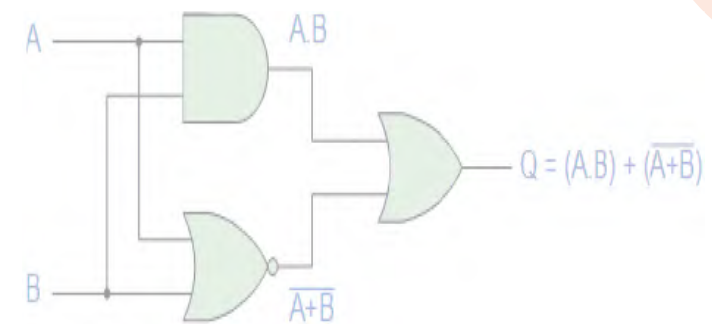


Digital Design and Microprocessor



Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur

Phone : 5639122/6634373/6635046/6630088

Website- <https://www.moecdc.gov.np>

Email- info@moecdc.gov.np

**Technical and Vocational Stream
Learning Resource Material**

**Digital Design and Microprocessor
(Grade 10)
Computer Engineering**



**Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur**

Publisher: Government of Nepal
Ministry of Education, Science and Technology
Curriculum Development Centre
Sanothimi, Bhaktapur

© Publisher
Layout by Khados Sunuwar

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted, in any other form or by any means for commercial purpose without the prior permission in writing of Curriculum Development Centre.

Preface

The curriculum and curricular materials have been developed and revised on a regular basis with the aim of making education objective-oriented, practical, relevant and job oriented. It is necessary to instill the feelings of nationalism, national integrity and democratic spirit in students and equip them with morality, discipline, self-reliance, creativity and thoughtfulness. It is essential to develop linguistic and mathematical skills, knowledge of science, information and communication technology, environment, health and population and life skills in students. It is also necessary to bring the feeling of preserving and promoting arts and aesthetics, humanistic norms, values and ideals. It has become the need of the present time to make them aware of respect for ethnicity, gender, disabilities, languages, religions, cultures, regional diversity, human rights and social values to make them capable of playing the role of responsible citizens with applied technical and vocational knowledge and skills. This learning resource material for computer engineering has been developed in line with the Secondary Level computer engineering Curriculum with an aim to facilitate the students in their study and learning on the subject by incorporating the recommendations and feedback obtained from various schools, workshops, seminars and interaction programs attended by teachers, students, parents and concerned stakeholders.

In bringing out the learning resource material in this form, the contribution of the Director General of CDC Mr. Yubaraj Paudel and members of the subject committee Dr. Baburam Dawadi, Dr. Sarbim Sayami, Mrs. Bibha Sthapit, Mrs. Trimandir Prajapati is highly acknowledged. This learning resource material is compiled and prepared by Mr. Bimal Thapa, Mr. Bikesh Shrestha, Mr. Nabin Maskey. The subject matter of this material is edited by Mr. Badrinath Timsina and Mr. Khilanath Dhamala. Similarly, the language is edited by Mr. Binod Raj Bhatta. CDC extends sincere thanks to all those who have contributed to developing this material in this form.

This learning resource material contains a wide coverage of subject matters and sample exercises which will help the learners to achieve the competencies and learning outcomes set in the curriculum. Each chapter in the material clearly and concisely deals with the subject matters required for the accomplishment of the learning outcomes. The Curriculum Development Centre always welcomes creative and constructive feedback for the further improvement of the material.

Table of Content

Unit	Content	Page No.
1.	Number System and Binary Arithmetic Operations	1-15
2.	Concept of Logic Gates	16-25
3.	Boolean Algebra and Karnaugh Map	26-48
4.	Binary Arithmetic and Combinational Logic	49-77
5.	Introduction to Microprocessor and its Components	78-96

Guidelines to Teachers

A. Facilitation Methods

The goal of this course is to combine the theoretical and practical aspects of the contents needed for the subject. The nature of contents included in this course demands the use of practical or learner focused facilitation processes. Therefore, the practical side of the facilitation process has been focused much. The instructor is expected to design and conduct a variety of practical methods, strategies or techniques which encourage students engage in the process of reflection, sharing, collaboration, exploration and innovation new ideas or learning. For this, the following teaching methods, strategies or techniques are suggested to adopt as per the course content nature and context.

Brainstorming

Brainstorming is a technique of teaching which is creative thinking process. In this technique, students freely speak or share their ideas on a given topic. The instructor does not judge students' ideas as being right or wrong, but rather encourages them to think and speak creatively and innovatively. In brainstorming time, the instructor expects students to generate their tentative and rough ideas on a given topic which are not judgmental. It is, therefore, brainstorming is free-wheeling, non-judgmental and unstructured in nature. Students or participants are encouraged to freely express their ideas throughout the brainstorming time. Whiteboard and other visual aids can be used to help organize the ideas as they are developed. Following the brainstorming session, concepts are examined and ranked in order of importance, opening the door for more development and execution. Brainstorming is an effective technique for problem-solving, invention, and decision-making because it taps into the group's combined knowledge and creative ideas.

Demonstration

Demonstration is a practical method of teaching in which the instructor shows or demonstrates the actions, materials, or processes. While demonstrating something the students in the class see, observe, discuss and share ideas on a given topic. Most importantly, abstract and complicated concepts can be presented into visible form through demonstration. Visualization bridges the gap between abstract ideas and concrete manifestations by utilizing the innate human ability to think visually. This enables students to make better decisions, develop their creative potential, and obtain deeper insights across a variety of subject areas.



Peer Discussion

Peer conversation is a cooperative process where students converse with their peers to exchange viewpoints, share ideas, and jointly investigate subjects that are relevant or of mutual interest. Peer discussion is an effective teaching strategy used in the classroom to encourage critical thinking, active learning, and knowledge development. Peer discussions encourage students to express their ideas clearly, listen to opposing points of view, and participate in debate or dialogue, all of which contribute to a deeper comprehension and memory of the course material. Peer discussions also help participants develop critical communication and teamwork skills by teaching them how to effectively articulate their views, persuasively defend their positions, and constructively respond to criticism.

Peer conversation is essential for professional growth and community building outside of the classroom because it allows practitioners to share best practices, work together, and solve problems as a group. In addition to expanding their knowledge horizon and deepening their understanding, peer discussions help students build lasting relationships and a feeling of community within their peer networks.

Group Work

Group work is a technique of teaching where more than two students or participants work together to complete a task, solve a problem or discuss on a given topic collaboratively. Group work is also a cooperative working process where students join and share their perspectives, abilities, and knowledge to take on challenging job or project. Group work in academic contexts promotes active learning, peer teaching, and the development of collaboration and communication skills. Group work helps individuals to do more together than they might individually do or achieve.

Gallery Walk

Gallery walk is a critical thinking strategy. It creates interactive learning environment in the classroom. It offers participants or students a structured way to observe exhibition or presentation and also provides opportunity to share ideas. It promotes peer-to-peer or group-to-group engagement by encouraging participants to observe, evaluate and comment on each other's work or ideas. Students who engage in this process improve their communication and critical thinking abilities in addition to their comprehension of the subject matter, which leads to a deeper and more sophisticated investigation of the subjects at hand.

Interaction

The dynamic sharing of ideas, knowledge, and experiences between people or things is referred to as interaction, and it frequently takes place in social, academic, or professional settings. It includes a broad range of activities such as dialogue, collaboration or team work, negotiation, problem solving, etc. Mutual understanding, knowledge sharing, and interpersonal relationships are all facilitated by effective interaction. Interaction is essential for building relationships, encouraging learning, and stimulating creativity in both in-person and virtual contexts. Students can broaden their viewpoints, hone their abilities, and jointly achieve solutions to difficult problems by actively interacting with others.

Project Work

Project work is a special kind of work that consists of a problematic situation which requires systematic investigation to explore innovative ideas and solutions. Project work can be used in two senses. First, it is a method of teaching in regular class. The next is: it is a research work that requires planned investigation to explore something new. This concept can be presented in the following figure.



Project work entails individuals or teams working together to achieve particular educational objectives. It consists of a number of organized tasks, activities, and deliverables. The end product is important for project work. Generally, project work will be carried out in three stages. They are:

- Planning
- Investigation
- Reporting

B. Instructional Materials

Instructional materials are the tools and resources that teachers use to help students. These resources/materials engage students, strengthen learning, and improve conceptual comprehension while supporting the educational goals of a course or program. Different learning styles and preferences can be accommodated by the variety of instructional

resources available. Here are a few examples of typical educational resource types:

- Daily used materials
- Related Pictures
- Reference books
- **Slides and Presentation:** PowerPoint slides, keynote presentations, or other visual aids that help convey information in a visually appealing and organized manner.
- **Audiovisual Materials:** Videos, animations, podcasts, and other multimedia resources that bring concepts to life and cater to auditory and visual learners.
- **Online Resources:** Websites, online articles, e-books, and other web-based materials that can be accessed for further reading and research.

Maps, Charts, and Graphs: Visual representations that help learners understand relationships, patterns, and trends in different subjects.

Real-life Examples and Case Studies: Stories, examples, or case studies that illustrate the practical application of theoretical concepts and principles.

C. Assessment

Formative Test

Classroom discussions: Engage students in discussions to assess their understanding of concepts.

Quizzes and polls: Use short quizzes or polls to check comprehension during or after a lesson.

Homework exercises: Assign tasks that provide ongoing feedback on individual progress.

Peer review: Have students review and provide feedback on each other's work.

Summative Test

Exams: Conduct comprehensive exams at the end of a unit or semester.

Final projects: Assign projects that demonstrate overall understanding of the subject.

Peer Assessment

Group projects: Evaluate individual contributions within a group project.

Peer feedback forms: Provide structured forms for students to assess their peers.

Classroom presentations: Have students assess each other's presentations.

Objective Test

Multiple-choice tests: Use multiple-choice questions to assess knowledge.

True/False questions: Assess factual understanding with true/false questions.

Matching exercises: Evaluate associations between concepts or terms.

Portfolio Assessment

Compilation of work: Collect and assess a variety of student work samples.

Reflection statements: Ask students to write reflective statements about their work.

Showcase events: Organize events where students present their portfolios to peers or instructors.

Observational Assessment

Classroom observations: Observe students' behavior and engagement during class.

Performance observations: Assess practical skills through direct observation.

Field trips: Evaluate students' ability to apply knowledge in real-world settings.



Abbreviation

Bits- Binary Digits

XOR- Exclusive OR

XNOR- Exclusive NOR

SOP- Sum Of Product

POS- Product Of Sum

K-Map- Karnaugh Map

ALU- Arithmetic and Logical Unit

CU- Control Unit

PC- Program Counter

BCD- Binary Coded Decimal

ADC- Analog to Digital Converter

DAC- Digital to Analog Converter

IC- Integrated Circuit

RISC- Reduced Instruction Set Computer

CISC- Complex Instruction Set Computer

DSP- Digital Signal Processor

GPU- Graphics Processing Unit

CPU- Central Processing Unit

RAM- Random Access Memory

ROM- Read Only Memory

HDD- Hard Disk Drive

SDD- Solid State Drive

CD- Compact Disk

DVD- Digital Versatile Disk

IR- Instruction Register

MU- Memory Unit

AC- Auxiliary Carry

SP- Stack Pointer

MAR- Memory Address Register

MBR- Memory Buffer Register

ALE- Address Latch Enable

NTR- Interrupt Request



Number system and Binary arithmetic operations

Unit 1

1.1 Introduce Numbering Concept

Computer is an electronic data processing machine. It processes all types of data in the binary digits such as 1 or 0. It also performs mathematical calculation in different format. Number system is the most important in the computer system or day to day activities for counting and calculating any numeric problems. A number system is the series of number used to represent the value of any things. It is also used to count any countable things. The technique to represent and work with numbers is called number system.

1.2 Different Types of Numbering System

There are different types of numbering system. The numbering system is classified into different categories on the basis of number of digits used. Each numbering system has a base also called a radix. The base or the radix defines the number of digits that is used by the numbering system.

Some of the commonly used number system in computer are

1. Binary number system
2. Decimal number system
3. Octal number system
4. Hexadecimal number system

1.2.1 Decimal Numbers

Decimal number system is the commonly used number system even in our daily life. It uses all the digits from 0 to 9. The base of decimal number system is 10 as it uses ten digits from 0 to 9. Decimal number system is written as $(954)_{10}$, $(5490)_{10}$.

1.2.2 Binary Numbers

Binary number system is used by computer system to process data. It uses two digits, 0 and 1. The instructions are represented through electric signals in two two-state system – on and off. On is represented as 1 and off as 0, though 0 is not actually no signal but signal at a lower voltage. Each binary digit is also called a bit. The base

of binary number system is 2 as it uses two digits 0 and 1. Binary number system is written as $(101110)_2$, $(1011)_2$.

1.2.3 Octal Numbers

Octal number system uses all the digits from 0 to 7. The base of octal number system is 8 as it uses eight digits from 0 to 7. Octal number system is written as $(657)_8$, $(4320)_8$.

1.2.4 Hexadecimal Numbers

Hexadecimal number system uses all the digits from 0 to 9 as well as alphabets from A to F. The base of hexadecimal number system is 16 as it uses ten digits from 0 to 9 and 6 alphabets from A to F. Hexadecimal number system is written as $(F6A3)_{16}$, $(CA4E)_{16}$.

Number System	Base	Symbols/Digits Used
Binary	2	0 and 1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

1.3 Number Conversion

The process of converting a numbering system from one base to another is called number conversion. Binary numbering system can be converted to decimal numbering system and vice versa. Similarly all other numbering system can be converted to one another.

1.3.1 Decimal Integer to Binary and Binary to Decimal

Decimal Integer to Binary

Steps to convert Decimal integer into its Binary equivalent:

- Divide the decimal integer by 2.
- Take the remainder and record it on the side.
- Divide the quotient by 2.
- Repeat until the decimal integer cannot be divided further or remainder is 0.
- Record the remainders in reverse order and you get the resultant binary number.

Example

Convert the Decimal number 56 into its Binary equivalent.

2	56	Remainder
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$$(56)_{10} = (111000)_2$$

Binary to Decimal

Steps to convert Binary into its Decimal equivalent:

- Start from the rightmost bit of integer part of binary number.
- Take the rightmost bit of integer part of binary number and multiply by 2^n where n is the current position beginning at 0 and increasing by 1 each time.
- Sum each terms of product until all bits have been used. The result will be the equivalent integer decimal number.

Example

Convert the Binary number 11011 to its Decimal equivalent.

Solution: $(11011)_2$

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

$$1 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$$

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

$$27$$

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

1.3.2 Decimal Fractions to Binary Conversion

Steps to convert Decimal fractions into its Binary equivalent:

- Multiply the fractional part of the decimal number by 2.
- Integral part of resultant decimal number will be first digit of fractional binary number.
- Repeat step a using only fractional part of decimal number obtained from step b.

- d. Integral part of resultant decimal number will be second digit of fractional binary number.
- e. Repeat the above steps for upto five iterations.

Example

Convert the Decimal fractions 79 into its Binary equivalent.

Fractional number	Multiplier	Result	Answer
.79	2	1.58	1
.58	2	1.16	1
.16	2	0.32	0
.32	2	0.64	0
.64	2	1.28	1

$$(.79)_{10} = (11001)_2$$

Convert Decimal number (57. 47) into its Binary equivalent.

$$(57. 47)_{10} = (?)_2$$

Converting integer part of the given Decimal number

2	57	Remainder
2	28	1
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$$(57)_{10} = (111001)_2$$

Converting fractional part of the given Decimal number

Fractional number	Multiplier	Result	Answer
.47	2	0.94	0
.94	2	1.88	1
.88	2	1.76	1

.76	2	1.52	1
.52	2	1.04	1

$$(.47)_{10} = (01111)_2$$

$$(57).47_{10} = (111001.01111)_2$$

1.3.3 Octal to Decimal and Decimal to Octal Conversion

Octal to Decimal

Steps to convert Octal Integer into its Decimal Equivalent

- Start from the rightmost number of integer part of octal number.
- Take the rightmost number of integer part of octal number and multiply by 8^n where n is the current position beginning at 0 and increasing by 1 each time.
- Sum each terms of product until all numbers have been used. The result will be the equivalent integer decimal number.

Example

Convert the octal number 502 to its Decimal equivalent.

Soln: $(502)_8$

$$5 \times 8^2 + 0 \times 8^1 + 2 \times 8^0.$$

$$5 \times 64 + 0 \times 8 + 2 \times 1.$$

$$320 + 0 + 2.$$

$$322.$$

$$(502)_8 = (322)_{10}$$

Decimal to Octal

Steps to Convert Decimal Integer into its Octal Equivalent

- Divide the decimal number by 8.
- Take the remainder and record it on the side.
- Divide the quotient by 8.
- Repeat until the decimal number cannot be divided further.
- Record the remainders in reverse order and you get the resultant octal number.

Example

Convert the Decimal number 445 into its octal equivalent.

Soln: $(445)_{10}$

8	445	Remainder
8	54	5
8	6	6
	0	6

$$(445)_{10} = (665)_8$$

1.4 Introduction to 1's Complement

Complements are used in the digital computers in order to perform and simplify the subtraction operation and for the logical manipulations. To simplify the binary subtraction operation complement of numbers are used. In computer system 1's complement and 2's complement are used for binary subtraction.

1's complement of the given binary number can be obtained by subtracting it from its higher order bits. It can also be obtained by inverting all bits in it, i.e., transforming the 0 bit to 1 and the 1 bit to 0.1s'. Complement is simply a Bitwise NOT gate.

1's complement of 1010 is 0101. (Inverting 1 to 0 and 0 to 1)

Or, $1111 - 1010 = 0101$ (Here 1111 is the higher order bits of 1010)

1's complement of 11001 is 00110. (Inverting 1 to 0 and 0 to 1)

Or, $11111 - 11001 = 00110$ (Here 11111 is the higher order bits of 11001)

Subtraction of two Binary Numbers using 1's Complement

Algorithm

1. Make equal number of bits if the bits are not equal.
2. Calculate 1's complement of the subtrahend, that is the number that is to be subtracted.
3. Add 1's complement with the minuend, that is the main number.
4. a. If the result contains the overflow bit, the sign is positive and the final result is obtained by removing the overflow bit and adding it with the remaining number.
b. If the result does not contain overflow bit, the sign is negative and the final result is obtained by calculating 1's complement of the number.

Example 1: Subtract 1101-110 by using 1's Complement

Step 1: Make equal number of bits.

1 1 0 1 (Minuend)

0 1 1 0 (Subtrahend)

Step 2: 1's complement of 0110 is 1001.

Step 3: Add 1's complement with the minuend.

$$1101 + 1001 = 10110$$

Step 4: The result contains the overflow bit, so the sign is positive and the final result is obtained by removing the overflow bit and adding it with the remaining number.

$$0110 + 1 = 0111$$

$$\text{Hence, } 1101 - 110 = 111$$

Example 2: Subtract 10110-11101 by using 1's Complement.

Step 1: Number of bits are already equal.

10110 (Minuend)

11101 (Subtrahend)

Step 2: 1's complement of 11101 is 00010.

Step 3: Add 1's complement with the minuend.

$$10110 + 00010 = 11000$$

Step 4: The result does not contain overflow bit, so the sign is negative and the final result is obtained by calculating 1's complement of the resulting number.

1's complement of 11000 is 00111

$$\text{Hence } 10110 - 11101 = -111$$

1.5 Introduction to 2's Complement

2's complement of a binary number can be obtained by simply inverting the given number and add 1 to the least significant bit (LSB) of given result. So, in other words, 2's complement of a binary number is obtained by adding 1 to the 1's complement of the binary number.

2's complement of 1010

1's complement of 1010 is 0101

$$2's \text{ complement of } 1010 \text{ is } 0101 + 1 = 0110$$

2's complement of 11001

1's complement of 11001 is 00110

2's complement of 11001 is $00110+1=00111$

Subtraction of two Binary Numbers using 2's Complement

Algorithm

1. Make equal number of bits if the bits are not equal.
2. Calculate 2's complement of the subtrahend, that is the number that is to be subtracted.
3. Add 2's complement with the minuend, that is the main number.
- 4.a. If the result contains the overflow bit, the sign is positive and the final result is obtained by removing the overflow bit.
- b. If the result does not contain overflow bit, the sign is negative and the final result is obtained by calculating 2's complement of the number.

Example 1: Subtract 1101-101 by using 2's complement.

Step 1: Make equal number of bits.

1101 (Minuend)

0101 (Subtrahend)

Step 2: 2's complement of 0101 is 1011

Step 3: Add 2's complement with the minuend.

$$1101 + 1011 = 11000$$

Step 4: The result contains the overflow bit, so the sign is positive and the final result is obtained by removing the overflow bit that is 1000

$$\text{Hence } 1101-101 = 1000$$

Example 2: Subtract 10110-11011 by using 2's Complement.

Step 1: Number of bits are already equal.

10110 (Minuend)

11011 (Subtrahend)

Step 2: 2's complement of 11011 is 00101.

Step 3: Add 2's complement with the minuend.

$$10110 + 00101 = 11011$$

Step 4: The result does not contain overflow bit, so the sign is negative and the final result is obtained by calculating 2's complement of the resulting number.

2's complement of 11011 is 00101

Hence $10110 - 11011 = -101$

1.6 Introduction Binary Addition

Binary addition is similar to decimal numbers addition. The rules of binary addition are:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	0 with carry 1

In last case, a binary addition of $1 + 1 = 10$ i.e. 0 is written in the given column and a carry of 1 over to the next column.

Steps of Binary Addition

1. Align the numbers that you have to add in the same way you align while adding decimal numbers.
2. Start with the two numbers from the right end position.
3. Add the numbers by following the rules of binary addition as given above.
4. After adding first column from the right position, add second column and so on.
5. Repeat the above steps until all the columns are finished.

Add Binary Number $(111)_2$ and $(110)_2$

1. To add these two numbers, let us first align these two numbers.

$$\begin{array}{r} 111 \\ + 110 \\ \hline \end{array}$$

2. Now, we will start adding two numbers from the right end position.
3. Adding 1 and 0 gives 1. There is not any carry.

$$\begin{array}{r} 111 \\ + 110 \\ \hline \end{array}$$

1

4. Now add two numbers from second column from the right end position. Adding 1 and 1 gives 10. So write 0 in the answer and take 1 as carry.

Carry 1
1 1 1
+ 1 1 0
0 1

5. Now add numbers from first column along with carry. Adding 1+1+1 gives 11.

1
1 1 1
+ 1 1 0
1 1 0 1

So answer is 1101

Examples

1111+111
1111
+111
10110

Hence, 1111 + 111 = 10110

101+110
1 0 1
+ 1 1 0
1 0 1 1

Hence, 101 + 110 = 1011

1.7 Introduction Binary Subtraction

Subtracting binary number is very similar to subtract decimal numbers. The rules for subtracting binary numbers are

A	B	A-B
0	0	0
1	0	1
1	1	0

0	1	1 with borrow 1 from left column
---	---	----------------------------------

Subtract binary number $(100)_2$ and $(10)_2$

- To subtract these two numbers, let us first align these two numbers.**

$$\begin{array}{r} 100 \\ - 10 \\ \hline \end{array}$$

- Now, we will start subtracting two numbers from the right end position.**

Subtracting 0 and 0 gives 0.

$$\begin{array}{r} 100 \\ - 10 \\ \hline 0 \end{array}$$

- Now subtract two numbers from second column from the right end position. Subtract 0 and 1.**

1 cannot be subtracted from 0. So we borrow 1 from the near left column. Then it becomes 10 and we calculate $10-1$ which gives 1.

$$\begin{array}{r} 100 \\ - 10 \\ \hline 10 \end{array}$$

- Now there is not any number in the left most column because 1 has already gone as borrow and no digits remain to subtract here.**

So answer is 10.

Examples

$$\begin{array}{r} 1011 \\ - 110 \\ \hline 101 \end{array}$$

Hence, $1011 - 110 = 101$.

$$\begin{array}{r} 10110 \\ - 101 \\ \hline 10001 \end{array}$$

Hence, $10110 - 101 = 10001$.

1.8 Introduction Binary Multiplication

Binary multiplication is similar to decimal multiplication. It is simpler than decimal multiplication because only 0s and 1s are involved. The rules of binary multiplication

are:

A	B	A×B
0	0	0
0	1	0
1	0	0
1	1	1

Example

11101×101

			1	1	1	0	1
				×	1	0	1
			1	1	1	0	1
	0		0	0	0	0	×
1	1		1	0	1	×	×
1	0	0	1	0	0	0	1

Hence $11101 \times 101 = 10010001$

$$11001 \times 110$$

				1	1	0	0	1
					×	1	1	0
				0	0	0	0	0
		1		1	0	0	1	×
	1	1		0	0	1	×	×
1	0	0		1	0	1	1	0

Hence $11001 \times 110 = 10010110$

Exercise

Choose the correct answer from the given alternatives.

1. Which of the following number system has base 2?
a. Decimal b. Binary c. Octal d. Hexadecimal
2. Which of the following number system has base 16?
a. Decimal b. Binary c. Octal d. Hexadecimal
3. Bit stands for
a. Binary International Technology b. Binary Information Technology
c. Binary Digit d. Binary System
4. Decimal number system uses digits.
a. 9 b. 8 c. 10 d. 16
5. 1's complement of 10111 is
a. 1000 b. 10101 c. 01001 iv. 01000
6. 2's complement of 11011 is
a. 00101 b. 00100 c. 101 d. 100
7. $10111+10100$ is
a. 101010 b. 101011 c. 10111 d. 100011
8. Octal number system uses digits from
a. 0 to 10 ii. 0 to 8 c. 0 to 7 d. 0 to 9
9. 10101×110 is
a. 101010 b. 101011 c. 10111 d. 1111110
10. The binary equivalent of $(133)_{10}$ is
a. 11001010 b. 10000101 c. 1100111 d. 1111100
11. What is the octal equivalent of binary number 110?
a. 6 b. 7 c. 4 d. 5

12. Which of the following is most suitable for binary subtraction?
- 9's complement
 - 10's complement
 - 1's complement
 - 5's complement

Write short answer to the following questions.

- What is number system? Explain the different types of number system.
- Define 1's complement and 2's complement with examples.
- Convert the given number system as indicated.
 - $(89)_{10} = (?)_2$
 - $(280)_{10} = (?)_2$
 - $(800.417)_{10} = (?)_2$
 - $(11010)_2 = (?)_{10}$
 - $(111001)_2 = (?)_{10}$
 - $(11011)_2 = (?)_{10}$
 - $(654.432)_{10} = (?)_2$
 - $(310.2654)_{10} = (?)_2$
 - $(854.320)_{10} = (?)_2$
 - $(653)_8 = (?)_{10}$
 - $(327)_8 = (?)_{10}$
 - $(1543)_8 = (?)_{10}$
 - $(943)_{10} = (?)_8$
 - $(3098)_{10} = (?)_8$
 - $(813)_{10} = (?)_8$
- Perform the following binary addition:
 - $(10110)_2 + (1001)_2$
 - $(10111)_2 + (10101)_2$
 - $(10100)_2 + (11011)_2$
 - $(11010)_2 + (11010)_2$
 - $(101111)_2 + (111111)_2$
 - $(111101)_2 + (101110)_2$
- Perform the following binary subtraction:
 - $(11010)_2 - (10010)_2$
 - $(11010)_2 - (1010)_2$
 - $(110100)_2 - (1101)_2$
 - $(1101100)_2 - (101011)_2$
 - $(11010111)_2 - (10111)_2$
 - $(1110011)_2 - (101010)_2$
- Perform the following binary multiplication:
 - $(1110)_2 \times (11)_2$
 - $(1011)_2 \times (10)_2$
 - $(1010111)_2 \times (101)_2$
 - $(110110)_2 \times (111)_2$
 - $(110110)_2 \times (110)_2$
 - $(1011100)_2 \times (100)_2$
- Perform the following binary simplification:
 - $(111 \times 10)_2 + (11)_2$
 - $(10 \times 101)_2 + (1011)_2$
 - $(10101 + 110)_2 - (1100)_2$
 - $(110011)_2 \times (101)_2$
- Perform the following binary subtraction using 1's complement and 2's complement.

- a. 1101-110 b. 101-1101
- c. 1101-11 d. 100-1101

9. Subtract 3 from 7 using 1's complement and 2's complement.

Write long answer to the following questions.

1. Explain the different types of number system with examples.
2. Write the steps to convert decimal integer into binary with an example.

Project Work

1. Explain on chart paper “number system conversion” and paste in your class room.
2. Prepare a presentation file on a topic “Different types of number system” and demonstrate in your class as a group work.
3. Prepare a Power Point Presentation file on a topic “**Binary calculation**”



Concept of Logic Gates

2.1 Notations

Boolean Expression

A boolean expression is an expression that produces results in terms of boolean value, that is, in a value of either true or false. A Boolean expression is a logical statement that is either TRUE or FALSE. A Boolean expression is the combination of the Boolean constants, Boolean-typed variables and Boolean valued operators. The example of Boolean expression is $F=AB+A'C+BC'$

Boolean Function

Boolean functions are mathematical expressions involving Boolean variables that take on values of true or false, typically represented as 1 and 0, respectively. These functions are fundamental in digital logic design, computer science, and engineering, as they form the basis of digital circuits and various computational processes.

2.2 Concept of Gate and Truth Table

Logic gates are the basic building blocks of electronic digital circuits. Logic gates will accept one or more than one input and produces only one output and are based on Boolean algebra. . Some circuits may have only a few logic gates, while others, such as microprocessors, may have millions of them. Logic gates are commonly used in Integrated Circuits (IC). There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

A truth table generally shows the working mechanism of digital circuits. It represents a table having all combinations of possible inputs and their corresponding result. A truth table is a tabular representation of all the combinations of values for inputs and their corresponding outputs. It is a mathematical table that shows all possible outcomes that would occur from all possible scenarios. Truth tables are usually used for logic problems as in Boolean algebra and electronic circuits.

2.2.1 Inverter

Inverter gate is also known as NOT gate. It is the type of logic gate which accepts only one input and produces one output. The output produced by the Inverter gate will be the reverse of the input. The output produced will be false if the input is true, and true, if the input is false.

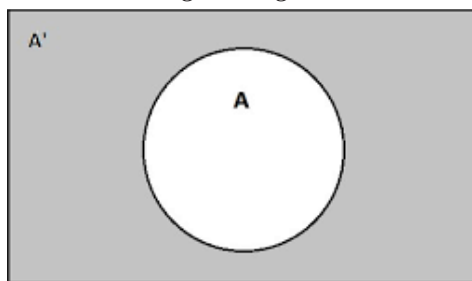
The Boolean expression of Inverter gate is $Z=A'$.

A	$Z=A'$
T	F
F	T

Truth table



Logical diagram



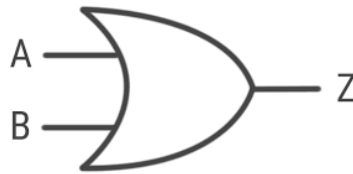
Venn Diagram

2.2.2 OR Gate

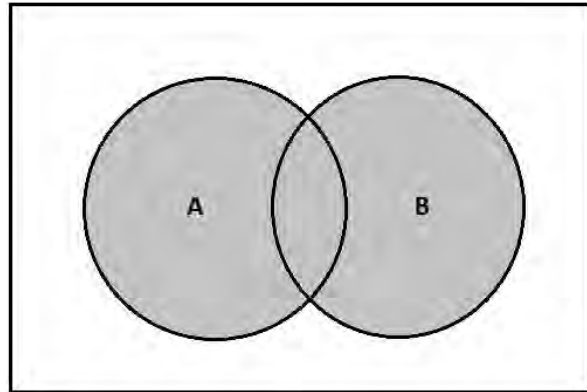
It is the type of logic gate which accepts two or more than two inputs and produces only one output. The result produced by the OR gate will be true if at least one input is true, otherwise false. That means in OR gate, the result will only be false if all inputs are false. The Boolean expression of OR gate is $Z=A+B$.

A	B	$Z=A+B$
F	F	F
F	T	T
T	F	T
T	T	T

Truth table



Logical diagram



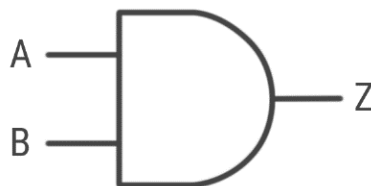
Venn Diagram

2.2.3 AND Gate

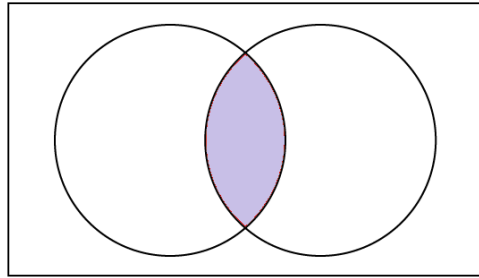
It is the type of logic gate which accepts two or more than two inputs and produces only one output. The result produced by the AND gate will be true if all the inputs are true, otherwise false. If one input is false, the result produced by the AND gate will be false. The Boolean expression of AND gate is $Z=A.B$.

A	B	$Z=A.B$
F	F	F
F	T	F
T	F	F
T	T	T

Truth table



Logical diagram



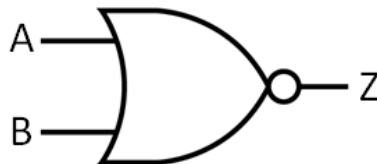
Venn Diagram

2.2.4 NOR Gate

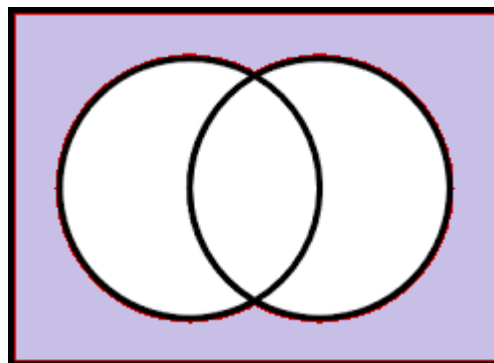
It is the type of logic gate which accepts two or more than two inputs and produces only one output. The result produced by the NOR gate will be the compliment of Inverter gate. The result produced by the NOR gate will be true if all the inputs are false, otherwise false. The Boolean expression of NOR gate is $Z=(A+B)'$

A	B	$Z=(A+B)'$
F	F	T
F	T	F
T	F	F
T	T	F

Truth table



Logical diagram



Venn diagram

2.2.5 NAND Gate

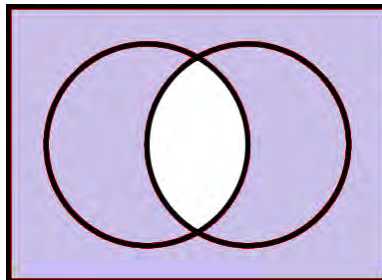
It is the type of logic gate which accepts two or more than two inputs and produces only one output. The result produced by the NAND gate will be the compliment of AND gate. The result produced by the NAND gate will be true if at least one input is false, otherwise false. The Boolean expression of AND gate is $Z=(A.B)'$.

A	B	$Z=(A.B)'$
F	F	T
F	T	T
T	F	T
T	T	F

Truth table



Logical diagram



Venn diagram

2.2.6 Universal Gates

Universal gates are the logic gates by using which all other logic gates can be designed. NAND and NOR gates are known as universal gates because by using NAND gate only and NOR gate only, all other logical gates can be designed. This is beneficial because NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

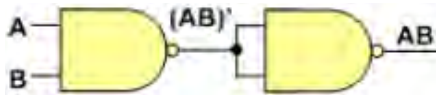
NAND Gate as an Universal Gate

NAND is known as universal gate because by using only these NAND gates, AND, OR, and Inverter gates can be designed.

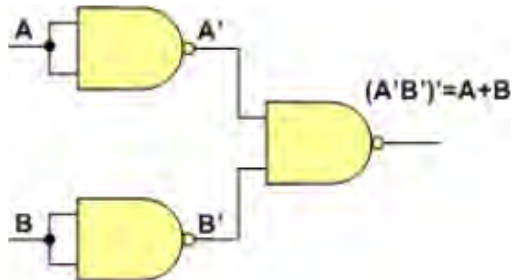
a. Designing Inverter gate using only NAND Gate



b. Designing AND gate using only NAND Gates



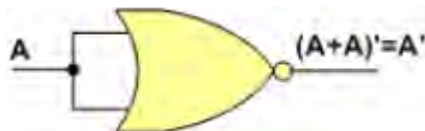
c. Designing OR using only NAND Gates



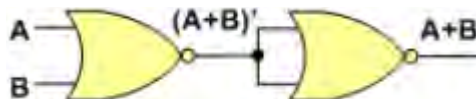
NOR gate as an Universal Gate

NOR is known as universal gate because by using only these NOR gates, AND, OR, and NOT gates can be designed.

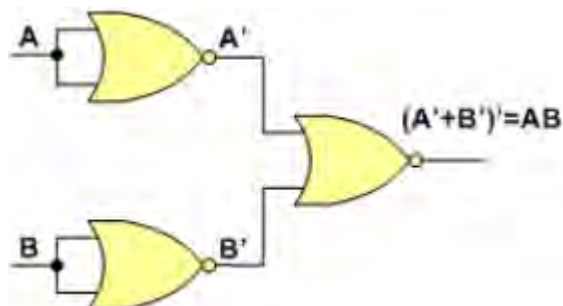
a. Designing Inverter gate using only NOR Gate



b. Designing OR using only NOR Gates



c. Designing AND Using only NOR Gates

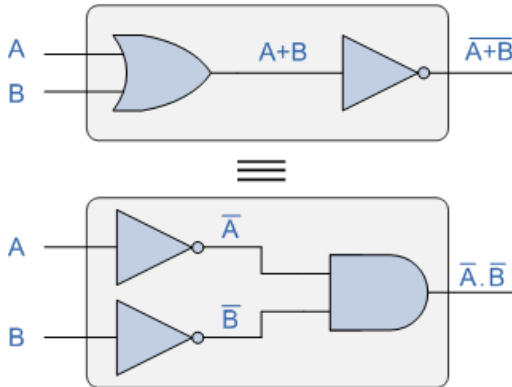


2.3 De Morgan's Theorem

Theorem 1

The complement of the sum of the two or more Boolean variables is equal to the product of its individual complement.

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



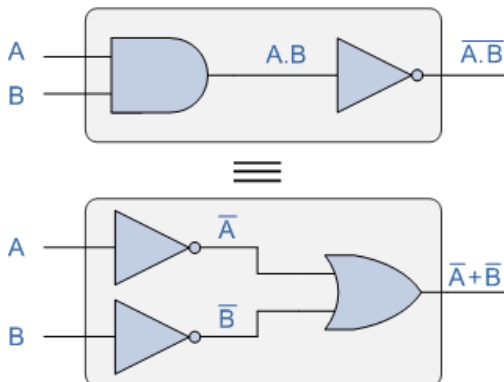
Proof using truth table

A	B	A+B	$(A+B)'$	A'	B'	$A' \cdot B'$
F	F	F	T	T	T	T
F	T	T	F	T	F	F
T	F	T	F	F	T	F
T	T	T	F	F	F	F

Theorem 2

The complement of the product of two or more Boolean variables is equal to the sum of its individual complement.

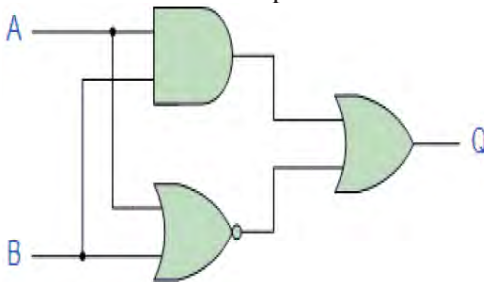
$$(A \cdot B)' = A' + B'$$



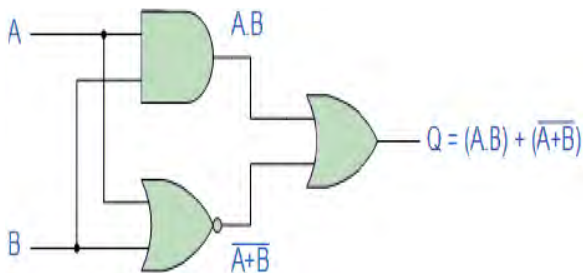
Proof using truth table

A	B	$A.B$	$(A.B)'$	A'	B'	$A'+B'$
F	F	F	T	T	T	T
F	T	F	T	T	F	T
T	F	F	T	F	T	T
T	T	T	F	F	F	F

1. Find the Boolean expression and truth table of the following logic circuit.



Boolean expression of the logic circuit is calculated as follows.

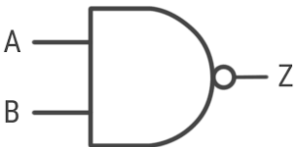

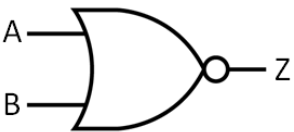



Truth table of the Boolean expression is shown as below.

A	B	$A.B$	$A+B$	$(A+B)'$	$(A.B)+(A+B)'$
F	F	F	F	T	T
F	T	F	T	F	F
T	F	F	T	F	F
T	T	T	T	F	T

Exercise

Choose the correct answer from the given alternatives.

1. Which of the following is known as universal gate?
a. OR b. NOT c. NAND d. AND
2. Which of the following is the statement of De Morgan's theorem?
a. $A.A'=0$ b. $(A.B)' = A' + B'$ c. $A+1=1$ d. $A+B=B+A$
3. Which of the following gate accepts only one input?
a. AND b. OR c. Inverter d. NOR
4. Which of the following gate produce the result True if all inputs are True, otherwise False?
a. XOR b. AND c. NOT d. NAND
5. Which of the following is the Boolean expression of AND gate?
a. $Z=A+B$ ii. $Z=A'$ c. $Z=A-B$ d. $Z=A.B$
6. The compliment of OR gate and Inverter forms the gate.
a. NOTOR b. NOR c. NAND d. NORT
7. Which of the following is the logical diagram of NAND gate?
a.  b. 
c.  d. 
8. Which of the following is the Boolean expression of NOR gate?
a. $F=A.B$ b. $F=A+B$ c. $F=(A+B)'$ d. $F=A'$

9. Which logic gate produces a high output only when all its input are high?
- a. OR b. NOT c. AND d. NAND

Write short answer to the following questions.

1. What is universal gate? Why NAND and NOR are called universal gates?
2. List the different types of universal gates. Make OR gate using NAND gate.
3. Write the truth table and draw the logical diagram for the following Boolean expression.
 - a. $F = C + (BC)'$
 - b. $F = AB + A'C + BC'$
 - c. $F = (A.B) + (A+B)'$
 - d. $AB(A + B)(B + B')$
 - e. $AB + A(B+C) + B(B+C)$

Write long answer to the following questions.

1. Explain different types of logic gates with truth table, logical diagram and Venn diagram.
2. State and prove De Morgan's theorem for two input variables.

Project Work

1. Prepare a presentation file on a topic "Logic Gates" and demonstrate in your class as a group work.
2. Prepare a Power Point Presentation file on a topic "DeMorgan's Theorem"



Unit 3

Boolean Algebra and Karnaugh Map

3.1 Boolean Relationships Simplifications

The most practical use of Boolean algebra is in the simplification of logic circuits. The logic circuit's function is converted into Boolean form and different laws of Boolean algebra are applied. This will convert the expression in simpler form by reducing the number of terms and/or arithmetic operations. The simplified expression may be converted back into circuit form for a logic circuit performing the same function with fewer components. If the equivalent function is achieved with fewer components, the reliability of output increases, output will be achieved quickly and cost of development also decreases.

Boolean Algebra

Boolean algebra is a form of mathematics developed by English mathematician George Boole. Boole created a system by which certain logical statements can be written in mathematical terms. Boolean algebra is used to analyze and simplify the digital logic circuits. Boolean logic is a form of algebra where all values are represented in the form of either True or False. These values of True and False are used to test the conditions that selection and iteration are based around. Boolean logic is referred to as digital logic which is considered as the heart of all the modern digital computers.

Boolean Values

In computer science, a boolean or bool is a data type that has two possible values, that is either True, or False. True is denoted by 1 and False is denoted by 0. It is named after the English mathematician and logician George Boole, whose algebraic and logical systems are used in all modern digital computers.

Boolean Expression

A Boolean expression is an expression that results in a Boolean value, that is, in a value of either true or false. A Boolean expression is a logical statement that is either TRUE or FALSE. A Boolean expression may be composed of a combination of the Boolean constants true or false, Boolean-typed variables and Boolean valued operators. The example of Boolean expression is $F = AB + A'C + BC'$

Digital Design and Microprocessor/Grade 10

Laws of Boolean Algebra

The main aim of any logic design is to simplify the logic as much as possible so that the final design will become easy. In order to simplify the logic, the Boolean equations and expressions representing that logic must be simplified. Different laws and theorems are proposed to simplify the Boolean equations and expression, Using these laws and theorems, it becomes very easy to simplify or reduce the logical complexities of any Boolean expression or function. The different laws of Boolean algebra are:

1. Associative Law

The output produced by the Boolean expression will be same if the Boolean variables are associated separately with the help of parenthesis.

- $(A+B)+C=A+(B+C)$
- $(A.B).C=A.(B.C)$

A	B	C	A+B	(A+B)+C	B+C	A+(B+C)
F	F	F	F	F	F	F
F	F	T	F	T	T	T
F	T	F	T	T	T	T
F	T	T	T	T	T	T
T	F	F	T	T	F	T
T	F	T	T	T	T	T
T	T	F	T	T	T	T
T	T	T	T	T	T	T

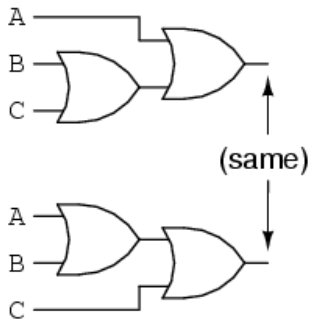
Proof of associative law for AND

A	B	C	A.B	(A.B).C	B.C	A.(B.C)
F	F	F	F	F	F	F
F	F	T	F	F	F	F
F	T	F	F	F	F	F
F	T	T	F	F	T	F
T	F	F	F	F	F	F

T	F	T	F	F	F	F
T	T	F	T	F	F	F
T	T	T	T	T	T	T

Associative property of addition

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



2. Commutative Law

The output produced by the Boolean expression will be same if the position of Boolean variables are interchanged.

- $A + B = B + A$
- $A \cdot B = B \cdot A$

Proof of commutative law for OR

A	B	A+B	B+A
F	F	F	F
F	T	T	T
T	F	T	T
T	T	T	T

Proof of commutative law for AND

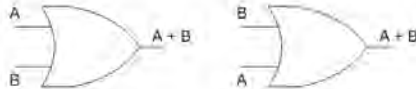
A	B	A.B	B.A
F	F	F	F
F	T	F	F
T	F	F	F
T	T	T	T

Hence commutative law is verified.

Commutative Law

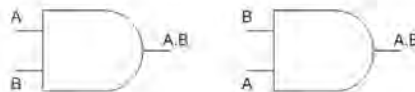
- Commutative Law for Addition

$$A + B = B + A$$



- Commutative Law for Multiplication

$$A.B = B.A$$



3. Distributive Law

Distributive law permits the multiplying or factoring out of an expression. Distributive law states that the multiplication of two variables and adding the result with a variable will result in same value as multiplication of addition of the variable with individual variables.

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

A	B	C	BC	A+BC	A+B	A+C	(A+B).(A+C)
F	F	F	F	F	F	F	F
F	F	T	F	F	F	T	F
F	T	F	F	F	T	F	F
F	T	T	T	T	T	T	T
T	F	F	F	T	T	T	T
T	F	T	F	T	T	T	T
T	T	F	F	T	T	T	T
T	T	T	T	T	T	T	T

The addition of two variables and multiplying the result with a variable will result in same value as addition of multiplication of the variable with individual variables.

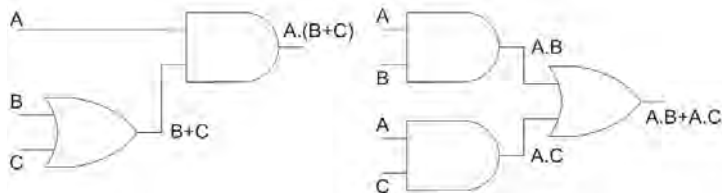
$$A.(B+C)=A.B+A.C$$

A	B	C	B+C	A.(B+C)	A.B	A.C	A.B+A.C
F	F	F	F	F	F	F	F
F	F	T	T	F	F	F	F
F	T	F	T	F	F	F	F
F	T	T	T	F	F	F	F
T	F	F	F	F	F	F	F
T	F	T	T	T	F	T	T
T	T	F	T	T	T	F	T
T	T	T	T	T	T	T	T

Hence distributive law is verified.

Distributive Law

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



4. Identity Law

The Boolean variables when anded with 1 or ored with 0 will be the Boolean variable itself.

- $A.1=A$
- $A+0=A$

A	A.1
0	0
1	1

A	A+0
0	0
1	1

5. Idempotent Law

The sum of the Boolean variables is equal to the Boolean variable itself.

The product of the Boolean variables is equal to the Boolean variable itself.

- $A+A=A$

- $A.A=A$

A	$A+A$
0	0
1	1

A	$A.A$
0	0
1	1

6. Complement Law

The sum of the Boolean variable and its complement is always 1.

The product of the Boolean variable and its complement is always 0.

The double complement of the Boolean variable is equal to the Boolean variable itself.

- $A+A'=1$

- $A.A'=0$

- $(A')'=A$

A	A'	$A+A'$
0	1	1
1	0	1

A	A'	$A.A'$
0	1	0
1	0	0

A	A'	$(A')'$
0	1	0
1	0	1

7. Absorption Law

This law enables a reduction in a complicated expression to a simpler one by absorbing like terms. Absorption law states that one boolean variables can be absorbed by other boolean variables.

- $A.(A+B) = A$
- $A + AB = A$

A	B	A+B	A.(A+B)
F	F	F	F
F	T	T	F
T	F	T	T
T	T	T	T

A	B	A.B	A+A.B
F	F	F	F
F	T	F	F
T	F	F	T
T	T	T	T

Hence absorption law is verified.

3.2 Sum of Product

Sum of Product is generally termed as SOP. The Sum of Product form represent a Boolean expression where several product terms involving AND operations are added(ORed) up. This is the most popular form used in the design of digital systems. In SOP format, each of the elements is a combination (ANDing) of some variables or their complement and all of them are combined by means of OR operator. In other words, SOP form is a combination of product(AND) terms that are added(OR) together.

For example the sum of product form is represented as

$$F=A'B+AB+AB'$$

Min Term

The term that is True for a minimum number of combination of inputs is referred as Min Term. AND gate also gives True only, when all of the inputs are true. So we can say min

terms are AND of input combinations. When a Boolean function or logical expression is expressed in the SSOP (Standard Sum of Product) Form or canonical form, then each term of the expression is called a minterm.

A minterm is represented as m_i , where, i is an integer in between 0 and $2^{(n-1)}$. Here, " n " is the number of variables in the expression. Therefore, minterms can be denoted as $m_0, m_1, m_2, m_3, \dots$. In a minterm, a variable will appear in its complemented form if its value is equal to 0 and the variable will appear in its un-complimented form if its value is equal to 1.

In the following table, there are three inputs. So there will be 8 different combinations of inputs. Each combination has a min terms denoted by small m and its decimal combination number written in subscript. Each of these min terms will be only true for the specific input combination.

X	Y	Z	Min Term (m)
0	0	0	$m_0 = X'Y'Z'$
0	0	1	$m_1 = X'Y'Z$
0	1	0	$m_2 = X'Y Z'$
0	1	1	$m_3 = X'YZ$
1	0	0	$m_4 = XY'Z'$
1	0	1	$m_5 = XY'Z$
1	1	0	$m_6 = XYZ'$
1	1	1	$m_7 = XYZ$

Types of Sum Of Product (SOP) Forms

Different forms of Sum of Product are

- Canonical SOP Form
- Non-Canonical SOP Form
- Minimal SOP Form

Canonical SOP Form

Canonical SOP is the standard form of sum of product. It is formed by adding (ORing) the minterms of the function for which the output is True. Hence, canonical SoP form is also called as sum of min terms form.

Canonical SOP expression is represented by summation sign \sum and minterms in the braces for which the output is True.

For example, let us consider a truth table as given below.

X	Y	Z	F	Min Term (m)
0	0	0	0	m0=X'Y'Z'
0	0	1	0	m1=X'Y'Z
0	1	0	0	m2=X'Y Z'
0	1	1	1	m3=X'YZ
1	0	0	1	m4=XY'Z'
1	0	1	0	m5=XY'Z
1	1	0	1	m6=XYZ'
1	1	1	1	m7=XYZ

For this function the canonical SOP expression is written as

$$F = \sum (m_3, m_4, m_6, m_7)$$

which means that the function is True for the min terms {3, 4, 6, 7}.

By expanding the summation we get,

$$F = m_3 + m_4 + m_6 + m_7$$

Now putting min terms in the expression

$$F = X'YZ + XY'Z' + XYZ' + XYZ$$

Canonical form contains all inputs which can be either in complemented or non-complemented in its product terms.

Non-Canonical SOP Form

Non-Canonical SOP Form is the non-standardized form of SOP expressions. The product terms are not the min terms but they are simplified by using laws of Boolean algebra. In this form each product term may or may not contain all the variables that are used in the function. Let's take the above function in canonical form as an example.

$$F = X'YZ + XY'Z' + XYZ' + XYZ$$

$$F = X'YZ + XY'Z' + XY(Z' + Z)$$

$$F = X'YZ + XY'Z' + XY(1)$$

$$F = X'YZ + XY'Z' + XY$$

In this form the first and second term contain all the variables of the function and the last term contains only two variables. This expression is still in Sum of Product form but it is non-canonical or non-standardized form.

Minimal SOP Form

Minimal SOP Form is the simplified expression of the sum of the product. It is a type of non-canonical form. This is made simplified by using different laws of Boolean expression and it can simply be done by using K-map (Karnaugh map). Minimal SOP form is preferred because it uses the minimum number of gates and input lines. It is commercially beneficial because of its small size, fast speed, and low fabrication cost.

Let's take an example of the function given above in canonical form.

A	B	C	F	Min Term (m)
0	0	0	0	$m_0 = A'B'C'$
0	0	1	1	$m_1 = A'B'C$
0	1	0	1	$m_2 = A'BC'$
0	1	1	1	$m_3 = A'BC$
1	0	0	0	$m_4 = AB'C'$
1	0	1	1	$m_5 = AB'C$
1	1	0	0	$m_6 = ABC'$
1	1	1	0	$m_7 = ABC$

The canonical SOP of the above function is

$$F = A'B'C + A'BC' + A'BC + AB'C$$

The K map is given below.

F		BC			
		00	01	11	10
A	0	0	1	1	1
	1	0	1	0	0

According to the K-map, the output expression will be

$$F=B'C+A'B$$

This is the most simplified expression for the above function. This expression requires only two 2-input AND gates and one 2-input OR gate. The canonical form needs four 3-input AND gates and one 4-input OR gate, which is relatively more costly than minimal form implementation.

(The detail of K-map is explained in the topic K-map below)

3.3 Product of Sum(POS)

Product of the sum is generally termed as POS. The Product of Sum form refers to a Boolean expression where several products (AND) of the different sum (OR) of inputs are taken. These are not arithmetic product and sum but they are logical Boolean AND and OR respectively. In POS form, first Add (OR) of multiple variables are performed and then product (AND) operation between them are done.

For example The Product of Sum form is represented as

$$F = (A'+B')(A+B)$$

Max Term

The maxterm is referred to as the term that is true for a maximum number of combinations of inputs. OR gate also provides false for just one input combination. Thus Max term is OR of any complemented otherwise non-complemented inputs.

A max term is often represented as M_i , where, i is an integer in between 0 and $2^{(n-1)}$. Here, "n" is the number of variables in the expression. Therefore, max terms can be denoted as $M_0, M_1, M_2, M_3, \dots$. In a max term, a variable will appear in its complemented form if its value is equal to 1 and the variable will appear in its un-complimented form if its value is equal to 0.

In the following table, there are three inputs, which have 8 different combinations. Each combination has a max terms denoted by capital M and its decimal combination number written in subscript.

X	Y	Z	Max Term (M)
0	0	0	$M_0 = X + Y + Z$
0	0	1	$M_1 = X + Y + Z'$
0	1	0	$M_2 = X + Y' + Z$
0	1	1	$M_3 = X + Y' + Z'$
1	0	0	$M_4 = X' + Y + Z$
1	0	1	$M_5 = X' + Y + Z'$
1	1	0	$M_6 = X' + Y' + Z$
1	1	1	$M_7 = X' + Y' + Z'$

Types of Product of Sum

There are three types of POS form.

- Canonical POS
- Non-canonical POS
- Minimal POS

Canonical POS Form

The canonical POS is also named as a product of max term. In canonical POS form, all the variables are present in a complimented or un-complimented form in each maxterm. The expression is denoted by \prod and the max terms in the bracket are taken when the output is false.

For example, let us consider a truth table as given below.

X	Y	Z	F	Max Term (M)
0	0	0	0	$M_0 = X + Y + Z$
0	0	1	1	$M_1 = X + Y + Z'$
0	1	0	1	$M_2 = X + Y' + Z$
0	1	1	1	$M_3 = X + Y' + Z'$
1	0	0	0	$M_4 = X' + Y + Z$
1	0	1	1	$M_5 = X' + Y + Z'$
1	1	0	0	$M_6 = X' + Y' + Z$
1	1	1	0	$M_7 = X' + Y' + Z'$

For this function the canonical POS expression is written as

$$F = \prod (M_0, M_4, M_6, M_7)$$

By expanding the above equation, we get

$$F = M_0, M_4, M_6, M_7$$

By substituting the max terms in the above equation we can get the below expression

$$F = (X+Y+Z) (X'+Y+Z)(X'+Y'+Z)(X'+Y'+Z')$$

Canonical POS form contains all inputs either complemented or non-complemented in its product terms.

Non Canonical POS

Non canonical POS is the expression of the product of sum which is not in normal form. In this form each sum term may or may not contain all the variables of the function. Let us consider the following function which is in canonical POS form.

$$\begin{aligned} F &= (X+Y+Z) (X'+Y+Z) \\ &= XX'+XY+XZ+X'Y+YY+YZ+X'Z+YZ+ZZ \\ &= 0+XY+XZ+X'Y+YY+YZ+X'Z+YZ+Z \\ &= X(Y+Z) + X'(Y+Z) + Y(1+Z) + Z \\ &= (Y+Z) (X+X') + Y(1) + Z \\ &= (Y+Z) (0) + Y+Z \\ &= Y+Z \end{aligned}$$

The above final expression is still in the form of Product of Sum, however, it is in the form of non-canonical.

Minimal POS

Minimal POS form is the simplified expression of the product of the sum. It is a type of non-canonical form. This is simplified by using different laws of Boolean algebra and it can also be done by using K-map (Karnaugh map). Minimal POS form is preferred because it uses the minimum number of gates and input lines. It is commercially beneficial because of its small size, fast speed, and low fabrication cost.

A	B	C	F	Max Term (M)
0	0	0	0	$M_0 = A+B+C$
0	0	1	1	$M_1 = A+B+C'$
0	1	0	1	$M_2 = A+B'+C$
0	1	1	1	$M_3 = A+B'+C'$
1	0	0	0	$M_4 = A'+B+C$
1	0	1	1	$M_5 = A'+B+C'$
1	1	0	0	$M_6 = A'+B'+C$
1	1	1	0	$M_7 = A'+B'+C'$

The canonical form of the above table is

$$F = (A+B+C)(A'+B+C)(A'+B'+C)(A'+B'+C')$$

The K map of the function is

		BC			
		00	01	11	10
A	0	0	1	1	1
	1	0	1	0	0

Simplified expression using K-map

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

The obtained expression is the minimal product of sum form. It is still Product of Sum expression But it needs only 2 inputs two OR gates and a single 2 input AND gate. However, the canonical form needs 4 OR gates of 3 inputs and 1 AND gate of 4 inputs.

(The detail of K-map is explained in the topic K-map below)

Conversion from Minimal POS to Canonical POS form

The canonical form of POS contains max terms. The max terms contains every input either complemented or non-complemented. So we will add every sum term with the product of complemented and non-complemented missing input.

Let us consider the following example which is in minimal POS form.

$$F = (A'+B')(B+C)$$

In $(A'+B')$ term C input is missing. So let us add (CC') with it. In $(B+C)$ term A input is

missing. So let us add (AA') with it.

$$F = (A' + B' + CC')(B + C + AA')$$

$$F = (A' + B' + C)(A' + B' + C')(A + B + C)(A' + B + C)$$

This expression is now in canonical form.

Conversion from Minimal SOP form to Canonical SOP form

Let us consider the following example which is in minimal SOP form

$$F = AB + AC + BC$$

In (AB) term C input is missing. So let us add $(C + C')$ with it. In (AC) term B input is missing. So let us add $(B + B')$ with it. In (BC) term A input is missing. So let us add $(A + A')$ with it.

$$F = AB(C + C') + A(B + B')C + (A + A')BC$$

$$F = ABC + ABC' + ABC + AB'C + A'BC + A'BC$$

$$F = ABC + ABC' + AB'C + A'BC$$

This expression is now in canonical form.

Conversion from Canonical POS to SOP

The product of Sum expression can be converted into Sum of Product form only if the expression is in canonical form. Canonical POS and canonical SOP can be converted into one another. The following are the steps to convert Canonical POS to SOP.

- Change the operation sign to Σ .
- Find the missing indexes of the terms.
- Write the product form of the noted terms.

Example

The POS canonical form is

$$F = (A + B + C)(A' + B + C)(A' + B' + C)(A' + B' + C')$$

In canonical form each sum term is a max term. So it can be written as

$$F = \prod(M_0, M_4, M_6, M_7)$$

The remaining combinations of inputs are minterms of the function for which its output is true. To convert it into SOP expression let us change the symbol to summation (Σ) and use the remaining minterm.

$$F = \Sigma(m_1, m_2, m_3, m_5)$$

Now we will expand the summation sign to form canonical SOP expression.

$$F = A'B'C + A'BC' + A'BC + AB'C$$

Min terms are complement of Max terms for the same combination of inputs.

Conversion of SOP form to POS form

The following are the steps to convert Canonical SOP to POS.

- Change the operation sign to \prod .
- Find the missing indexes of the terms.
- Write the sum form of the noted terms.

Example

The SOP canonical form is

$$x'y'z' + zy'z' + xy'z + xyz' + xyz$$

In canonical SOP form each product term is min term. So it can be written as

$$\sum x, y, z (m_0, m_2, m_3, m_5, m_7)$$

The remaining combinations of inputs are maxterms of the function for which its output is true. To convert it into POS expression let us change the symbol to \prod and use the remaining maxterm.

$$F = \prod(M_1, M_4, M_6)$$

Now we will expand to form canonical POS expression.

$$F = (A+B+C')(A'+B+C)(A'+B'+C)$$

Difference between SOP and POS

SOP	POS
SOP stands for Sum of Product.	POS stands for Product of Sum.
SOP is the method of representing Boolean expressions as sum of product terms.	POS is the method of representing Boolean expressions as product of sum terms.
SOP uses minterms. Minterm is product of Boolean variables either in normal form or complemented form.	POS uses maxterms. Maxterm is sum of Boolean variables either in normal form or complemented form.
It is sum of minterms. Minterms are represented as 'm'	It is product of maxterms. Maxterms are represented as 'M'
SOP is formed by considering all the minterms, whose output is HIGH(1)	POS is formed by considering all the maxterms, whose output is LOW(0)

While writing minterms for SOP, input with value 1 is considered as the variable itself and input with value 0 is considered as complement of the input.	While writing maxterms for POS, input with value 1 is considered as the complement and input with value 0 is considered as the variable itself.
--	---

KarnaughMaps(K-Map)

Digital electronic device deals with discrete valued digital signals which is in the form of either 0 or 1. In many digital circuits and practical problems, it will better and economical if the expressions contains minimum variables. So simplification of Boolean expression is an important part of the design of a digital electronic system.

Different laws of Boolean algebra and DeMorgan's theorems can be used to simplify the Boolean expression. But the process becomes tedious and difficult for large number of variables. So, we can easily minimize Boolean expressions of 3, 4 variables using Karnaugh Maps (K-map) without using any law of Boolean algebra.

Karnaugh Maps (K-map) was introduced by Maurice Karnaugh in 1953. A Karnaugh map (K-map) is a simple method used to simplify the algebraic expressions. A K-map can be thought of as a special version of a truth table that makes it easier to map out parameter values and produces a simplified Boolean expression. K-map can take two forms:

- Sum of product (SOP)
- Product of Sum (POS)

Steps to simplify Expression using K-map

1. Select the K-map according to the number of variables. The number of cells in K-map is equal to the total number of possible input variables combinations as is the number of rows in the truth table. For two variables, the number of cells is $2^2=4$, for three variables, the number of cells is $2^3=8$, for four variables, the number of cells is $2^4=16$ and so on.
2. Find the minterms or maxterms as given in the expression.
3. For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).
4. For POS put 0's in blocks of K-map respective to the max terms (1's elsewhere).
5. Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements in one group.
6. From the groups made in step 5 find the product terms and sum them up

for SOP form.

Example 1

Simplify the following 3-variable Boolean function in SOP form using K-Map.

$$F(P,Q,R)=\sum m(0,1,3,5,7)$$

The K-Map representation of the given Boolean function is shown in figure.

		QR			
		00	01	11	10
P	0	1	1	1	
	1		1	1	

The simplification of K-map is done as per the following steps:

- There are no any isolated 1s.
- The minterm m_1 forms a 4-square with minterms m_3 , m_5 , and m_7 . Make it and read it as R.
- The minterm m_0 forms a 2-square with the minterm m_1 . Make it and read it as $P'Q'$.
- Write all the product terms in SOP form.

Thus, the simplified SOP expression is, $F=R+P'Q'$

Example 2

Simplify the following 3-variable Boolean function in POS form using K-Map.

$$F(A,B,C)=\prod M(1,2,4,6)$$

The POS K-map representation of the given Boolean function is shown in Figure.

		BC			
		00	01	11	10
A	0		0		0
	1	0			0

The simplification of K-map is done as per the following steps:

- The maxterm M_1 has no adjacency. Thus, keep it as it is and read it as $(A+B+C')$
- The maxterm M_2 has only one adjacency M_6 . Hence, expand the maxterm M_2 into a 2-square with the maxterm M_6 and read the 2-square as $(B'+C)$
- The maxterm M_4 also has only one adjacency M_6 . Hence, expand the maxterm M_4

into a 2-square with the maxterm M_6 and read the 2-square as $(A'+C)$

- Write all the sum terms in POS form.

Therefore, the simplified POS expression is,

$$F=(A+B+C')(B'+C)(A'+C)$$

3.4 Algebraic Simplifications

Boolean Algebra Simplification is the way of simplifying boolean expressions. Boolean expression can be simplified by using different laws of boolean algebra and De-Morgan's theorem. This can also be done by using K-map as discussed earlier. Simplification of boolean expression makes the digital circuits significantly cost-effective, simpler to design and implement, and low power consuming.

Some of the examples of Boolean Algebraic simplifications are given below.

1. Simplify the following Boolean expression by using laws of Boolean algebra.

a. $F = A'.B.C + A' + A'.B'.C$

Solution

Expression	Rules used
$F = A'.B.C + A' + A'.B'.C$	Original Expression
$F = A'.B.C + A'.B'.C + A'$	Commutative law
$F = A'.C(B+B') + A'$	Distributive law
$F = A'.C(1) + A'$	Complement law)
$F = A'C + A'$	Identity law
$F = A'$	Absorption law

b. $F = C + (BC)'$

Expression	Rules Used
$C + (BC)'$	Original Expression
$C + (B' + C')$	DeMorgan's Law
$(C + C') + B$	Commutative, Associative Laws
$1 + B$	Complement Law
1	Identity Law

c. $F = A'.B.C + A' + A'.B'.C$

Expression	Rules used
$A'.B.C + A' + A'.B'.C$	Original Expression
$A'.B.C + A'.B'.C + A'$	Commutative law
$A'.C(B+B') + A'$	Distributive law
$A'.C(1) + A'$	Complement law
$A'.C + A'$	Identity law
A'	Absorption law

d. $F = (A+B)(A+C)$

Expression	Rules used
$F = (A+B)(A+C)$	Original Expression
$F = A.A + A.C + B.A + B.C$	Distributive law
$F = A + A.C + B.A + B.C$	Idempotent law
$F = A(1+C) + B.A + B.C$	Distributive law
$F = A(1) + B.A + B.C$	Identity law
$F = A + B.A + B.C$	Identity law
$F = A(1+B) + B.C$	Distributive law
$F = A(1) + B.C$	Identity law
$F = A + B.C$	Identity law

2. Minimize the following Boolean expression using K-map

$$F = A'BC + A'BC' + AB'C' + AB'C$$

The K-Map representation of the given Boolean function is shown in figure.

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	1	1	0	0

The simplification of K-map is done as per the following steps:

- There are no any isolated 1s.
- The minterm m_4 forms a 2-square with minterms m_5 . Make it and read it as $A'B$.
- The minterm m_3 forms a 2-square with the minterm m_2 . Make it and read it as AB'
- Write all the product terms in SOP form.

Thus, the simplified SOP expression is, $F=A'B+AB'$

Exercise

Choose the correct answer from the given alternatives.

- Boolean algebra is a form of mathematics developed by
a. Charles Babbage b. George Bool c. Lady Augusta
d. Alan Turing
- Which of the following statement illustrates idempotent law?
a. $A.A'=0$ b. $(A.B)'=A'+B'$ iii. $A+A+A=A$ d. $A+B=B+A$
- The Boolean expression $F=XY'Z+XYZ+X'Y'Z$ is in form.
a. Sum of Product b. Linear c. Product of sum d. Complex
- Min terms are of input combinations.
a. XOR b. AND c. OR d. NAND
- Karnaugh Maps (K-map) was introduced by
a. Charles Karnaugh b. George Bool
c. Charles Babbage d. Maurice Karnaugh
- There arecells in 3 variable K-map.
a. 6 b. 8 c. 12 d.16
- Which of the following expression illustrates complement law?
a. $(A.B)'=A'+B'$ b. $A.A'=1$ c. $A+A'=1$ d. $(A+B)'=A'.B'$
- The Boolean expression $F=(X+Y+Z')(X+Y'+Z')$ is in form.
a. Sum of Product b. Linear c. Product of sum iv. Complex
- What will be the simplified form of $F=(A+A')+(A.1)$?
a. A b. 1 c. A' d. 0

Write short answer to the following questions.

1. Differentiate between SOP and POS form.
2. Simplify the following Boolean expression by using laws of Boolean algebra.
 - a. $A'BC + AC$
 - b. $AB + AB'$

- c. $AB(A + B)(B + B)$
 - d. $AB + A(B+C) + B(B+C)$
 - e. $AB' + A(B+C)' + B(B+C)'$
 - f. $(BC' + A'D)(AB' + CD')$
 - g. $AB + A(CD + CD')$
 - h. $A'B + ABC' + ABC$
3. Define minterm and maxterm.
 4. Convert the Boolean expression $F = A'BC + A'B' + AB'CD'$ into standard SOP form.
 5. Convert the Boolean expression $F = (A+B'+C)(B'+C+D')(A+B'+C'+D')$ into standard POS form.
 6. Convert the SOP expression to an equivalent POS expression.
 $A'B'C' + A'BC' + A'BC + AB'C + ABC$
 7. Simplify the following Boolean expression by using K-map.
 - a. $F = A'B'C' + A'BC' + AB'C' + AB'C + ABC' + ABC$.
 - b. $\Phi = (A + B + C') + (A + B' + C') + (A' + B' + C) + (A' + B' + C')$
 8. Convert the following truth table into its POS form using K-Map.

A	B	Output(F)
0	0	0
0	1	1
1	0	1
1	1	0

Write long answer to the following questions.

1. Explain the various laws of Boolean algebra.
2. Explain the different types of POS form.
3. Explain K-map. How can it be used to simplify Boolean expression?

Project Work

1. Prepare a chart paper on a topic “Karnaugh map” in your class as a group work.
2. Prepare a Power Point Presentation file on a topic “Laws of Boolean algebra”

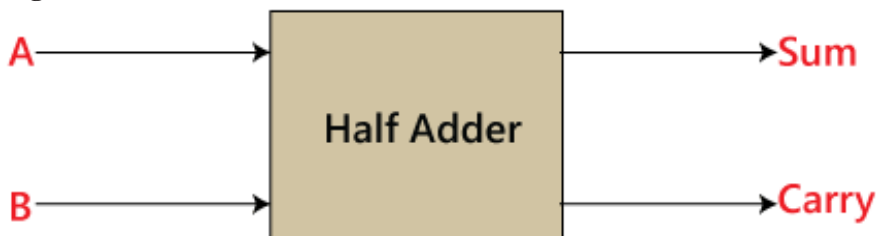
Introduction

The logic circuits that is formed by combining different types of logic gates is called Combinational Logic circuits. The output of the combinational circuit is determined from the present combination of inputs, regardless of the previous input. The input variables, logic gates, and output variables are the basic components of the combinational logic circuit. One of the main applications of combinational circuits is to perform basic mathematical operations with binary numbers. Some of the examples of combinational logic circuits are Adder, Subtractor, Decoder, Encoder, Multiplexer, and De-multiplexer.

4.1 Half Adder

Half adder is one of the simple of all adder circuits. It is considered as a basic building block for other complex adder circuits such as full adders and multiple-bit adders. It performs binary addition of two single-bit inputs and provides two outputs. The input variables are called augend and addend bits and output variables are called sum and carry bits. The Sum output is the least significant bit (LSB) of the result, which is the XOR of the two inputs. The XOR gate is used for the addition operation for binary digits as it produces result “1” in the Sum output only when one of the inputs is “1”. The Carry output is the most significant bit (MSB) of the result which indicates whether there was a carry-over from the addition of the two inputs. The Carry output is the AND of the two inputs. AND gate is used as it generates a “1” in the Carry output only when both inputs are “1”. So ,to construct half adder, one AND gate and one XOR gate is used.

Block diagram of Half Adder



Truth table

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

In the above truth table,

- The input states are A and B, and the output states 'sum' and 'carry'.
- The carry output is 0 in case where both the inputs are not 1.
- The least significant bit of the sum is defined by the 'sum' bit.

Construction of Half Adder Circuit

Half Adder circuit contains two inputs and two outputs. The **augend** and **addend** bits are input states, and **carry** and **sum** are output states. The half adder is designed with the help of the following two logic gates.

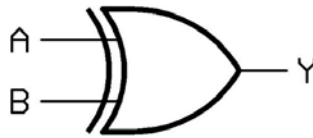
- XOR Gate
- AND Gate

XOR Gate

The Sum bit is generated with the help of XOR Gate. XOR gate is also known as Exclusive OR gate. It is the type of logic gate which accepts exact two inputs and produces only one output. The result produced by the XOR gate will be 1(True) if both inputs are different, otherwise 0(False). When the inputs are same, then the output will be 0(False).

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

Truth table



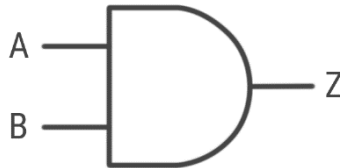
Logical diagram

AND Gate

The Carry bit is generated with the help of AND Gate. It is the type of logic gate which accepts two or more than two inputs and produces only one output. The result produced by the AND gate will be 1(True) if all the inputs are 1(True), otherwise 0(False). When one or more inputs of the AND gate is 0(False), then only the output will be 0(False).

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

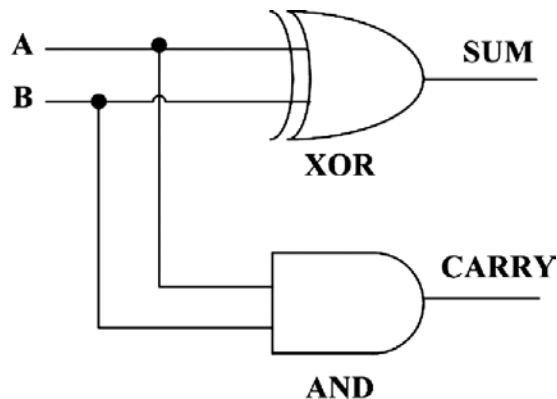
Truth table



Logical diagram

Half-Adder Logical Circuit

The Half Adder is designed by combining the 'XOR' and 'AND' gates and provide the sum and carry.



The logical expression for SUM is $A \oplus B$

The logical expression for CARRY is $A \text{ AND } B$

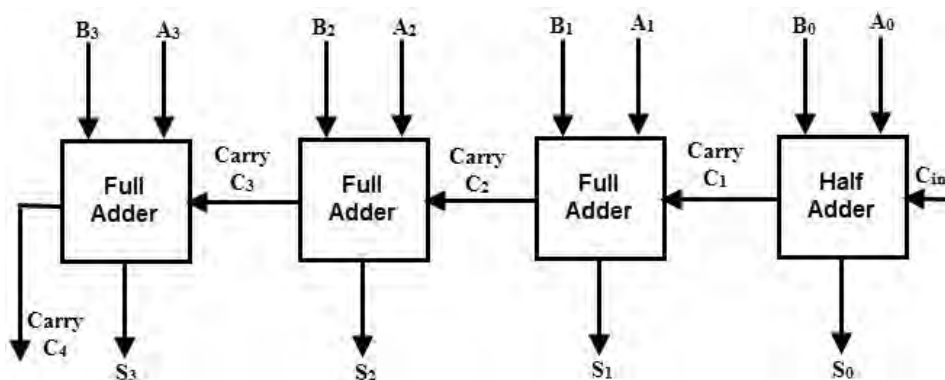
Applications of Half Adder

Some of the applications of half adder are

- It is used in ALU (Arithmetic Logic Unit) of computer processors to add binary bits.
- It is used to design full adder circuit.
- It is used in calculating machine like calculators.
- It is used to calculate addresses and tables.

4.2 Binary Adder

A binary adder is a circuit which is used to add two binary numbers. It takes two binary numbers as input and gives the sum of the two binary numbers and carry as output. Binary Adders can be implemented using various configuration. The common Configuration is using Half Adder and Full Adder Circuit. A half adder adds two single-digit binary numbers and produces the sum and carry output. A full adder is used to extend this function to use an additional carry input from the previous bit position, allowing it to add three input bits (two operands and the carry from the previous addition) to produce the sum and carry output. So a Binary Adder can be used to add two binary numbers of any length. It is generated using full-adder circuits connected in linear order. The output carries from one full-adder is linked to the input carry of the next full-adder.



- The 'A' and 'B' are the augend, and addend bits are defined by the subscript numbers. The subscripts start from right to left, and the lower-order bit is defined by subscript '0'.

- The C_1 , C_2 , and C_3 are the carry inputs which are connected together using Full Adder. The C_4 is the carry output produced by the last Full-Adder.
- The C_1 of the half Adder is connected to the next Full-Adder.
- The S_0 , S_1 , S_2 , and S_3 are the sum outputs that produce the sum of augend and addend bits.
- The inputs for the input variable 'A' and 'B' are fetched from different source registers. For example, the bit for the input variable 'A' comes from register 'R1', and a bit for the input variable 'B' comes from register 'R2'.
- The outcome produced by adding both input variables is stored into either third register or to one of the source registers.

Applications of Binary Adder

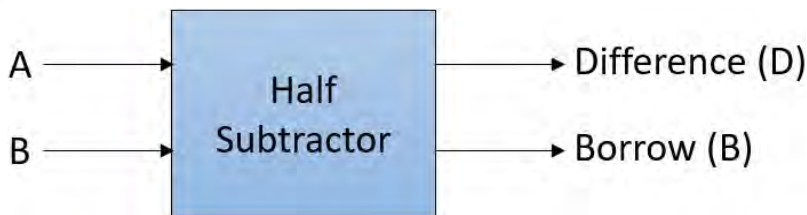
Some applications of the binary adder are

- ALU uses binary adders for addition.
- Binary adders are used in the digital gadgets to perform arithmetic operations.
- Microcontrollers use the binary adders for the increment purposes.

4.3 Half Subtractor

The half subtractor is also a building block for subtracting two binary numbers. It has two inputs and two outputs. It performs binary subtraction of two single bit inputs and provides two outputs. The input variables are augend and addend bits and output variables are sum and carry bits. In the subtraction ($A-B$), A is called as Minuend and B is called as Subtrahend. The 'Difference' and 'Borrow' are two output states of the half subtractor. The Difference output is the subtraction between the two input bits, while the Borrow output indicates whether bit is necessary to borrow during the subtraction. The half subtractor can be designed using logic gates such as XOR and NOT gates. The Difference output is the XOR of the two inputs, while the Borrow output is the NOT of input A and the AND of inputs A and B.

Block diagram of Half Subtractor



Truth table

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

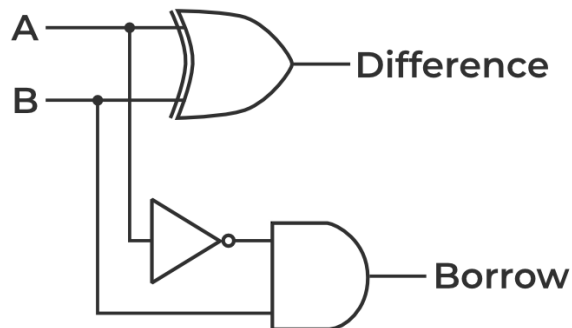
Construction of Half Subtractor Circuit

Half Subtractor circuit contains two inputs and two outputs. The **minuend** and **subtrahend** bits are the input states, and **difference** and **borrow** are the output states. The half subtractor is designed with the help of the following three logic gates.

- XOR Gate
- AND Gate
- Inverter or NOT gate

Half Subtractor Logical Circuit

The Half Subtractor is designed by combining the logic gates 'XOR', 'AND', and 'NOT'. It provide the Difference and Borrow.



The logical expression for Difference is $A \oplus B$

The logical expression for Borrow is $\neg A \cdot B$

Application of Half Subtractor

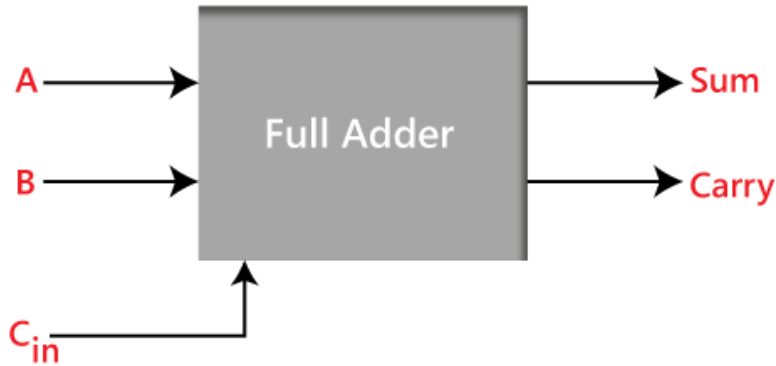
- Half subtractor is used in ALU (Arithmetic Logic Unit) of processors.

- Half subtractor is also used to decrease the force of radio signals or audio signals.
- Half subtractor is also used to increase or decrease operators.
- Half subtractor can also be used in amplifiers to manage the sound distortion.

4.4 Full Adder

Full adder is the combinational logic circuit that can add two binary digits and a carry bit. It produces a sum bit and a carry bit as output. So, Full Adder is a digital circuit that can perform the addition of three binary inputs.

Block Diagram of Full Adder



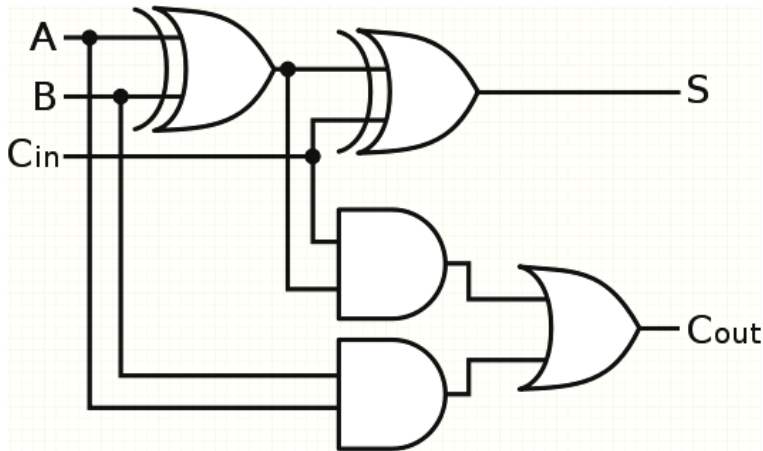
Full adder takes three inputs namely A, B, and C_{in} . Here, A and B are the two binary digits, and C_{in} is the carry bit from the previous binary addition. The sum output of the full adder is obtained by XORing the bits A, B, and C_{in} . While the carry output bit is obtained using AND and OR operations.

Truth table

Inputs			Outputs	
A	B	C_{in}	S (Sum)	C_{out} (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

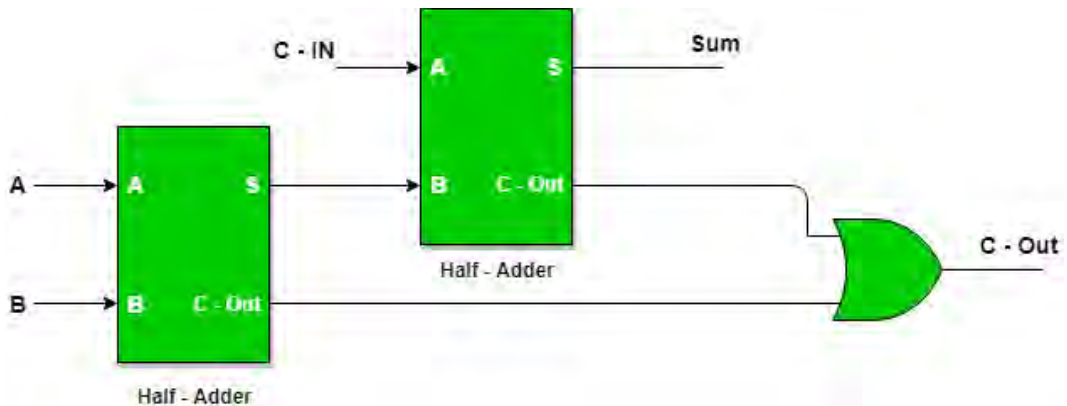
1	1	0	0	1
1	1	1	1	1

Full Adder logical circuit:



Implementation of Full Adder Using Half Adders:

2 Half Adders and an OR gate is required to implement a Full Adder.



Applications of Full Adder

Some of the applications of full adder are following:

- Full adders is the important part of CPU as it is used for arithmetic and logical operations.
- Full Adders are also implemented in calculator to perform calculations.
- Full adders are capable of generating the sum and carry of numbers, so they are used

in the generation of memory addresses.

- Full Adders are also used in Multiplexers to design the circuits and represent them for different uses.

4.5 Full Subtractor

A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, and borrow of the previous adjacent lower minuend bit. Full subtractor has three inputs and two outputs. So, Full Subtractor is a combinational circuit that can perform the subtraction of three binary inputs.

Block Diagram of Full Subtractor

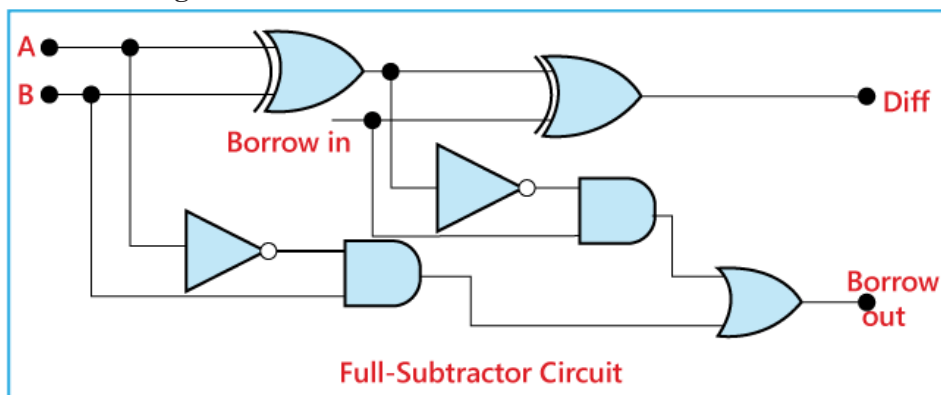


The three inputs A, B and B_{in} , denote the minuend, subtrahend, and previous borrow, respectively. The two outputs, D and B_{out} represent the difference and output borrow, respectively.

Truth table

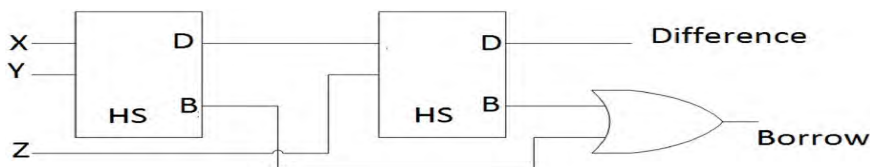
Inputs			Outputs	
A	B	B_{in}	D (Difference)	B (Borrow)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Full Subtractor logical circuit:



Implementation of Full Subtractor using Half Subtractors

2 Half Subtractors and an OR gate is required to implement a Full Subtractor.



Applications of Full Subtractor

Some of the applications of full subtractor are

- Full subtractors are used in ALU (Arithmetic Logic Unit) in computers CPUs.
- Full subtractors are widely used to perform arithmetical operations like subtraction in electronic calculators and many other digital devices.
- Full subtractors are used in microcontrollers for arithmetic subtraction.
- Full subtractors are used in timers and program counters (PC).
- Full subtractors are also used in Digital Signal Processing and networking based systems.

4.6 Code Converters

A digital code is a piece of data or information which represented in binary form. It is in the form of strings of 0s and 1s. The availability of a different codes for the same discrete elements of information results in the use of different codes in different digital systems. Sometimes the output of one system can be used as the input to another. So if the system uses different digital code, the two system must be made compatible by performing the conversion of code. A conversion circuit must

be used between two systems if each uses different codes for the same information. A code converter is a digital electronic circuit that is used to convert a digital code of one form to another in order to make two systems compatible. It is simply a code translator which translates a code from one format to another. Some of the examples of code converters are binary to decimal converter, BCD to Excess-3 converter, decimal to binary converter, etc.

Types of Code Converter

Binary to Decimal Converter

Binary to Decimal Converter as the name implies, is used to convert data from binary format to decimal format. The input to the binary to decimal converter is a number represented in binary format of 0s and 1s. The converter uses an algorithm to convert the input binary number into an equivalent decimal number and generates a decimal code as output.

BCD to Decimal Converter

BCD to Decimal Converter can convert a Binary Coded Decimal (BCD) number into an equivalent decimal number.

Decimal to BCD Converter

A Decimal to BCD (Binary Coded Decimal) converter is a type of code convert that converts a decimal number into its equivalent 4-bit binary code, called BCD code.

Binary to Gray Code Converter

A Binary to Gray code converter is a type of code converter that can translate a binary code into its equivalent gray code. The binary to gray code converter accepts a binary number as input and produces a corresponding gray code as output.

BCD to Excess 3 Converter

BCD to Excess 3 Converter is used to convert a binary-coded decimal number into an equivalent excess-3 code.

Applications of Code Converters

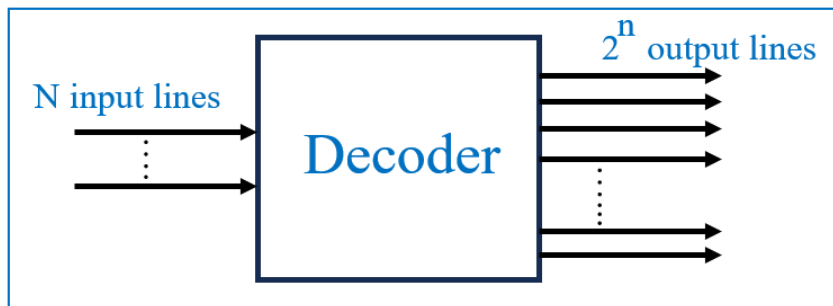
Some of the applications of code converters are

- Code converters are used in ADC (Analog-to-Digital Converters) and DAC (Digital-to-Analog Converters).
- The code converters are used as interface between two digital devices or systems that use different encoding schemes.

- The code converters are used to perform modulation and encoding tasks in digital communication systems.
- Code converters are also used in display devices like seven segment displays, to convert binary codes into human readable form.

4.7 Decoder

A decoder is a combinational logic circuit that converts a binary code into a set of outputs. Decoder converts an N bit binary input code into M output channels in such a way that only one output channel is activated for each one of the possible combinations of inputs. So decoder converts N input lines into a maximum of 2^n output lines. The Decoder performs the reverse operation of the Encoder.

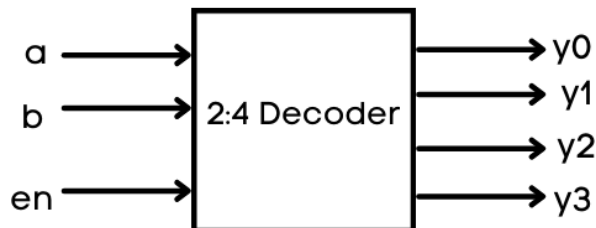


Types of Decoder

There are several types of decoder. Based on the input and output lines present, decoders may be classified into the following three types.

2 to 4 Decoder

It is the type of decoder that has 2 input lines and 4 (2^2) output lines. The block diagram of the 2 to 4 decoder is shown below.



When the decoder is enabled with the help of enable input en , then one of the four outputs will be active for each combination of inputs. The operation of this 2 to 4 line decoder can be analyzed with the help of its truth table which is given below.

Inputs			Outputs			
en	A	B	y ₃	y ₂	y ₁	y ₀
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Using this truth table, the Boolean expression for each output can be derived.

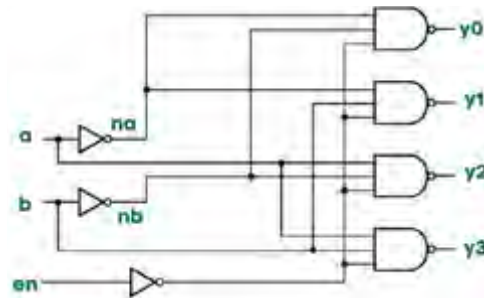
$$y_0 = en \cdot a' \cdot b'$$

$$y_1 = en \cdot a' \cdot b$$

$$y_2 = en \cdot a \cdot b'$$

$$y_3 = en \cdot a \cdot b$$

Each output term contains products of input variables, so it can be implemented with the help of AND gates. The logic circuit diagram of the 2 to 4 decoder is shown below.

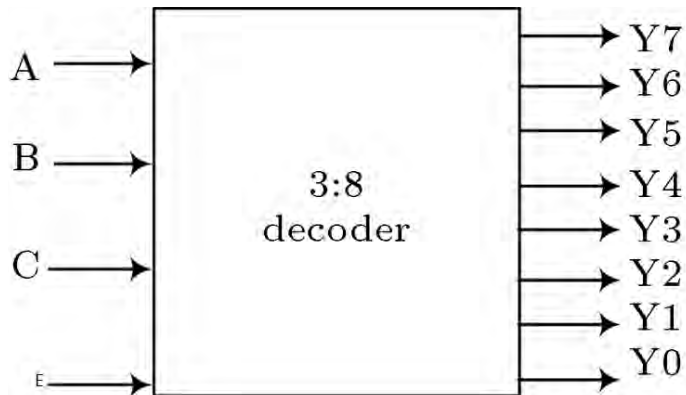


The operation of logic circuit of 2 to 4 decoder is as follows.

- When enable input (en) is inactive, i.e. set to 0, none of the AND gates will function.
- When enable input (en) is made active by setting it to 1, then the circuit works as explained below.
 - When a = 0 and b = 0, the AND gate 1 becomes active and produces output y₀.
 - When a = 0 and b = 1, the AND gate 2 becomes active and produces output y₁.
 - When a = 1 and b = 0, the AND gate 3 becomes active and produces output y₂.
 - When a = 1 and b = 1, the AND gate 4 becomes active and produces output y₃.

3 to 8 decoder

It is the type of decoder that has 3 input lines and 8 (2^3) output lines. The functional block diagram of the 3 to 8 decoder is shown below.



When this decoder is enabled with the help of enable input E, then one of the eight outputs will be active for each combination of inputs. The operation of this 3-line to 8-line decoder can be analyzed with the help of its truth table which is given below.

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

Using this truth table, the Boolean expression for each output can be derived.

$$Y_0 = EA'B'C'$$

$$Y_1 = EA'B'C$$

$$Y_2 = EA'BC'$$

$$Y_3 = EA'BC$$

$$Y_4 = EAB'C'$$

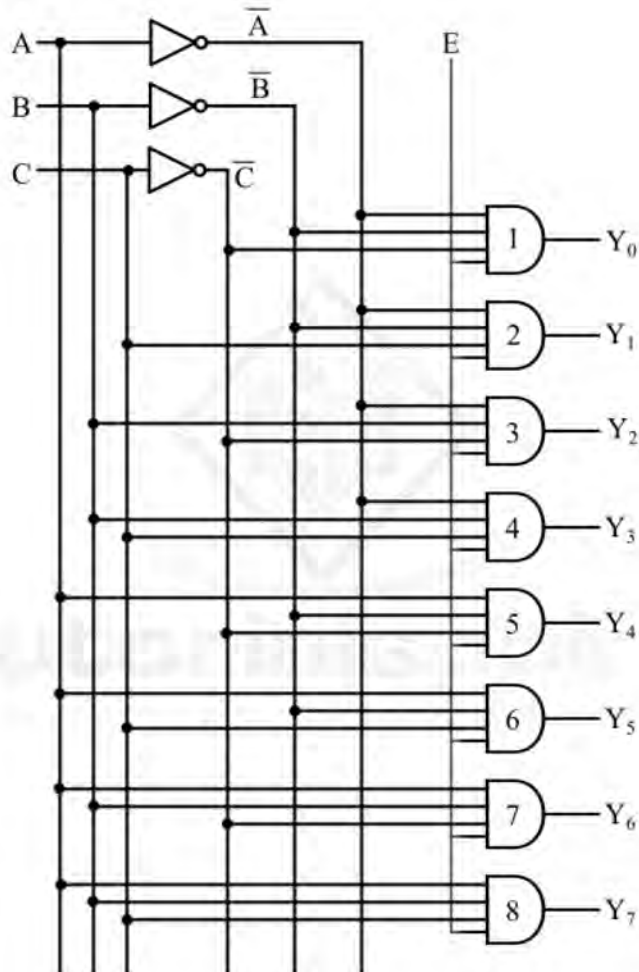
$$Y_5 = EAB'C$$

$$Y_6 = EABC'$$

$$Y_7 = EABC$$

As we can see, each output term contains products of input variables, hence they can be implemented with the help of AND gates. Therefore, the logic circuit diagram of the 3 to 8 decoder is shown

Each output term contains products of input variables, so it can be implemented with the help of AND gates. The logic circuit diagram of the 3 to 8 decoder is shown below.



3 to 8 Decoder

The operation of logic circuit of the 3 to 8 decoder is described as follows:

- When enable input (E) is inactive, i.e. set to 0, none of the AND gates will function.
- When enable input (E) is made active by setting it to 1, then the circuit works as described below.
 - When $A = 0$, $B = 0$, and $C = 0$, the AND gate 1 becomes active and produces output Y_0 .
 - When $A = 0$, $B = 0$, and $C = 1$, the AND gate 2 becomes active and produces output Y_1 .
 - When $A = 0$, $B = 1$, and $C = 0$, the AND gate 3 becomes active and produces output Y_2 .
 - When $A = 0$, $B = 1$, and $C = 1$, the AND gate 4 becomes active and produces output Y_3 .
 - When $A = 1$, $B = 0$, and $C = 0$, the AND gate 5 becomes active and produces output Y_4 .
 - When $A = 1$, $B = 0$, and $C = 1$, the AND gate 6 becomes active and produces output Y_5 .
 - When $A = 1$, $B = 1$, and $C = 0$, the AND gate 7 becomes active and produces output Y_6 .
 - When $A = 1$, $B = 1$, and $C = 1$, the AND gate 8 becomes active and produces output Y_7 .

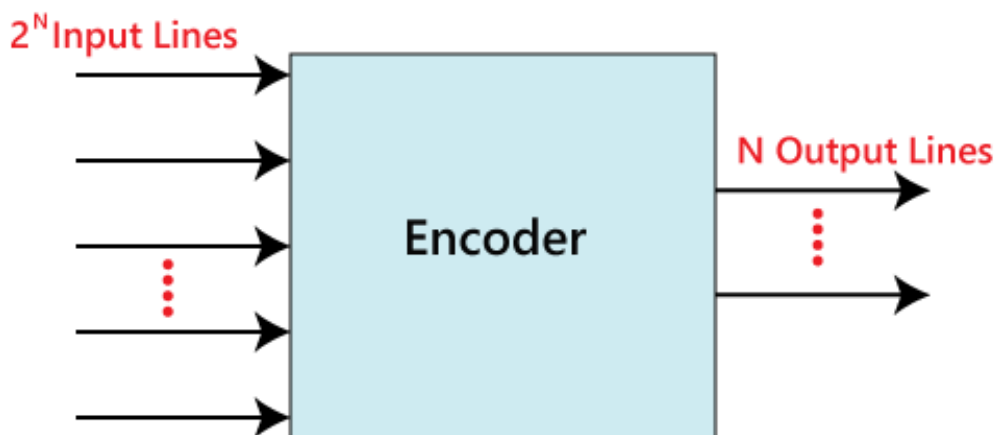
Applications of Decoders

Some important applications of decoders are

- Decoders are used for code conversions.
- Decoders are widely used in memory systems of computers.
- Decoders are used for de-multiplexing or data distribution.
- Decoders are used in data routing applications where very short propagation delay is required.
- Decoders are also used for timing or sequencing purposes.
- Decoders are also used to turn on and off digital devices at a specific time.

4.8 Encoder

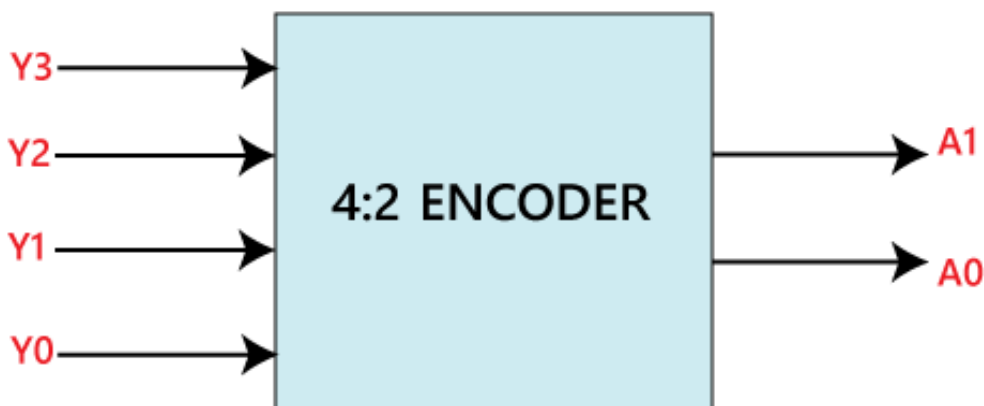
An Encoder is a combinational circuit that converts the readable information into a coded format for processing. An encoder converts binary information in the form of a 2^N input lines into N output lines, which represent N bit code for the input. The binary information is passed in the form of 2^N input lines. The output lines define the N-bit code for the binary information. At a time, only one input line is activated. The Encoder performs the reverse operation of the Decoder.



Types of Encoder

4 to 2 line Encoder

A 4 to 2 Encoder is a type of encoder which has 4 (2^2) input lines and 2 output lines. It produces an output code depending on the combination of input lines. The block diagram of a 4 to 2 Encoder is shown in the following figure.



Truth Table

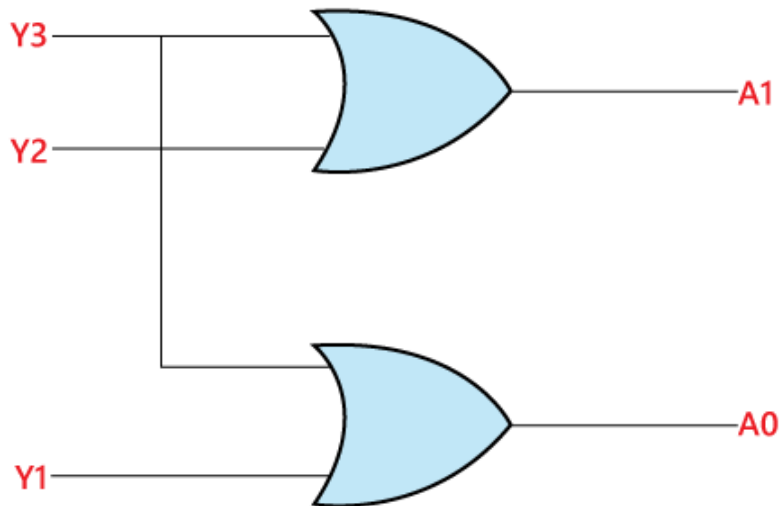
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Using this truth table, the Boolean expression for each output can be derived.

$$A_1 = Y_3 + Y_2$$

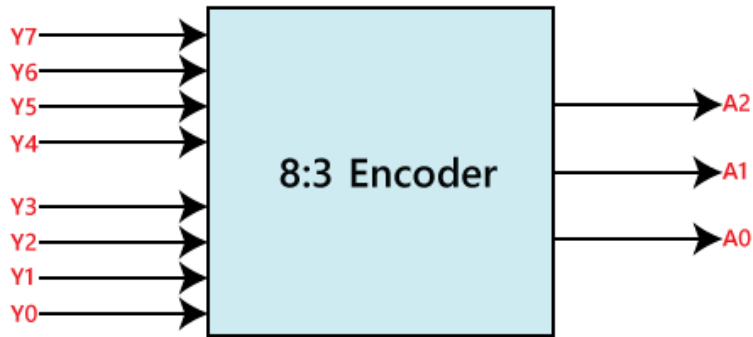
$$A_0 = Y_3 + Y_1$$

Each output term contains sum of input variables, so it can be implemented with the help of OR gates. The logic circuit diagram of the 4 to 2 encoder is shown below.



8 to 3 Line Encoder

A 8 to 3 Encoder is a type of encoder which has 8 (2^3) input lines and 3 output lines. It produces an output code (convert input information in a 3-bit format) depending on the combination of input lines. Therefore 8 to 3 line Encoder is also known as Octal to Binary Encoder. The block diagram of a 8 to 3 Encoder is shown in the following figure.



Truth Table

The following truth table describes the working of an 8 to 3 encoder.

Inputs								Outputs		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_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

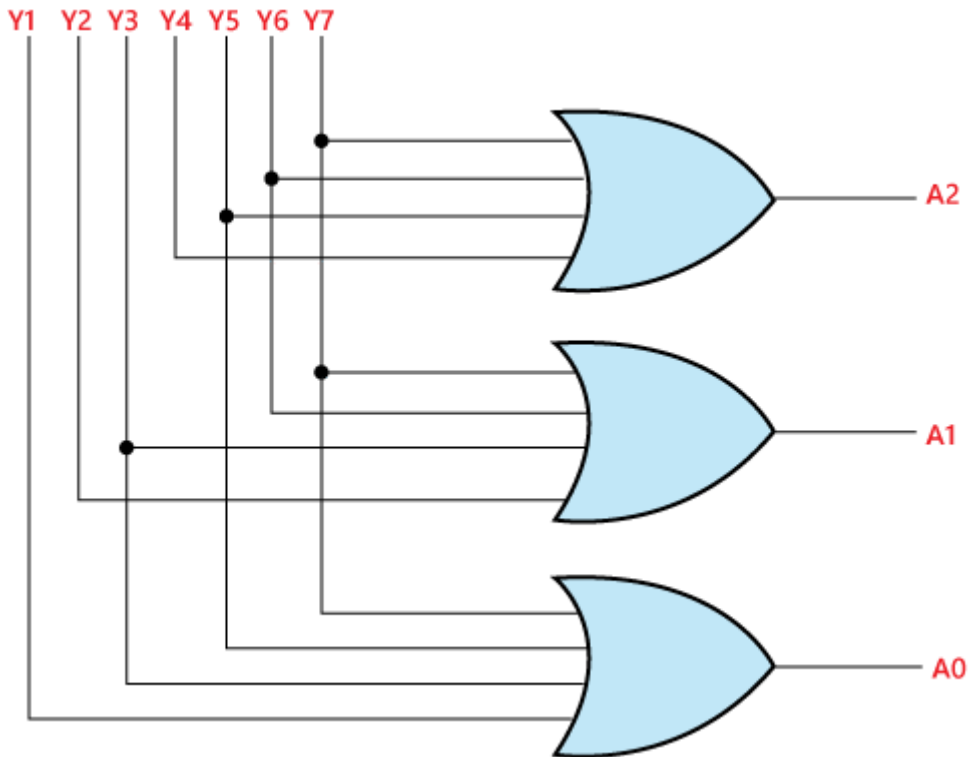
Using this truth table, the Boolean expression for each output can be derived.

$$A_2 = Y_4 + Y_5 + Y_6 + Y_7$$

$$A_1 = Y_2 + Y_3 + Y_6 + Y_7$$

$$A_0 = Y_7 + Y_5 + Y_3 + Y_1$$

Each output term contains sum of input variables, so it can be implemented with the help of OR gates. The logic circuit diagram of the 8 to 3 encoder is shown below.



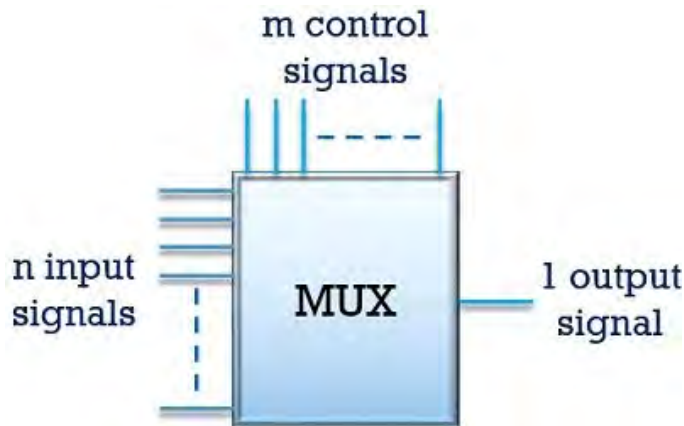
Applications of Encoder

Some applications of encoders are:

- Encoders are used in digital clocks and timers.
- Encoders are used in data processing devices and storage systems.
- Encoders are used in calculators, measuring instruments, display devices, microprocessors, microcontrollers and embedded systems.
- Encoders are used to convert a octal number into its equivalent binary number.

4.9 Multiplexer

Multiplexer is a combinational logic circuit that is designed to accept multiple input signals and transfer only one of them through the output line. Multiplexer has 2^n input lines and a single output line. The binary information is received from the input lines and directed to the output line. On the basis of the values of the selection lines, one of these inputs will be connected to the output. The block diagram of multiplexer is shown below.

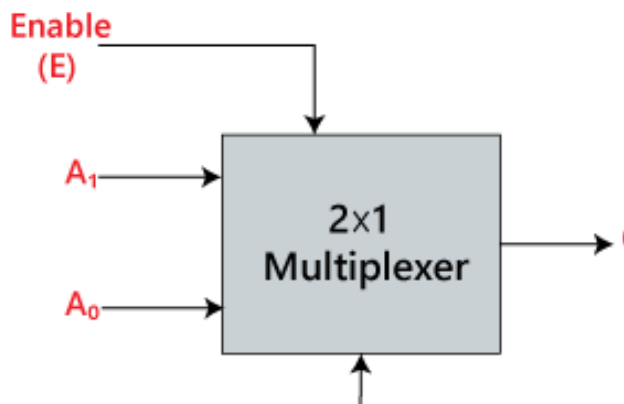


Types of Multiplexer

There are various types of the multiplexer which are as follows:

The 2×1 multiplexer

The 2×1 multiplexer has two data input lines. One is data select line and other is output line. The 2×1 multiplexer is used to connect two 1-bit data sources to a common designation. The two inputs are A_0 and A_1 , 1 selection line is S_0 and single output is Y . On the basis of the combination of inputs which are present at the selection line S^0 , one of these two inputs will be connected to the output. The block diagram of the 2×1 multiplexer is given below.



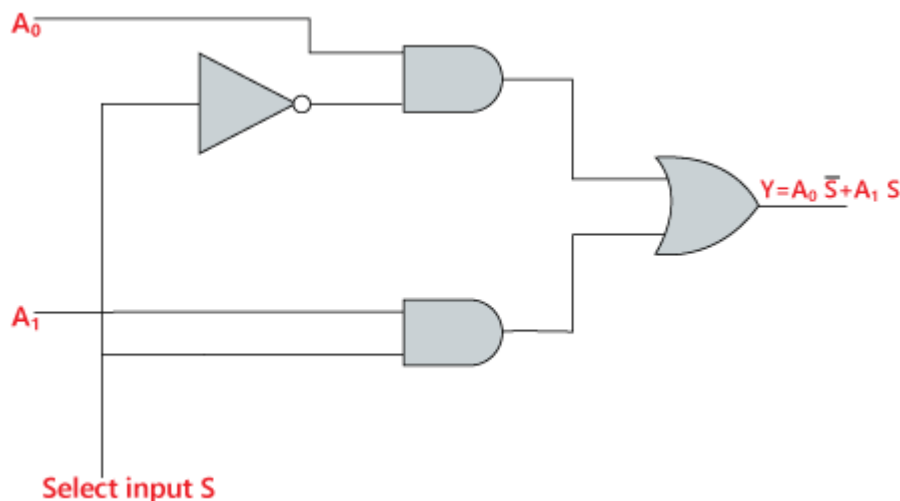
Truth table

Select Line (S)	Output (Y)
0	A_0
1	A_1

Using this truth table, the Boolean expression of the output can be derived.

$$Y = S_0' \cdot A_0 + S_0 \cdot A_1$$

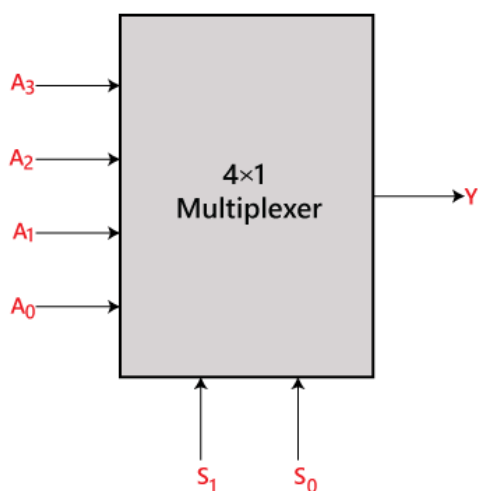
The logic circuit diagram of the 2 to 1 multiplexer is shown below:



4x1 multiplexer

The 4×1 multiplexer has four data input lines. Two data select line and one output line. The four inputs are A_0 , A_1 , A_2 and A_3 , two selection line is S_0 and S_1 and single output is Y . One of these four inputs will be connected to the output based on the combination of inputs present at these two selection lines.

The block diagram of 4×1 Multiplexer is shown below:



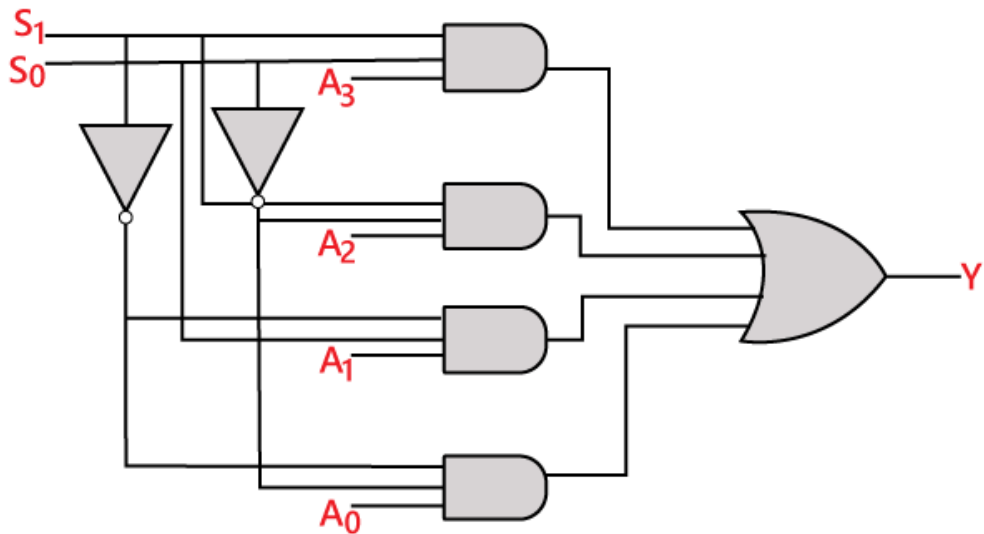
Truth Table

Selection Lines		Output
S1	S0	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

Using this truth table, the Boolean expression of the output can be derived.

$$Y = S_1' S_0' A_0 + S_1' S_0 A_1 + S_1 S_0' A_2 + S_1 S_0 A_3$$

The logic circuit diagram of the 4 x 1 multiplexer is shown below:



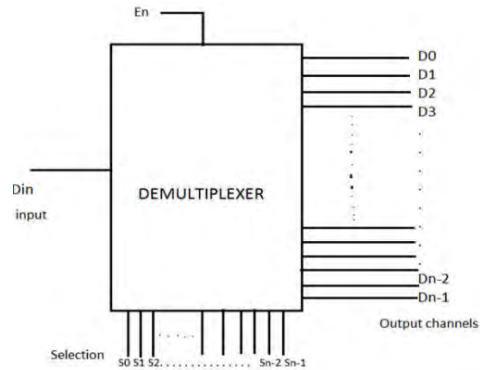
Applications of Multiplexer

Some of the applications of Demultiplexer are:

- Multiplexer is used to design various digital circuits like adders, subtractors logic gates etc.
- Multiplexer is used to convert the parallel data transmission to serial data transmission.
- Multiplexer is used in data routing to select one path from the different paths.
- Multiplexer is used in telecommunication system to convert different signals to one signal and then pass it through the communication channel.

4.10 Demultiplexer

Demultiplexer is a combinational logic circuit that is designed to accept single input signal and distribute it over several output lines. De-multiplexer has only one input line and 2^N output lines. The block diagram of Demultiplexer is shown below.

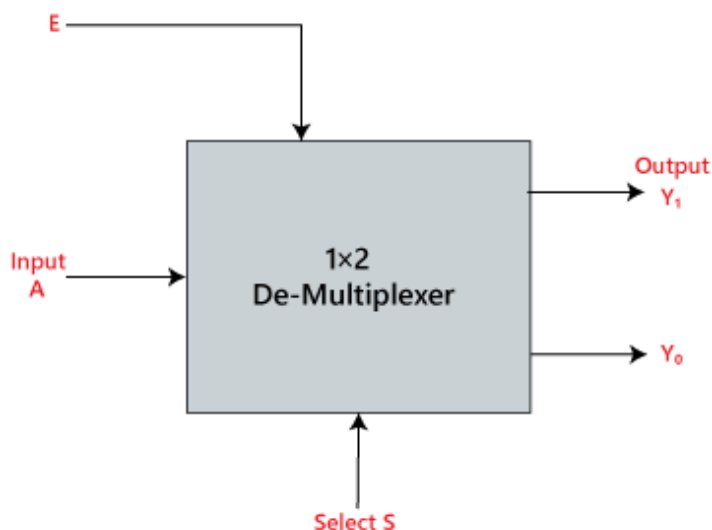


Types of Demultiplexer

There are various types of the multiplexer which are as follows:

1×2 Demultiplexer

The 1×2 Demultiplexer consists of one input line, one select line and two output lines. The logic level applied at the select line determines the output line to which the input data will be transmitted. The two outputs are Y_0 , and Y_1 , one selection line is S_0 , and single input is A. On the basis of the selection value, the input will be connected to one of the outputs. The block diagram of the 1×2 multiplexer is shown below:



Truth table

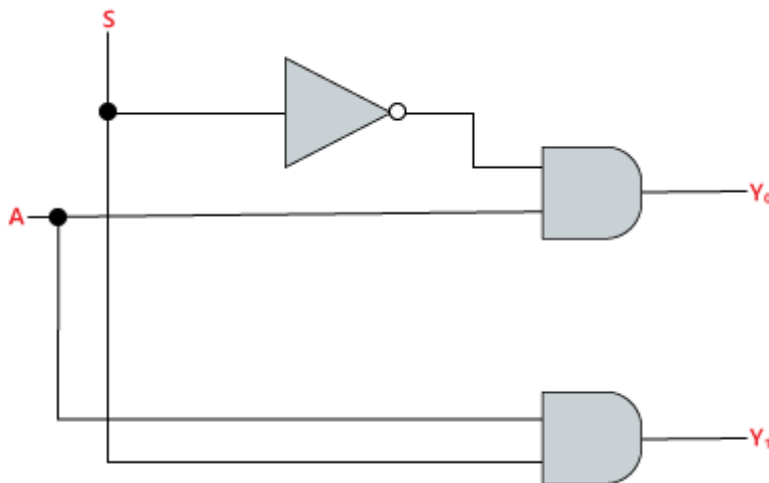
Select Line	Outputs	
S	Y_1	Y_0
0	0	A
1	A	0

Using this truth table, the Boolean expression of the output can be derived.

$$Y_0 = S_0' \cdot A$$

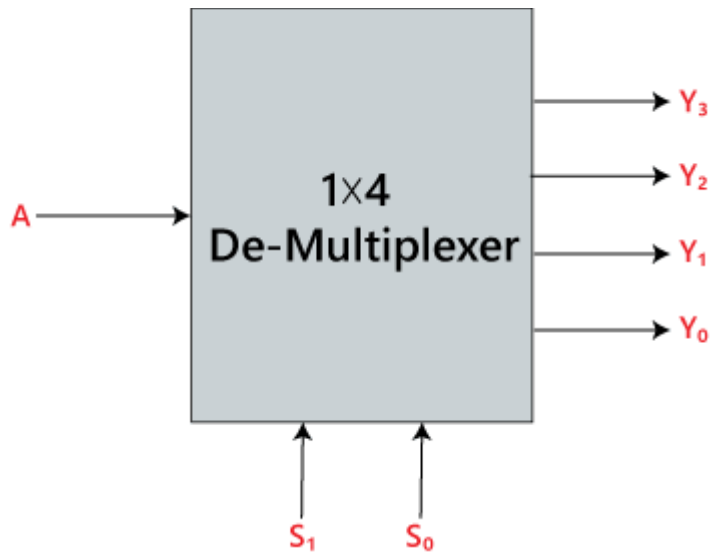
$$Y_1 = S_0 \cdot A$$

The logic circuit diagram of the 1x2 Demultiplexer is shown below:



1x4 De-multiplexer

The 1x4 Demultiplexer consists of one input line, two selection lines and four output lines. The logic level applied at the select line determines the output line to which the input data will be transmitted. The four outputs are Y_0 , Y_1 , Y_2 , and Y_3 , two selection line is S_0 and S_1 , and single input is A. On the basis of the selection value, the input will be connected to one of the outputs. The block diagram of the 1x4 multiplexer is shown below:



Truth Table

Select Line		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

Using this truth table, the Boolean expression of the output can be derived.

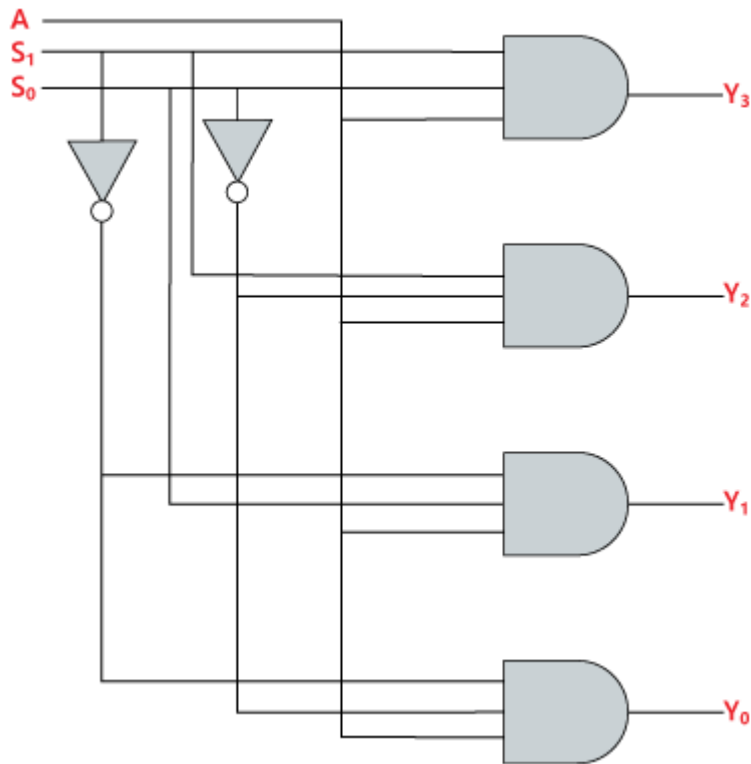
$$Y_0 = S_1' S_0' A$$

$$Y_1 = S_1' S_0 A$$

$$Y_2 = S_1 S_0' A$$

$$Y_3 = S_1 S_0 A$$

The logic circuit diagram of 1x 4 Demultiplexer is shown below:



Applications of Demultiplexer

Some of the applications of Demultiplexer are:

- Demultiplexer are used in several input and output devices for data routing.
- Demultiplexer are also used in digital control systems to select one signal from a mutual stream of signals.
- Demultiplexer can be used for generating Boolean functions.
- Demultiplexer can be used in serial to parallel converters.
- Demultiplexer can also be used to design automatic test equipment, etc.

Exercise

Choose the correct answer from the given alternatives.

1. Which combinational circuit is used to perform binary addition of two single-bit inputs and provides two outputs?
a. Full adder b. Half adder c. Half subtractor d. Register
2. Which of the following logic gates is used to design half adder?
a. XOR and AND gate b. XOR and OR gate
c. AND and OR gate d. XOR and NOT gate
3. Which combinational circuit can perform the subtraction of three binary inputs?
a. Full adder b. Half subtractor c. Full subtractor d. Full adder
4. The opposite of encoding is
a. multiplexing b. decoding c. demultiplexing d. adding
5. The encoder with 16 inputs have outputs.
a. 4 b. 8 c. 16 d. 32
6. Which of the following is required to implement a Full Adder?
a. 2 Half Adders and AND gate b. 1 Half Adder and AND gate
c. 2 Half Adders and XOR gate d. 2 Half Adders and OR gate
7. The primary function of Half adder is
a. to add two bit numbers without carry input
b. to add two bit numbers with carry input
c. to add two bit numbers with carry of previous addition
d. to add three bit numbers
8. What does Half subtractor calculate?
a. Difference and borrow b. Sum and carry
c. Product and quotient d. Input and output
9. 1x8 De-multiplexer has
a. 3 select input lines b. 4 select input lines
c. 6 select input lines d. 8 select input lines

Write short answer to the following questions.

1. Explain half adder with truth table and logical diagram.
2. Explain the working mechanism of half subtractor.
3. What is decoder? Explain the working mechanism of 2 to 4 decoder.
4. Differentiate between encoder and decoder.
5. Write the applications of encoder and decoder.
6. Differentiate between multiplexer and demultiplexer.
7. Construct the Full Adder with truth table and logical diagram.
8. Explain Encoder with its logical circuit diagram and truth table.

Write long answer to the following questions.

1. Explain the various types of code converters.
2. Design 8x1 multiplexer and explain it with the help of block diagram and logical diagram.
3. Explain 3 to 8 decoder with the help of truth table and logical diagram.
4. Explain the working mechanism of Full adder.

Project Work

1. Prepare a chart paper on a topic “Combinational Logic” and demonstrate the working mechanism of Half adder, Full adder, Half subtractor and Full subtractor in your class as a group work.
2. Prepare a Power Point Presentation file on a topic “ Multiplexer and Demultiplexer”



Unit 5 Introduction to Microprocessor and its Components

5.1 Definition of Microprocessor and its Applications

A microprocessor, as the name implies is the small micro chip designed to perform data processing, and controls the over all operations in a computing machines. It is generally built n the form of Integrated Circuit (IC). Microprocessor generally perform three basic functions, that is Input, Process and Output. It is the main component of a computer or digital device, responsible for executing instructions and performing various computing functions. So It is also referred to as the brain of the computer system. Microprocessors are used in a wide range of electronic devices, including personal computers, smartphones, embedded systems, and industrial equipment.

Applications of Microprocessors

Microprocessor is the essential component of modern electronic devices and systems. It is compact size, reliability, and effectiveness have increased its use in various applications from domestic to commercial and industrial applications.

- Microprocessors are used in various smart home appliances like washing machines, microwave, smart TV, digital-controlled refrigerators, automated home lighting system etc.
- Microprocessors are used in various industrial applications such as automation and operation of machine, robotics traffic system etc.
- Microprocessors are used in different electronic and computing devices like computers, laptops, tablets, smartphones, smart TV etc.
- Microprocessors are used in embedded systems, which are dedicated computing systems built into devices such as appliances, automotive systems, medical devices, and consumer electronics.

- Microprocessors are used in various network and communication devices like modems, routers, and many other network devices.
- Microprocessors are used in medical devices such as MRI machines, pacemakers, and insulin pumps for monitoring and controlling various functions.
- Microprocessors are used in satellites, radar systems, and military equipment to provide navigation, communication, and control.
- Microprocessors can be used in smart home devices like thermostats, doorbells, and security cameras, allowing for remote control and automation.

5.2 Types of Microprocessor

There are different types of microprocessors. Microprocessors are designed for specific applications and tailored to meet particular requirements. Some of the most common types of microprocessors are discussed below:

RISC (Reduced Instruction Set Computer) Processors

RISC microprocessors have a simplified instruction set due to which the execution time of the instruction is reduced. RISC microprocessors have more registers which reduces the interaction with memory for processing. So, they are known for their high performance and efficiency. They are commonly used in embedded systems, mobile devices, and networking equipment. Examples include ARM Cortex-A series processors.

CISC (Complex Instruction Set Computer) Processors

CISC processors have a more complex and extensive instruction set which reduces the number of instructions needed in a program. Since the instructions are less, it consumes less memory of RAM and few registers are needed. Since the instructions set are long and complex, it takes large time to process which decreases the speed of electronic devices. They are commonly found in general-purpose computers and servers. Examples include Intel x86 processors.

Digital Signal Processors (DSPs)

DSP microprocessors are designed to process the analog signals into digital form. It is used to process digital signals, such as audio, video, and sensor data. They are used in applications like audio processing, image and video processing, and telecommunications. Examples include the Texas Instruments TMS320 series and Analog Devices' Blackfin processors.

Graphics Processing Units (GPUs)

Graphics processing unit are specialized microprocessor which is used for both for personal and business computing. It is designed for parallel processing and can be used in a wide range of applications including graphics and video rendering. GPUs are also becoming more popular for use in creative production and artificial intelligence (AI). Examples include NVIDIA GeForce and AMD Radeon GPUs.

Microprocessors for Mobile Devices

These microprocessors are designed for portable devices such as smartphones and tablets. They focus on power efficiency and performance for tasks like web browsing, app execution, and multimedia playback. Examples include Qualcomm Snapdragon and Apple A-series processors.

5.3 Input/output

An input/output (I/O) microprocessor, is a component of a computer system which is mainly responsible for managing input and output operations between the Central Processing Unit (CPU) and various peripheral devices. Its main function is to handle data transfer and communication between the CPU and peripherals, which can include devices such as hard drives, keyboards, mouse and network interfaces.

Functions and Features of an I/O Microprocessor

There are various functions and features of an I/O processor.

- **Interface Management**

I/O processor is used to create and manage communication between the CPU and various input/output devices. The different types of protocols and data formats that are required for communication are handles by the I/O processor.

- **Data Buffering**

I/O processor have memory buffers which are used to store data temporarily during input/output operations. Buffers are used to store data temporarily as it moves between CPU and peripherals. Buffering helps in maintaining the data transfer rates between devices with different data transfer rates and smooth out the operations.

- **Interrupt Handling**

I/O processor can manage and handle interrupts generated by input/output devices. I/O processor trigger an interrupt and informs the CPU when a device needs attention, allowing the CPU to respond promptly and coordinate the data transfer.

- **Data Conversion**

I/O processor is responsible for the conversion of data format between the format used by the device and the format expected by the CPU. This includes tasks like data compression/decompression, byte swapping, and data encoding/decoding.

- **Error Handling**

I/O processors are also responsible for detecting and handling the errors that occur during input/output operations. The different types of error detection mechanism such as checksums or parity bits are used and suitable actions are taken in case of errors.

- **Address Decoding**

I/O processors decode addresses generated by the CPU to identify the specific peripheral or memory location with which the CPU wants to communicate.

5.4 Memory

Memory is generally the place where data and information are stored for temporary or permanent period. In a microprocessor or any digital computing system, memory refers to the electronic components used to store data, instructions, and program code that are necessary for the functioning of the system. Memory in a microprocessor is essential for temporarily holding data and instructions during processing and for storing programs and data more permanently when the system is powered off. Here are the primary types of memory used in a microprocessor-based system:

- **Registers:** Registers are the smallest and fastest type of memory chip that present within the microprocessor. They are used to store the data which CPU is processing and those data which are ready for the processing. The data stored in the register are directly. Register are directly accessed by the CPU and are used for performing arithmetic and logical operations, as well as for holding intermediate results.
- **Cache Memory:** Cache memory is an intermediate memory which is placed between CPU and main memory. Its purpose is to store frequently accessed data and instructions from RAM to reduce the CPU's access time and improve processing speed. Cache memory is divided into multiple levels (L1, L2, L3) with varying sizes and access speeds.
- **Random Access Memory (RAM):** RAM is the main memory of a microprocessor-based system. RAM stores data and programs that are currently running in the computer system. The data stored in the RAM will be lost or erased when the

computer system is switched off. So RAM can store data only for temporary period. RAM is volatile memory, as its contents are lost when the power is turned off. RAM comes in various types, including Dynamic RAM (DRAM) and Static RAM (SRAM).

- **Read-Only Memory (ROM):** ROM is also the main memory of the computer system which stores data and programs that are necessary to boot the computer and initialize hardware components. The programs that are stored in ROM are called firmware. ROM is Read Only Memory as the information can only be read and cannot be modified, updated and overwritten like RAM. ROM is non volatile memory as it retains its data even when the power is turned off.
- **External Storage Devices:** Microprocessor-based systems also supports and interact with different external storage devices, such as Hard Disk Drives (HDDs), Solid-State Drives (SSDs), optical drives(CD/DVD), and USB flash drives. These devices provide long-term storage for user data, applications, and files. These devices are also known as auxiliary memory or backup memory.
- **Virtual Memory:** Virtual memory is an artificial memory which does not physically exist in the computer system. It is the memory management technique which takes the required space from the hard drive or SSD and converts into working memory(RAM). It allows for efficient utilization of memory resources and enables the execution of larger programs.

5.5 Processing Unit

The processing unit is the primary component of a computer or microprocessor-based system. It is responsible for executing the given set of instructions and performing computations. It is considered as the "brain" of the computer system because it interprets and executes instructions and controls the overall operation of the entire system. The CPU consists of several key components, each playing a specific role in the processing of data. The main components of a processing unit are:

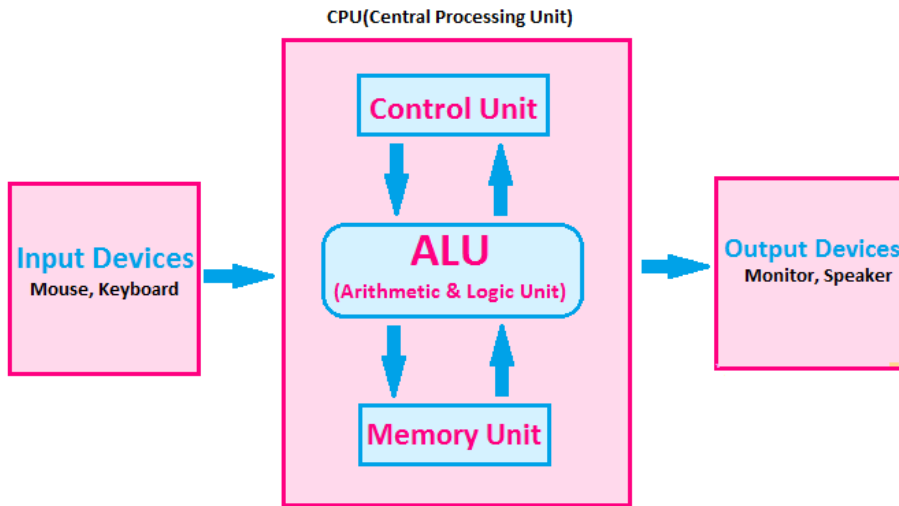
- **Arithmetic Logic Unit (ALU):** The ALU is responsible for performing arithmetic and logical operations on data. It can process arithmetic operations such as addition, subtraction, multiplication, division, and logical operations (AND, OR, XOR), and comparisons. The ALU operates on data that is stored in the registers.
- **Control Unit (CU):** The Control Unit is also known as the nervous system of computer because it controls and coordinates all the activities of the computer system.

It fetches instructions from RAM, decodes the instructions, and sends control signals to ALU for performing arithmetic and logical operations.

- **Registers:** Registers are small, high-speed storage locations within the CPU used for temporarily holding data and instructions during processing. They are used by the ALU for performing calculations and by the control unit for managing instructions. Common types of registers include the program counter (PC), instruction register (IR), and general-purpose registers (e.g., accumulator).
- **Clock:** The clock is a timing mechanism that synchronizes the operation of various components within the CPU. It generates clock pulses at a fixed frequency, defining the rate at which instructions are executed. The clock ensures that different components operate in harmony.
- **Cache Memory:** Cache memory is a small, high-speed memory located between the CPU and the main memory (RAM). Its purpose is to store frequently accessed data and instructions from RAM, reducing the CPU's access time and improving processing speed. Cache memory is divided into multiple levels (L1, L2, L3) with varying sizes and access speeds.
- **Instruction Fetch and Decode Unit:** This unit is responsible for fetching instructions from memory, decoding them to determine the operation to be performed, and providing the necessary control signals to execute the instructions.
- **Data Bus and Address Bus:** These are communication channels that allow the CPU to exchange data and addresses with other parts of the computer system, including memory and I/O devices. The data bus carries data, while the address bus carries memory addresses.

5.6 Arithmetic and Logical Unit, Control Unit, Registers

The Arithmetic and Logical Unit (ALU), Control Unit, and Registers are the three main components of the Central Processing Unit (CPU). These are the primary processing unit of a computer or microprocessor-based system. Each of these components plays a crucial role in executing instructions and performing computations.



CPU Block Diagram & Architecture

Arithmetic and Logical Unit (ALU)

The ALU is responsible for performing arithmetic and logical operations on data. It can process arithmetic operations such as addition, subtraction, multiplication, division, and logical operations (AND, OR, XOR), and comparisons. The ALU operates on data that is stored in the registers. When an instruction specifies an operation, the control unit directs the ALU to operate, taking data from registers and producing a result. The ALU operates on data stored in registers, typically taking two operands at a time. It can perform both integer and floating-point operations, depending on the design of the CPU. After an operation is executed, the ALU stores the result in a register or sends it to another destination, such as memory or another register.

Control Unit (CU):

The Control Unit is also known as the nervous system of computer because it controls and coordinates all the activities of the computer system. It plays a crucial role in the fetching, decoding, and execution of instructions. The CU fetches instructions from memory, typically following the program counter (PC) to determine the next instruction's address. It then decodes instructions to determine the specific operation to be performed and the operands involved. The CU also generates control signals that direct other components within the CPU and control the operation of the computer. These signals include signals to the ALU, memory, and registers. In some CPU architectures, the control unit uses microcode, a set of low-level instructions, to interpret complex instructions and manage the execution of simpler micro-operations.

Registers:

Registers are small, high-speed storage locations within the CPU used for temporarily holding data and instructions during processing. There are several types of registers, each serving a specific purpose. Some of the common types of registers are

Program Counter (PC): It holds the memory address of the next instruction to be fetched.

Instruction Register (IR): It stores the currently fetched instruction.

General-Purpose Registers: These registers are used for various purposes, including holding data operands during arithmetic and logical operations.

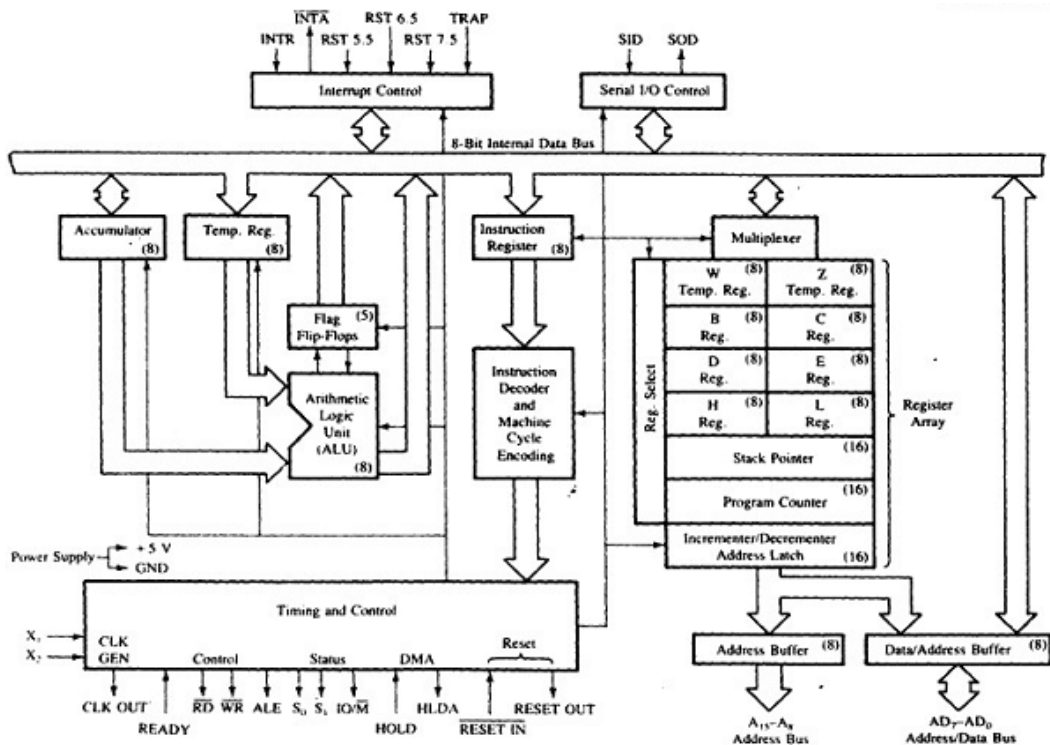
Accumulator: In some CPU architectures, the accumulator is a special register used as a target for arithmetic operations.

Data Access: Registers provide extremely fast access to data, making them ideal for operations requiring low latency.

Context Switching: Registers are also used during context switching, where the CPU switches between different tasks or processes, preserving their state.

1.7 8085 bus Structure and Internal Architecture

The 8085 microprocessor is an 8-bit microprocessor developed by Intel in the late 1970s. It was a popular microprocessor used in early personal computers and embedded systems. The 8085 microprocessor has a simple and well-defined internal architecture, which includes various components and registers. Here is an overview of the structure and internal architecture of the 8085 microprocessor:



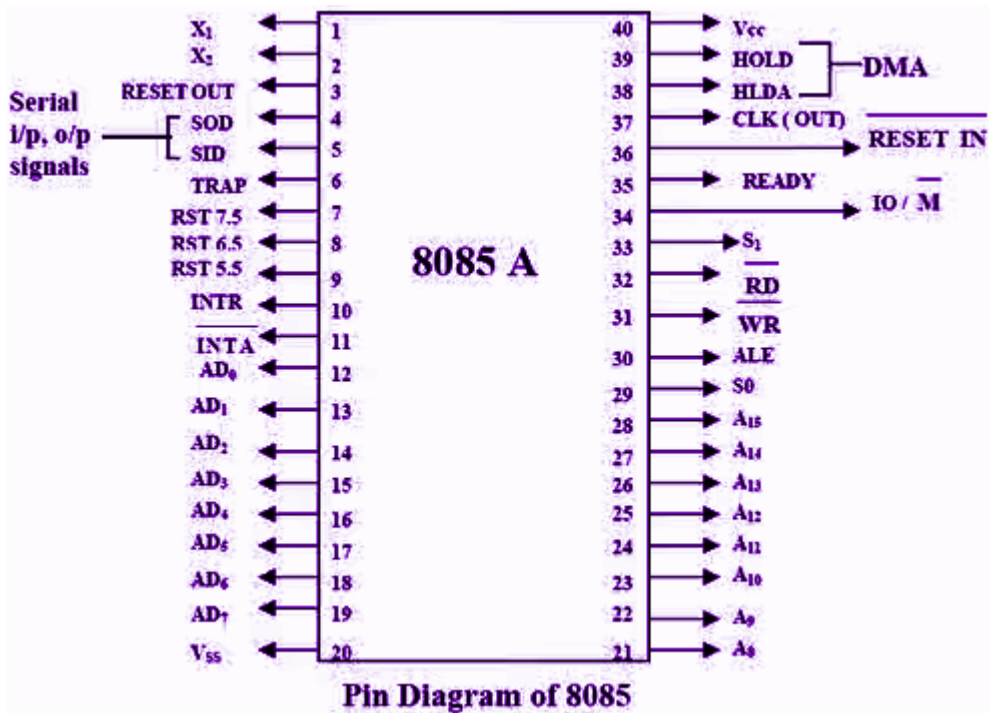
- i. **Accumulator (A):** The Accumulator is an 8-bit register used for most of the arithmetic and logic operations. It is often the target and source of data during these operations.
- ii. **General-Purpose Registers (B, C, D, E, H, L):** The 8085 includes six additional 8-bit registers, organized as three pairs: BC, DE, and HL. These registers can be used for data storage and manipulation.
- iii. **Flag Register (F):** The Flag Register is a 8-bit register that contains five condition flags that reflect the results of arithmetic and logical operations. The flags include:
 - a. **Sign (S):** Indicates the sign of the result (set if negative).
 - b. **Zero (Z):** Set if the result is zero.
 - c. **Auxiliary Carry (AC):** Used for BCD (Binary Coded Decimal) arithmetic.
 - d. **Parity (P):** Set if the number of set bits in the result is even.
 - e. **Carry (CY):** Indicates if a carry occurred during arithmetic operations.
- iv. **Program Counter (PC):** The Program Counter is a 16-bit register that stores the memory address of the next instruction to be executed. It is automatically incremented

after each instruction fetch.

- v. **Stack Pointer (SP):** The Stack Pointer is a 16-bit register used to manage the stack. It points to the memory location where the top of the stack is located.
- vi. **Memory Address Register (MAR):** The MAR holds the memory address of the data or instruction being accessed in memory.
- vii. **Memory Buffer Register (MBR):** The MBR temporarily holds data read from or written to memory.
- viii. **Instruction Register (IR):** The IR holds the current instruction being executed.
- ix. **Control Unit:** The Control Unit generates control signals to coordinate the operation of various components within the microprocessor. It includes a clock generator for synchronization.
- x. **Arithmetic and Logic Unit (ALU):** The ALU performs arithmetic and logical operations on data from registers and memory. It uses the Accumulator and the general-purpose registers as operands.
- xi. **Instruction Decoder:** The instruction decoder interprets the opcodes of fetched instructions and generates control signals to execute them.
- xii. **Address Bus:** The 8085 uses a 16-bit address bus to access memory and peripherals, allowing it to address up to 64K (64,000) memory locations.
- xiii. **Data Bus:** The data bus is an 8-bit bidirectional bus used for transferring data between the microprocessor and memory or peripheral devices.
- xiv. **Interrupt Control Unit:** The 8085 microprocessor has support for interrupts, and the Interrupt Control Unit handles external and internal interrupts.
- xv. **Serial I/O Control:** The 8085 includes serial input and output control lines for communication with external serial devices.

5.8 Pin Configuration of 8085

The 8085 microprocessor has a specific pin configuration that defines how it interacts with the external world, including memory, input/output devices, and other peripherals. It consists of 40 pins which are arranged in seven categories. The categories are Address bus, Data bus, Control signals and status signals, Power supply and frequency, Reset signals, DMA signals, and serial input/output ports. The pin configuration of the 8085 microprocessor are



Address Bus and Data Bus

This group consists of sixteen pins which are numbered from A0-A15. The address bus from A8 to A15 are unidirectional. So in this pin, the bits can flow in only one direction from the microprocessor unit to other peripheral devices. The data bus from A0 to A7 are bidirectional. So in this pin the bits can flow in both the directions simultaneously.

Control Status Signals

Control status signals are used to signify the type of operations that is to be performed. There are three control and three status signals.

RD (Read Control Output): This signal is generated by the microprocessor to indicate that it wants to read data from memory or an I/O device.

WR (Write Control Output) : This signal is generated by the microprocessor to indicate that it wants to write data to memory or an I/O device.

ALE (Address Latch Enable) : This signal is used to indicate the new operation. When the pulse becomes high, it denotes the address and when the pulse becomes low, it indicates that data has been received.

Power Supply

There are two power supply pins. These pins provide the necessary power connections for the microprocessor. VCC is typically connected to +5V, and VSS is the ground reference.

Clock Frequency

There are three clock signals.

X1 and X2: These pins are used for connecting an external crystal oscillator or clock source to provide the system clock signal.

CLK: This pin receives the system clock signal, which synchronizes the operations of the microprocessor.

Interrupts and Peripheral Initiated Signals

The 8085 has interrupt signals that is used to stop the execution of a program. The INTA signal is used by the microprocessor to acknowledge Interrupt Request. INTR stands for interrupt request. External devices use this pin to request an interrupt, prompting the microprocessor to respond to an interrupt service routine.

Reset Signals

RESET IN : This pin outputs a low signal when the program counter is reset.

RESET OUT: This pin outputs a high signal when the microprocessor is in the reset state.

DMA Signals

Hold and Hold Acknowledge pins are used for the Hold and Hold Acknowledge signals, allowing external devices to gain control of the bus temporarily.

Serial I/O Ports

There are two serial signals, SID and SOD, that are used for serial communication.

SOD (Serial output data line) :The SIM instruction sets/resets the output SOD.

SID (Serial input data line): When a RIM instruction is executed, the data on this line is loaded into the accumulator.

5.9 Description of each Block: Registers, Flag, Data and Address Bus, Timing, and Control with Interrupts

8085 microprocessor is an 8-bit microprocessor which is developed by Intel. The following is a description of each block/component:

Registers

Registers in the 8085 are small, high-speed storage locations within the CPU that hold data and instructions temporarily during processing. The 8085 microprocessor has several types of registers:

Accumulator (A): Used for most arithmetic and logical operations.

General-Purpose Registers (B, C, D, E, H, L): Used for data storage and manipulation.

Stack Pointer (SP): Points to the top of the stack.

Program Counter (PC): Stores the memory address of the next instruction to be executed.

Flag Register (F): Stores condition flags (Sign, Zero, Auxiliary Carry, Parity, Carry) reflecting the results of operations.

Flag Register:

The Flag Register (F) is a 5-bit register in the 8085 microprocessor that contains condition flags indicating the results of arithmetic and logical operations. These flags include:

Sign (S): Set if the result is negative.

Zero (Z): Set if the result is zero.

Auxiliary Carry (AC): Used for BCD arithmetic.

Parity (P): Set if the number of set bits in the result is even.

Carry (CY): Indicates if a carry occurred during arithmetic operations.

Data and Address Bus

The Data Bus is an 8-bit bidirectional bus (D0-D7) used to transfer data between the microprocessor and memory or I/O devices.

The Address Bus consists of 16 lines (A0-A15) and is used to send memory addresses and I/O port addresses. It can address up to 64K (64,000) memory locations.

Timing and Control

The Timing and Control unit generates control signals to synchronize the operation of various components within the microprocessor. It includes a clock generator for synchronization. Key control signals include RD (Read), WR (Write), ALE (Address Latch Enable), and various status signals like S0 and S1.

Interrupts

The 8085 microprocessor supports a variety of interrupts, which are signals generated by external devices to request the CPU's attention. There are five interrupt lines:

NTR (Interrupt Request): Used for external hardware interrupts.

RST7.5, RST6.5, RST5.5: Hardware interrupt lines that trigger predefined interrupt service routines (ISRs).

TRAP: Used for non-maskable interrupts.

Interrupt Control: The microprocessor can be configured to enable or disable interrupts based on program requirements. When an interrupt is acknowledged, the CPU saves its current state, branches to the appropriate ISR, and executes the interrupt service routine.

These blocks/components collectively allow the 8085 microprocessor to execute instructions, perform arithmetic and logical operations, manage data, address memory and I/O devices, and respond to interrupts from external devices. Proper utilization of these components is crucial for effective programming and operation of the microprocessor in various applications.

5.10 Introduction to Addressing Modes

Addressing modes are a fundamental concept in computer architecture and assembly language programming. They define how a microprocessor or CPU accesses operands or data for processing within instructions. Different addressing modes provide flexibility in specifying the source and destination of data, allowing programmers to write efficient and versatile code. The different addressing modes are:

Immediate Addressing:

In immediate addressing, the operand or data is directly specified within the instruction itself. The value is typically a constant or immediate data.

Example

MVI K,20F(20F is copied into the register K)

MOV R1, #25 (Move the value 25 into register R1).

Register Addressing

In register addressing, the operand is in one of the CPU's internal registers. The data is copied from one register into another.

Example

ADD R2, R3 (Add the contents of registers R2 and R3).

MOV K,B (Data in register B is copied to register K)

Direct Addressing

In direct addressing, the data is copied from the specified address to register.

LDA 9FFFH (Data at address 9FFFH is loaded in the accumulator)

Indirect Addressing

In indirect addressing the data is transferred from one register to another by using the address pointed by the register.

Example

MOV K,B (Move the value stored at the memory address pointed to by register B into register K).

Implied Addressing

In implied addressing mode, it does not require any operand. The data is specified by the opcode itself.

Example

CMP

These addressing modes allow assembly language programmers to write efficient code by specifying how data is accessed and manipulated within instructions. The choice of addressing mode depends on the specific task and the architecture of the microprocessor or CPU being used. Programmers select the most appropriate mode to optimize code readability and performance.

Exercise

Choose the correct answer from the given alternatives.

1. What is a microprocessor?
 - a. A type of memory storage device
 - b. A small computer used for personal tasks
 - c. A programmable integrated circuit that serves as the central processing unit (CPU) of a computer
 - d. An input device for data entry
2. RISC stands for
 - a. Remove Instruction Set Computer
 - b. Random Integrated Set Computer
 - c. Reduced Instruction Set Computer
 - d. Reduced Integrated Set Computer
3. Which of the following is an intermediate memory that is placed between CPU and main memory?
 - a. Register
 - b. Cache
 - c. RAM
 - d. ROM
4. Which component of a microprocessor is responsible for executing instructions and performing arithmetic and logic operations?
 - a. Control Unit
 - b. Memory Unit
 - c. Arithmetic Logic Unit (ALU)
 - d. Input Unit
5. Which component of microprocessor controls and coordinates all the activities of the computer system?
 - a. Register
 - b. ALU
 - c. MU
 - d. Output Unit
6. Which of the following is a common microprocessor architecture used in embedded systems, mobile devices, and networking equipment?
 - a. MISC
 - b. RISC
 - c. CISC
 - d. FISC
7. What is the purpose of the Clock Signal in a microprocessor?
 - a. To synchronize the activities of the microprocessor.

- b. To control the power supply.
 - c. To store data temporarily.
 - d. To interface with external memory.
8. Which of the following memory does not physically exist in the computer system?
 - a. Virtual b. RAM c. ROM d. Cache
 9. The instruction MOV A, M is an example of addressing mode.
 - a. Indirect b. Direct c. Register d. Immediate
 10. What is the primary function of the Data Bus in a microprocessor?
 - a. To transfer memory addresses.
 - b. To provide control signals.
 - c. To carry data between the CPU and memory or I/O devices.
 - d. To manage the clock signal.
 11. How many pins are there in 8085 microprocessor?
 - a. 40 b. 42 c. 44 d. 46
 12. What indicates to the address of instruction which is to be executed next time?
 - a. MAR b. MBR c. IR iv. PC
 13. Which of the following memory is used to data permanently?
 - a. ROM b. RAM c. Register d. Cache

Write short answer to the following questions.

1. What is a microprocessor? Write any four applications of it.
2. Explain any three types of microprocessor.
3. Explain the basic components of a microprocessor?
4. What is memory? Explain any two types of memory used in microprocessor-based system.
5. Write the different types of addressing modes in 8085. Explain them briefly.
6. Explain the concept of data bus and address bus in a microprocessor.
7. Differentiate between RISC and CISC microprocessors?
8. What is flag in microprocessor? Explain its types.

Write long answer to the following questions.

1. Explain the various functions of microprocessor.
2. Explain different types of memory that are present in microprocessor based system.
3. Describe the major components of a microprocessor, including the ALU, control unit, registers, and their roles in processing data and instructions?
4. Draw the structure of 8085 microprocessor and explain its components in brief.
5. Explain the various pin configuration of the 8085 microprocessor with labeled diagram.
6. What is addressing modes? Explain different types of addressing modes with examples.

Project Work

To introduce students or individuals to microprocessors and their components, you can use a combination of theoretical and hands-on activities. Here are some activities that can help you achieve this:

1. **Interactive Presentation:** Start with a presentation that introduces the basic concepts of microprocessors and their components. Use diagrams and visual aids to illustrate key points.
2. **Component Identification:** Provide a set of common microprocessor components (e.g., CPU, RAM, ROM, ALU, registers) and ask participants to identify and label each component.

Reference

Book References

"Digital Design" by M. Morris Mano and Michael D. Ciletti

"Digital Logic and Computer Design" by M. Morris Mano

"Contemporary Logic Design" by Randy H. Katz and Gaetano Borriello

"Microprocessor Architecture, Programming and Applications with the 8085" by Ramesh S. Gaonkar

"The Intel Microprocessors" by Barry B. Brey

"Advanced Microprocessors and Peripherals" by A.K. Ray and K.M. Bhurchandi

Web References

<https://www.geeksforgeeks.org>

<https://www.tutorialspoint.com>

<https://dribbble.com>

<https://en.wikipedia.org>

<https://www.w3schools.com>