open switch	Closed switch

* The operation of a logic gate can be easily understood with the help of "truth table".
* A truth table is a table that shows all the input & output possibilities of a logic circuit. i.e., the truth table indicates the outputs for different possibilities of the inputs.
* The number of input combinations will equal 2^N for an N-input truth table.

Classification of Logic Gates:


Logic Gates

Basic gates (NOT, AND, OR)

Universal gates (NAND, NOR)

Other gates (EXOR, EXNOR)

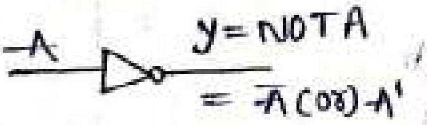
Basic Gates:

* NOT Gate (or) Inverter:

→ NOT gate is a logic circuit with only one input and one output and carries out the "NOT" operation.

→ The inverter (or) NOT circuit performs a basic logic function called "inversion" (or) "complementation".

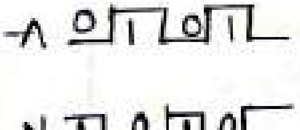
→ The inverter changes one logic level to its opposite.



NOT Logic Symbol

Input A	output $y = \bar{A}$
0	1
1	0

Truth Table



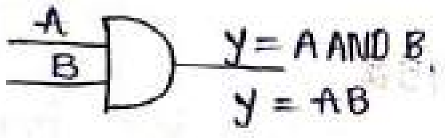
Waveform

→ NOT gate IC Number : 7404

The AND Gate:

* The AND gate performs logical multiplication, more commonly known as the AND function.

* The AND gate may have two (or) more inputs and a single output.

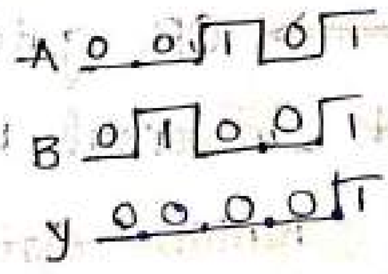


Logic symbol for a two input AND gate

Truth Table:

Inputs		output
A	B	Y=AB
0	0	0
0	1	0
1	0	0
1	1	1

wave-form:



Logic: In AND gate, all inputs must be high to get a high output.

* AND gate IC number is 7408

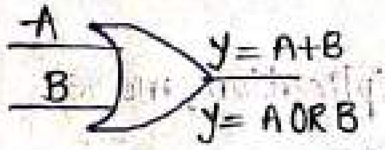
The OR Gate:

* The OR gate performs logical addition, more commonly known as the OR function.

* The OR gate is a logic circuit to perform the OR operation.

* The OR gate has two (or) more input signals but only one output signal.

Logic: If any one of the input is high, the output is high.

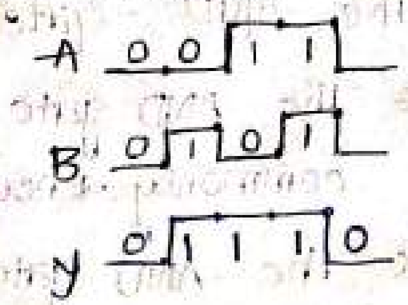


Logic symbol for a two input OR gate

Truth Table:

Inputs		output
A	B	y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

waveforms:



⟹ OR gate IC number is 7432

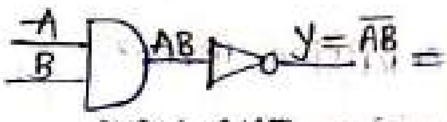
* Universal Logic Gates:

NAND Gate:

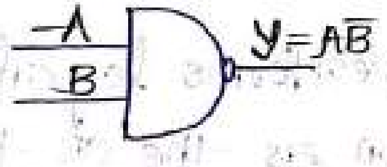
⟶ The term NAND is a contraction of NOT-AND and implies an AND-function with inverted output

⟶ The NAND gate has two (or) more inputs but only one output.

Logic:- All inputs must be high to get a low output.



Logic structure of a NAND gate



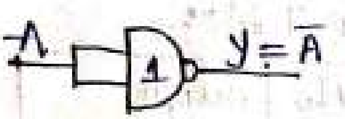
Logic symbol for a two input NAND gate

Truth Table:

Inputs		Output
A	B	y = $\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

⟶ NAND gate is a universal gate it can be used to construct an NOT gate, AND gate, OR gate and other gates.

(i) NOT gate:



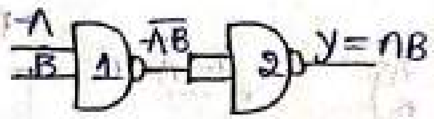
Input is A

Output is

$y=\overline{A\cdot A}$

$y=\overline{A}$

(ii) AND Gate:

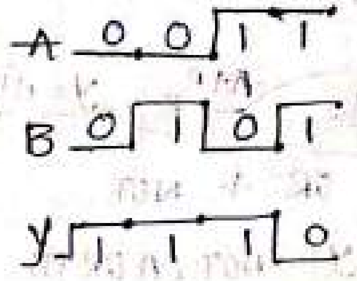


$y=\overline{\overline{AB}\cdot\overline{AB}}$

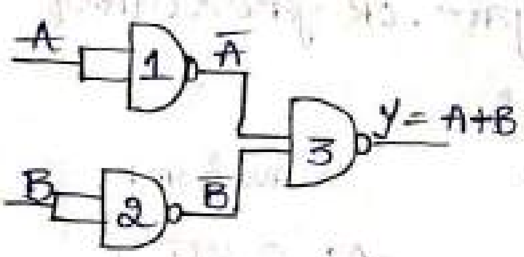
$y=\overline{\overline{AB}}$

$y=AB$

Waveform:



(iii) OR Gate:

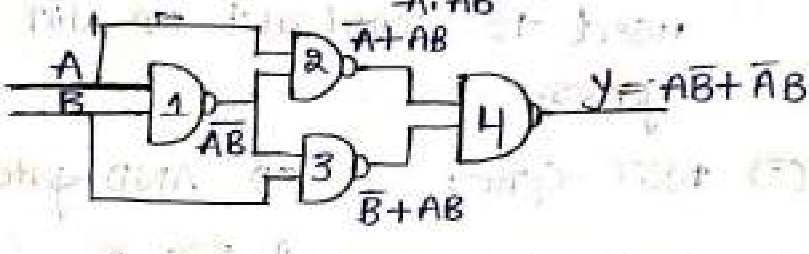


$y=\overline{\overline{A}\cdot\overline{B}}$

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

$y=A+B$

(iv) EX-OR Gate:



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

$y=\overline{\overline{A}+AB}+\overline{\overline{B}+AB}$

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

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

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

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

$y=A\oplus B$

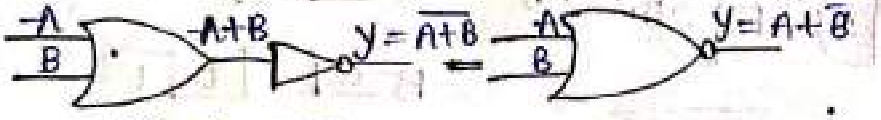
⟹ NAND gate IC number is 7400

NOR Gate:

⟹ The term NOR is a contraction of NOT-OR and implies an OR-function with an inverted output.

⟹ The NOR gate has two (or) more input signals but only one output signal.

Logic: All inputs must be low to get a high output



OR + NOT

y = NOT (A OR B)

Logic structure of a NOR gate

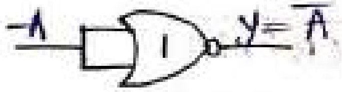
Logic symbol for a two input NOR gate

Truth Table:

Inputs		output
A	B	$y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

⟶ NOT gate is a universal logic gate it can be used to construct an NOT gate, OR gate & other gates.

(i) NOT Gate:



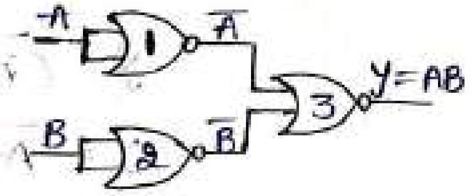
input is A

Output is

$y = \overline{A+A}$

$y = \overline{A}$

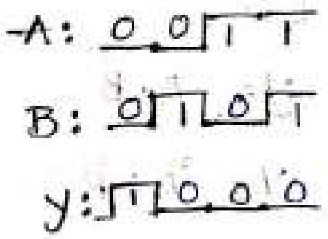
(ii) AND gate:



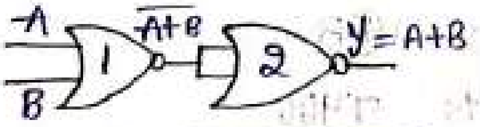
$y = \overline{\overline{A}+\overline{B}} = \overline{\overline{A}}.\overline{\overline{B}}$

$y = A.B$

Waveform:



(iii) OR-Gate:

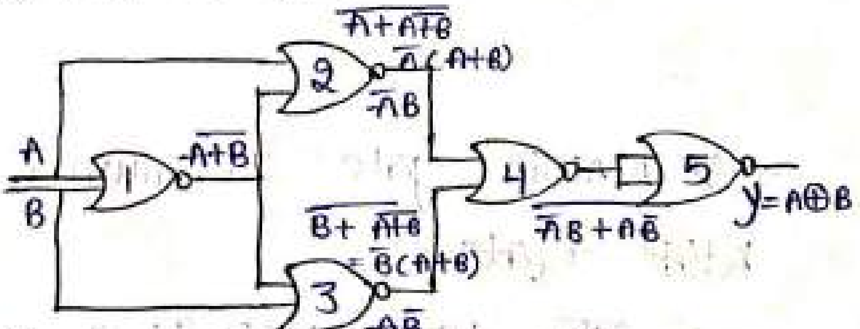


$y = \overline{\overline{A+B}+\overline{A+B}}$

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

$y = A+B$

(iv) EX-OR Gate:



i.e., $y = \overline{\overline{(\overline{A}B+A\overline{B})}}$

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

$y = A \oplus B$

⟹ NOR gate IC number 7402

Other Gates:

Exclusive OR Gate (or) EX-OR Gate:

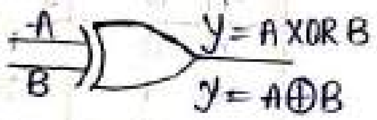
⟹ An exclusive OR gate is a logic circuit, which has two (or) more inputs and one output.

Logic: The output is a logic 1 only when there are odd number of 1's at the input.

⟹ It acts like as an "Odd number of 1's detector in the input". It is also called "stair case switch".

⟶ It is mostly used in "parity generation and detection".

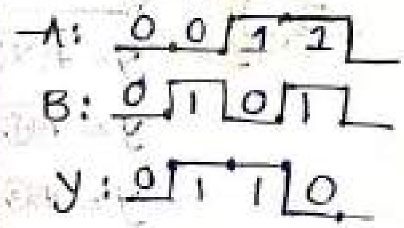
Logic Symbol:



Truth Table:

Inputs		output
A	B	$y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Waveforms:



⟶ IC Number of EX-OR gate is 7486

⟶ Boolean function of 2 input EX-OR operation is

$$\boxed{y = A \oplus B = \bar{A}B + A\bar{B}}$$

⟶ From the truth table, we can say when both the inputs are same then output becomes low (or) Logic '0'.

i.e., If $A = B \rightarrow y = 0$ and

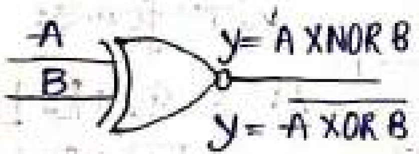
If $A \neq B \rightarrow y = 1$

The Exclusive NOR gate (or) EX-NOR-Gate:

→ It is also called "Equivalence gate" (or) "coincidence logic circuit".

Logic: The EX-NOR gate has two (or) more inputs but only one output. When both the inputs are same then output becomes High (or) logic 1.

→ It acts like as an "even number of 1's detector" when number of input variables are even and also called "odd number of 1's detector" when number of input variables are odd.

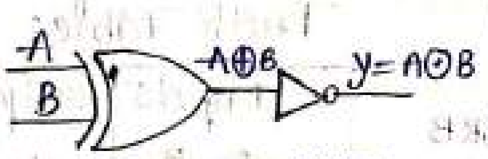


$y = \overline{A\ XOR\ B}$

$y = \overline{A \oplus B}$

$y = A \odot B$

Logic Symbol for EXNOR Gate



EX-OR + NOT

Logic Structure for EX-NOR gate

Truth Table:

Inputs		output
A	B	$y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

⟹ Boolean function of 2-input EX-NOR Operation is as, $y = A \odot B = \overline{A \oplus B} = \overline{\bar{A}B + A\bar{B}}$

$$\boxed{\therefore y = \bar{A}\bar{B} + AB = A \odot B}$$

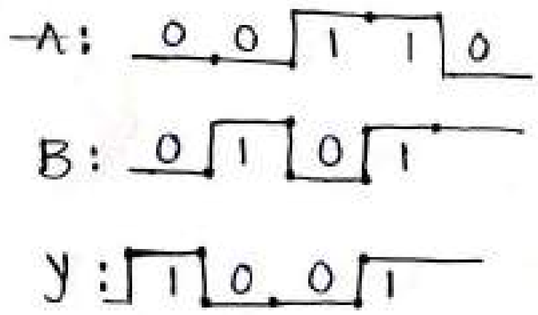
⟹ "EXNOR" operation like an exor gate followed by an "INVERTER".

→ If $A=B \Rightarrow y=1$ and

If $A \neq B \Rightarrow y=0$.

→ EX-NOR gate IC number is 74266.

Waveform:



Note:

* $A \oplus A \oplus A \oplus \ldots\ldots$ upto n terms

$= 0$, when $n =$ even

$= A$, when $n =$ odd

* $A \odot A \odot A \odot \ldots\ldots$ upto n terms

$= 1$, when $n =$ even

$= A$, when $n =$ odd

UNIT-II

Gate Level Minimization

* The Map Method.

→ This Method is developed by Karnaugh in 1953

→ K-map method is used to simplify boolean algebraic expressions

→ The "Karnaugh map" is a graphical chart which provides a systematic method for simplifying and manipulating the Boolean expressions (or) to convert a truth table to its corresponding logic circuit in a simple, orderly process

→ In this technique, the information contained in a truth table (or) available in sop (or) pos form is represented on K-map. It contains boxes called cells.

→ It is generally used up to 6-variables

→ In an n-variable K-map there are 2^n cells

→ "Gray Code" has been used for the identification of cells.

* Two-Variable K-map:

→ 2 variable K-map contains $2^2 = 4$ cells

→ Four minterms (or) Maxterms

A \ B	$\overline{B}$ ← 0	B ← 1
0 → $\overline{A}$	m_0	m_1
1 → A	m_2	m_3

(For SOP)

A \ B	B ← 0	$\overline{B}$ ← 1
A ← 0	M_0	M_1
$\overline{A}$ ← 1	M_2	M_3

(For POS)

* Three-Variable K-map:-

→ 3 variable K-map contains $2^3 = 8$ cells

→ Eight minterms (or) Maxterms

A \ BC	$\overline{B}\overline{C}$ ← 00	$\overline{B}C$ ← 01	BC ← 11	$B\overline{C}$ ← 10
$\overline{A}$ ← 0	m_0	m_1	m_3	m_2
A ← 1	m_4	m_5	m_7	m_6

(For SOP)

A \ BC	$(B+C)$ ← 00	$(B+\overline{C})$ ← 01	$(\overline{B}+\overline{C})$ ← 11	$(\overline{B}+C)$ ← 10
A ← 0	M_0	M_1	M_3	M_2
$\overline{A}$ ← 1	M_4	M_5	M_7	M_6

(For POS)

* Four-Variable K-map:-

→ 4 Variable K-map contains $2^4 = 16$ cells

→ 16 minterms (or) Maxterms.

$AB \backslash CD$	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ ← 00	m_0	m_1	m_3	m_2
$\overline{A}B$ ← 01	m_4	m_5	m_7	m_6
AB ← 11	m_{12}	m_{13}	m_{15}	m_{14}
$A\overline{B}$ ← 10	m_8	m_9	m_{11}	m_{10}

(For SOP)

$AB \backslash CD$	$C+D$ 00	$C+\overline{D}$ 01	$\overline{C}+\overline{D}$ 11	$\overline{C}+D$ 10
$A+B$ ← 00	M_0	M_1	M_3	M_2
$A+\overline{B}$ ← 01	M_4	M_5	M_7	M_6
$\overline{A}+\overline{B}$ ← 11	M_{12}	M_{13}	M_{15}	M_{14}
$\overline{A}+B$ ← 10	M_8	M_9	M_{11}	M_{10}

(For POS)

* Five-variable k-map:

→ Five-variable k-map contains $2^5 = 32$ cells

→ 32 minterms (or) Maxterms

→ Here, we have $f(A, B, C, D, E)$

For $A=0$

$BC \backslash DE$	$\overline{D}\overline{E}$	$\overline{D}E$	DE	$D\overline{E}$
$\overline{B}\overline{C}$	m_0	m_1	m_3	m_2
$\overline{B}C$	m_4	m_5	m_7	m_6
BC	m_{12}	m_{13}	m_{15}	m_{14}
$B\overline{C}$	m_8	m_9	m_{11}	m_{10}

[From (0–15)]

For $A=1$

$BC \backslash DE$	$\overline{D}\overline{E}$	$\overline{D}E$	DE	$D\overline{E}$
$\overline{B}\overline{C}$	m_{16}	m_{17}	m_{19}	m_{18}
$\overline{B}C$	m_{20}	m_{21}	m_{23}	m_{22}
BC	m_{28}	m_{29}	m_{31}	m_{30}
$B\overline{C}$	m_{24}	m_{25}	m_{27}	m_{26}

[From (16–31)]

(For SOP)

For A=0(A)

BC \ DE	$D+E$	$D+\bar{E}$	$\bar{D}+\bar{E}$	$\bar{D}+E$
$B+C$	M_0	M_1	M_3	M_2
$B+\bar{C}$	M_4	M_5	M_7	M_6
$\bar{B}+\bar{C}$	M_{12}	M_{13}	M_{15}	M_{14}
$\bar{B}+C$	M_8	M_9	M_{11}	M_{10}

For A=1 ($\bar{A}$)

BC \ DE	$D+E$	$D+\bar{E}$	$\bar{D}+\bar{E}$	$\bar{D}+E$
$B+C$	M_{16}	M_{17}	M_{19}	M_{18}
$B+\bar{C}$	M_{20}	M_{21}	M_{23}	M_{22}
$\bar{B}+\bar{C}$	M_{27}	M_{28}	M_{31}	M_{30}
$\bar{B}+C$	M_{24}	M_{25}	M_{27}	M_{26}

(For pos)

Simplification of logical Function using k-map.

Simplification of logical Function with k-map is based on the principle combining terms in adjecent cells.

Note:- If the Boolean function is in Sop Form, the cells are identified by '1'.

If the Boolean function is in pos form the cells are identified by '0'.

Looping:- The process for combining these '1's (or) 0's is called Looping.

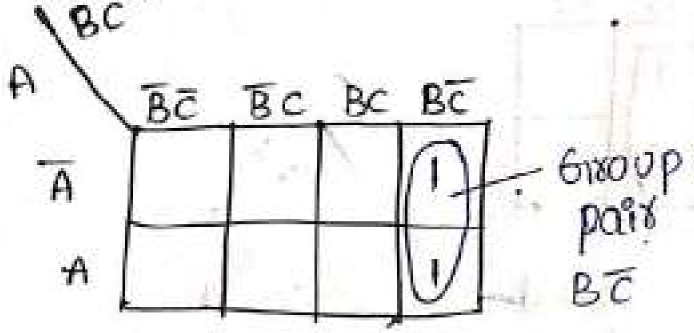
⇒ Groops are made up of 2, 4, 8, 16 and so on

⇒ By folding k-map over its edges, the number of 1's (or) 0's are overlaping forms the Group.

looping group of '2' (pair)

⇒ looping a pair of adjacent 1's in a K-map eleminates the one variable that appearing in both complemented and uncomplemented form

Eg:-

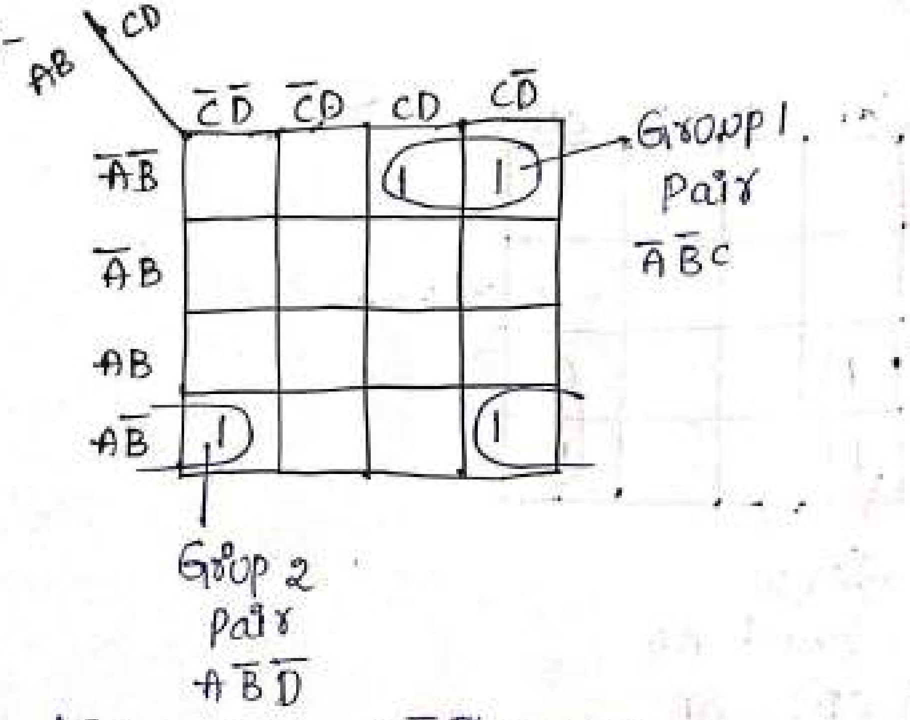


A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$				1
A				1

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

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

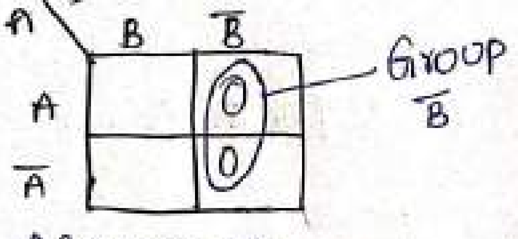
Eg:-



AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$			1	1
$\bar{A}B$				
AB				
$A\bar{B}$	1			1

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

Eg:-



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

$f(A,B) = \bar{B}$

looping Groops of 4(four) (Quad)

looping a Quad of 1's or d's eliminates 2 variables and that apper in both complemented and uncomplemented form.

$A \backslash BC$	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$		1	1	
A		1	1	

Quad

C

$f(A,B,C) = C.$

Eg:-

$AB \backslash CD$	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB	1			1
$A\bar{B}$	1			1

Group

Quad $A\bar{D}$

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

Eg:-

$A \backslash BC$	$B+C$	$B+\bar{C}$	$\bar{B}+\bar{C}$	$\bar{B}+C$
A	0			0
$\bar{A}$	0			0

$\therefore f(A,B,C) = C.$

Group quad C

looping Groups of 8 (Octet)

looping an octet of 1's or 0's eliminating the 3 - Variables that appears in both complemented and uncomplemented form.

For example,

AB \ CD	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$				
$\overline{A}B$	1	1	1	1
AB	1	1	1	1
$A\overline{B}$				

→ octet B

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

Example:-

AB \ CD	$C+D$	$C+\overline{D}$	$\overline{C}+\overline{D}$	$\overline{C}+D$
$A+B$	0	0	0	0
$A+\overline{B}$				
$\overline{A}+\overline{B}$				
$\overline{A}+B$	0	0	0	0

→ Group octet B

1) Minimize the expression by using K-map

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

(or)

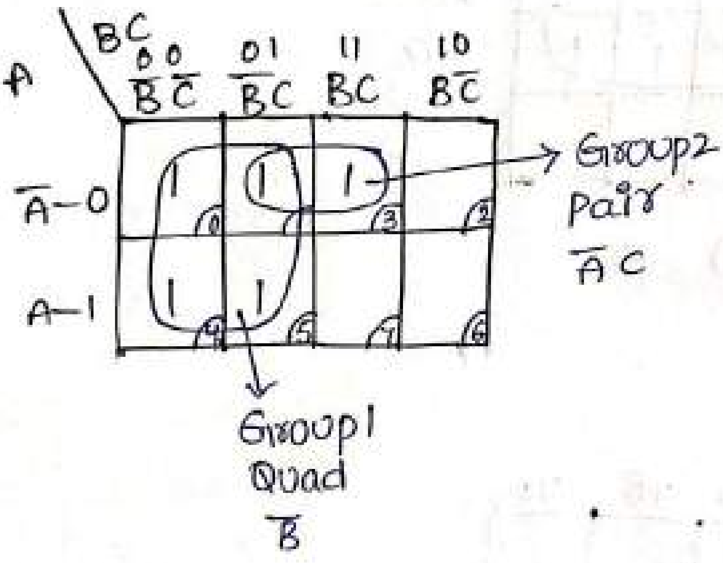
$$Y = \Sigma m(0,1,3,4,5)$$

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

* The given boolean function contains 3 variables

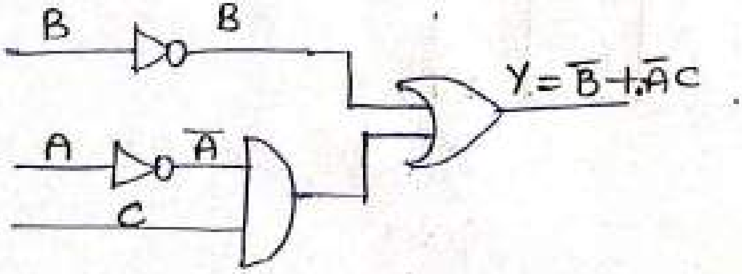
* 3 variable K-map contains 8 cells

$$n = 3 \Rightarrow 2^n = 2^3 = 8 \text{ cells}$$



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

logic diagram :



) Reduce the following the function using K-map technique

$f(A,B,C,D) = \Sigma m(0,1,4,8,9,10)$

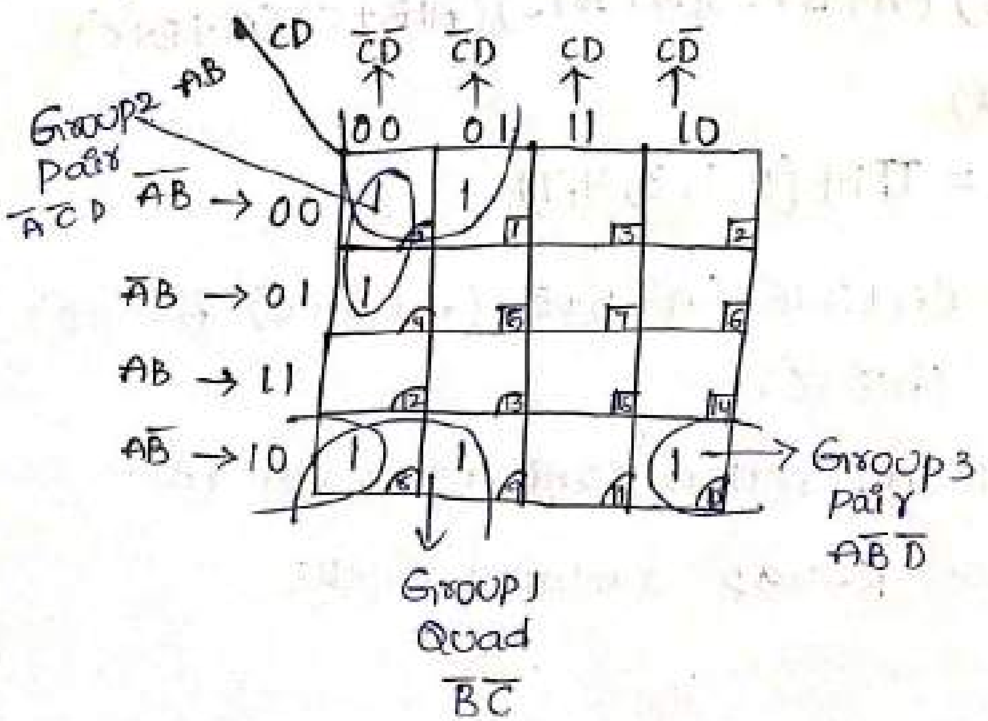
* The Given Functions min-terms and it is in Sum of product form (SOP)

* $f(A,B,C,D) = m_0 + m_1 + m_4 + m_8 + m_9 + m_{10}$

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

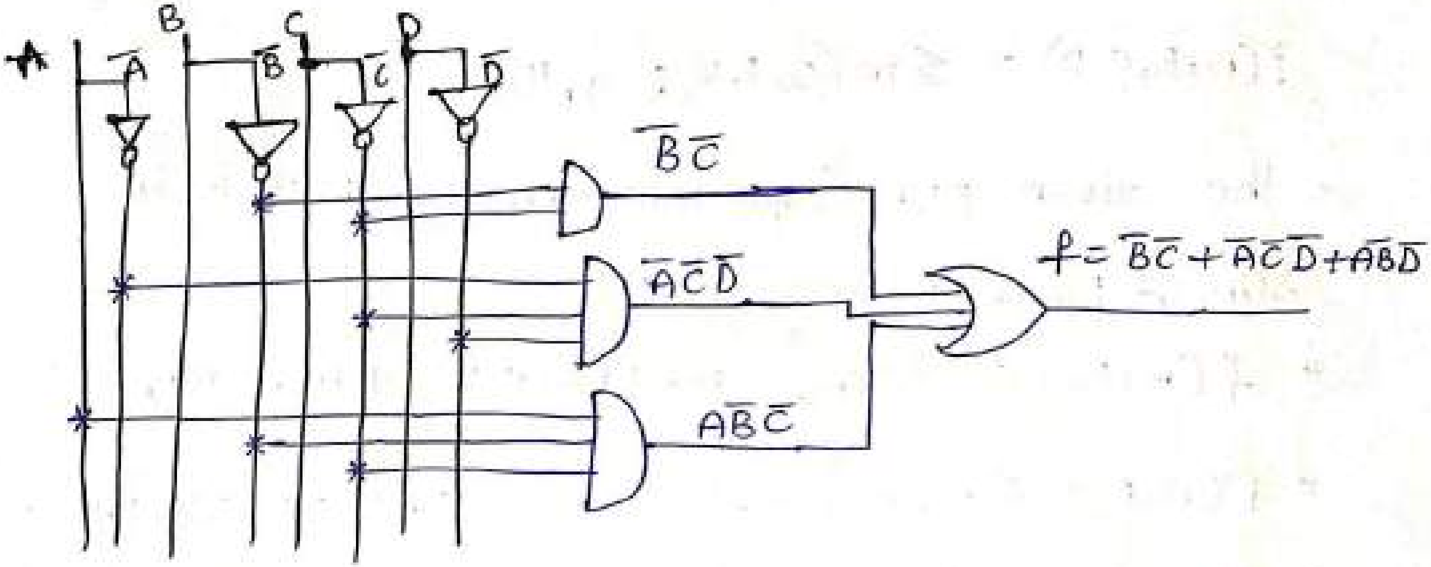
* The given boolean function contains 4 Variables

* 4-variable K-map contains 16 cells.


CD
AB
$\overline{C}\overline{D}$
$\overline{C}D$
CD
$C\overline{D}$
00
01
11
10
Group2
Pair
$\overline{A}\overline{C}D$
$\overline{A}\overline{B} \rightarrow 00$
$\overline{A}B \rightarrow 01$
$AB \rightarrow 11$
$A\overline{B} \rightarrow 10$
1
1
1
1
1
1
0
1
3
2
4
5
7
6
12
13
15
14
8
9
11
10
Group3
Pair
$A\overline{B}\overline{D}$
Group1
Quad
$\overline{B}\overline{C}$

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

logic diagram :


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

3) Minimize the expression By using K-map

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

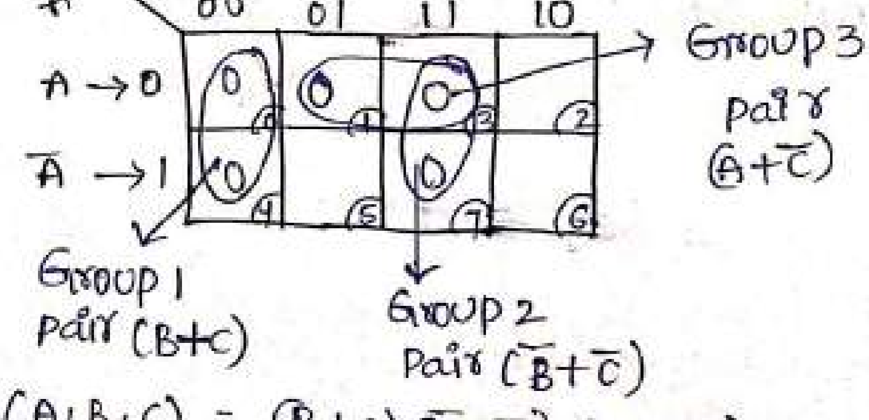
(or)

$Y(A,B,C) = \Pi M(0,1,3,4,7)$

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

* The Boolean function contains 3 variables

* 3 variable K-map contains 8 cells



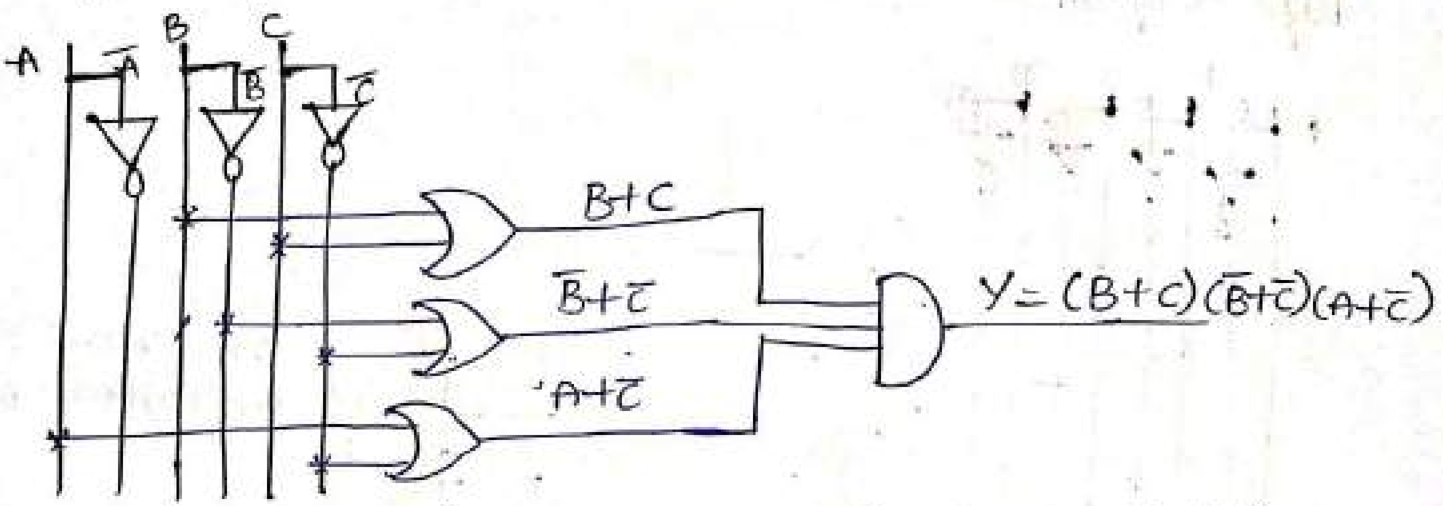
Group 3
pair
$(A+\overline{C})$

Group 1
pair $(B+C)$

Group 2
Pair $(\overline{B}+\overline{C})$

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

logic diagram :-



4) Reduce the following function using K-map technique

$f(A,B,C,D) = \Pi M(0,2,3,8,9,12,13,15)$

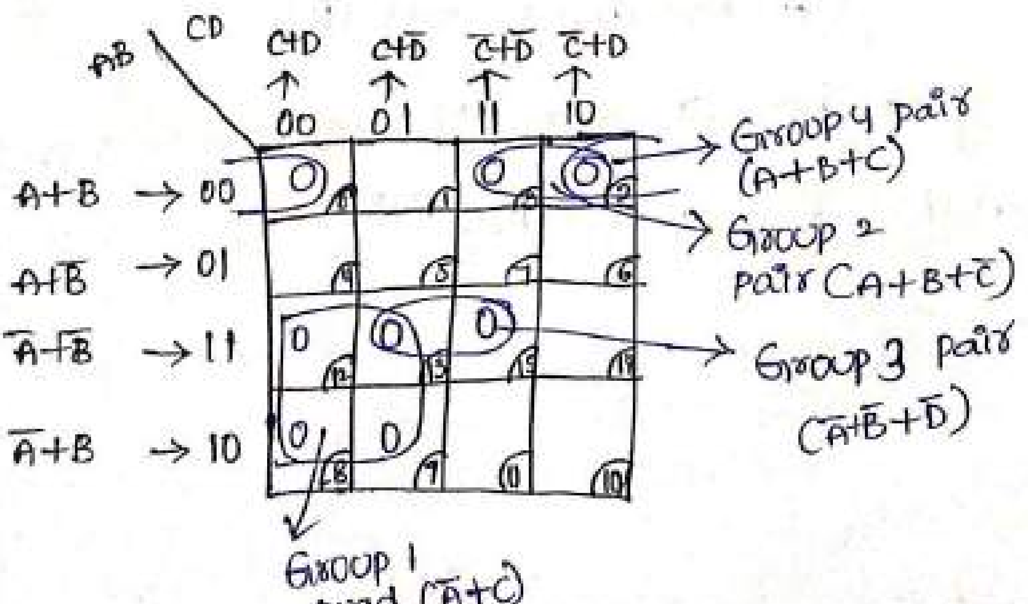
The Given function contains Max-terms and it is in POS Form

$f(A,B,C,D) = \Pi M(M_0 \times M_2 \times M_3 \times M_8 \times M_9 \times M_{12} \times M_{13} \times M_{15})$

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

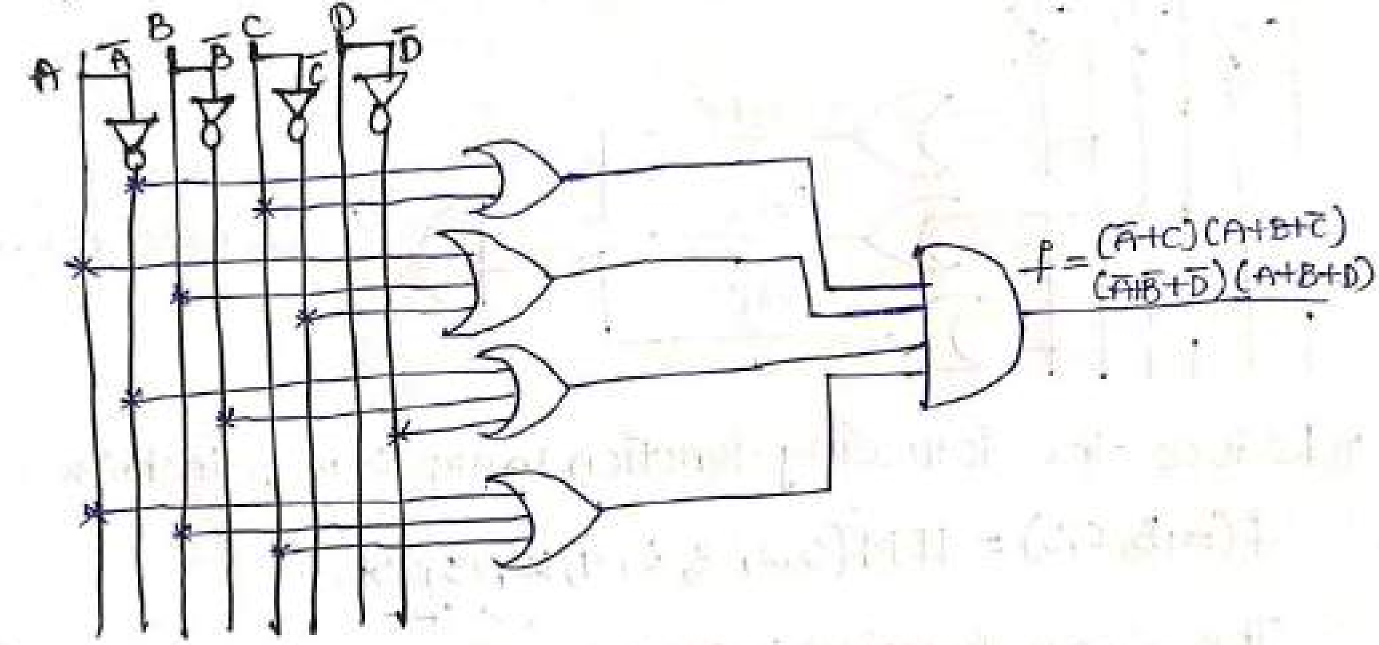
* The Given Boolean function contains 4 variables

* 4-variable K-map contains 16 cells.



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

logic diagram:-


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

5) Reduce the Boolean function in pos using k-map technique

$f(A,B,C,D) = \Pi M(4,6,10,12,13,15)$

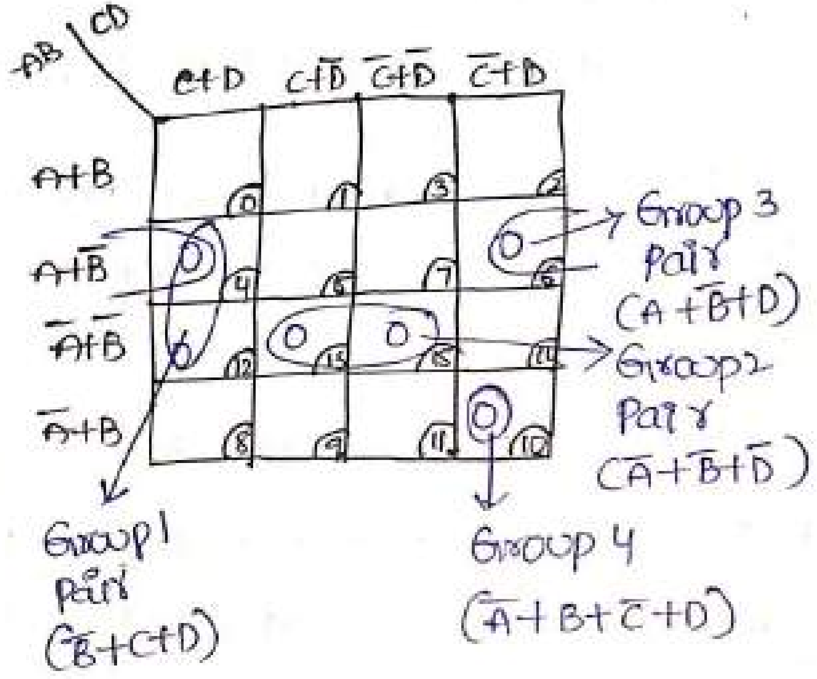
⇒ The Given function is in pos form contains Maxterms

$f(A,B,C,D) = M_4 \times M_6 \times M_{10} \times M_{12} \times M_{13} \times M_{15}$

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

⇒ The given boolean function contains 4-variables

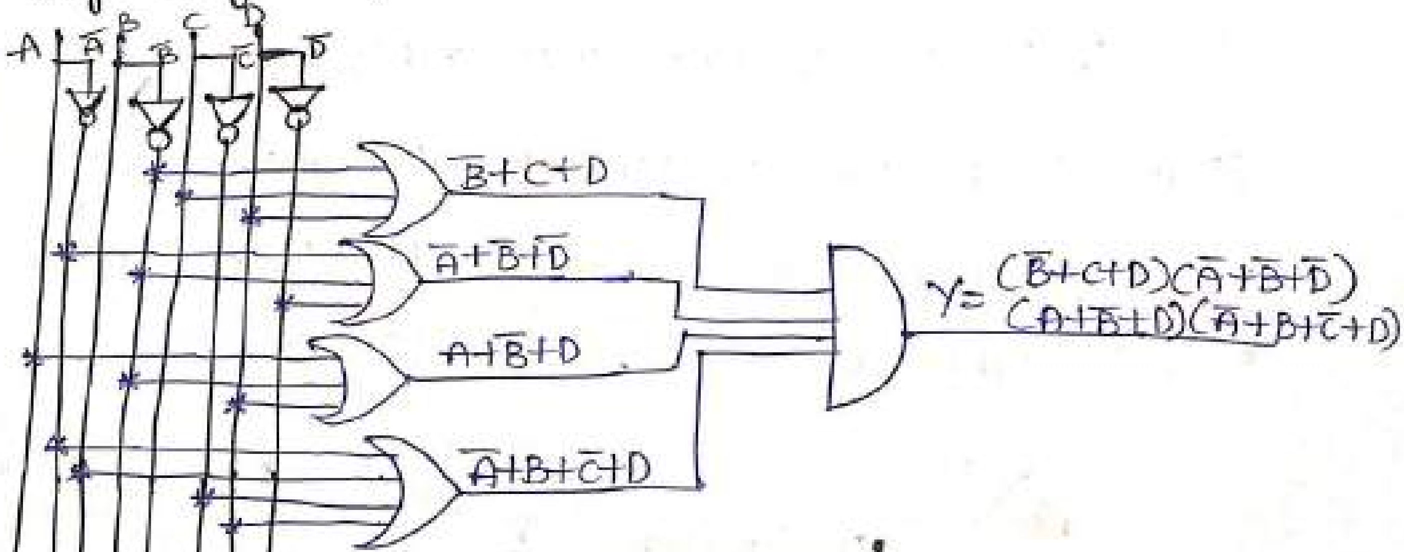
⇒ 4-variable k-map contains 16-cells



AB \ CD	C+D	C+D̄	C̄+D̄	C̄+D
A+B	0	1	3	2
A+B̄	0 4	5	7	0 6
Ā+B̄	0 12	0 13	0 15	14
Ā+B	8	9	11	0 10

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

Logic diagaram :-



⟹ Don't care map entries :

- Some logic circuits can be designed so that there are certain input conditions for which there are no specified output levels, usually because these input conditions will never occur
- So, a circuit designer is free to make the output for any "don't care" condition either a 0 or 1 in order to produce the simplest output expression

Eg:-

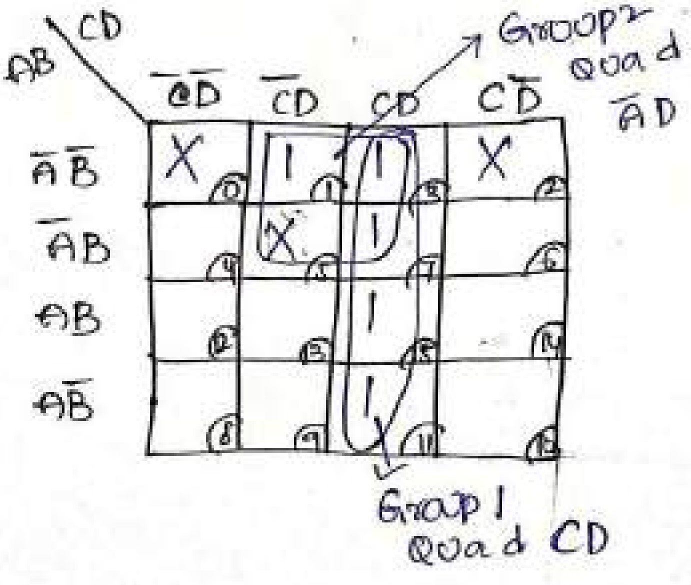
1) In terms of SOP and Don't care conditions

$f(A,B,C,D) = \Sigma m(1,3,7,11,15) + d(0,2,5)$

Sol:- $f(A,B,C,D) = \Sigma m(1,3,7,11,15) + d(0,2,5)$

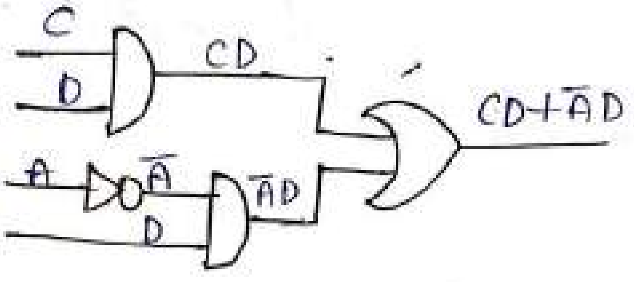
$f(A,B,C,D) = m_1 + m_3 + m_7 + m_{11} + m_{15} + d(0,2,5)$

- The given boolean function contains 4 variables
- 4-variable K-map contains 16 cells



Here By using don't care entry '5' the expression is Simplified (or) variable is reduced

logic diagram:-



2) In terms of pos and don't care condition

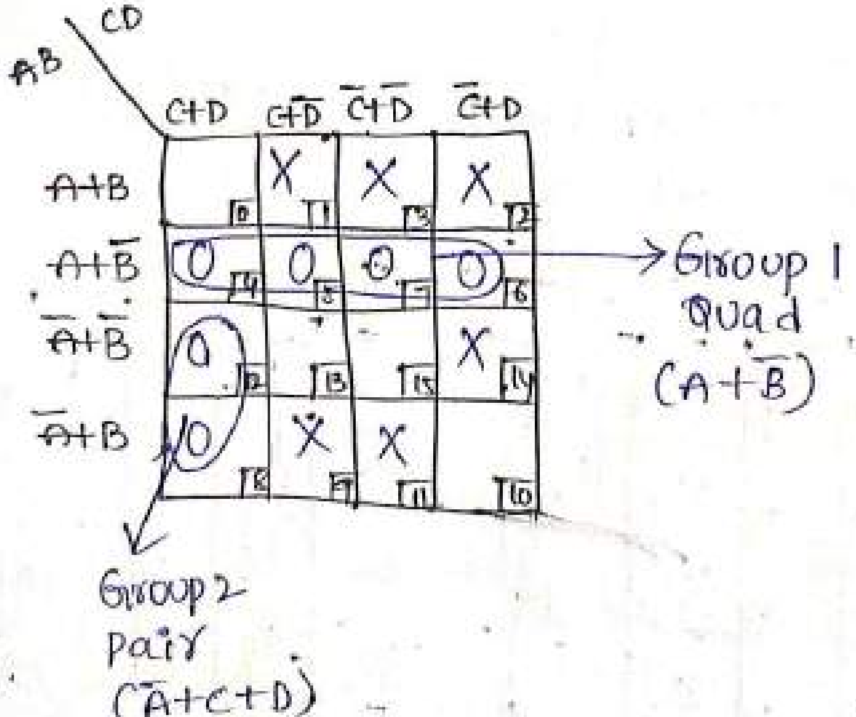
$f(A,B,C,D) = \pi M(4,5,6,7,8,12)d(1,2,3,9,11,14)$

$f(A,B,C,D) = \pi M(4,5,6,7,8,12)\ d(1,2,3,9,11,14)$

$f(A,B,C,D) = (M_4 \times M_5 \times M_6 \times M_7 \times M_8 \times M_{12}) \cdot d(1,2,3,9,11,14)$

* The given variable contains 4 Variables

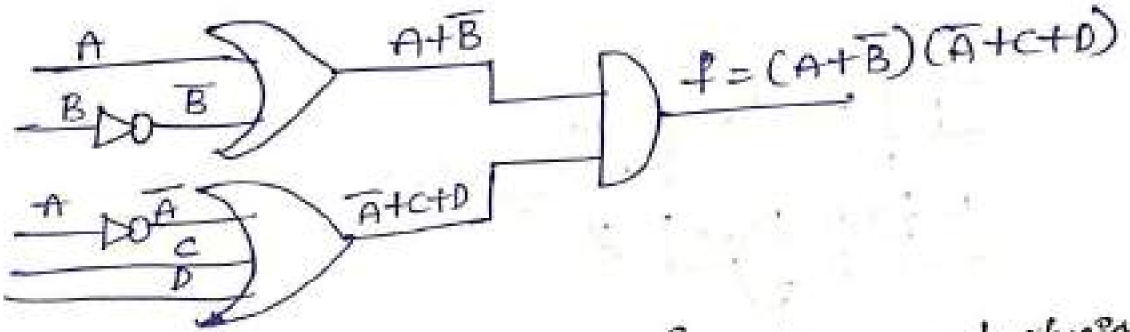
* 4 Variable K-map contains 16 cells



AB \ CD	$C+D$	$C+\bar{D}$	$\bar{C}+\bar{D}$	$\bar{C}+D$
$A+B$	$\boxed{0}$	X $\boxed{1}$	X $\boxed{3}$	X $\boxed{2}$
$A+\bar{B}$	0 $\boxed{4}$	0 $\boxed{5}$	0 $\boxed{7}$	0 $\boxed{6}$
$\bar{A}+\bar{B}$	0 $\boxed{12}$	$\boxed{13}$	$\boxed{15}$	X $\boxed{14}$
$\bar{A}+B$	0 $\boxed{8}$	X $\boxed{9}$	X $\boxed{11}$	$\boxed{10}$

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

logic diagram:-



3) Implement the Following function using K-map technique

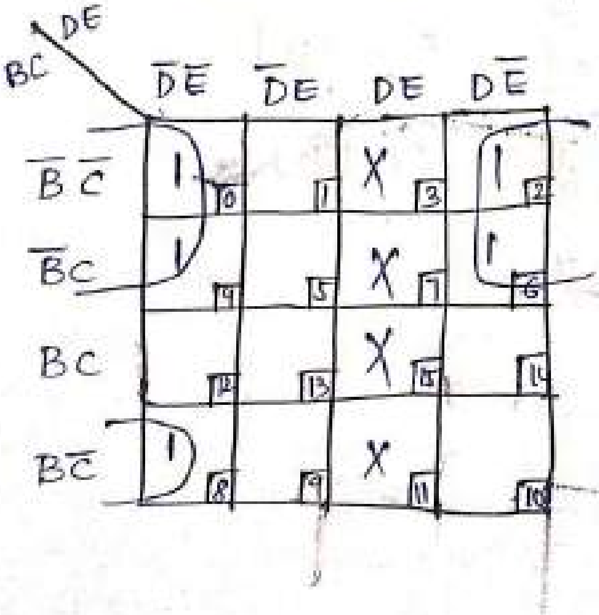
$f(A,B,C,D,E) = \sum m(0,2,4,6,8,16,18,20,22,24,26,28,30) + d(3,7,11,15,19,23,27,31)$

$f(A,B,C,D,E) = \sum m(0,2,4,6,6,16,18,20,22,24,26,28,30) + d(3,7,11,15,19,23,27,31)$

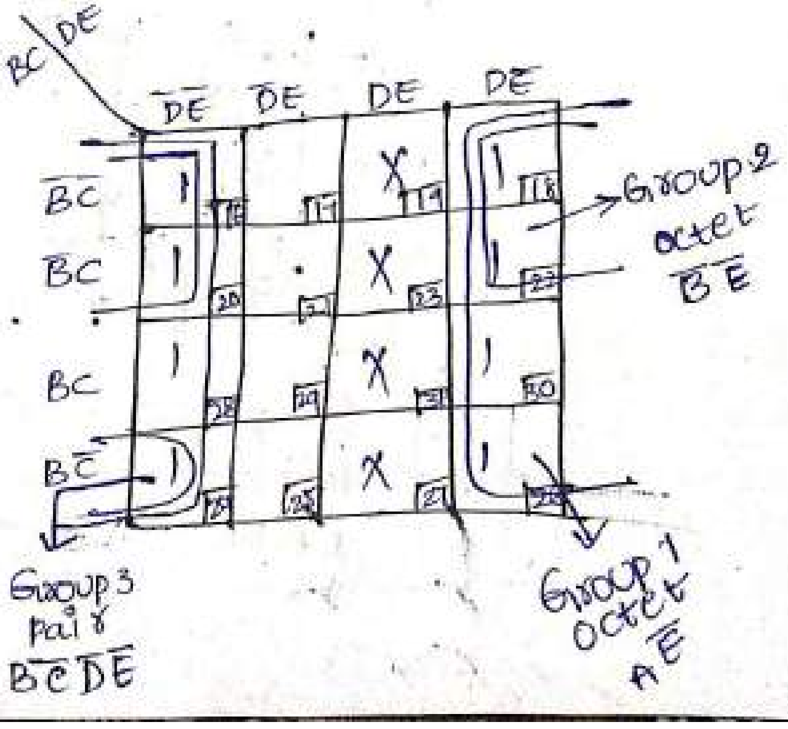
$f(A,B,C,D,E) = (m_0+m_2+m_4+m_6+m_8+m_{16}+m_{18}+m_{20}+m_{22}+m_{24}+m_{26}+m_{28}+m_{30}) + d(3,7,11,15,19,23,27,31)$

* It contains 5-variables
* 5 variable K-map contains 32 cells

for A=0 ($\bar{A}$)



BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	1 0	1	X 3	1 2
$\bar{B}C$	1 4	5	X 7	1 6
BC	12	13	X 15	14
$B\bar{C}$	1 8	9	X 11	10



BC \ DE	$\bar{D}\bar{E}$	$\bar{D}E$	DE	$D\bar{E}$
$\bar{B}\bar{C}$	1 16	17	X 19	1 18
$\bar{B}C$	1 20	21	X 23	1 22
BC	1 28	29	X 31	1 30
$B\bar{C}$	1 24	25	X 27	1 26

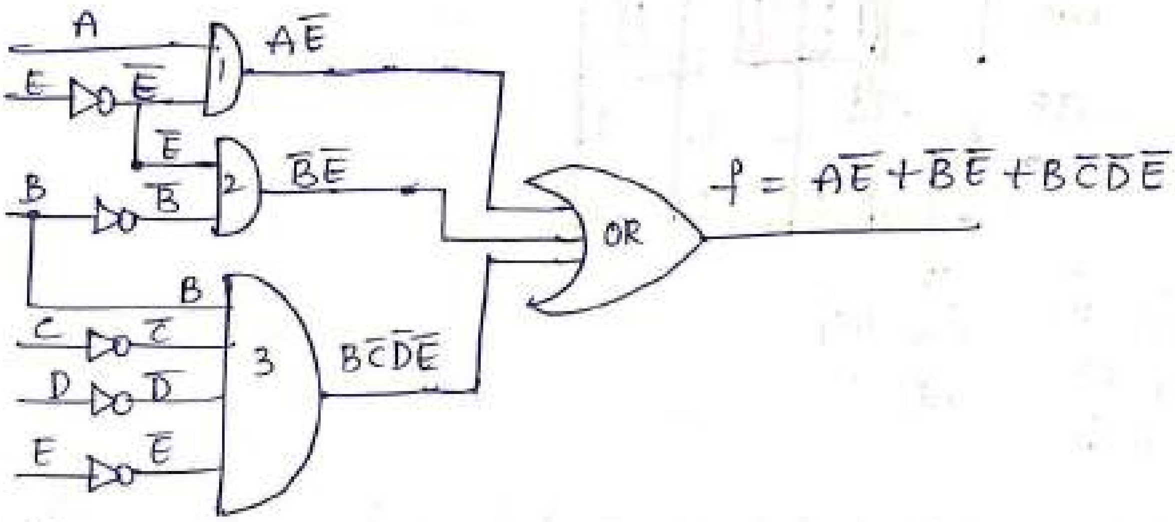
Group 2 octet $\bar{B}\bar{E}$

Group 3 pair $B\bar{C}\bar{D}\bar{E}$

Group 1 octet $A\bar{E}$

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

logic diagram :-



4) Using K-map obtain minimal sop's and minimal pos forms of the function

Sop:- $f(A,B,C,D) = \Sigma m(1,2,3,5,6,7,8,13)$

* The given boolean function is in Sop Form and it contains minterms

* $f(A,B,C,D) = m_1 + m_2 + m_3 + m_5 + m_6 + m_7 + m_8 + m_{13}$

* the given boolean function contains 4-variables

* 4 variable K-map Contains 16 cells

5) Using K-map Determine the minimal Sum of product expression and Realize the Simplified expression using Only Nand Gates.

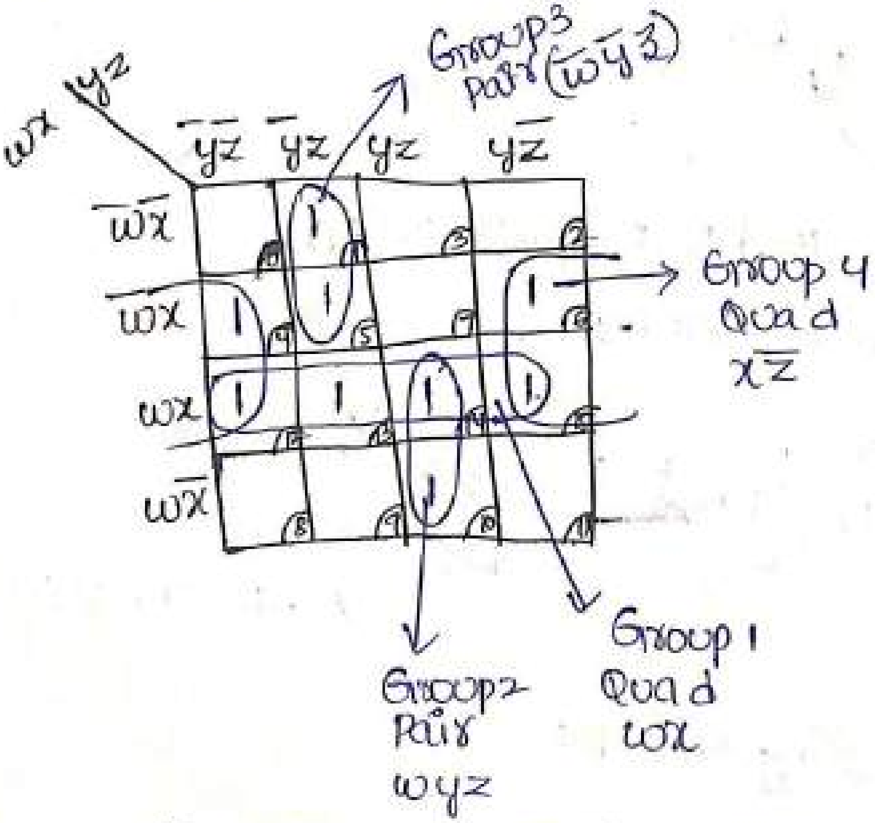
$f(w,x,y,z) = \Pi M(0,2,3,7,8,9,10)$

Given, $f(w,x,y,z) = \Pi M(0,2,3,7,8,9,10)$

In Sop Form:

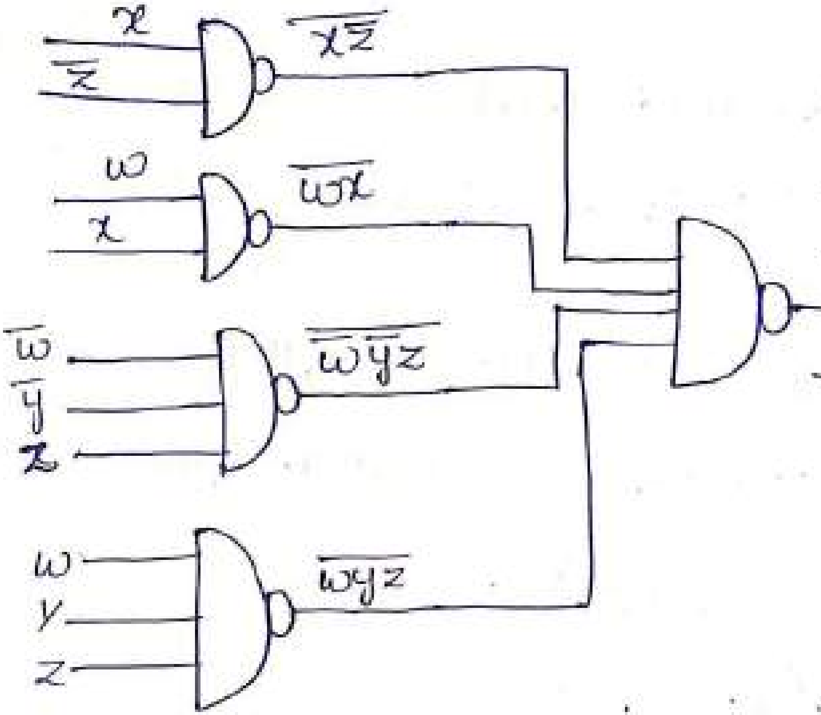
$f(w,x,y,z) = \Sigma m(1,4,5,6,11,12,13,14,15)$

$f(w,x,y,z) = m_1 + m_4 + m_5 + m_6 + m_{11} + m_{12} + m_{13} + m_{14} + m_{15}$



$\therefore f(w,x,y,z) = x\bar{z} + wx + \bar{w}\bar{y}z + wyz$

Logic diagram:-



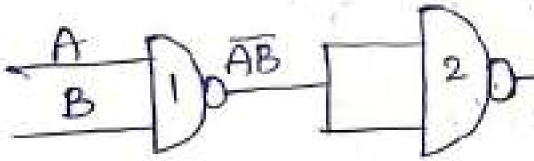
$f = \overline{\overline{x\bar z}\cdot\overline{wx}\cdot\overline{\bar w\bar y z}\cdot\overline{wyz}}$

$f = \overline{\overline{x\bar z}} + \overline{\overline{wx}} + \overline{\overline{\bar w\bar y z}} + \overline{\overline{wyz}}$

$f = x\bar z + wx + \bar w\bar y z + wyz$

* NAND and NOR implemitation.-

AND using NAND :-

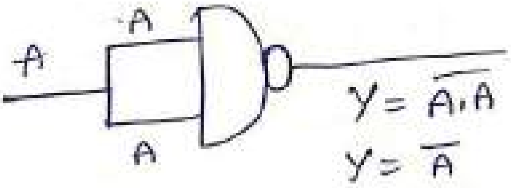


$Y = \overline{\overline{AB}\cdot\overline{AB}}$

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

$Y = AB$ (AND Gate)

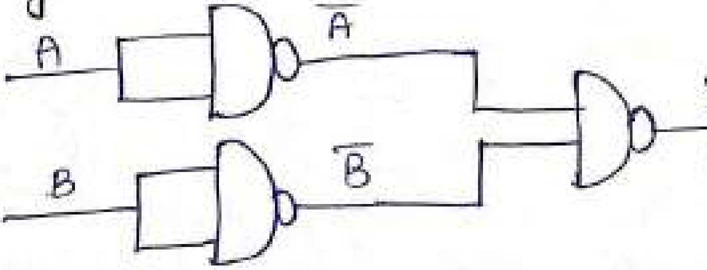
NOT using NAND:-



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

$Y = \bar A$ (NOT Gate)

OR using NAND :-

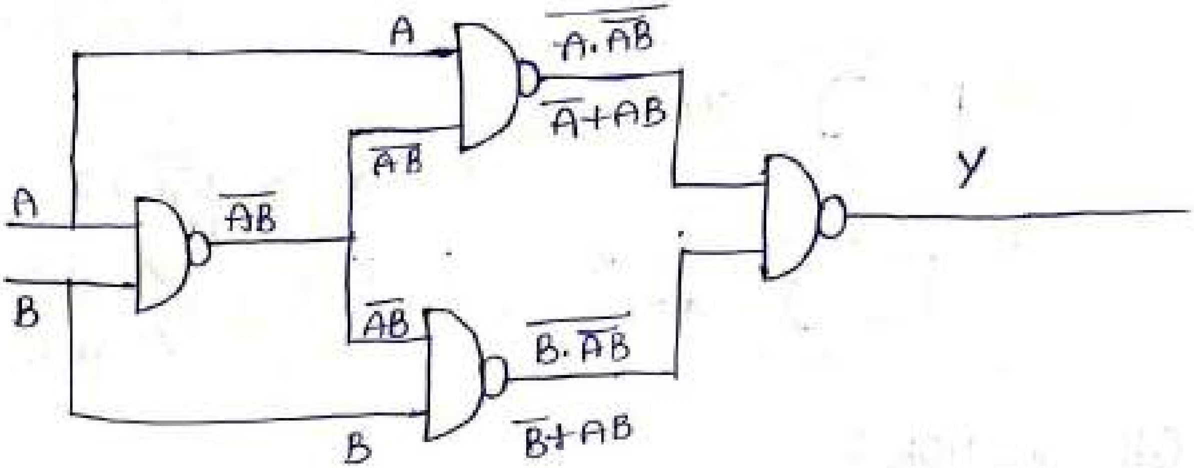


$Y = \overline{\bar A\cdot\bar B}$

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

$Y = A + B$. (OR Gate)

EX-OR using NAND :-



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

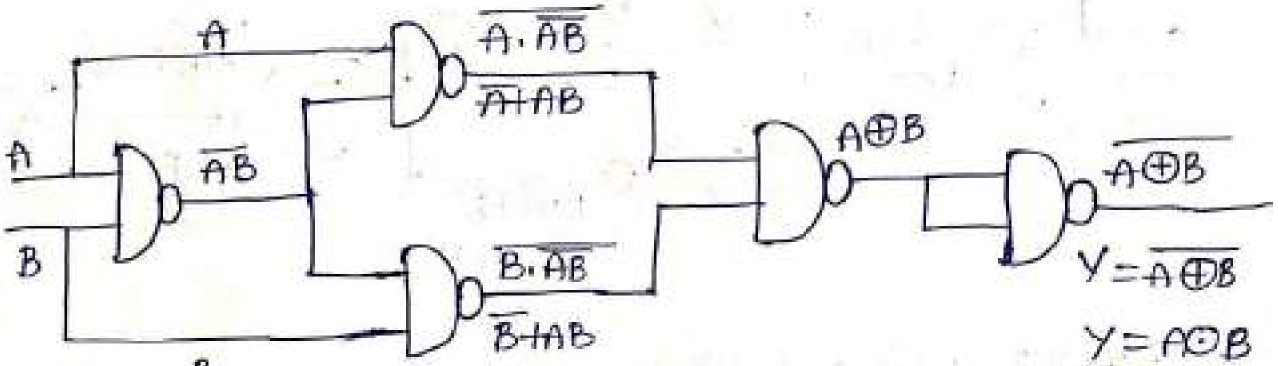
$$Y = \overline{\overline{A}}\cdot\overline{AB} + \overline{\overline{B}}\cdot\overline{AB}.$$

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

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

$$Y = A \oplus B$$

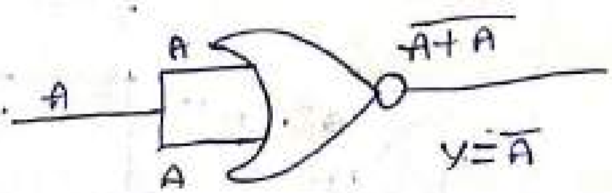
* EX-NOR using NAND :-



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

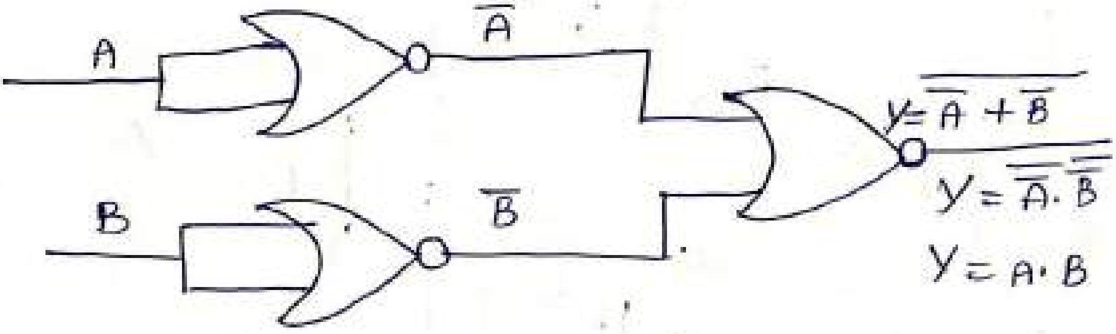
$Y = A \odot B$

NOT using NOR :-

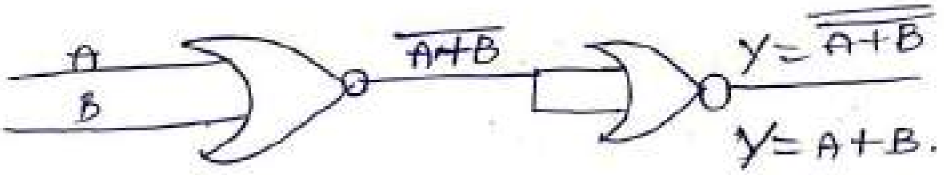


$Y = \overline{A}$

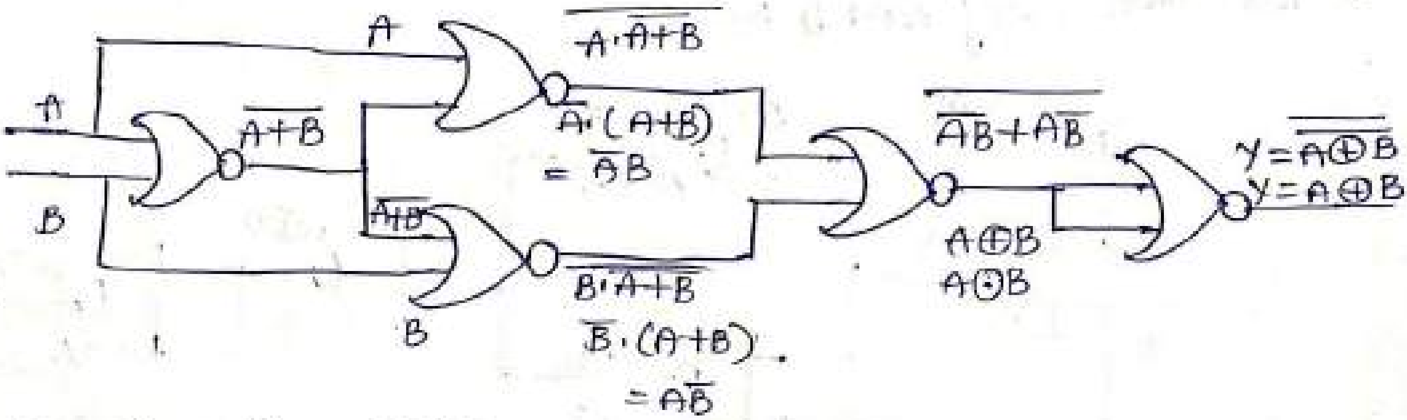
AND using NOR :-


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

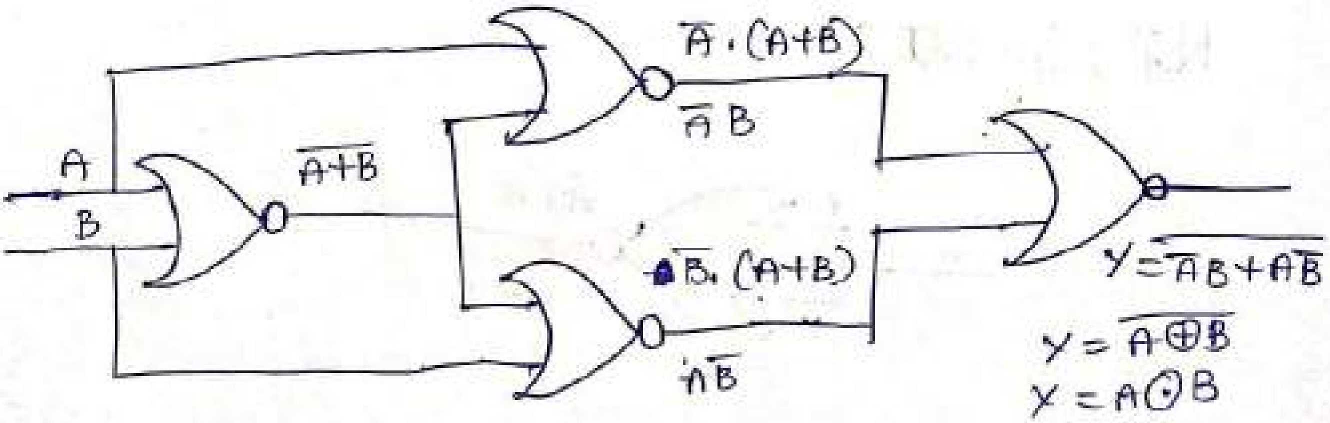
OR using NOR :-



Ex-OR using NOR :-


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

EXNOR using NOR :-


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

* If any function is given how to Realize by using NAND and NOR Implementation follows bellow 3 steps:

Step1 :- We need to complement whatever the expression is given

Step2 :- Apply de-morgan's theorem

Step3 :- Double complement to get final expression

Problems :-

1) Realize the given function by using NAND Gate

$Y = A+BC$

1) Apply complement on both sides

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

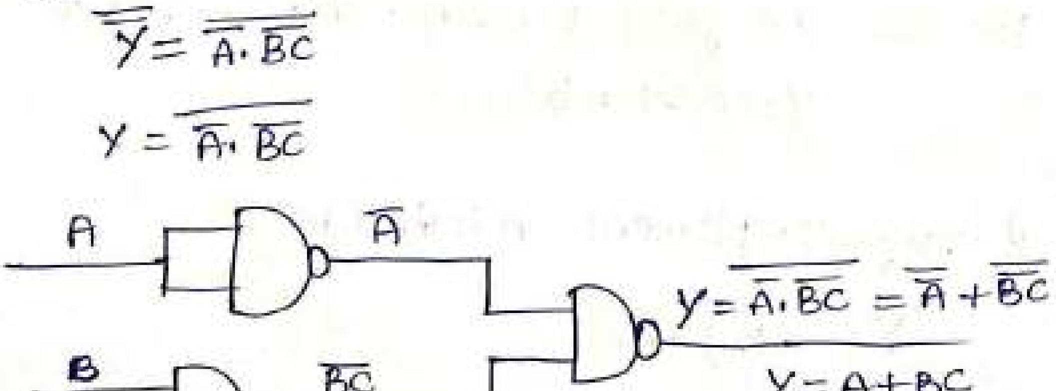
2) Apply de-morgan's theorem

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

3) Apply Double complement on both sides

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

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


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

∴ $\boxed{Y = A+BC}$

2) Realize the given function using NAND gates

$$Y = AB + CD$$

1) Apply complement on both sides

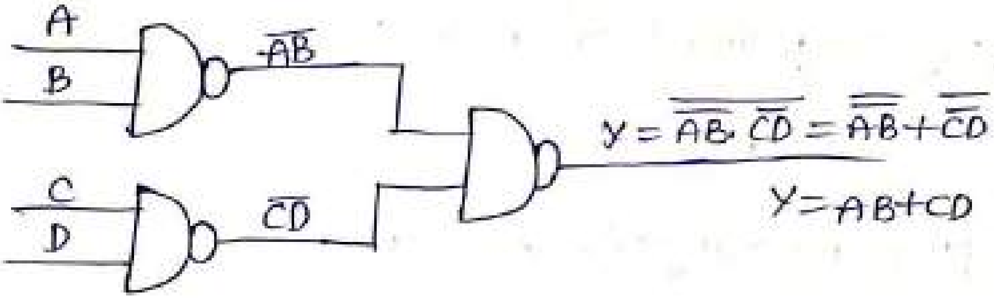
$$\overline{Y} = \overline{AB + CD}$$

2) Apply De-morgan's theorem

$$\overline{Y} = \overline{AB} \cdot \overline{CD}$$

3) Again Apply complement on both sides

$$\overline{\overline{Y}} = \overline{\overline{AB} \cdot \overline{CD}}$$


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

$$\therefore \boxed{Y = AB + CD}$$

3) Realize the given function using NAND gates

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

1) Apply complement on both sides

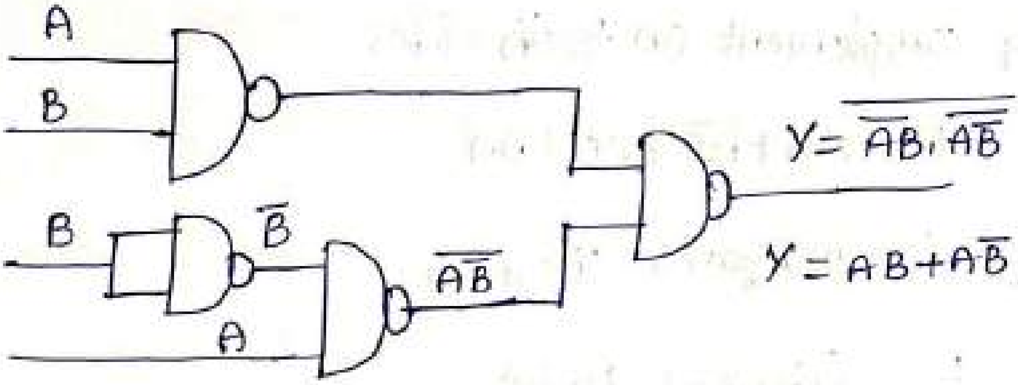
$$\overline{Y} = \overline{AB + A\overline{B}}$$

2) Apply de-morgan's theorem

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

3) Again apply complement on both sides

$\overline{\overline{Y}}, \overline{\overline{AB} \cdot \overline{A\bar{B}}}$



4) Realize the given function using NAND Gates

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

1) Apply complement on both sides

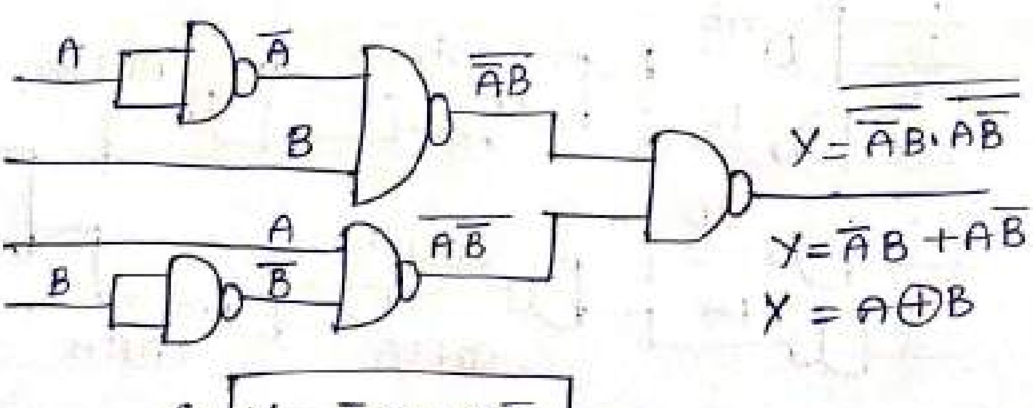
$\overline{Y} = \overline{\bar{A}B + A\bar{B}}$

2) Apply de-morgan's theorem

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

3) Again apply complement on both sides

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


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

∴ $\boxed{Y = \bar{A}B + A\bar{B}}$

5) Find the minimum number of 2 input NAND gate required to implement the Boolean function

$$F = AB + BC + CD + DA$$

1) Apply complement on Both sides

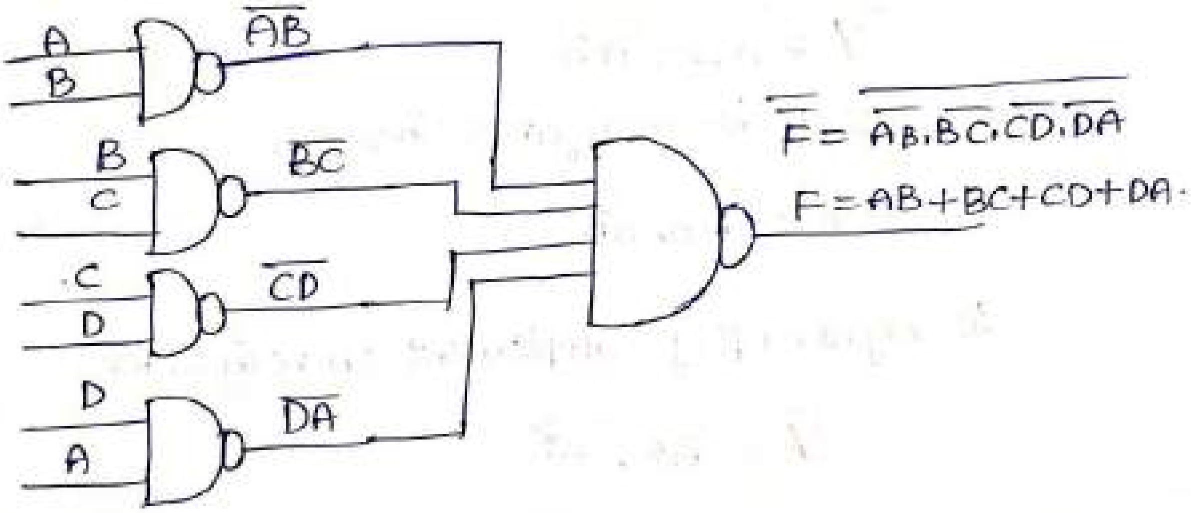
$$\overline{F} = \overline{AB+BC+CD+DA}$$

2) Apply de-morgan's theorem

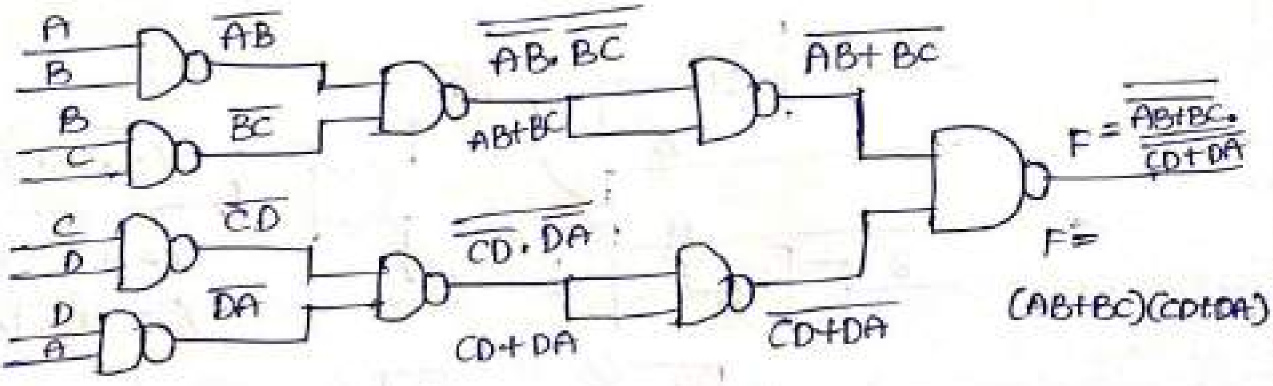
$$\overline{F} = \overline{AB}\cdot\overline{BC}\cdot\overline{CD}\cdot\overline{DA}$$

3) Again apply complement on both sides

$$\overline{\overline{F}} = \overline{\overline{AB}\cdot\overline{BC}\cdot\overline{CD}\cdot\overline{DA}}$$


A
B
$\overline{AB}$
B
C
$\overline{BC}$
C
D
$\overline{CD}$
D
A
$\overline{DA}$
$\overline{\overline{F}} = \overline{\overline{AB}\cdot\overline{BC}\cdot\overline{CD}\cdot\overline{DA}}$
$F = AB+BC+CD+DA$.

⇒


A
B
$\overline{AB}$
B
C
$\overline{BC}$
$\overline{\overline{AB}\cdot\overline{BC}}$
$AB+BC$
$\overline{AB+BC}$
C
D
$\overline{CD}$
D
A
$\overline{DA}$
$\overline{\overline{CD}\cdot\overline{DA}}$
$CD+DA$
$\overline{CD+DA}$
$F = \overline{\overline{AB+BC}\cdot\overline{CD+DA}}$
$F =$
$(AB+BC)(CD+DA)$

6) Realize the given function by using NOR gates

$F(A+B)(B+C)$

1) Apply complement on Both sides

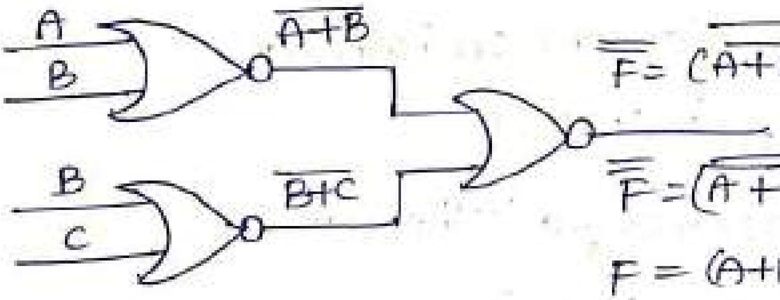
$\overline{F} = \overline{(A+B)(B+C)}$

2) Apply de-morgan's theorem

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

3) Apply again complement on both sides

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



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

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

$F = (A+B)(B+C)$

7) Realize the Given function using NOR gates

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

Step 1:- Apply complement on both sides

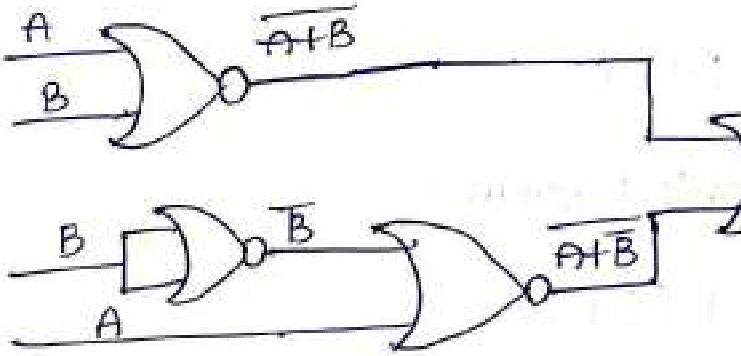
$\overline{F} = \overline{(A+B)(A+\overline{B})}$

2 :- Apply de-morgan's theorem

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

3) Again Apply complement on both sides

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



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

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

8) Realize the Given Function by using NOR gates

$$F = x(w+z)$$

1) Apply complement on both sides

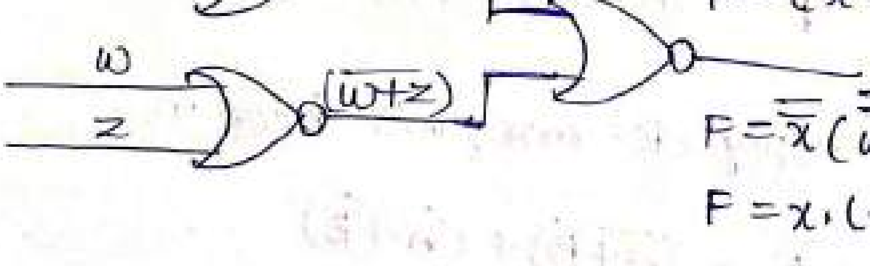
$$\overline{F} = \overline{x(w+z)}$$

2) Apply De-morgan's theorem

$$\overline{F} = \overline{x} + (\overline{w+z})$$

3) Again Apply complement on both sides

$$\overline{\overline{F}} = \overline{\overline{x} + (\overline{w+z})}$$



$\overline{\overline{F}} = \overline{(\overline{x} + (\overline{w+z}))}$

$F = \overline{\overline{x}}(\overline{\overline{w+z}})$

$F = x\cdot(w+z)$

* Other 2 level NAND and NOR implementations

If the Boolean function is in sop & pos form then the function can be implemented in 2 levels

Level 1	Level 2.
AND	OR
OR	AND
NAND	NAND
NOR	NOR

Note:-

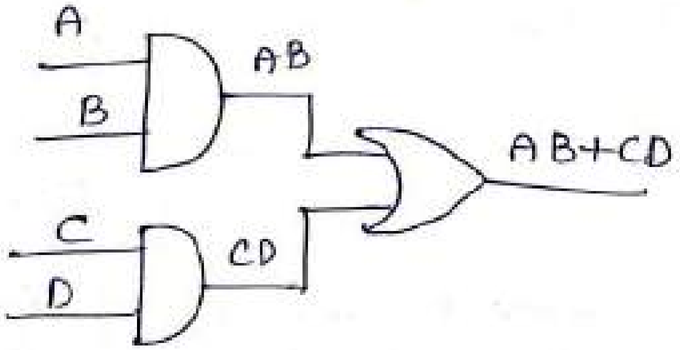
AND - OR ——— NAND - NAND

OR - AND ——— NOR - NOR

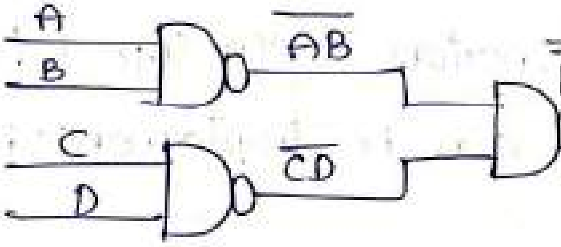
For example:-

1) $F = AB + CD$ (sop form)

⟹



⇒

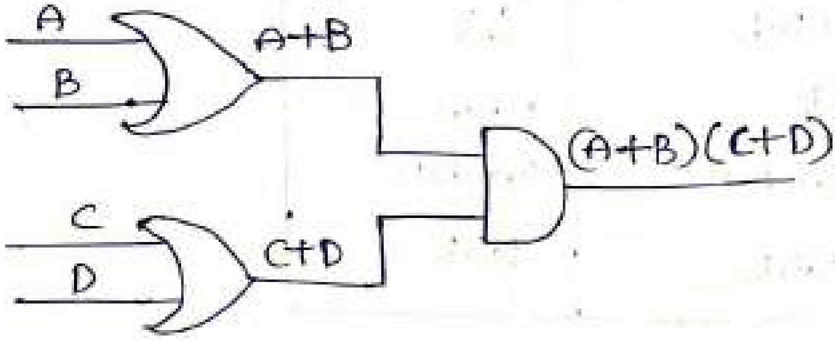


$\overline{\overline{F}} = \overline{\overline{AB}\cdot\overline{CD}} = \overline{\overline{AB}} + \overline{\overline{CD}}$

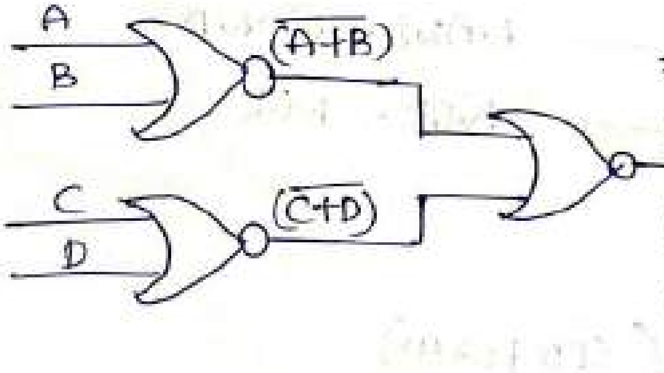
$F = AB+CD$

2) $F = (A+B)(C+D)$ (pos-form)

⇒



⇒



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

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

$F = (A+B)(C+D)$

* Exclusive -OR Function (or) Ex-OR gate :

⇒ An Exclusive OR gate is a logic circuit, which has two (or) more inputs and one output

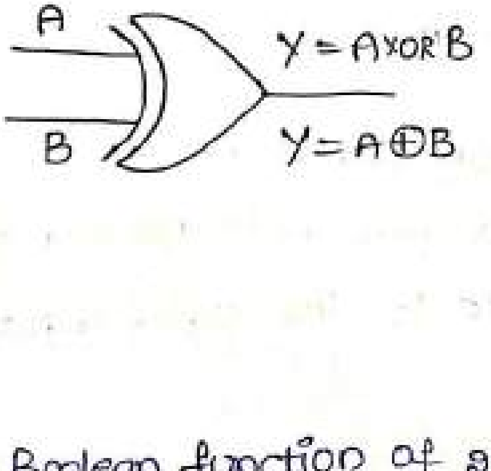
⇒ IC Number of EX-OR gate is 7486

⇒ It acts like as an "odd number of 1's detector in the input". It is also called as "stair case switch"

⇒ It is mostly used in "parity generation and detection".

Logic :- The output is a logic 1 only when there are odd number of 1's at the input

Logic Symbol :-



Truthtable :

Inputs		output
A	B	$Y = A\oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

→ Boolean function of 2-input Ex-OR operation is

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

⇒ From the truth table, we can say when both the inputs are same then output becomes Low (or) Logic '0'.

i.e., if $A=B \Rightarrow Y=0$ and if $A\neq B \Rightarrow Y=1$

* Propesties Of XOR Gates:-

Property ①: $A \oplus A = 0$: output is logic zero when inputs are same

Property ②: $A \oplus \overline{A} = 1$: output is logic 1 when inputs are different

Property ③: $A \oplus 1 = \overline{A}$: EX-OR as Inverter

when one input of EX-OR gate is connected to logic 1 we get the complement of the other input at the output of EX-OR gate

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

Input tied to Logic 1

other input

output is complement from of other input

Property ④: $A \oplus 0 = A$: EX-OR as Non-Inverter

when one input of EX-OR gate is connected to logic 0 we get the uncomplemente of the other input at the output of EX-OR gate

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

input is Grounded

output is uncomplemented form of the other input

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

Property ⑤ : EX-OR as Modulo 2 Adder

The exclusive-OR gate can be used as modulo 2 adder because its truth table is same as the truth table of modulo 2 adder

$$\begin{array}{lcl} 0+0=0 & & 0 \oplus 0 = 0 \\ 0+1=1 & \equiv & 0 \oplus 1 = 1 \\ 1+0=1 & & 1 \oplus 0 = 1 \\ 1+1=0 & & 1 \oplus 1 = 0 \end{array}$$

Property ⑥ : $(AB) \oplus (AC) = A(B \oplus C)$

A	B	C	AB	AC	AB⊕AC	B⊕C	A(B⊕C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1
1	1	0	1	0	1	1	1
1	1	1	1	1	0	0	0

=

Property ⑦ : if $A \oplus B = C$, then

$A \oplus C = B$

$B \oplus C = A$ and

$A \oplus B \oplus C = 0$

A	B	$A \oplus B = C$	$A \oplus C = B$	$B \oplus C = A$	$A \oplus B \oplus C = 0$
0	0	0	0	0	0
0	1	1	1	0	0
1	0	1	0	1	0
1	1	0	1	1	0

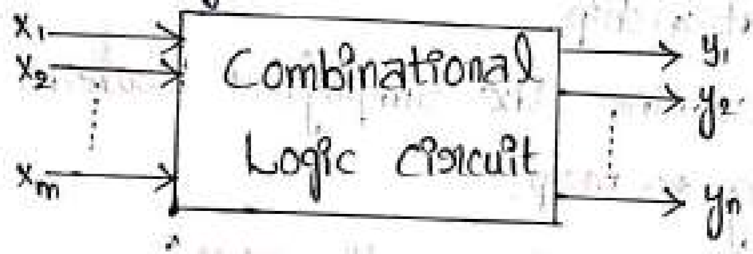
Unit - III

Combinational Logic

Introduction:

* In combinational logic circuits, the output is function of logic inputs at that instant of time.

* The general model of a combinational system can be represented by



-A combinational Logic circuit

* Here we have m inputs, $x_1, x_2 \dots x_m$ and n outputs $y_1, y_2 \cdots y_n$ all assumed to be functions of time.

* Adder, subtractor, comparator, multiplexer, demultiplexer, endocer, decoder are some of the examples of combinational logic circuits.

-Analysis and design procedure of Combinational Logic circuit:

Step 1: Observe the problem defination.

Step 2: Determine the required input and output variables.

Step 3: Assign Letters (or) symbols to the input variables.

Step 4: Make a truth table that define required relationship.

Step 5: Determine the simplified boolean expression using K-map.

Step 6: Draw the logic diagram.

Eg:- Let us consider Arithmetic circuits

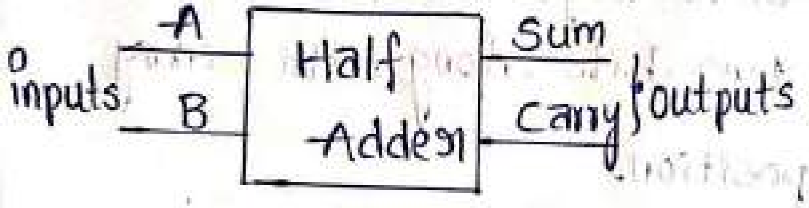
-Arithmetic circuits:

-The logic circuits performing the operations addition, subtraction, multiplication and division are known as 'Arithmetic circuits'.

Half Adder:

-A digital circuit that adds two binary bits is known as 'Half Adder'. The addition of two bits produces two outputs i.e., sum and carry.

Logic Symbol:

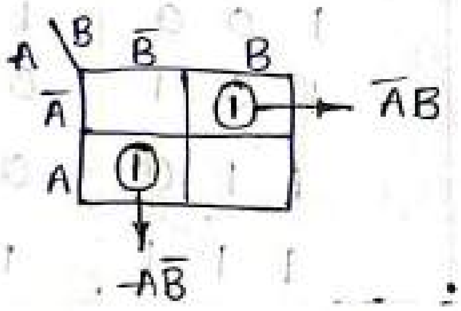


Truth Table of Half adder

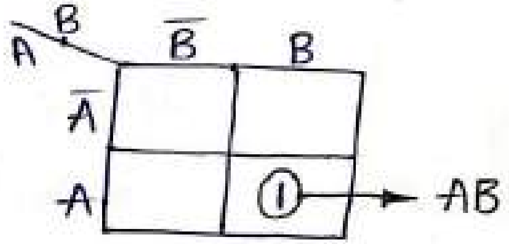
Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map simplification for carry & sum:

sum:



carry:

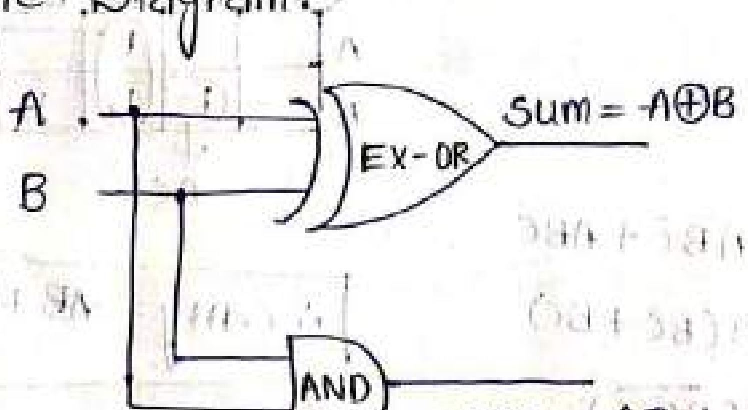


Sum = $\overline{A}B + A\overline{B}$

carry = AB

Sum = $A \oplus B$

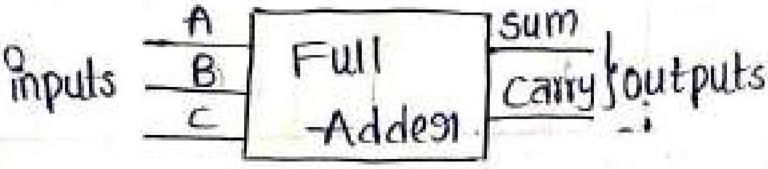
Logic Diagram:



Full - Adder:

- Full - Adder is a combinational logic circuit that adds three binary digits i.e., two bits along with carry from lesser significant position.

Logic Symbol:



Truth Table of Full adder

Inputs			outputs	
A	B	C	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

K-map simplification for sum and carry:

sum:

A \ BC	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	0	1 1	3	1 2
A	1 4	5	1 7	6

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

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

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

$$\boxed{\therefore\ sum = A \oplus B \oplus C}$$

carry:

A \ BC	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$			1	
A	-	1	1	1

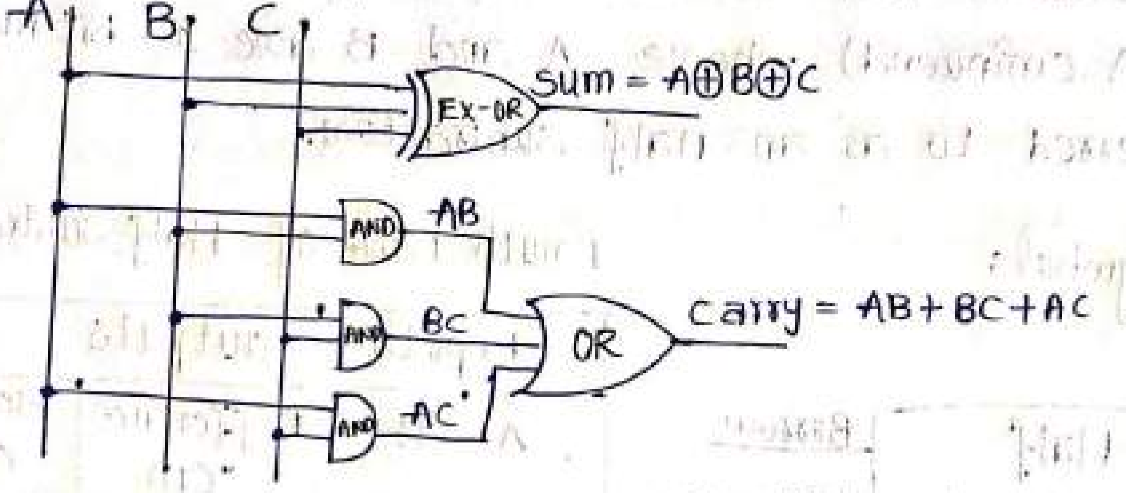
→ BC

→ AB

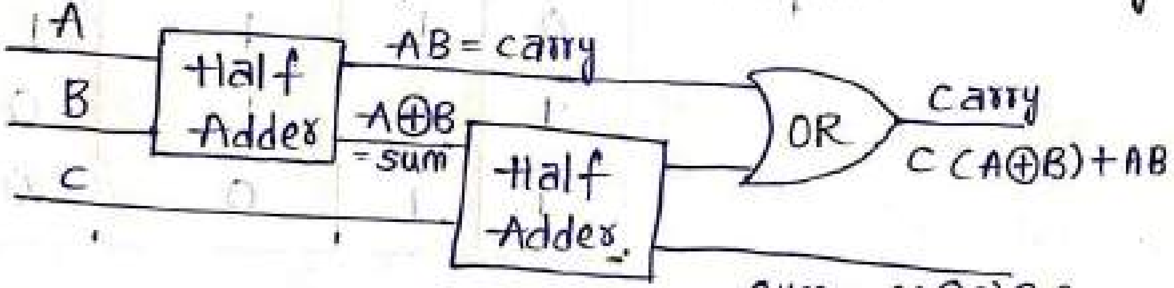
↓ AC

$$\boxed{\therefore\ carry = AB + BC + AC}$$

Logic diagram:



* Full adder using two Half adders and OR gate



Using two half adders & OR gate

$\therefore$ sum $= A \oplus B \oplus C$

carry $= (A\oplus B)C + AB$

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

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

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

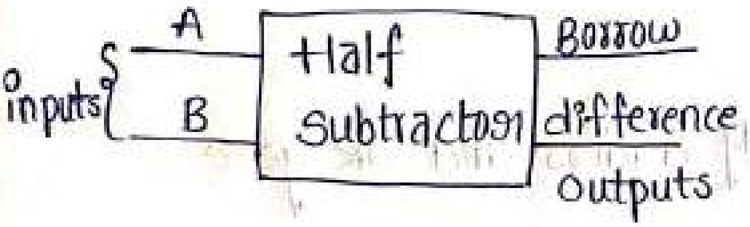
carry $= BC(A+\bar{A}) + AC(B+\bar{B}) + AB(C+\bar{C})$

carry $= AB + BC + AC$ $\quad [\because A+\bar{A} = 1]$

Half Subtractor:

- A logic circuit for the subtraction of B (subtrahend) from A (minuend) where A and B are 1-bit numbers is referred to as an half subtractor.

Logic Symbol:

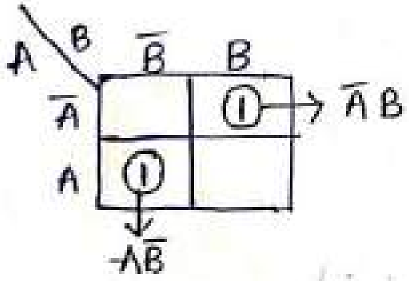


Truth Table of Half subtractor

Inputs		outputs	
A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-map simplification for half-subtractor:

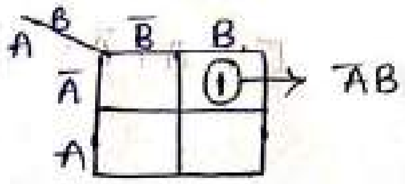
For Difference:



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

$\boxed{D = A \oplus B}$

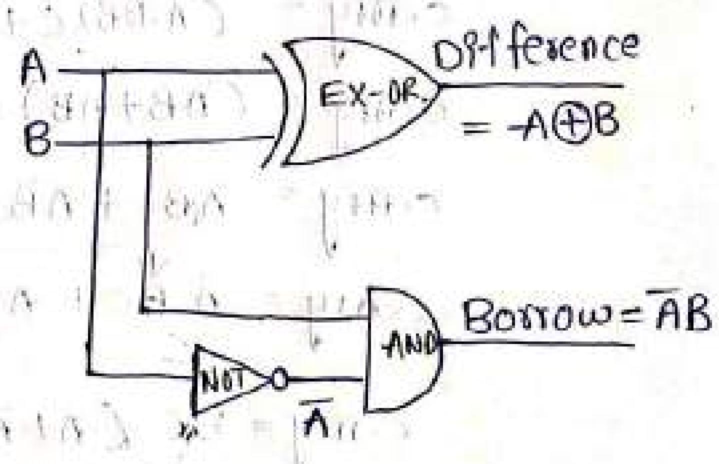
For Borrow:



Borrow = $\bar{A}B$

$\boxed{B = \bar{A}B}$

*Logic Diagram:

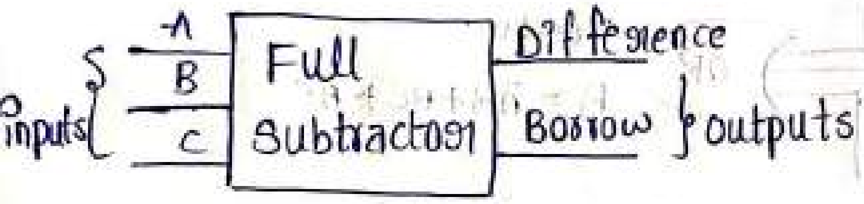


Full - Subtractor:-

* A full-subtractor is a combinational circuit that performs a subtraction between two bits, taking into account borrow of the lower significant stage.

* This circuit has three inputs & two outputs.

Logic Symbol:



Truth Table of a Full-subtractor:

Inputs			outputs	
A	B	C	Difference (D)	Borrow (B)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map simplification of Difference & Borrow:

For Difference:

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1 1	3	1 2
A	1 4	5	1 7	6

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

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

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

$$\boxed{D = A \oplus B \oplus C}$$

For Borrow:-

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$		1	1	1
A			1	

$\bar{A}C$

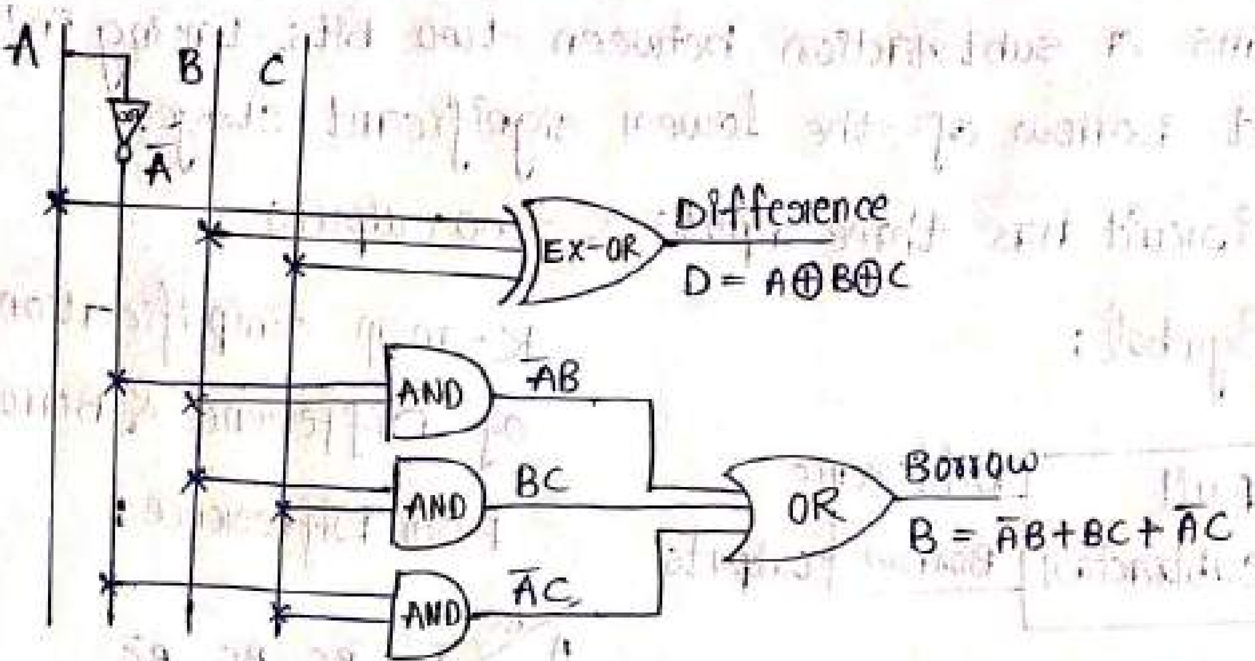
$\bar{A}B$

BC

$$\text{Borrow} = \bar{A}B + BC + \bar{A}C$$

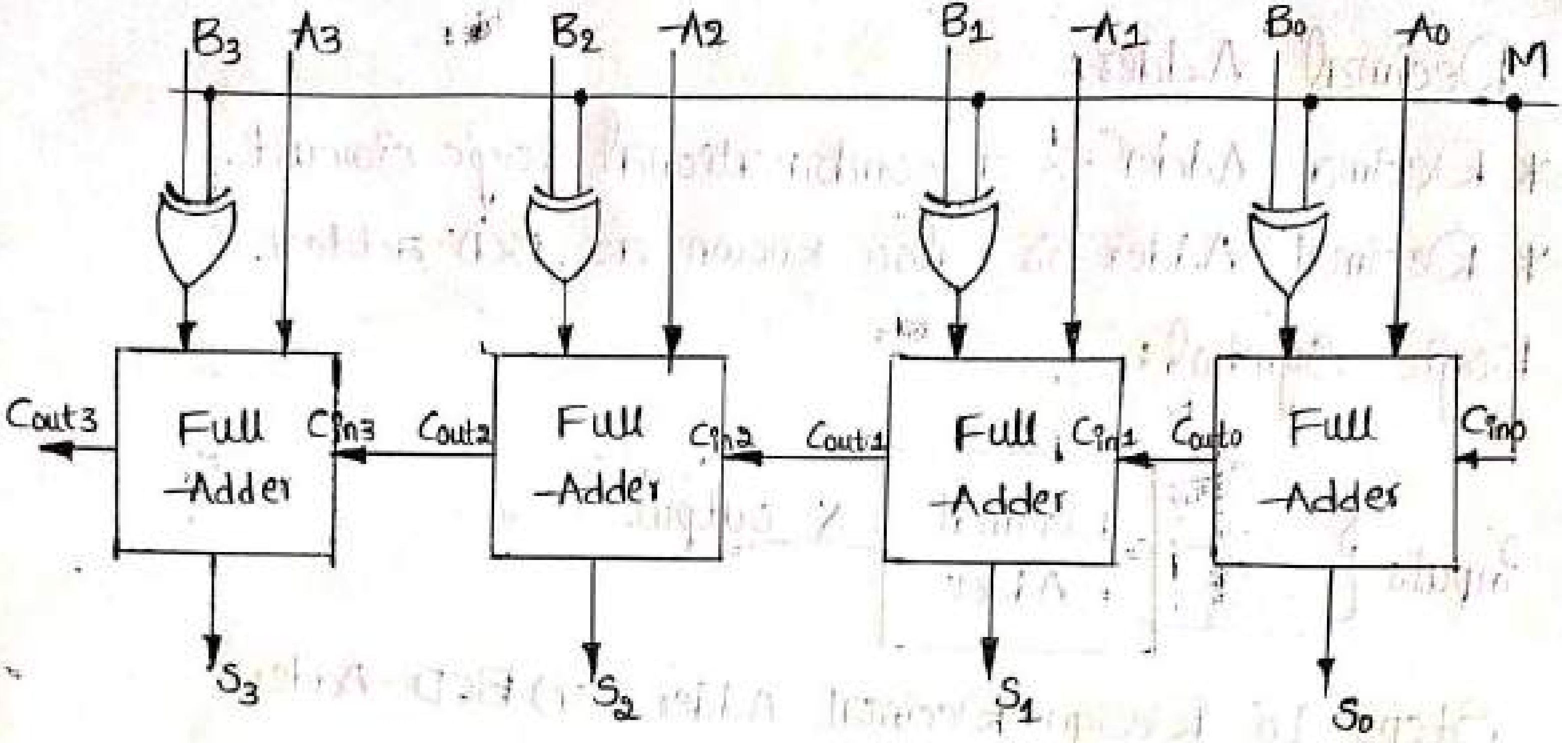
$$\boxed{B = \bar{A}B + BC + \bar{A}C}$$

Logic Diagram of a Full-Subtractor



Binary Adder-Subtractor:

The addition and subtraction operations can be combined into one circuit with one common binary-Adder. This is done by including an exclusive-OR gate with each full-adder, as shown below. The mode input M controls the operation of the circuit. when M=0, the circuit is an adder, and when M=1, the circuit becomes a subtractor. Each exclusive-OR gate receives input, M and one of the inputs of B. when M=0, we have B⊕0=B. The full-adders receive the value of B, the input carry is 0, and the circuit performs A plus B. when M=1, we have B⊕1 = $\bar{B}$ and C_{in} = 1. The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B, i.e., A−B.



4 - bit Adder/Subtractor

Addition:

Let us consider A = 1101, B = 0110

A : 1 1 0 1

B : + 0 1 1 0

1 0 0 1 1

∴ Sum is (10011)

Subtraction:

Let us consider A = 0101 and B = 0011

A : 0 1 0 1

B : - 0 0 1 1

0 0 1 0

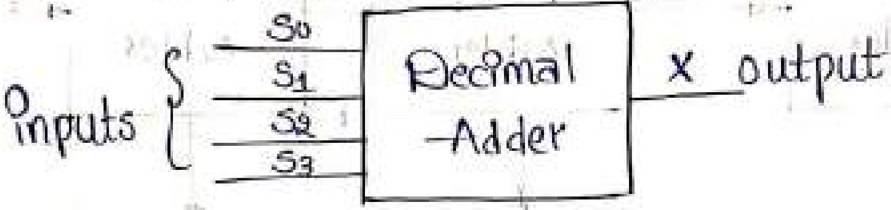
∴ The difference is (0010)

Decimal Adder:

* Decimal Adder is a combinational logic circuit.

* Decimal Adder is also known as BCD adder.

Logic Symbol:



Steps to Design Decimal Adder (or) BCD Adder:

step 1:- Consider 4-bit binary adder for initial addition

step 2:- We need a logic circuit to detect sum greaterthan 9.

step 3:- We need one-more 4-bit binary adder to add six (0110) in the sum, if sum is greaterthan 9 (or) carry is one (1).

Truth Table of a Decimal Adder :

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

Simplified expression using K-map:

S_3S_2 \ S_1S_0	$\overline{S_1}\,\overline{S_0}$	$\overline{S_1}S_0$	S_1S_0	$S_1\overline{S_0}$
$\overline{S_3}\,\overline{S_2}$	0	1	3	2
$\overline{S_3}S_2$	4	5	7	6
S_3S_2	1 12	1 13	1 15	1 14
$S_3\overline{S_2}$	8	9	1 11	1 10

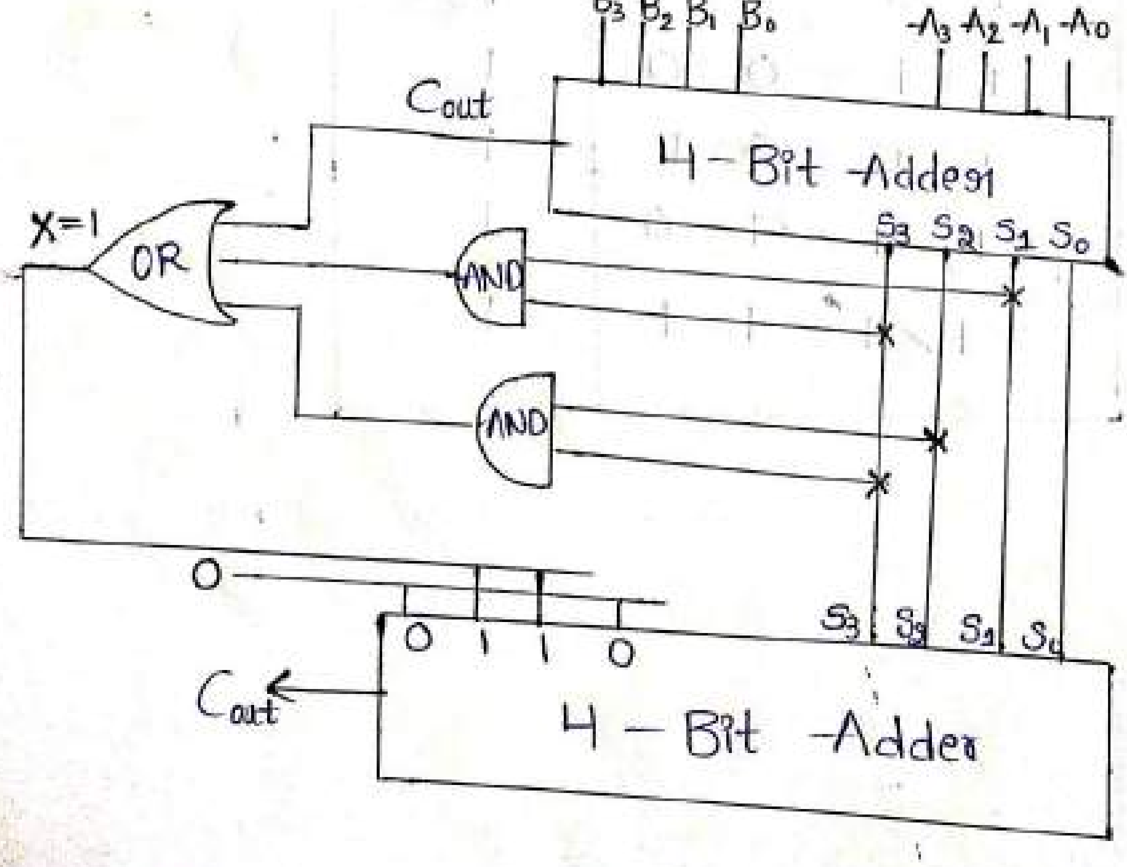
→ Group2 (Quad) $= S_3S_2$

Group1 (Quad) $= S_3S_1$

$$X = S_3S_1 + S_3S_2$$

$$\boxed{X = S_1S_3 + S_2S_3}$$

Logic Diagram of a Decimal Adder:



⟹ If X = 1 the sum is greaterthan '9' then add six (0110) to the sum bits.

⟹ If X = 0 the sum is lessthan (or) equal to '9' then the result is sum bit.

Eg:

$$\begin{array}{rr}
A = 8: & 1\ 0\ 0\ 0 \\
B = 5: & 0\ 1\ 0\ 1 \\
+ & \\
\hline
 & 1\ 1\ 0\ 1 \quad [\because 13 > 9] \\
\hline
 & 1\ 1\ 0\ 1 \\
 & +\ \underset{1}{0}\ 1\ 1\ 0 \\
\hline
 & 0\,0\,0\,1.\ 0\,0\,1\,1 \\
\hline
\end{array}$$

∴ The sum is (0001,0011)

*Binary Multiplier:

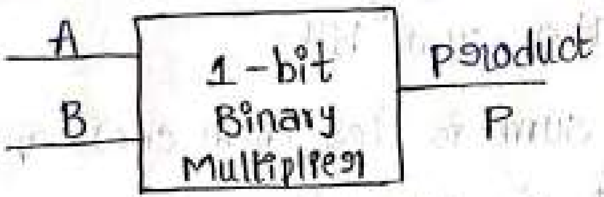
Binary multiplier is a combinational logic circuit which is used to perfoorm multiplication of two binary numbers.

Let us consider 1-bit binary multiplier.

One-bit Binary Multiplier:

It is a combinational logic circuit which performs the multiplication of two one-bit binary numbers.

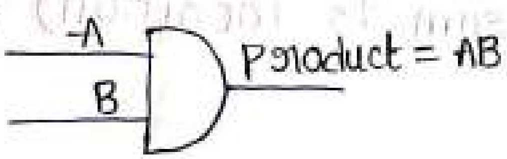
Logic Symbol:



Truth Table:

Inputs		Output
A	B	P
0	0	0
0	1	0
1	0	0
1	1	1

Logic Diagram:

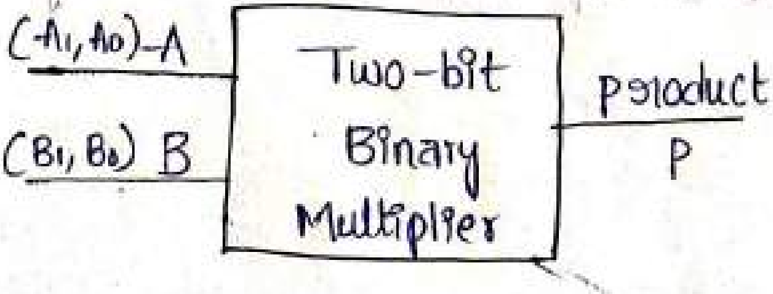


Let us consider two - bit binary multiplier

Two - bit Binary Multiplier:

Two - bit Binary multiplier is a combinational logic circuit which performs multiplication of two 2-bit binary numbers.

Logic Symbol:



$A : A_1 \; A_0$

$B : B_1 \; B_0$

$\times$

$C_1 \quad A_1B_1 \quad A_0B_0$

$C_2 \quad A_1B_1 \quad A_0B_1 \quad \times$

$P_3 \quad P_2 \quad P_1 \quad P_0$

Here, $P_0 = A_0B_0$

$P_1 = A_1B_0 + B_1A_0$

$P_2 = A_1B_1 + C_1$

$P_3 = C_2$

Eg:

$A : 1 \; 0$

$B : 1 \; 1$

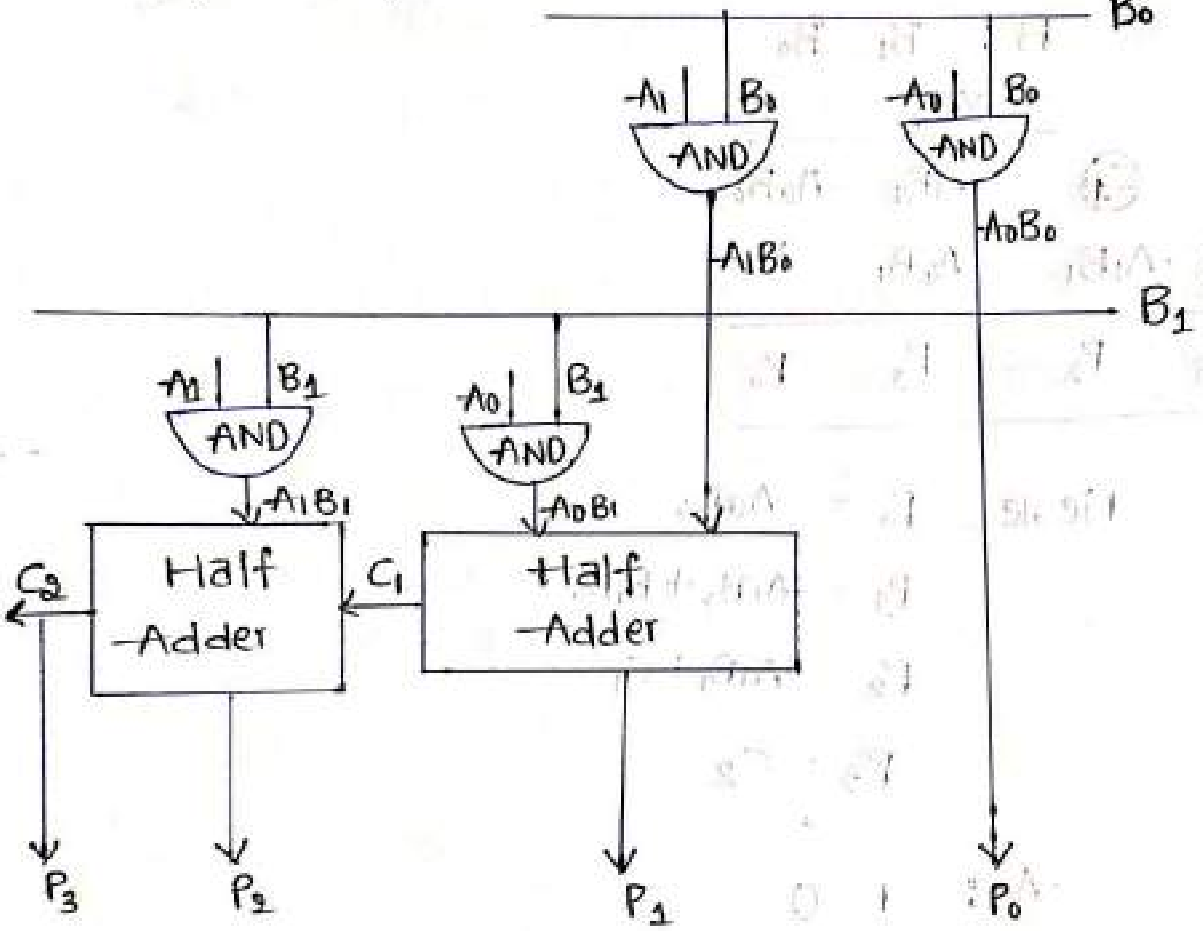
$\times$

$1 \; 0$

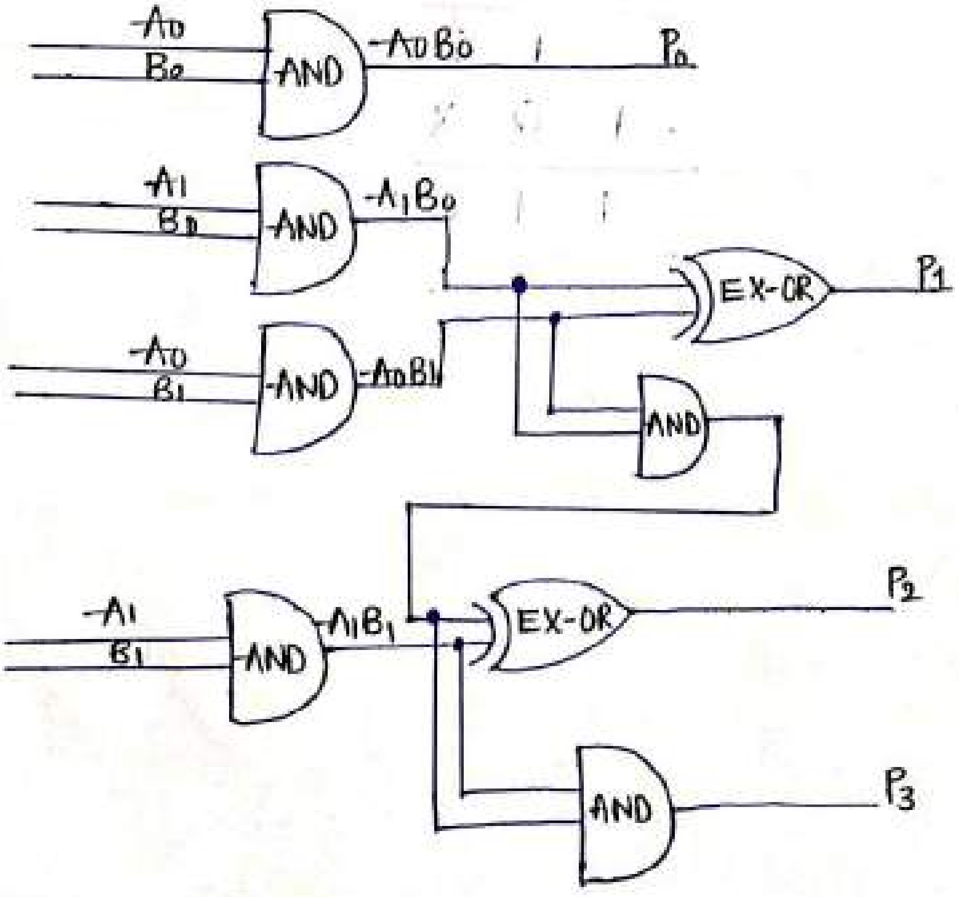
$1 \; 0 \; \times$

$1 \; 1 \; 0$

Logic Diagram:



(or)



Note:

For N-bit multiplier

Number of AND gates $= N^2$

Number of Half Adders required $= N$

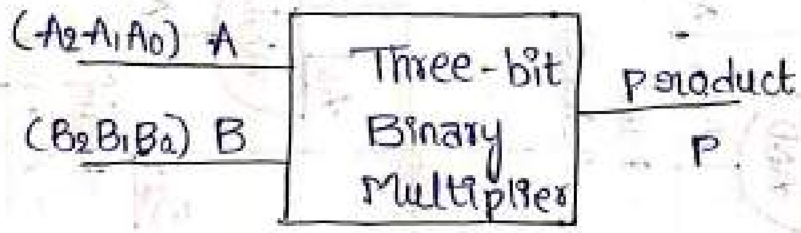
Number of Full Adders required $= N(N-2)$

* Let us consider three-bit binary multiplier

Three-bit Binary Multiplier:-

Three-bit binary multiplier is a combinational logic circuit which performs multiplication of two 3-bit binary numbers

Logic Symbol:



$$\begin{array}{ccccccc}
 & & & A: & A_2 & A_1 & A_0 \\
 & & & B: & B_2 & B_1 & B_0 \\
 & & & \times & & & \\
\hline
 & & & A_2B_0 & A_1B_0 & A_0B_0 & \\
 & & A_2B_1 & A_1B_1 & A_0B_1 & \times & \\
\boxed{C_4} & A_2B_2 & A_1B_2 & A_0B_2 & \times & \times & \\
 & \boxed{C_3} & \boxed{C_2} & \boxed{C_1} & & & \\
\hline
P_5 & P_4 & P_3 & P_2 & P_1 & P_0 & \\
\hline
\end{array}$$

* $P_0 = A_0B_0$

$P_1 = A_1B_0 + A_0B_1$

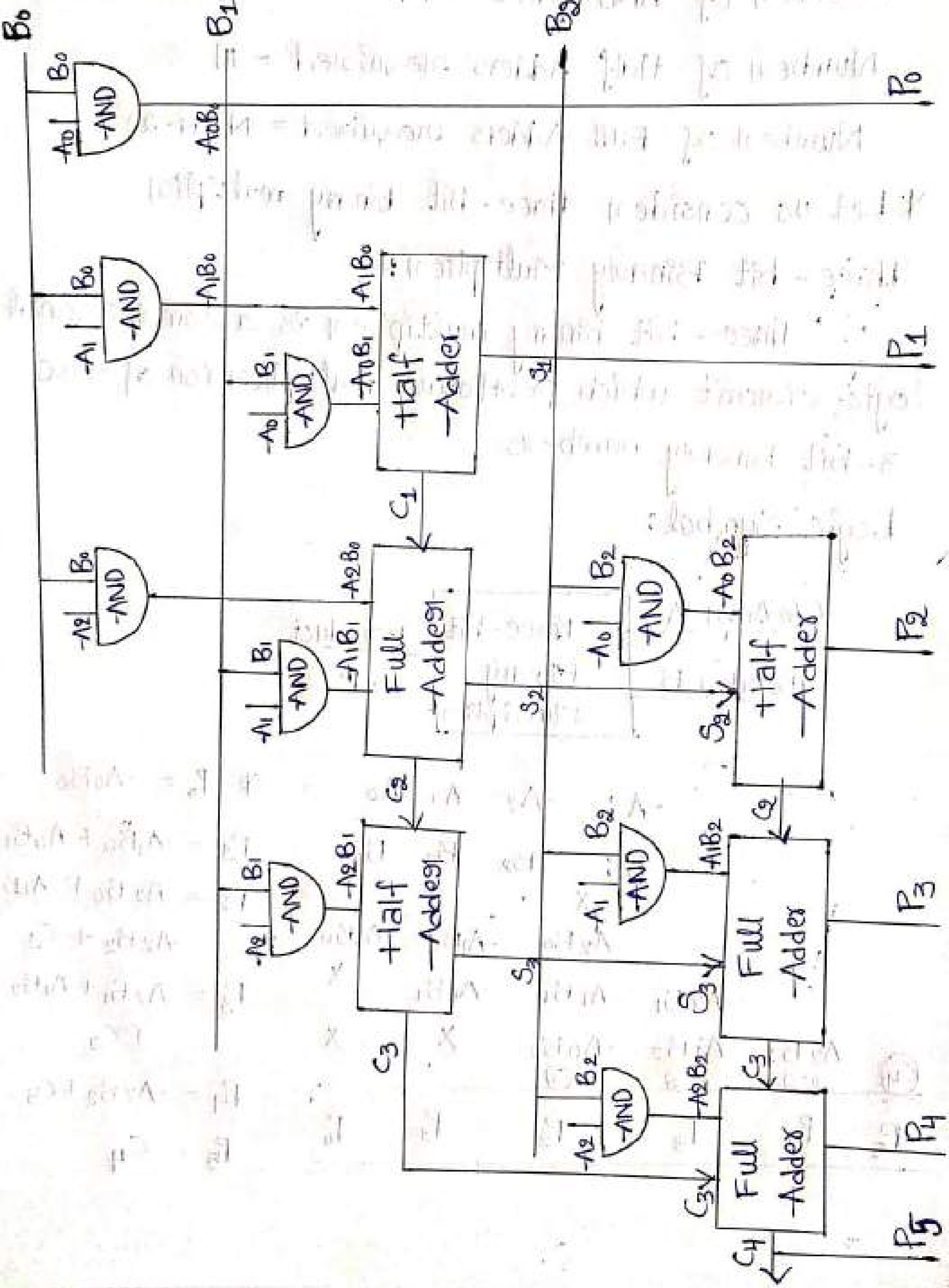
$P_2 = A_2B_0 + A_1B_1 + A_2B_2 + C_1$

$P_3 = A_2B_1 + A_1B_2 + C_2$

$P_4 = A_2B_2 + C_3$

$P_5 = C_4$

Logic Diagram:

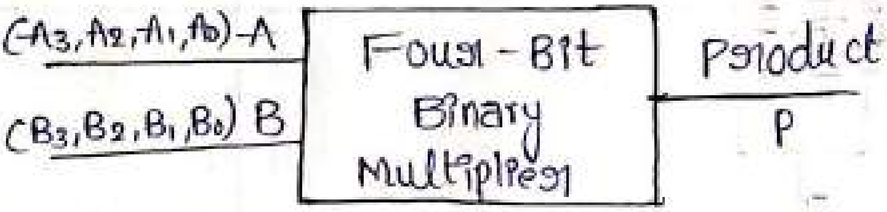


Let us consider 4- Bit Binary multiplier

Four - Bit Binary Multiplier:

Four - Bit Binary multiplier is a combinational circuit which performs multiplication of two four-bit binary numbers.

Logic Symbol:



				A :	A_3	A_2	A_1	A_0
				B :	B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0	
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	X	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2	X	X	
	A_3B_3	A_2B_3	A_1B_3	A_0B_3	X	X	X	
C_6	C_5	C_4	C_3	C_2	C_1			
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

$$P_0 = A_0B_0$$

$$P_1 = A_1B_0 + A_0B_1$$

$$P_2 = A_2B_0 + A_1B_1 + A_0B_2 + C_1$$

$$P_3 = A_3B_0 + A_2B_1 + A_1B_2 + A_0B_3 + C_2$$

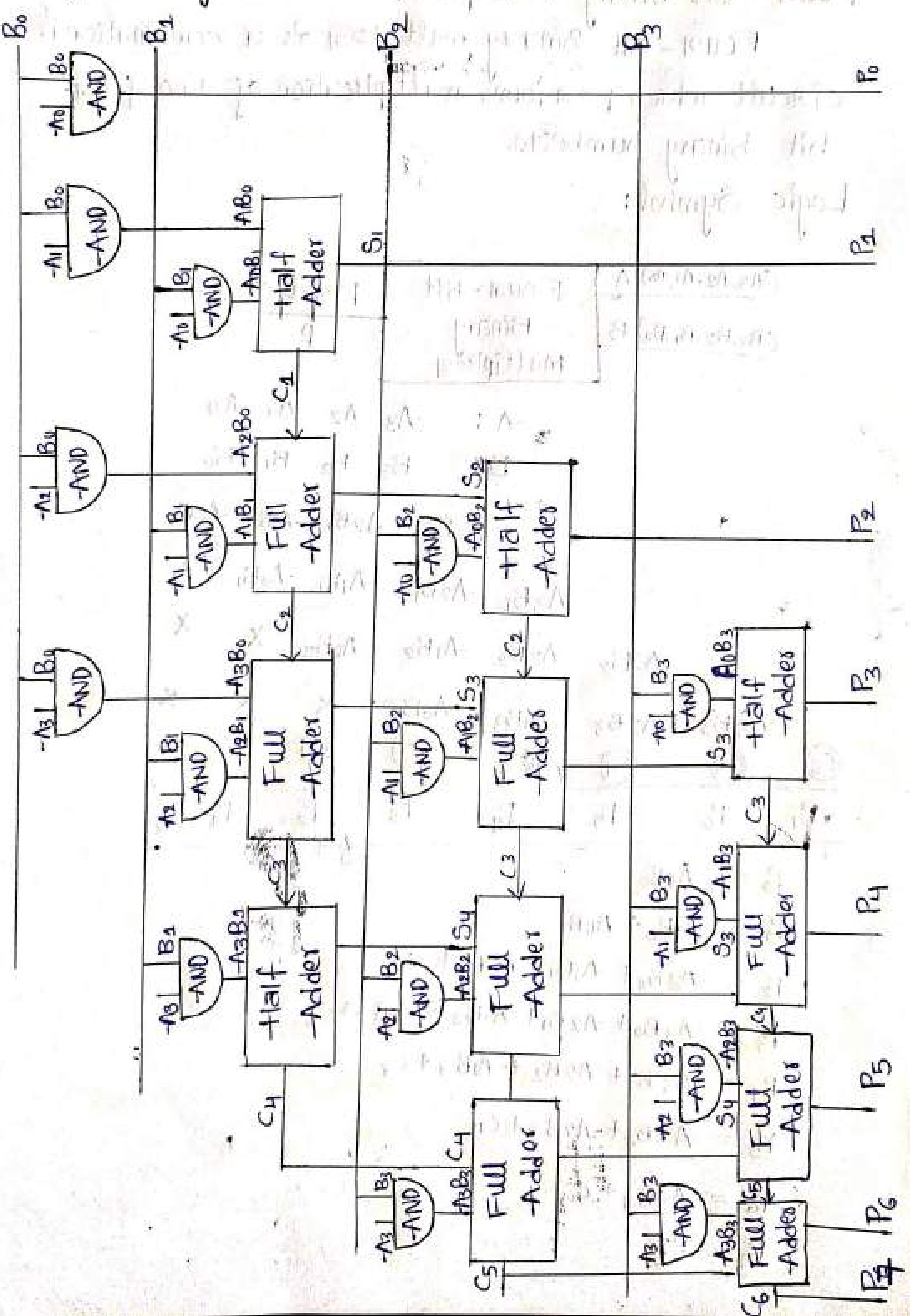
$$P_4 = A_3B_1 + A_2B_2 + A_1B_3 + C_3$$

$$P_5 = A_3B_2 + A_2B_3 + C_4$$

$$P_6 = A_3B_3 + C_5$$

$$P_7 = C_6$$

Logic Diagram:



* Magnitude Comparator:

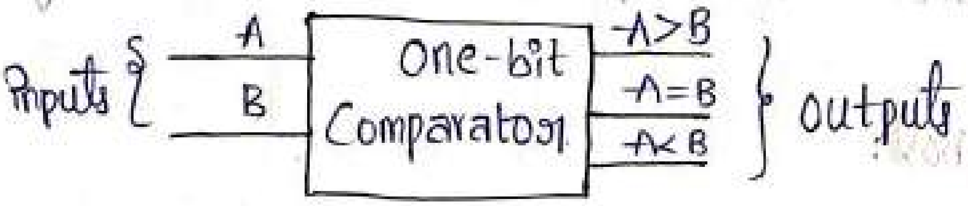
Magnitude comparator is a combinational logic circuit it compares two input binary quantities and generates outputs to indicate which one has the greater magnitude.

Let us consider 1 - bit comparator

One - Bit Comparator:

One- bit comparator is a combinational logic circuit that compares two 1- bit binary numbers.

Logic Symbol:



Truth Table of a one - bit Comparator

Inputs		Outputs		
A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

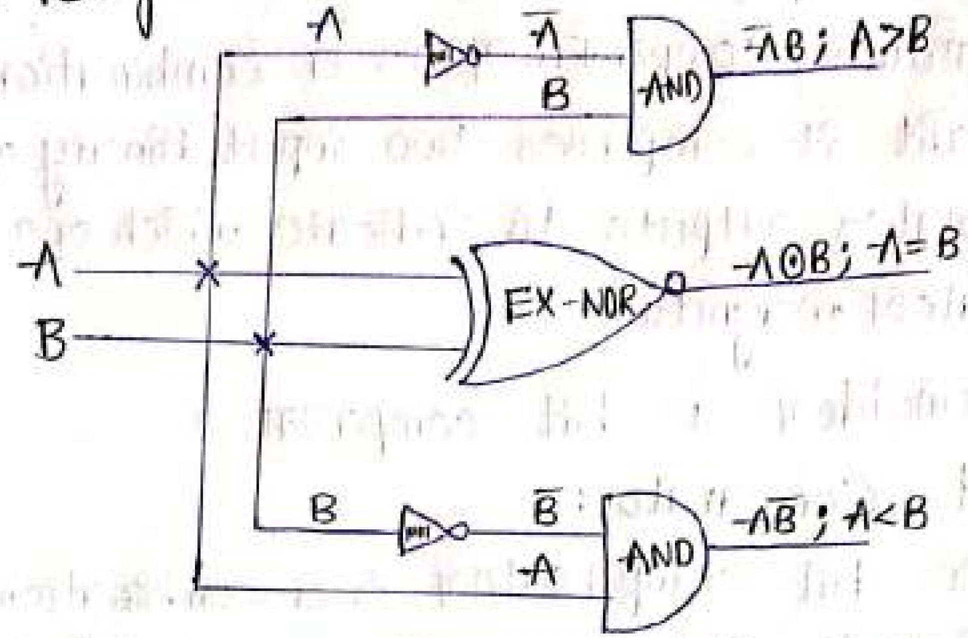
Output Expressions:

$A>B$; $A\bar{B}$

$A=B$; $\bar{A}\bar{B}+AB \Rightarrow A \odot B$

$A<B$; $\bar{A}B$

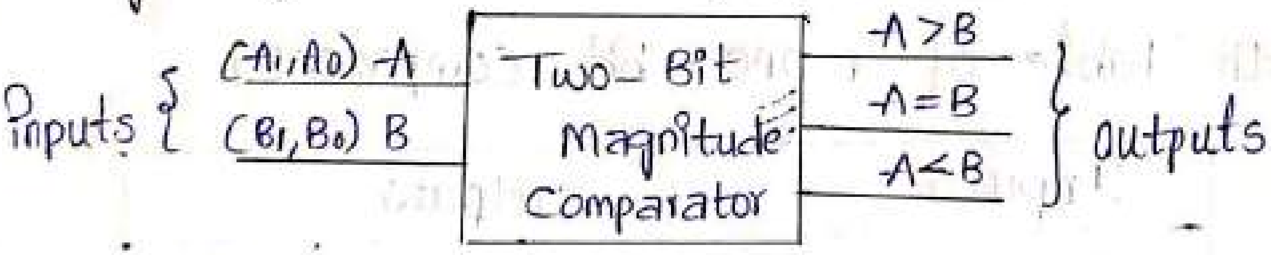
Logic Diagram:



Two-Bit Comparator:

Two-Bit comparator is a combinational logic circuit that compares two 2-bit binary numbers.

Logic Symbol:



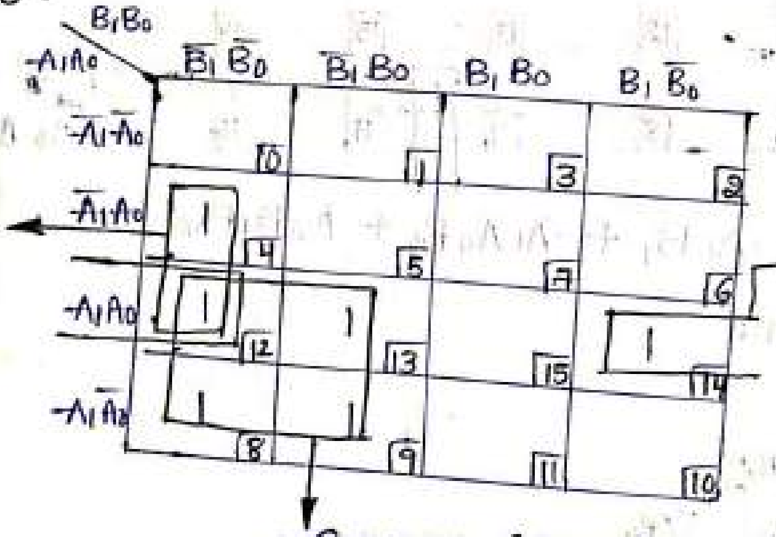
Truth Table:

Inputs (A)		Inputs (B)		Outputs		
A_1	A_0	B_1	B_0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1

A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Simplifying Expressions using K-Map:

$A>B$:



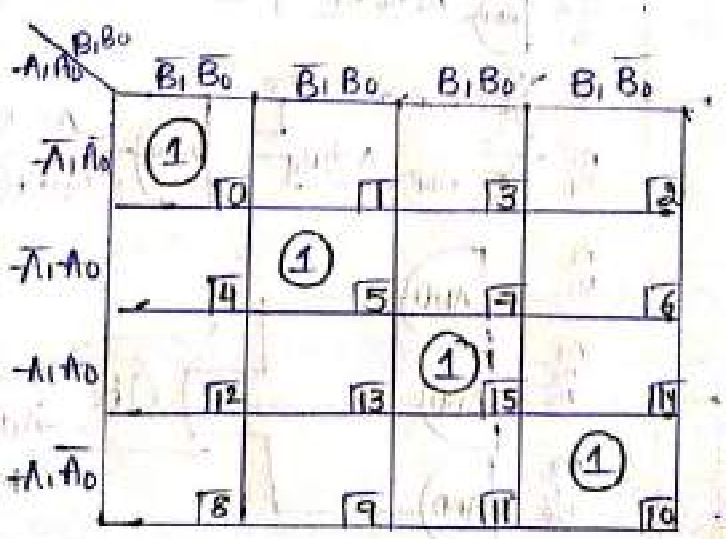
Group 2 (pair) $A_0\overline{B_1}\,\overline{B_0}$

Group 3 (pair) $A_1A_0\overline{B_0}$

Group 1 (Quad) = $A_1\overline{B_1}$:

$A>B$; $A_1\overline{B_1} + A_0\overline{B_1}\,\overline{B_0} + A_1A_0\overline{B_0}$

$A=B$:



$A = B;\ \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$A=B \Rightarrow \bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0) + A_1B_1(A_0B_0 + \bar{A}_0\bar{B}_0)$

$\Rightarrow (\bar{A}_0\bar{B}_0 + A_0B_0)(\bar{A}_1\bar{B}_1 + A_1B_1)$

$\Rightarrow (A_0 \odot B_0)(A_1 \odot B_1)$

$A < B$:

$A_1A_0 \backslash B_1B_0$	$\bar{B}_1\bar{B}_0$	$\bar{B}_1B_0$	B_1B_0	$B_1\bar{B}_0$
$\bar{A}_1\bar{A}_0$	0	1 1	1 3	1 2
$\bar{A}_1A_0$	4	5	1 7	1 6
A_1A_0	12	13	15	14
$A_1\bar{A}_0$	8	9	1 11	10

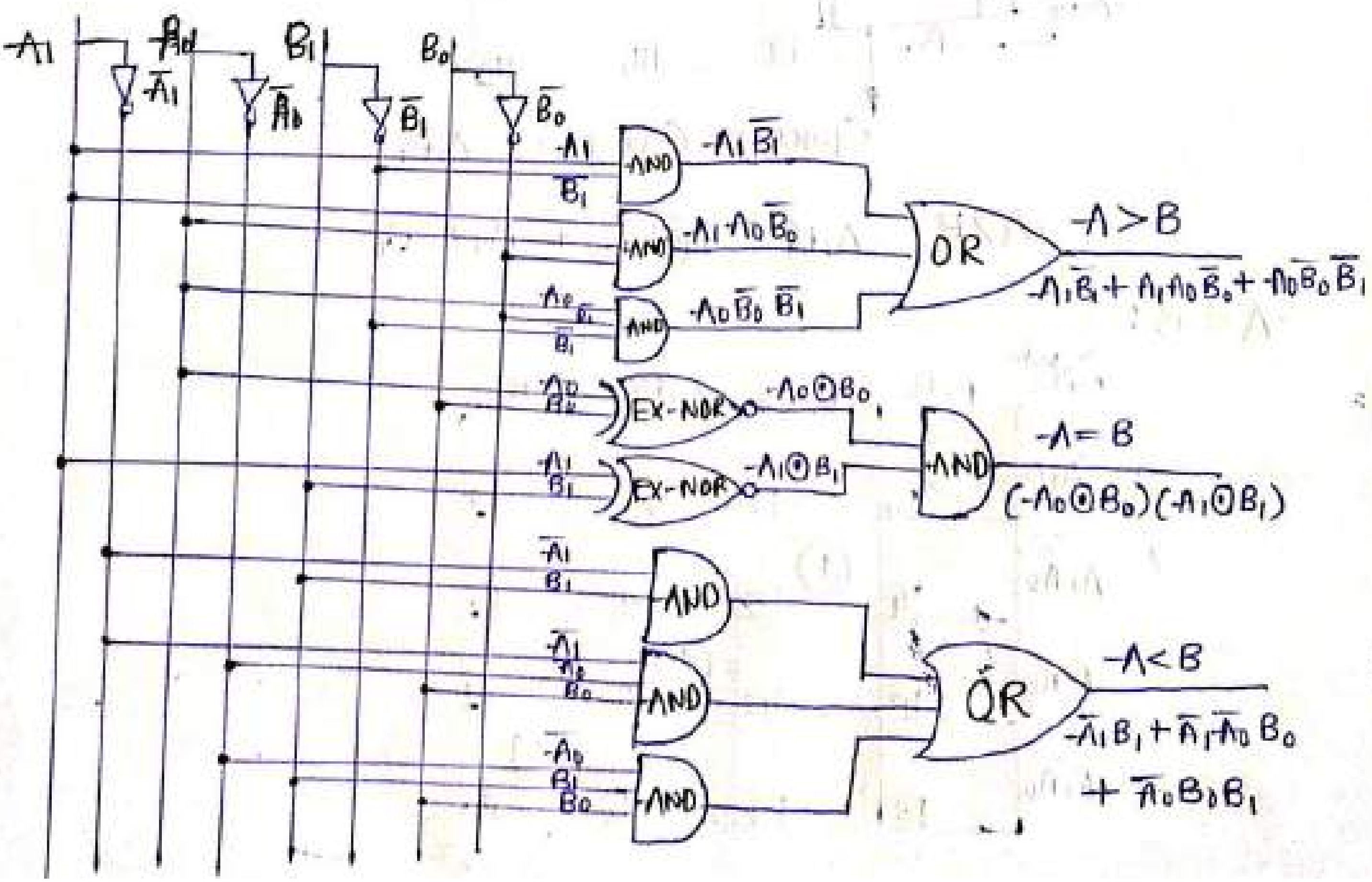
→ Group1 (Quad) = $\bar{A}_1B_1$

→ Group2 (pair) $\bar{A}_1\bar{A}_0B_0$

→ Group 3 (pair) $\bar{A}_0B_1B_0$

$A<B;\ \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$

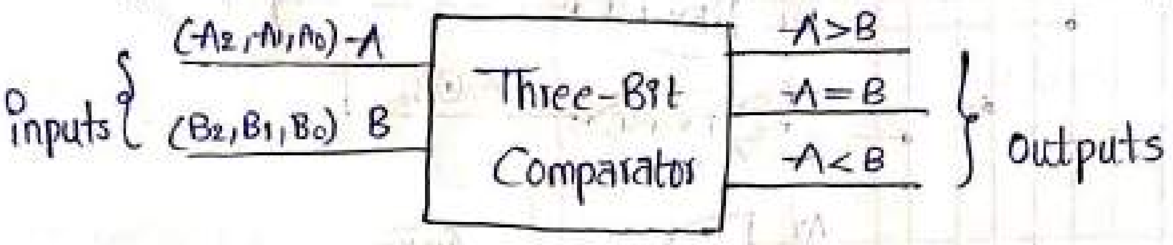
Logic Diagram:



Three - Bit Comparator:

Three - bit comparator is a combinational logic circuit that compares two 3- bit binary numbers.

Logic Symbol:



Truth Table :

Inputs			Outputs		
A_2, B_2	A_1, B_1	A_0, B_0	$A>B$	$A=B$	$A<B$
$A_2 > B_2$	X	X	1	0	0
$A_2 = B_2$	$A_1 > B_1$	X	1	0	0
$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0
$A_2 < B_2$	X	X	0	0	1
$A_2 = B_2$	$A_1 < B_1$	X	0	0	1
$A_2 = B_2$	$A_1 = B_1$	$\overline{A}_0 < B_0$	0	0	1

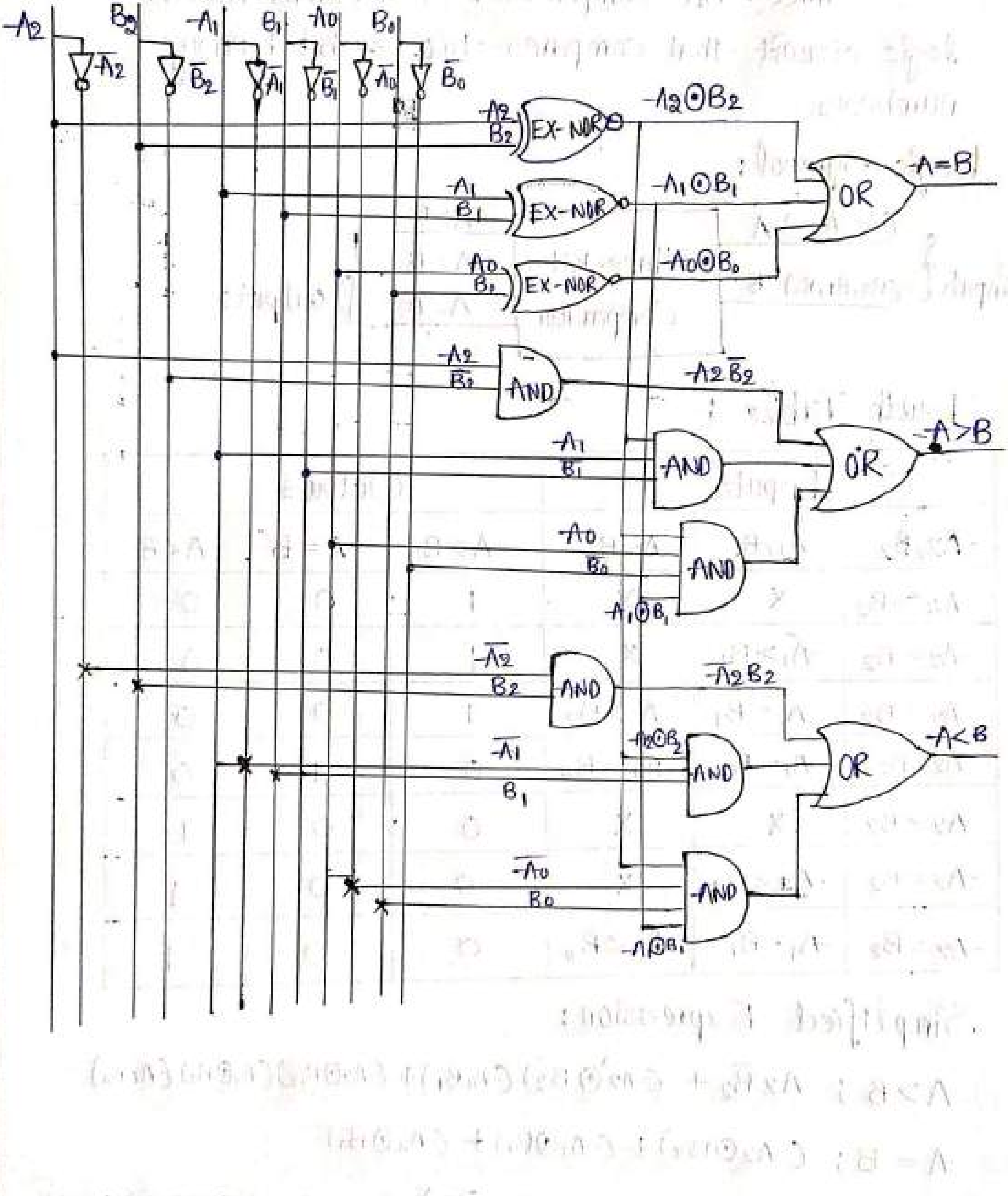
Simplified Expression:

$A > B$; $A_2 \overline{B}_2 + (A_2 \odot B_2)(A_1 \overline{B}_1) + (A_2 \odot B_2)(A_1 \odot B_1)(A_0 \overline{B}_0)$

$A = B$; $(A_2 \odot B_2) + (A_1 \odot B_1) + (A_0 \odot B_0)$

$A < B$; $\overline{A}_2 B_2 + (A_2 \odot B_2)(\overline{A}_1 B_1) + (A_2 \odot B_2)(A_1 \odot B_1)(\overline{A}_0 B_0)$

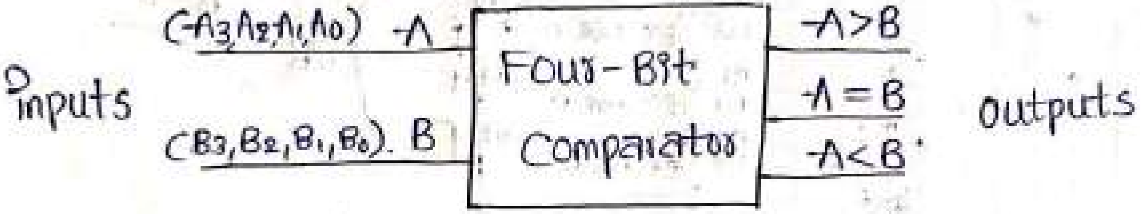
Logic Diagram:



Four-bit Comparator:

Four-bit comparator is a combinational logic circuit that compares two 4-bit binary numbers.

Logic Symbol:



Truth Table:

Inputs				Outputs		
A_3, B_3	A_2, B_2	A_1, B_1	$A_0 B_0$	$A > B$	$A = B$	$A < B$
$A_3 > B_3$	X	X	X	1	0	0
$A_3 = B_3$	$A_2 > B_2$	X	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	X	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0
$A_3 < B_3$	X	X	X	0	0	1
$A_3 = B_3$	$A_2 < B_2$	X	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	X	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	0	1

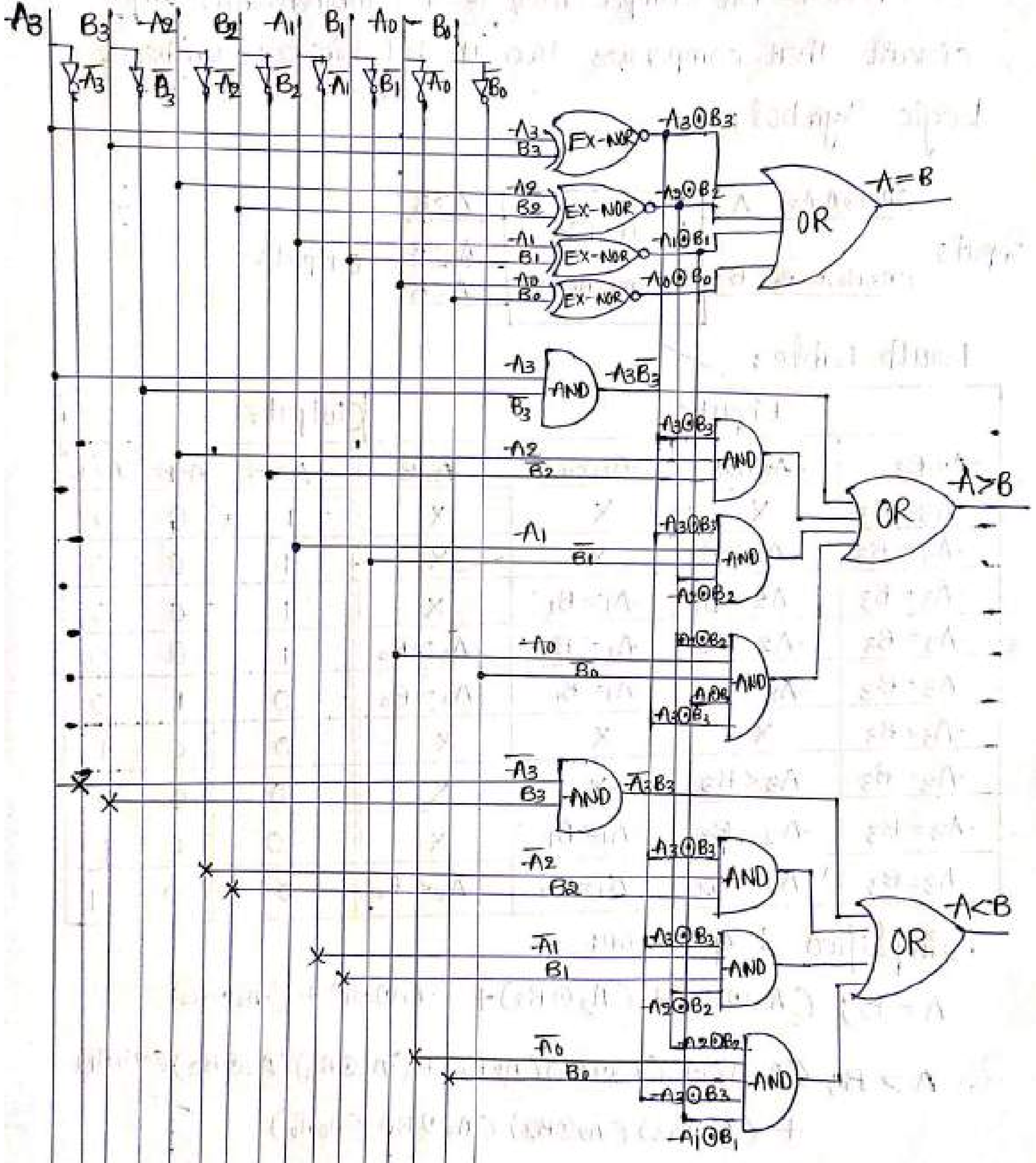
Simplified Expression:

$A = B$; $(A_3 \odot B_3) + (A_2 \odot B_2) + (A_1 \odot B_1) + (A_0 \odot B_0)$

$A > B$; $(A_3 \overline{B_3}) + (A_3 \odot B_3)(A_2 \overline{B_2}) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \overline{B_1})$
$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \overline{B_0})$

$A < B$; $(\overline{A_3} B_3) + (A_3 \odot B_3)(\overline{A_2} B_2) + (A_3 \odot B_3)(A_2 \odot B_2)(\overline{A_1} B_1)$
$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(\overline{A_0} B_0)$

Logic Diagram:


A3 B3 A2 B2 A1 B1 A0 B0
$\overline{A_3}$ $\overline{B_3}$ $\overline{A_2}$ $\overline{B_2}$ $\overline{A_1}$ $\overline{B_1}$ $\overline{A_0}$ $\overline{B_0}$
EX-NOR
A3⊙B3
A2⊙B2
A1⊙B1
A0⊙B0
OR
A=B
AND
$A_3\overline{B_3}$
A>B
$\overline{A_3}B_3$
A<B

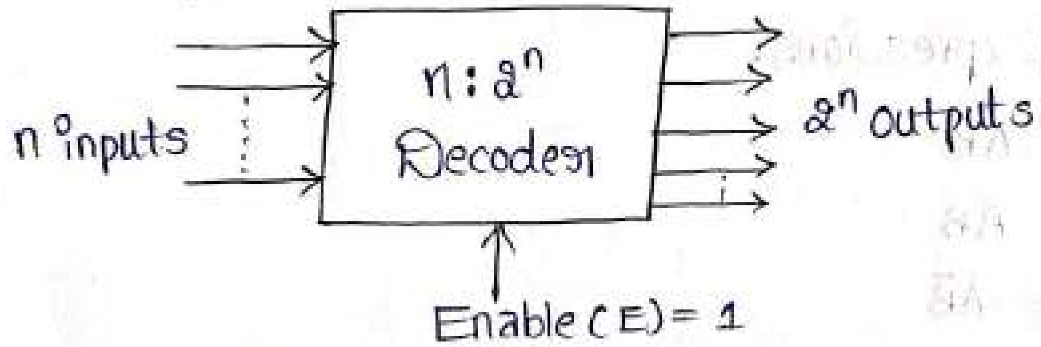
* Decoder:

Decoder is a combinational logic circuit that has multiple inputs and multiple outputs.

⟹ Decoder is used to convert binary data into other form of data.

⟹ Decoder has 'n' inputs and '2^n' outputs, and E = 1.

Logic Symbol:



⟹ If E = 1 the decoder is in active state.

⟹ If E = 0 the decoder is in inactive state.

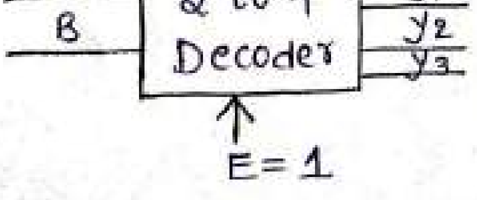
Types of Decoders:

1) 2 to 4 Decoder
2) 3 to 8 Decoder
3) 4 to 16 Decoder
4) 5 to 32 Decoder

Design 2 to 4 Decoder:

Two to four decoder is a combinational logic circuit which converts 2 inputs to 4 outputs.

Logic Symbol:



Truth Table:

Inputs		Output			
A	B	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Output Expressions:

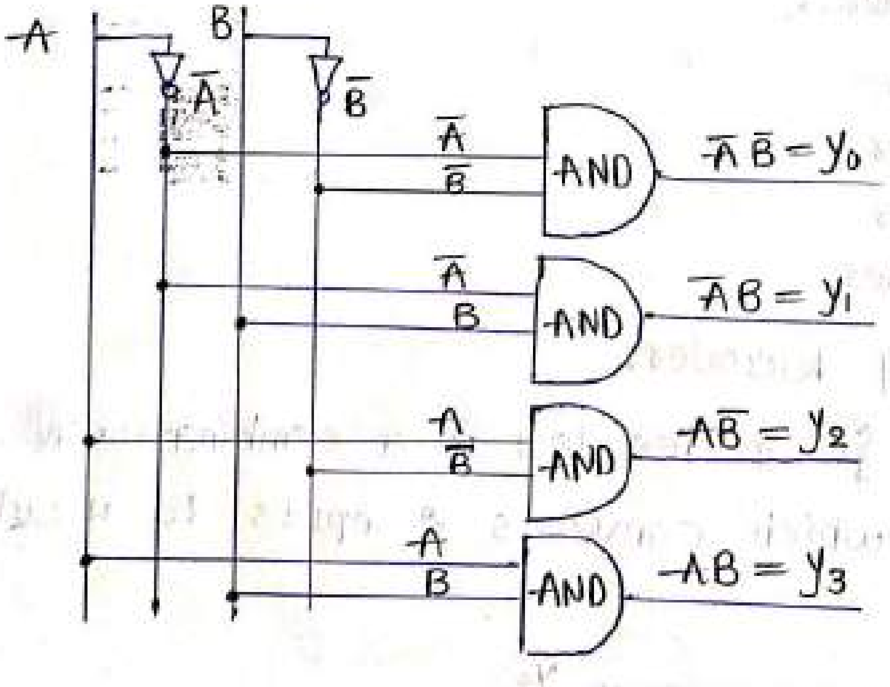
$y_0 = \overline{A}\overline{B}$

$y_1 = \overline{A}B$

$y_2 = A\overline{B}$

$y_3 = AB$

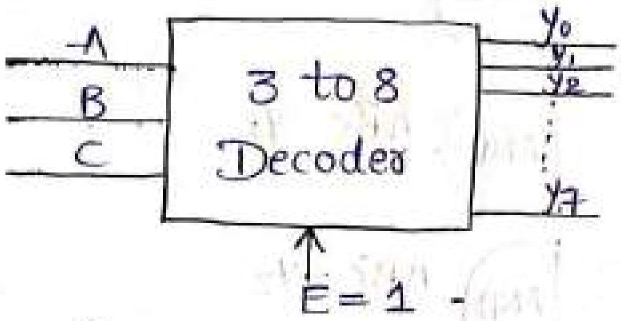
Logic Diagram:



Design 3 to 8 Decoder:

Three to Eight decoder is a combinational. logic circuit which converts 3 inputs to 8 outputs (2^3).

Logic Symbol:



Truth Table:

Inputs			Outputs							
A	B	C	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Output Expressions:

$y_0 = \bar{A}\bar{B}\bar{C}$

$y_1 = \bar{A}\bar{B}C$

$y_2 = \bar{A}B\bar{C}$

$y_3 = \bar{A}BC$

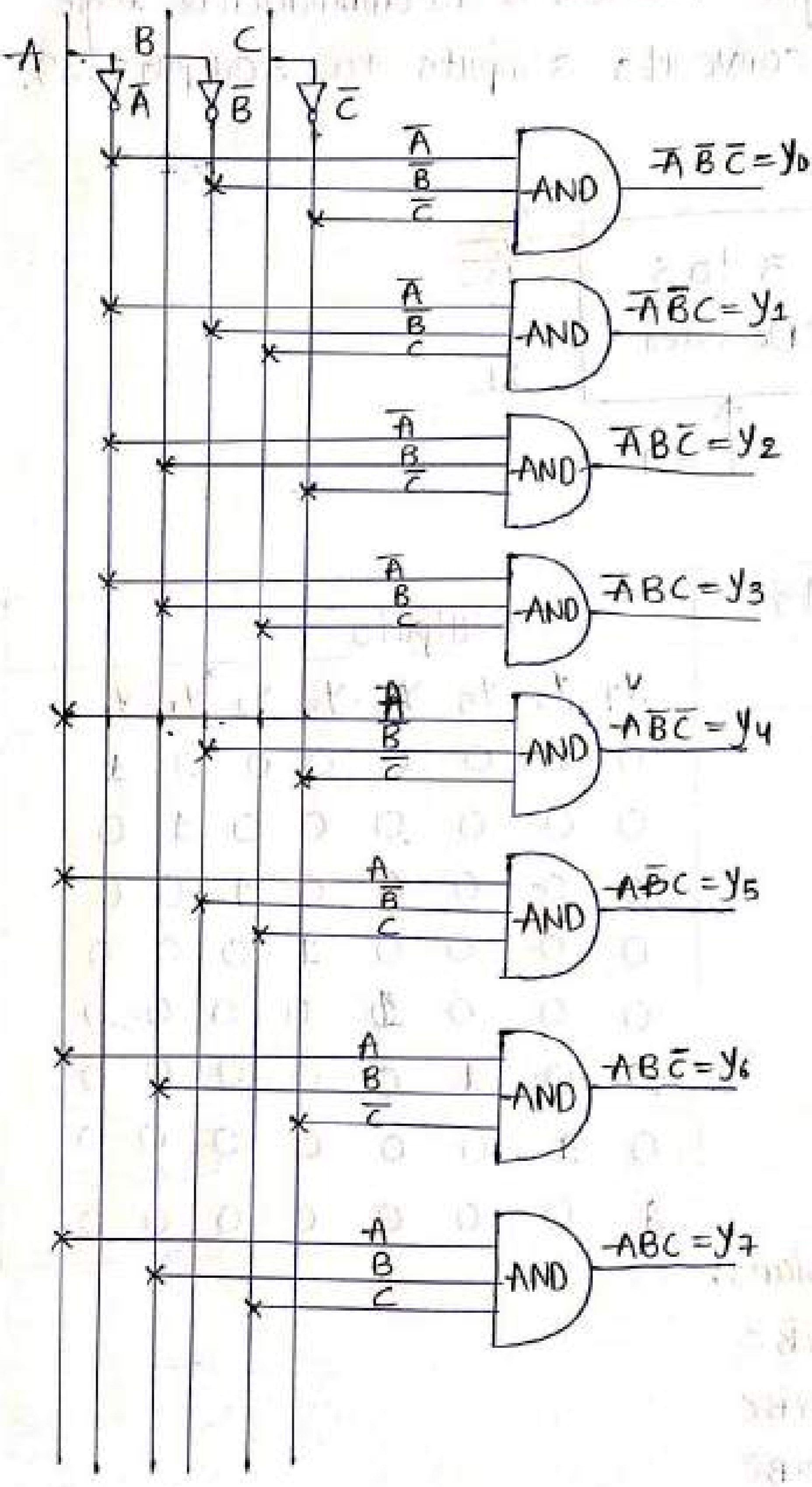
$y_4 = A\bar{B}\bar{C}$

$y_5 = A\bar{B}C$

$y_6 = AB\bar{C}$

$y_7 = ABC$

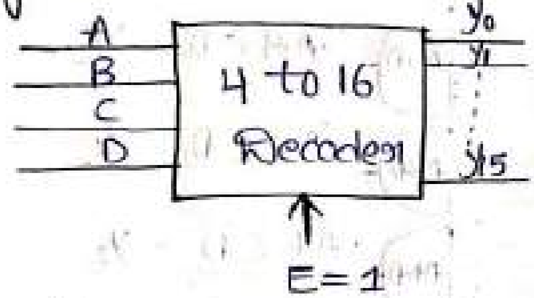
Logic Diagram:


A
B
C
$\overline{A}$
$\overline{B}$
$\overline{C}$
AND
$\overline{A}\,\overline{B}\,\overline{C} = y_0$
$\overline{A}\,\overline{B}C = y_1$
$\overline{A}B\overline{C} = y_2$
$\overline{A}BC = y_3$
$A\overline{B}\,\overline{C} = y_4$
$A\overline{B}C = y_5$
$AB\overline{C} = y_6$
$ABC = y_7$

Design 4 to 16 Decoder:

Four to sixteen decoder is a combinational logic circuit that converts 4 inputs to (24) 16 outputs.

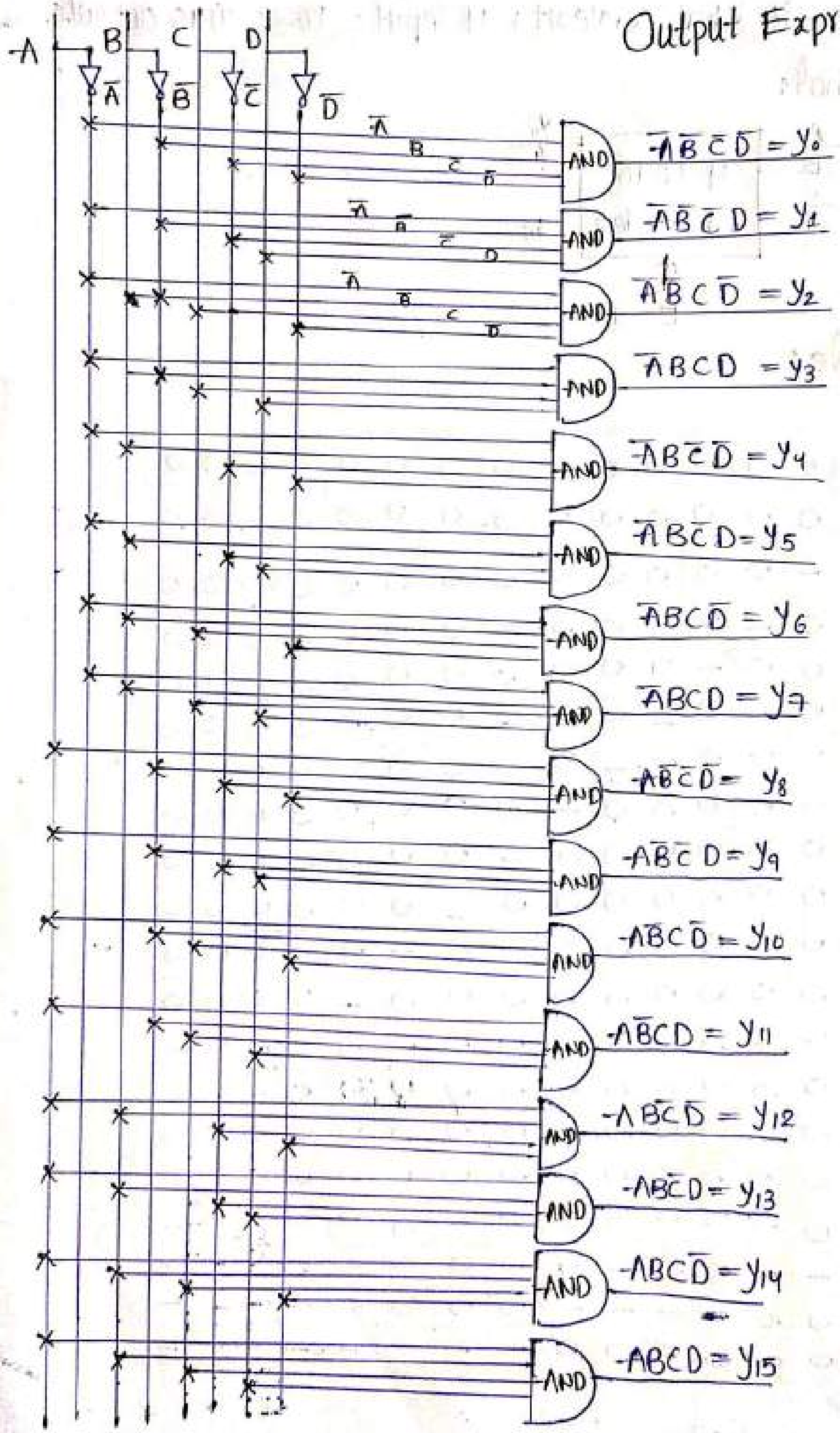
Logic Symbol:



Truth Table:

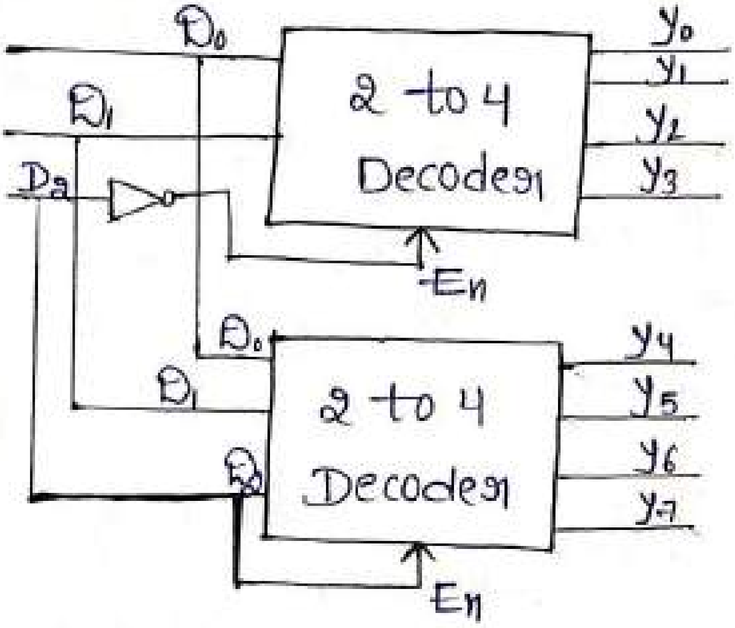
Inputs				Outputs															
A	B	C	D	y15	y14	y13	y12	y11	y10	y9	y8	y7	y6	y5	y4	y3	y2	y1	y0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0

Logic Diagram:



Design 3 to 8 Decoder by using 2 to 4 Decoder

Logic Symbol:



Truth Table:

Inputs			Outputs							
D_2	D_1	D_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Output Expressions:

$y_0 = \bar{A}\bar{B}\bar{C}$

$y_1 = \bar{A}\bar{B}C$

$y_2 = \bar{A}B\bar{C}$

$y_3 = \bar{A}BC$

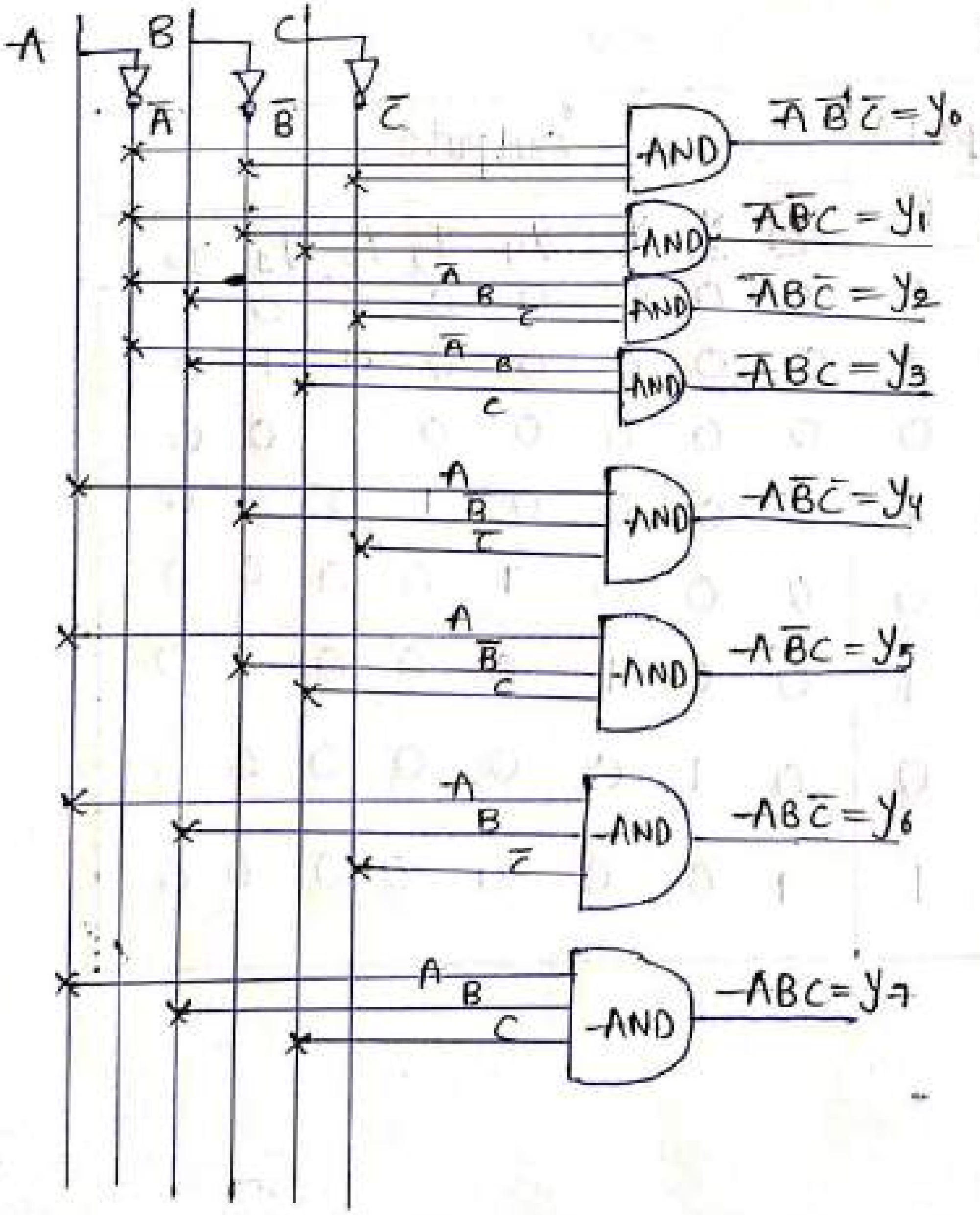
$y_4 = A\bar{B}\bar{C}$

$y_5 = A\bar{B}C$

$y_6 = AB\bar{C}$

$y_7 = ABC$

Logic Diagram:

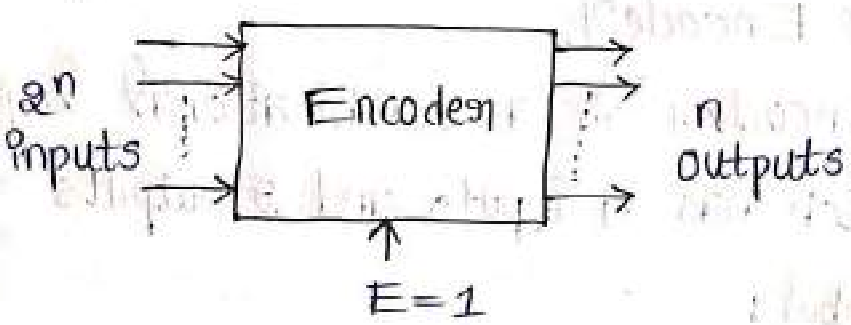


* Encoder:

Encoder is a combinational logic circuit that converts other form of data into binary data.

⟹ Encoder has 2^n inputs and 'n' outputs and also having one enable input (E) and the enable input is always high.

Logic Symbol:



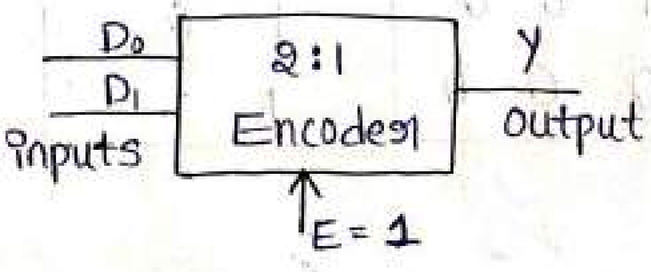
Types of Encoders:

1) 2:1 Encoder
2) 4:2 Encoder
3) 8:3 Encoder
4) 16:4 Encoder

Design of 2:1 Encoder:

2:1 Encoder is a combinational logic circuit which has 2 inputs and one output.

Logic Symbol:



Truth Table:

Inputs		Outputs
D_1	D_0	y
0	1	0
1	0	1

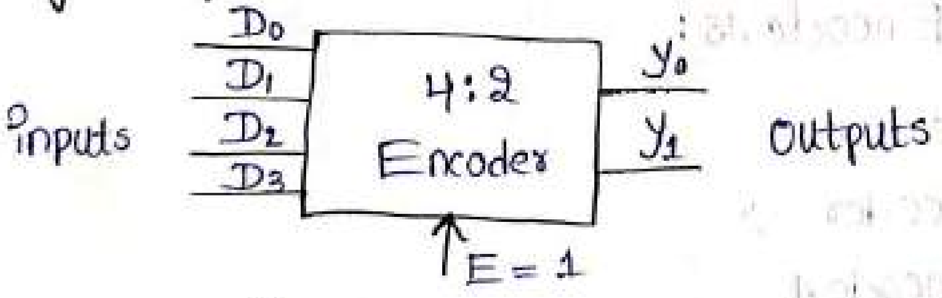
Output Expression:

$$y = D_1$$

Design of 4:2 Encoder.

4:2 Encoder is a combinational logic circuit which has 4 inputs and 2 outputs

Logic Symbol:



Truth Table:

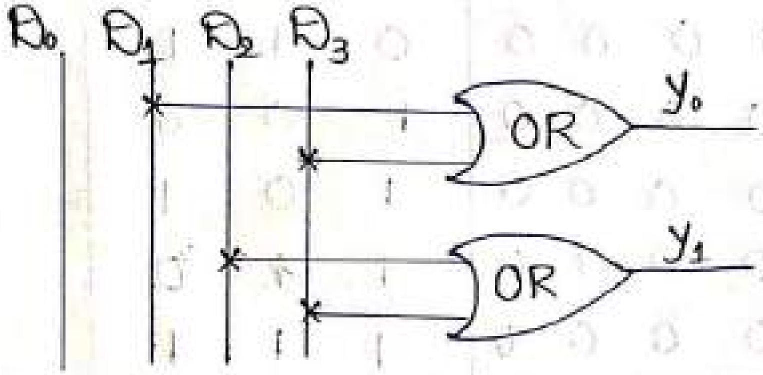
Inputs				Output	
D_3	D_2	D_1	D_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Output Expressions:

$Y_0 = D_1 + D_3$

$Y_1 = D_2 + D_3$

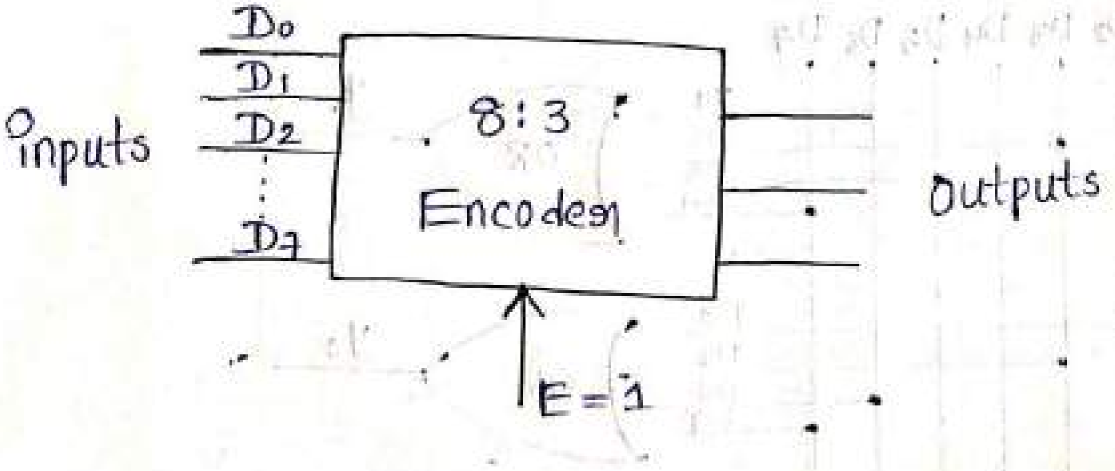
Logic Diagram:



Design of 8:3 Encoder :

8:3 Encoder is a combinational logic circuit which has 8 inputs and 3 outputs

Logic Symbol:



⟹ 8:3 Encoder is also known as Octal to Binary Encoder

Truth Table:

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

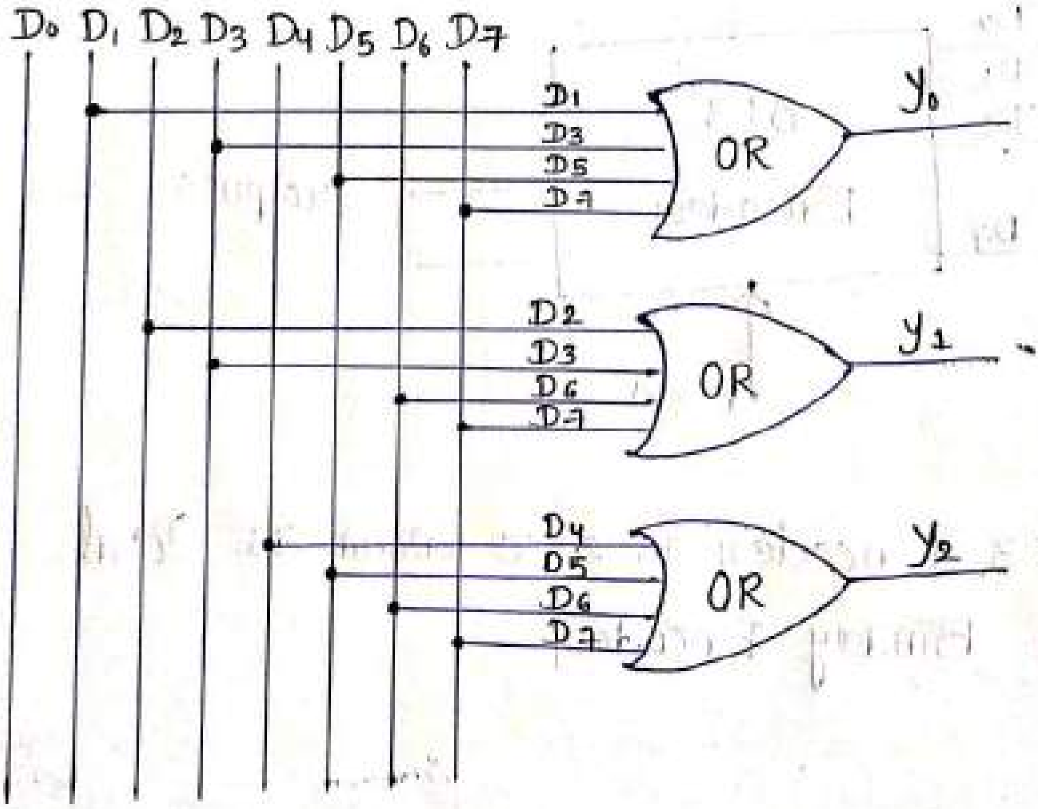
Output Expressions:

$$y_0 = D_1 + D_3 + D_5 + D_7$$

$$y_1 = D_2 + D_3 + D_6 + D_7$$

$$y_2 = D_4 + D_5 + D_6 + D_7$$

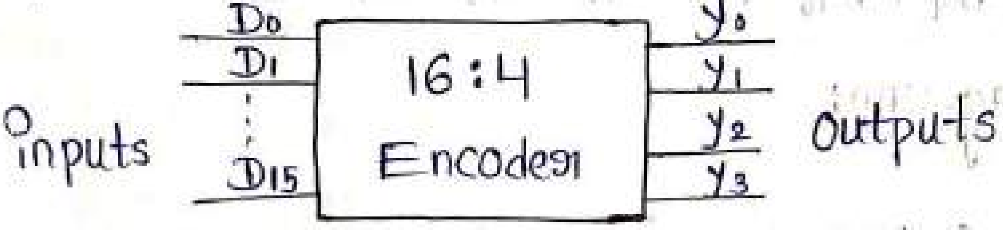
Logic Diagram:



Design of 16:4 Encoder:

16:4 Encoder is a combinational logic circuit which has 16 inputs and 4 outputs

Logic Symbol:



Truth Table:

Inputs																Outputs			
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Output Expressions:

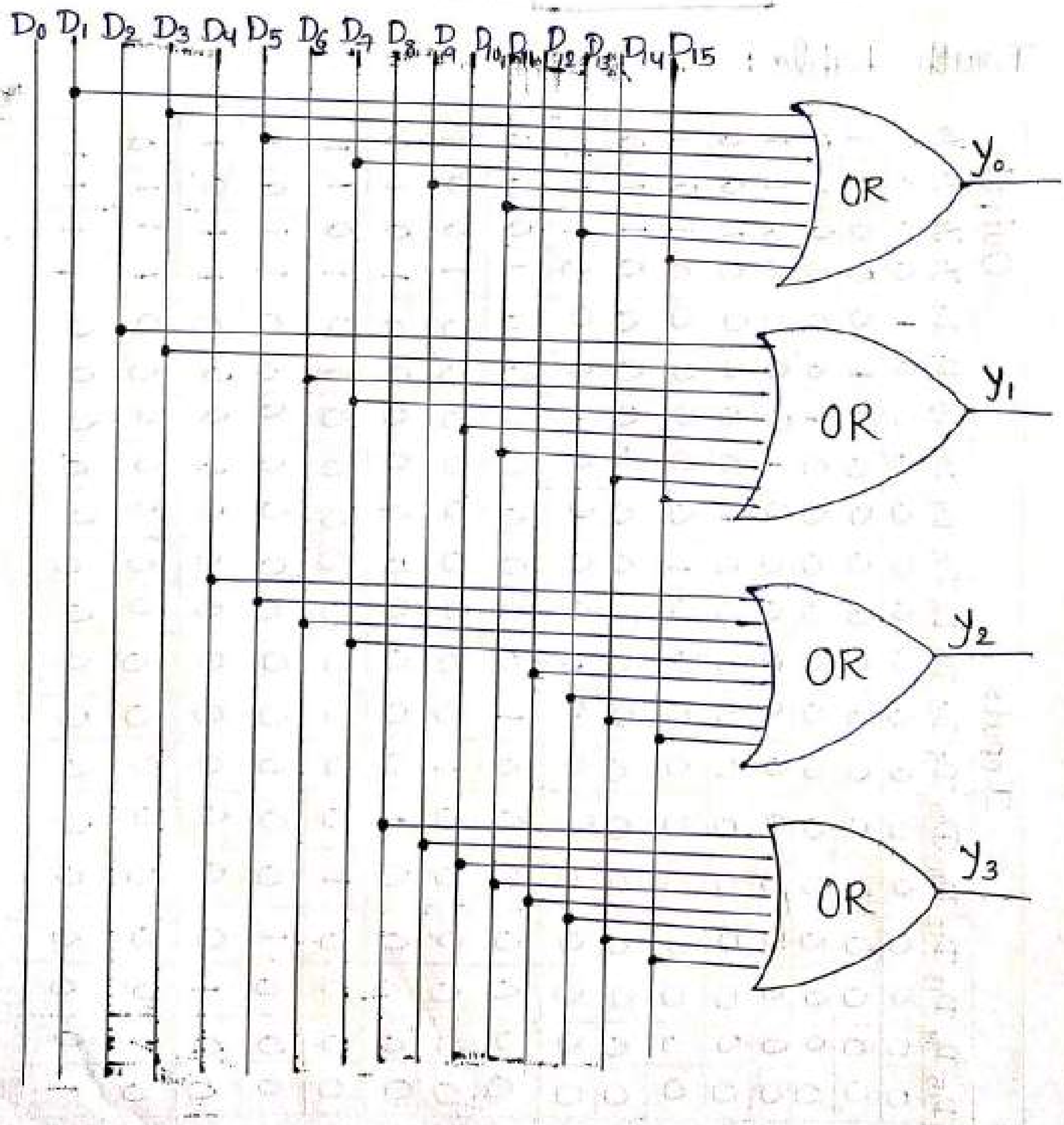
$y_0 = D_1 + D_3 + D_5 + D_7 + D_9 + D_{11} + D_{13} + D_{15}$

$y_1 = D_2 + D_3 + D_6 + D_7 + D_{10} + D_{11} + D_{14} + D_{15}$

$y_2 = D_4 + D_5 + D_6 + D_7 + D_{12} + D_{13} + D_{14} + D_{15}$

$y_3 = D_8 + D_9 + D_{10} + D_{11} + D_{12} + D_{13} + D_{14} + D_{15}$

Logic Diagram:



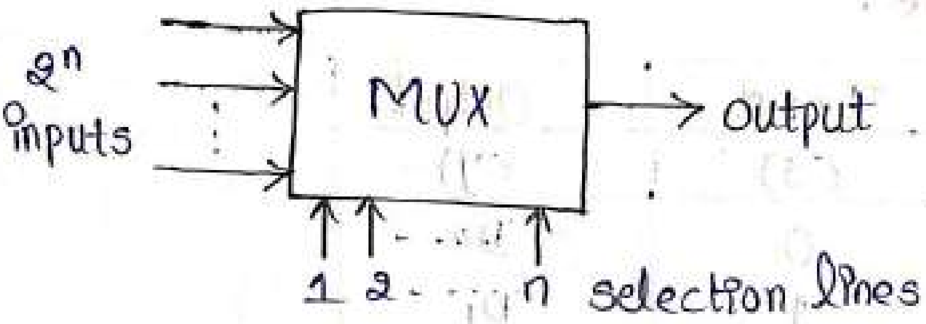
* Multiplexer:

Multiplexer is a combinational logic circuit that selects binary information from one of many lines and direct it to output line.

⟹ A multiplexer has '2^n' input lines, 'n'-selection lines and has only one output.

⟹ The multiplexer selects one input among several inputs based on the selection lines.

Logic Symbol:



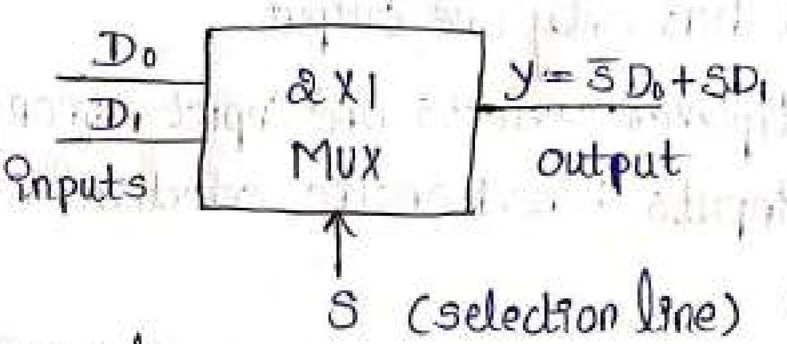
Types of Multiplexers:

1) 2X1 MUX
2) 4X1 MUX
3) 8X1 MUX
4) 16X1 MUX
5) 32X1 MUX
6) 64X1 MUX

Design Of 2X1 Multiplexer:

2X1 multiplexer is a combinational logic circuit which has 2 inputs, one output and one selection line.

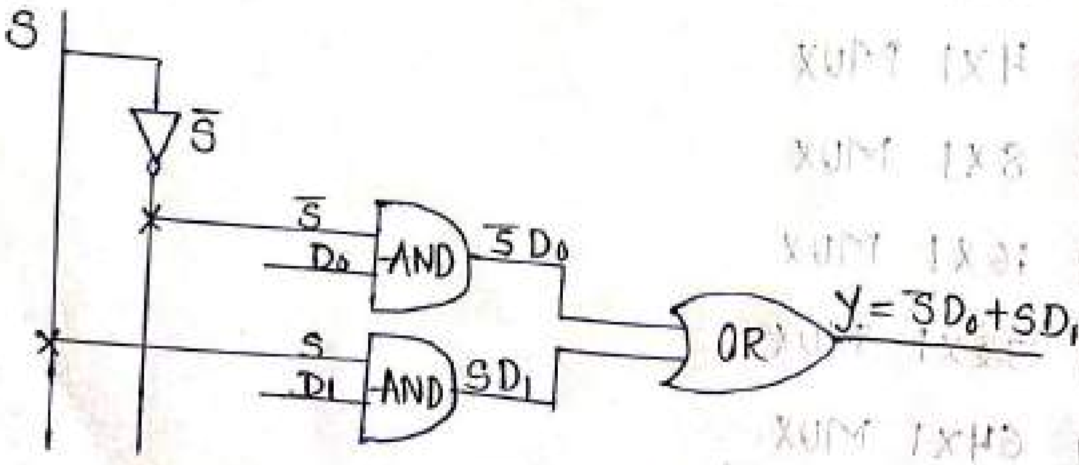
Logic Symbol:



Truth Table:

Input	Output
(S)	(Y)
0	D_0
1	D_1

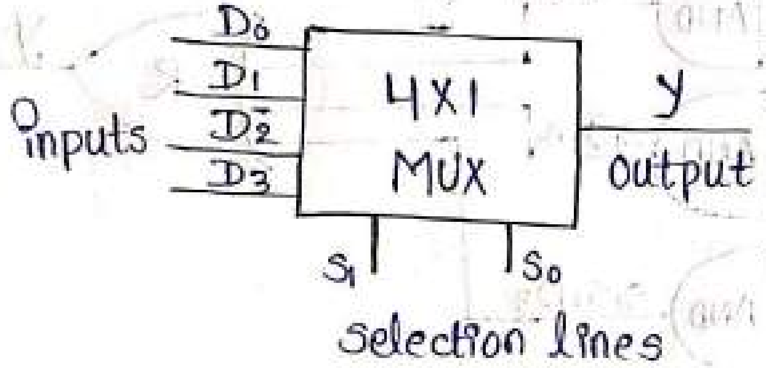
Logic Diagram:



Design of 4X1 multiplexer:

4X1 multiplexer is a combinational logic circuit which has 4 input lines, 2 selection lines and only one output line.

Logic Diagram:



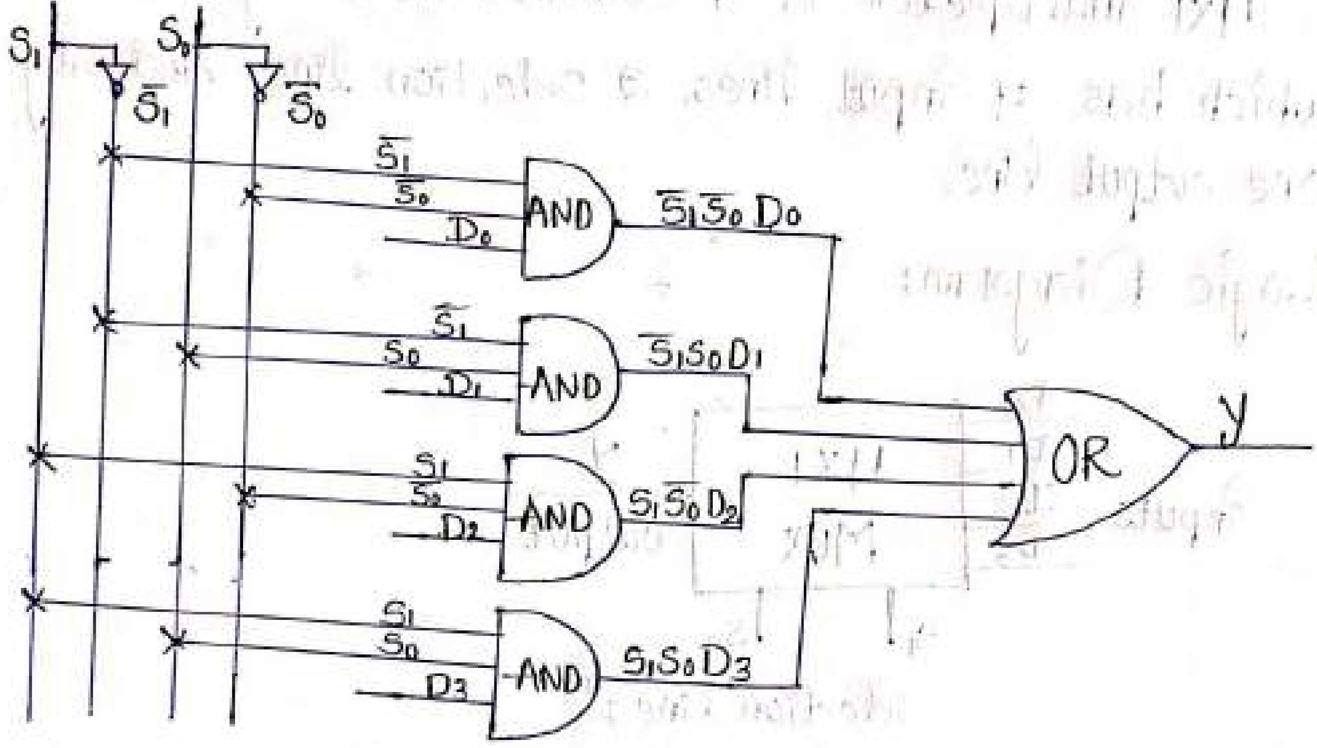
Truth Table:

Inputs		output
S_1	S_0	y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Output Expression:

$$y = \bar{S}_1\bar{S}_0D_0 + \bar{S}_1S_0D_1 + S_1\bar{S}_0D_2 + S_1S_0D_3$$

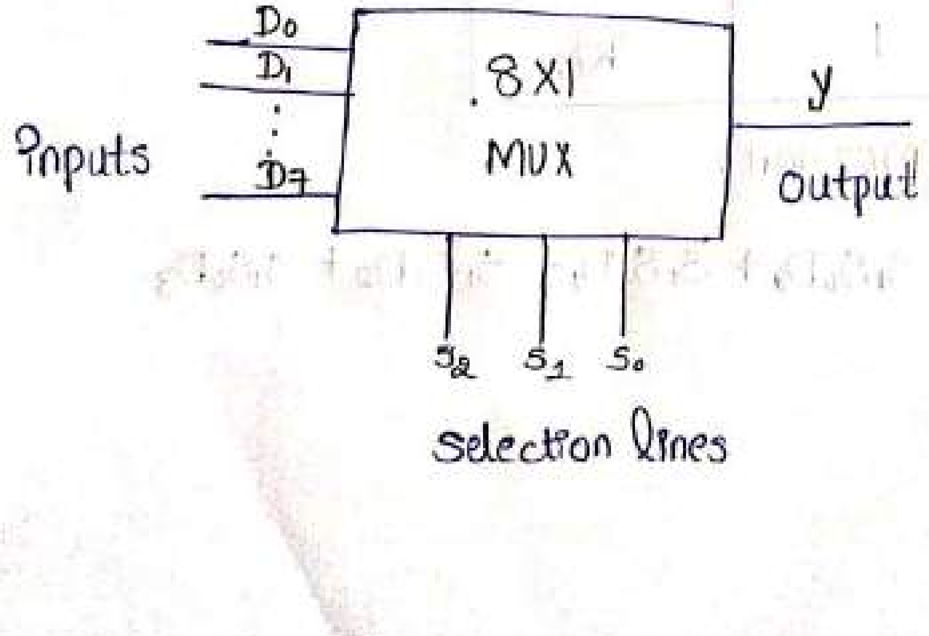
Logic Diagram:



Design of 8X1 Multiplexer:

8X1 Multiplexer is a combinational logic circuit which has 8 inputs; 3 selection lines and only one output line.

Logic Symbol:



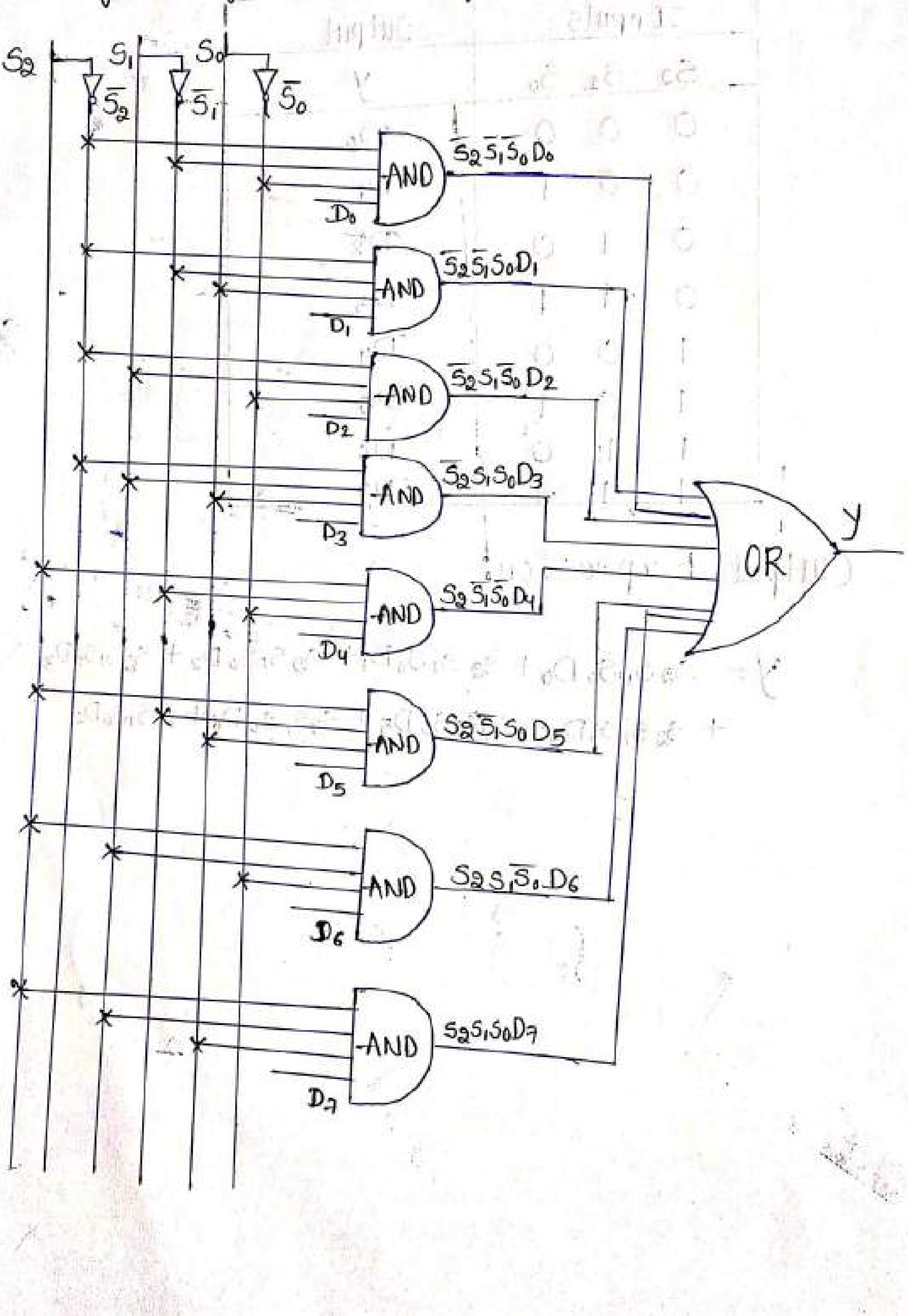
Truth Table :

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

Output Expression :-

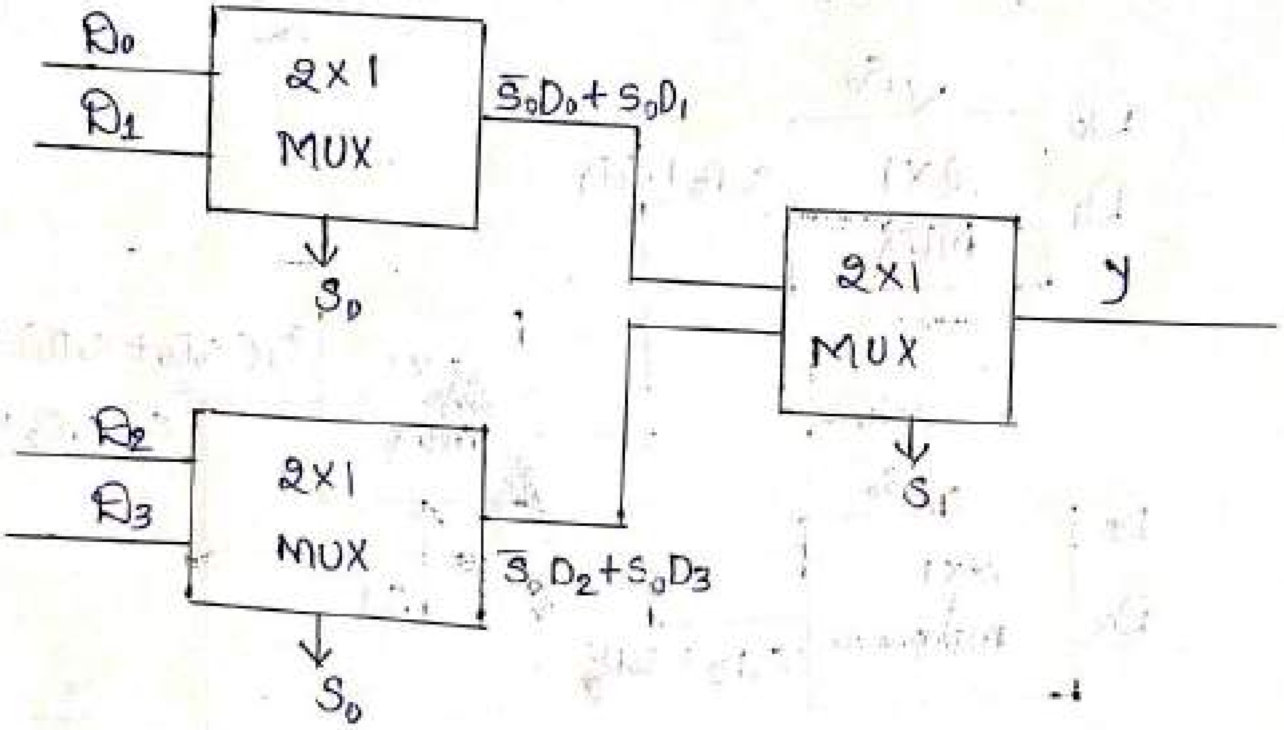
$$Y = \bar{S}_2\bar{S}_1\bar{S}_0D_0 + \bar{S}_2\bar{S}_1S_0D_1 + \bar{S}_2S_1\bar{S}_0D_2 + \bar{S}_2S_1S_0D_3$$
$$+ S_2\bar{S}_1\bar{S}_0D_4 + S_2\bar{S}_1S_0D_5 + S_2S_1\bar{S}_0D_6 + S_2S_1S_0D_7$$

Logic Diagram:



S_2
S_1
S_0
$\overline{S_2}$
$\overline{S_1}$
$\overline{S_0}$
AND
$\overline{S_2}\,\overline{S_1}\,\overline{S_0}D_0$
D_0
AND
$\overline{S_2}\,\overline{S_1}S_0D_1$
D_1
AND
$\overline{S_2}S_1\overline{S_0}D_2$
D_2
AND
$\overline{S_2}S_1S_0D_3$
D_3
AND
$S_2\overline{S_1}\,\overline{S_0}D_4$
D_4
AND
$S_2\overline{S_1}S_0D_5$
D_5
AND
$S_2S_1\overline{S_0}D_6$
D_6
AND
$S_2S_1S_0D_7$
D_7
OR
Y

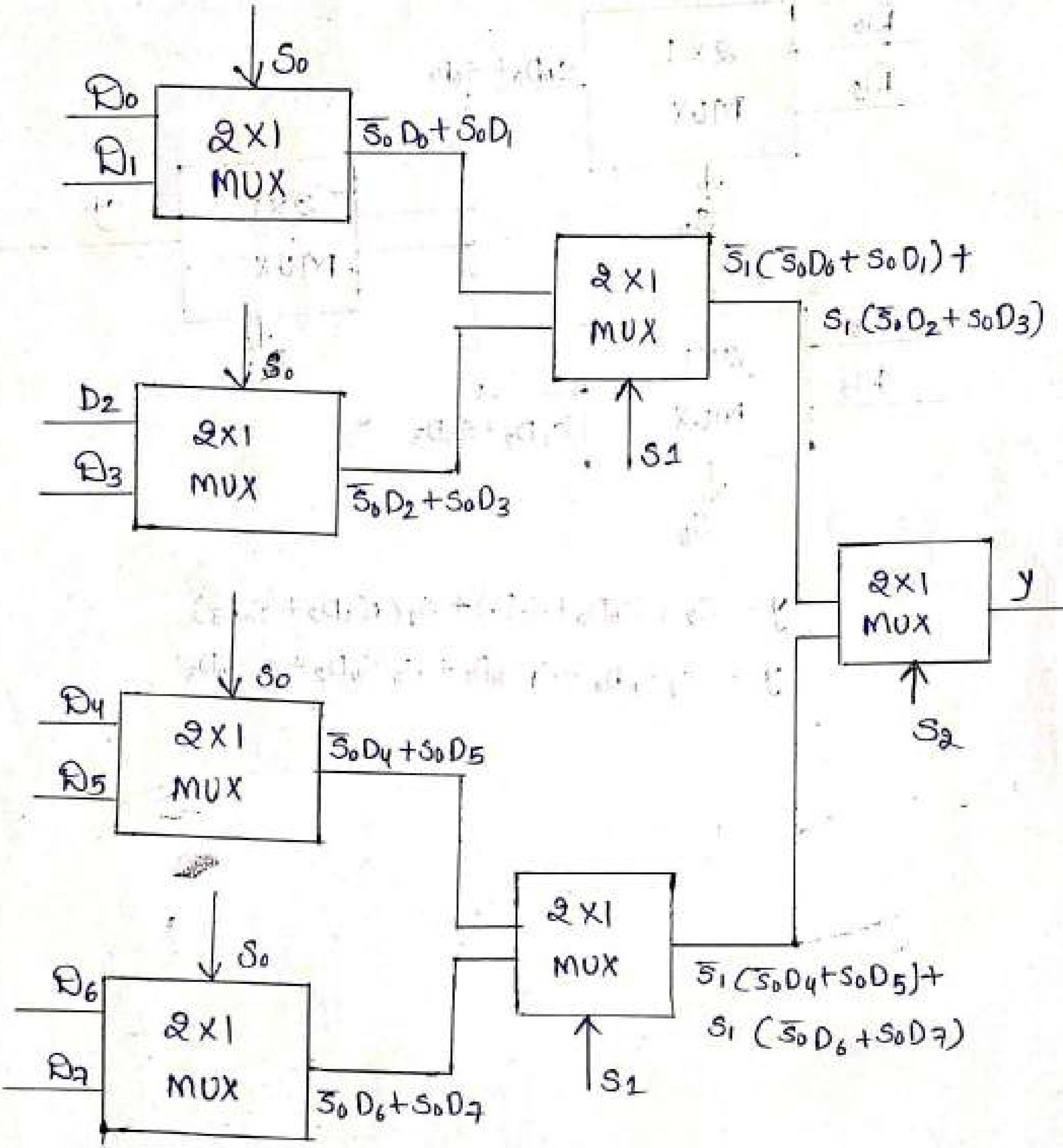
Design of 4X1 MUX by using 2x1 multiplexers:



$$y = \overline{S_1}(\overline{S_0}D_0+S_0D_1)+S_1(\overline{S_0}D_2+S_0D_3)$$

$$y = \overline{S_1}\,\overline{S_0}D_0+\overline{S_1}S_0D_1+S_1\overline{S_0}D_2+S_1S_0D_3$$

Design of 8x1 multiplexer using 2x1 multiplexer:



S_0
D_0
D_1
2X1 MUX
$\overline{S_0} D_0 + S_0 D_1$
S_0
D_2
D_3
2X1 MUX
$\overline{S_0} D_2 + S_0 D_3$
2X1 MUX
S_1
$\overline{S_1}(\overline{S_0} D_0 + S_0 D_1) +$
$S_1(\overline{S_0} D_2 + S_0 D_3)$
2X1 MUX
y
S_2
S_0
D_4
D_5
2X1 MUX
$\overline{S_0} D_4 + S_0 D_5$
S_0
D_6
D_7
2X1 MUX
$\overline{S_0} D_6 + S_0 D_7$
2X1 MUX
S_1
$\overline{S_1}(\overline{S_0} D_4 + S_0 D_5) +$
$S_1(\overline{S_0} D_6 + S_0 D_7)$

$$y = \overline{S_2}(\overline{S_1}\,\overline{S_0} D_0 + \overline{S_1} S_0 D_1 + S_1 \overline{S_0} D_2 + S_0 S_1 D_3) + S_2(\overline{S_1}\,\overline{S_0} D_4 + \overline{S_1} S_0 D_5 + S_1 \overline{S_0} D_6 + S_1 S_0 D_7)$$

$$y = \overline{S_2}\,\overline{S_1}\,\overline{S_0} D_0 + \overline{S_2}\,\overline{S_1} S_0 D_1 + \overline{S_2} S_1 \overline{S_0} D_2 + \overline{S_2} S_1 S_0 D_3 + S_2 \overline{S_1}\,\overline{S_0} D_4 + S_2 \overline{S_1} S_0 D_5 + S_2 S_1 \overline{S_0} D_6 + S_2 S_1 S_0 D_7$$