

**MODULE CODE: ETEDE401**

**TRADE:ELECTRONICS AND TELECOMMUNICATION**

**AND TE**

**Module name: DIGITAL ELECTRONICS FUNDAMENTALS**

**………………………………………………………**

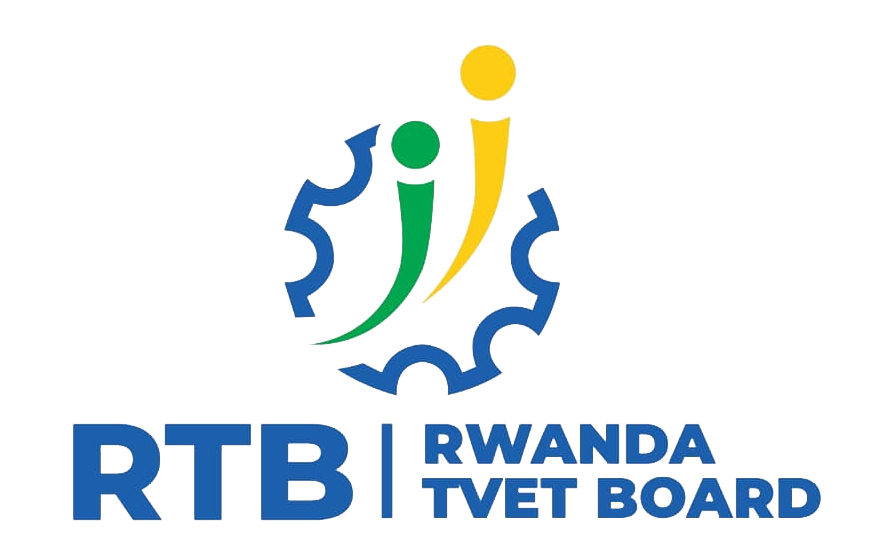
**TEACHER’S GUIDE**

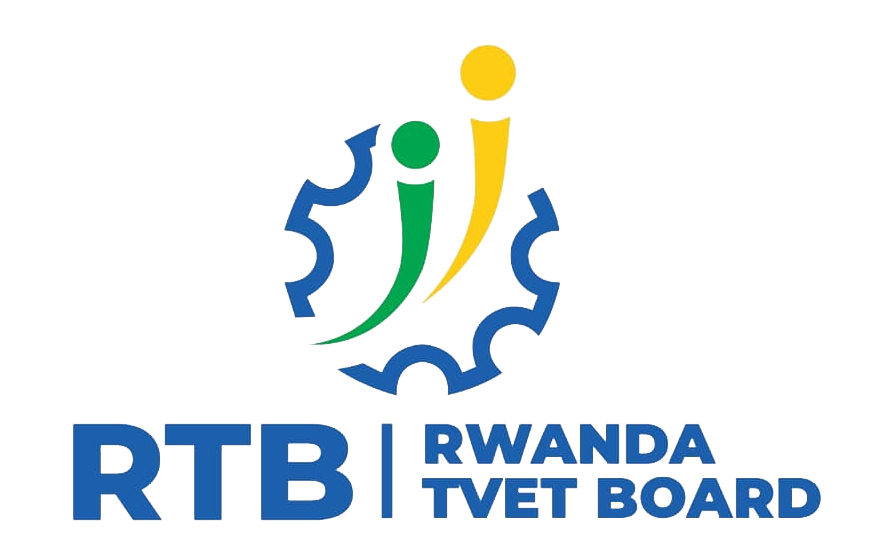
**RQF LEVEL 4**

**MODULE NAME: Digital electronics Fundamentals**

**2023**

2022

[Acronyms 6](#_Toc155192838)

[Introduction 7](#_Toc155192839)

Module Code and Title: ETEDE401- DIGITAL ELECTRONICS FUNDAMENTALS [1](#_Toc155192840)

Learning outcome 1: Apply digital numbers [2](#_Toc155192841)

Learning outcome 1 objectives: [3](#_Toc155192842)

Indicative content1.1: Definitions and applications of digital electronics [4](#_Toc155192843)

Theoretical learning Activity [4](#_Toc155192844)

Practical learning Activity [4](#_Toc155192845)

Indicative content1.2: Digital number systems [5](#_Toc155192846)

Numbering system [5](#_Toc155192847)

Digital system [6](#_Toc155192848)

Numbering system Conversion [6](#_Toc155192849)

Octal to binary Conversion [6](#_Toc155192850)

Theoretical learning Activity [8](#_Toc155192851)

Practical learning Activity [8](#_Toc155192852)

Indicative content1.3: Digital codes [9](#_Toc155192853)

Introduction to digital codes [9](#_Toc155192854)

Digital code conversion [10](#_Toc155192855)

Theoretical learning Activity [12](#_Toc155192856)

Practical learning Activity [12](#_Toc155192857)

Indicative content1.4: Digital arithmetic [14](#_Toc155192858)

Digital arithmetic operations [14](#_Toc155192859)

Complement of a Number [19](#_Toc155192860)

Theoretical learning Activity [20](#_Toc155192861)

Practical learning Activity [21](#_Toc155192862)

Learning outcome 1: formative assessment [21](#_Toc155192863)

Learning outcome 2: Apply logic gates [24](#_Toc155192864)

Learning outcome 2 objectives: [25](#_Toc155192865)

Indicative content2.1: Types of logic gates [26](#_Toc155192866)

Universal logic gates [28](#_Toc155192867)

Applications of logic gates [30](#_Toc155192868)

Theoretical learning Activity [30](#_Toc155192869)

Practical learning Activity [31](#_Toc155192870)

Indicative content 2.2: Apply Logic families [31](#_Toc155192871)

Types of logic families [31](#_Toc155192872)

Working principle of TTL [31](#_Toc155192873)

Types of TTL [32](#_Toc155192874)

Types of MOS Logic Families [33](#_Toc155192875)

Comparison of main logic families [33](#_Toc155192876)

Indicative content2.3: Apply Digital ICs [35](#_Toc155192877)

An integrated circuit [35](#_Toc155192878)

Limitations of Integrated Circuits [36](#_Toc155192879)

Classification of digital ICs [36](#_Toc155192880)

Applying digital ICs in electronic circuits [38](#_Toc155192881)

74HCXX and 74LSXX logic gates [38](#_Toc155192882)

Logic circuits simulation in software [41](#_Toc155192883)

Theoretical learning Activity [42](#_Toc155192884)

Practical learning Activity [42](#_Toc155192885)

**Learning outcome 2: formative assessment** [43](#_Toc155192886)

Learning outcome 3: Apply Boolean algebra [46](#_Toc155192887)

Learning outcome 3 objectives: [47](#_Toc155192888)

Indicative content3.1: Theorems of Boolean algebra [48](#_Toc155192889)

Description of Boolean algebra Laws [48](#_Toc155192890)

Demorgan’s theorems [49](#_Toc155192891)

Theoretical learning Activity [50](#_Toc155192892)

Practical learning Activity [50](#_Toc155192893)

Indicative content3.2: Standard Forms of Boolean Expressions [51](#_Toc155192894)

Boolean Expressions standard forms [51](#_Toc155192895)

Theoretical learning Activity [54](#_Toc155192896)

Practical learning Activity [54](#_Toc155192897)

Indicative content3.3: Boolean expressions simplification techniques [55](#_Toc155192898)

Using Karnaugh map (K-map) [55](#_Toc155192899)

Using Boolean algebra laws [59](#_Toc155192900)

Theoretical learning Activity [59](#_Toc155192901)

Practical learning Activity [60](#_Toc155192902)

**Learning outcome 3: formative assessment** [60](#_Toc155192903)

Learning outcome 4: Apply Fixed Logic Devices [63](#_Toc155192904)

Learning outcome 4 objectives: [63](#_Toc155192905)

Indicative content4.1: Apply Combinational logic circuits [65](#_Toc155192906)

Design and Working principle [65](#_Toc155192907)

Types of combinational logic circuits [65](#_Toc155192908)

Apply Comparator [69](#_Toc155192909)

Apply Multiplexer and De-multiplexer [70](#_Toc155192910)

Apply Encoder and Decoder [72](#_Toc155192911)

Applications of combinational logic circuits [74](#_Toc155192912)

Theoretical learning Activity [75](#_Toc155192913)

Practical learning Activity [75](#_Toc155192914)

Indicative content4.2: Multivibrators using ICs [75](#_Toc155192915)

Multivibrators using ICs [76](#_Toc155192916)

Theoretical learning Activity [77](#_Toc155192917)

Practical learning Activity [77](#_Toc155192918)

Indicative content4.3: Apply Sequential logic circuits [78](#_Toc155192919)

Design and Working principle [78](#_Toc155192920)

Types of Sequential logic circuits [79](#_Toc155192921)

Apply Flip-Flops [79](#_Toc155192922)

Apply Counters [83](#_Toc155192923)

Types of counters [83](#_Toc155192924)

Apply Shift Registers [88](#_Toc155192925)

Theoretical learning Activity [89](#_Toc155192926)

Practical learning Activity [89](#_Toc155192927)

**Learning outcome 4: formative assessment** [91](#_Toc155192928)

Learning outcome 5: Apply Programmable Logic Devices [95](#_Toc155192929)

Learning outcome 5 objectives: [96](#_Toc155192930)

Indicative content 5.1: Introduction to Programmable Logic Devices (PLD[) 97](#_Toc155192931)

PLD Working principle [97](#_Toc155192932)

Theoretical learning Activity [99](#_Toc155192933)

Practical learning Activity [100](#_Toc155192934)

Indicative content 5.2: Programming languages of PLDs [101](#_Toc155192935)

Description of Programming languages [101](#_Toc155192936)

Programming process [101](#_Toc155192937)

Functional Simulation [102](#_Toc155192938)

Timing Simulation [103](#_Toc155192939)

Device Programming (Downloading) [105](#_Toc155192940)

Theoretical learning Activity [106](#_Toc155192941)

Practical learning Activity [106](#_Toc155192942)

**Learning outcome 5: formative assessment** [107](#_Toc155192943)

[References 108](#_Toc155192944)

# 

**Acronyms**

**TVET:** Technical and Vocational Education and Training

**TTL:** Transistor - Transistor Logic

**PLD:** Programmable Logic Device

**MOS:** Metal Oxide Semiconductor

**IC:** Integrated Circuit

**ECL:** Emitter Coupled Logic

**DC:** Direct Current

**ETEDE:** Electronic and Telecommunication Digital Electronics

**RAM**: Random Access Memory

**ROM:** Read-Only Memory

**BCD:** Binary Code Decimal

**MUX:** Multiplex

**GSM:** Global System for Mobile Communication

**CPU:** Central Process Unit

**PWM:** Pulse Width Modulation

**DAC:** Digital Analogue Converter

**LED:** Light Emitting Diode

**SISO:** Serial -In -Serial Out

**SIPO:** Serial -In- Parallel Out

**PLA:** Programmable Logic Array

**GAL:** Generic Array Logic

**SPL:** Simple Programmable Device

**AMD:** Advanced Micro Device

# **Introduction**

Digital electronics is a branch of electronics that deals with the study of digital signals and the components that use or create them. Digital signals are discrete, meaning they can only have a limited number of values, unlike analog signals, which can have an infinite range of values. The most common digital signals are binary signals, which can only have two values, typically represented by high and low voltage levels. Basic components of digital electronics:

**Logic gates:** These are the building blocks of digital circuits. They perform basic logic operations on one or more input signals to produce an output signal. Common logic gates include AND, OR, NOT, NAND, NOR, XOR, and XNOR.

**Flip-flops:** These are circuits that can store a binary value (0 or 1). They are used to remember the state of a circuit even when the power is turned off.

**Counters:** These are circuits that count the number of times a particular event occurs. They are used in a variety of applications, such as timers and frequency dividers.

**Memory devices:** These are circuits that can store digital data. Common memory devices include RAM (Random Access Memory) and ROM (Read-Only Memory).

Digital electronics is used in a wide variety of applications, including:

**Computers:** Computers are the most common application of digital electronics. They use digital circuits to process information and perform calculations.

**Communications:** Digital circuits are used in a variety of communication devices, such as telephones, cell phones, and radios.

**Control systems:** Digital circuits are used in control systems for a variety of machines and devices, such as cars, airplanes, and industrial robots.

**Consumer electronics:** Digital circuits are used in a variety of consumer electronics devices, such as televisions, stereos, and video game consoles.

# **Module Code and Title: ETEDE401- DIGITAL ELECTRONICS FUNDAMENTALS**

|  |
| --- |
| **Learning Outcome 1: Apply digital numbers**  **Learning Outcome 2: Apply logic gates**  **Learning Outcome 3: Apply Boolean Algebra**  **Learning Outcome 4: Apply fixed logic devices**  **Learning Outcome 5:Apply programmable logic devices** |

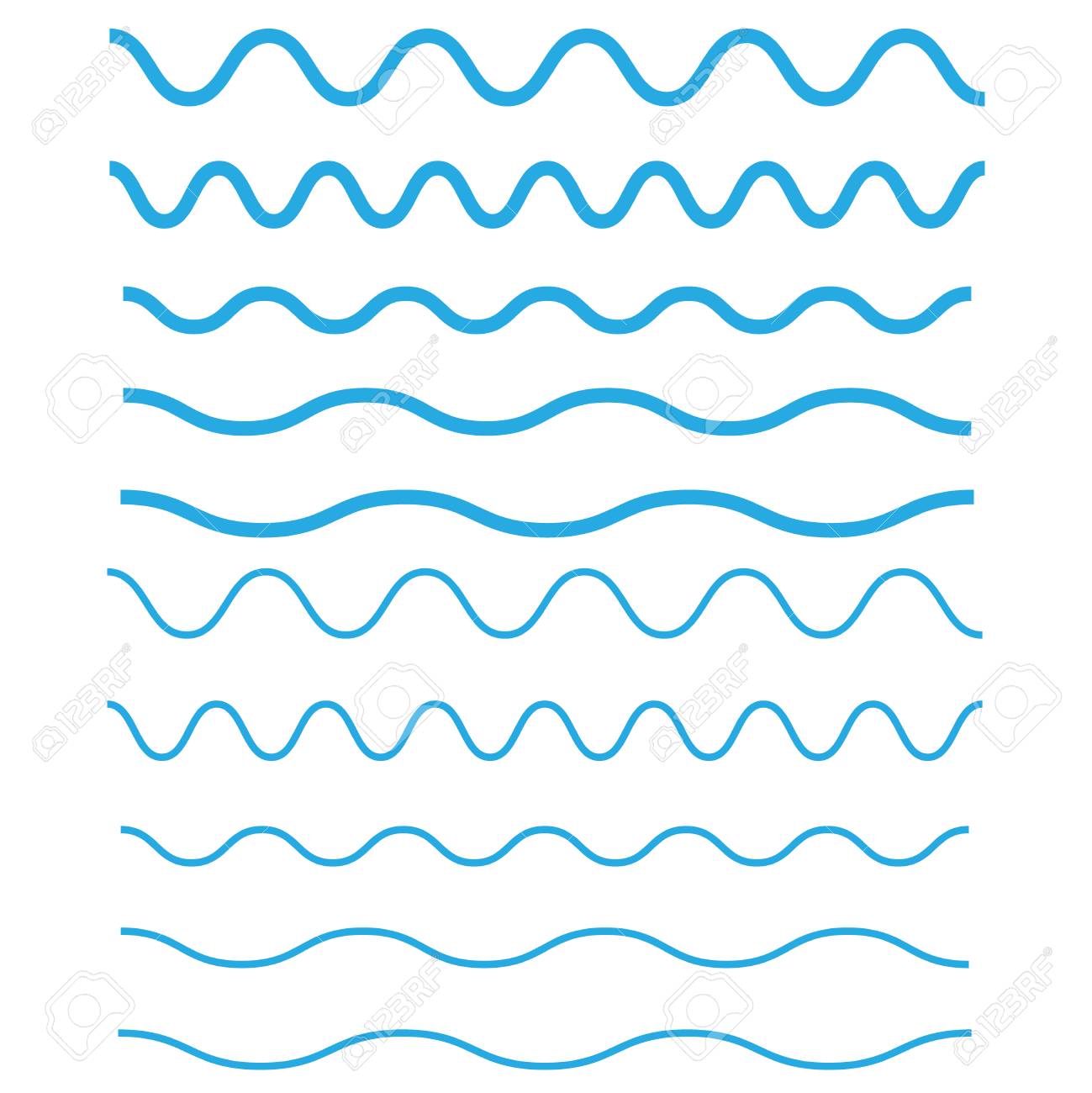
# **Learning outcome 1: Apply digital numbers**

|  |
| --- |
|  |

|  |
| --- |
| **Indicative contents**  **1.1 : Definitions and applications of digital electronics**  **1.2:** **Digital number systems**  **1.3:** **Digital codes**  **1.4:** **Digital arithmetic** |

|  |  |  |
| --- | --- | --- |
| **Duration:30hrs** | | |
| **Learning outcome 1 objectives****:**  By the end of the learning outcome, the trainees will be able to:  1. Describe clearly digital number systems according to their types  2. Identify properly digital codes according to their types  3. Apply clearly digital arithmetic according to the type of operators | | |
| **Resources** | | |
| **Equipment** | **Tools** | **Materials** |
| Computer  Projector  digital display  Analogue display | Calculator  Books | Flip chart  Markers  chalks  board  Internet |
| **Advance preparation:**   * Read the definition of digital and analogue electronics * Deeply research on the applications of digital electronics * Deeply research on the numbering systems types, conversion and arithmetic operations | | |

## **Indicative content1****.1: Definitions and applications of digital electronics**



|  |
| --- |
| * **Digital electronics** is the study of electronic circuits that are used to process and control digital signals. In contrast to analogue electronics, where information is represented by a continuously varying voltage, digital signals are represented by two discrete voltages or logic levels (0 and 1). * **A digital signal** is a signal that represents data as a sequence of discrete values; at any given time it can only take on, at most, one of a finite number of values * **An analogue signal** is any continuous-time signal representing some other quantity, i.e., analogous to another quantity. For example, in an analogue audio signal, the instantaneous signal voltage varies continuously with the pressure of the sound waves * **Analogue electronics** use relatively high voltage and high current as compared to digital electronics. The voltage and current used in digital electronics are extremely low. * **In analogue electronics**, high noise and distortion of signals is there. In digital electronics, there is very low noise and distortion of signals. * The most significant difference between analogue and digital electronics is that analogue electronics deals with continuously varying signals while the digital electronics deals with two state (binary) signals * **Digital electronics** are used in computers to store, process, and transmit data. They are used in the central processing units (CPUs) of computers, as well as in other components such as memory, storage, and input/output devices. |

### **Theoretical learning** **Activity**

* Learners within group of five brainstorm about difference between digital and analogue electronics

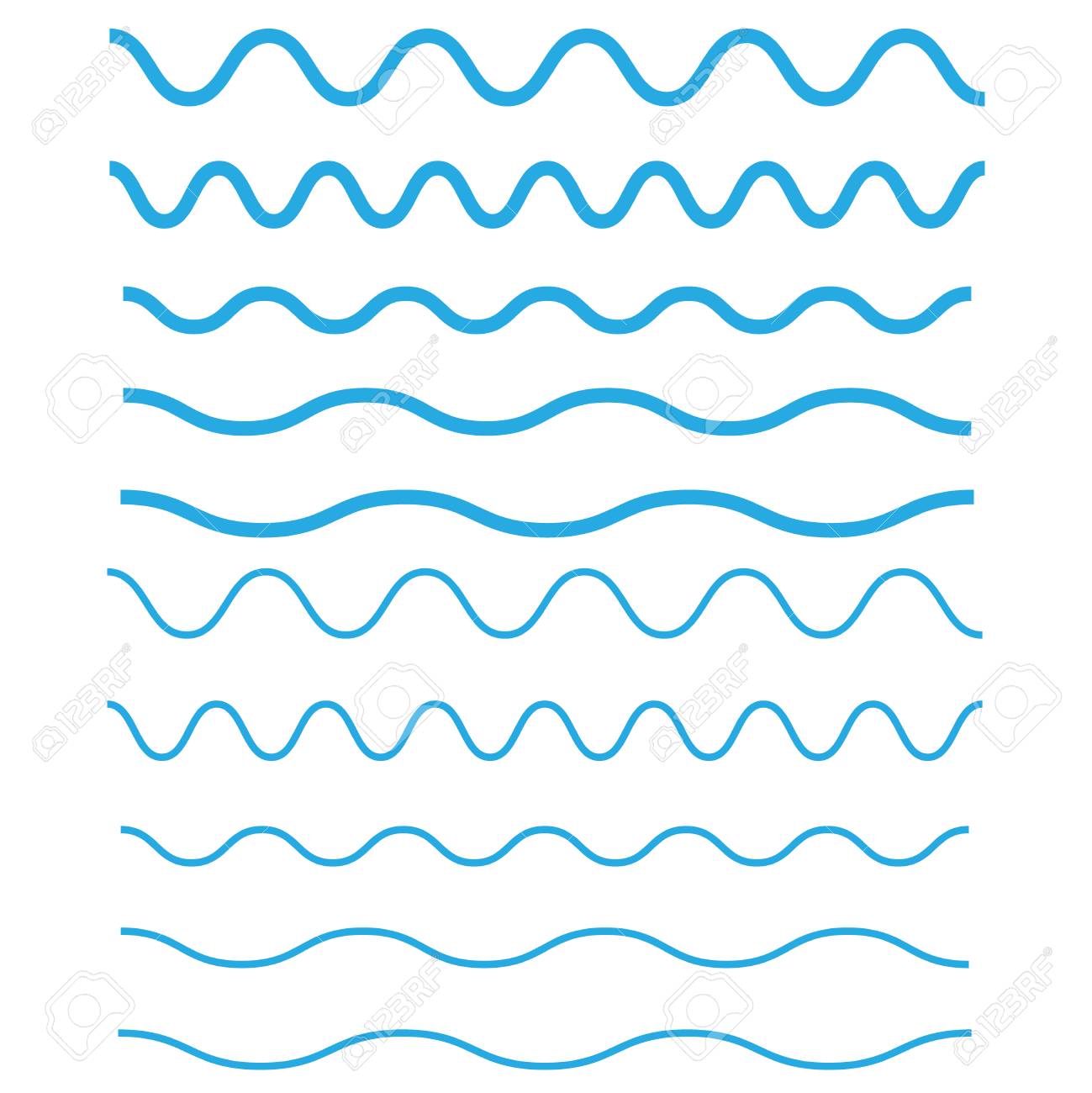
### **Practical learning** **Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * **Digital electronics** is the study of electronic circuits that are used to process and control digital signals. * **Analogue electronics** use relatively high voltage and high current as compared to digital electronics. The voltage and current used in digital electronics are extremely low |

## **Indicative content1****.2: Digital number systems**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Numbering system** It is defined as a system of writing to express numbers. It is the mathematical notation for representing numbers of a given set by using digits or other symbols in a consistent manner  Modern computers do not work with decimal numbers. Instead of, they process binary numbers, groups of Os and 1s. Why binary numbers?  Because electronic devices are most reliable when designed for two – states (Binary) operation either ON or OFF. People do not like working with binary numbers because they are very long.  Entering binary numbers into computer becomes tedious. Therefore, octal and hexadecimal numbers are widely used to compress long strings of binary numbers. Gate is a circuit with one or more input singles but only one output signal. It is used to construct logical circuits which are the building blocks of a computer. Therefore, it is necessary to study the basic operation of gates and logical circuits **Digital system** In general, in any number system there is an ordered set of symbols known as digits with rules defined for performing arithmetic operations like addition, subtraction, multiplication and division.   |  |  |  | | --- | --- | --- | | Number system | Base or radix | Symbol used | | Binary | 2 | 0 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 |  * Decimal numbering system   Decimal numbers are made of decimal digits: 0, 1, 2, 3, 4, 5, 6, 7 ,8 and 9 all decimal numbers are in the base 10.   * Binary numbering system   Binary numbers are made of binary digits(bits) 0 and 1  EX: (1011)2   * Octal numbering system   Octal numbers are made of octal digits : 0,2,3,4,5,6 and 7  EX: (326)8   * Hexadecimal numbering system   Hexadecimal numbers are made of hexadecimal digits: 1,2,3,4,5,6,7,8,9,A,B,C,D,E and F.  EX: 2BF **Numbering system Conversion****Octal to binary Conversion** Octal numbers can be converted into equivalent binary numbers by replacing each octal digit by its 3 – bit binary equivalent.  **Example:** convert 7368 into an equivalent binary number.  Solution: From the above table, the binary equivalents of 7, 3 and 6 are 111, 011 & 110 respectively. Therefore 736 8 = 1 11 0 1 11 1 0 2     * **Binary to octal**   Binary numbers can be converted into equivalent octal numbers by making groups of three bits starting from LSB and moving towards MSB for integer part of the number and then replacing each group of three bits by its octal representation. For fractional part the groupings of three bits are made starting from the binary point.  **Example 1. 15:** a) convert 1001102 to its octal equivalent.  **Solution:**  (100110)2 = (001001110)2 = 11 68 = 1168  **Octal number:**  0.101001102 = 0.101 001 1002 = 0.5 1 48 = 0.5148  **0 5 1 4**     * **Hexadecimal – to decimal**   Hexadecimal numbers can be converted to their equivalent decimal numbers.  **Example 1.16:** Obtain decimal equivalent of hexadecimal number  3A. 2F16 solution  = 3\*161+A\*160+2\*16-1+F\*16-2  = 48+10+2/16+15/162  =58.183610   * **Hexadecimal – to – binary conversion**   Hexadecimal numbers can be converted into equivalent binary numbers by replacing each hex digit by its equivalent 4 – bit binary numbers.  **Example:** convert 2 F 9 A 16 to its equivalent binary number  2F9A16 = 0010 1111 10 01 1010  **2 F 9 A**  = (0010 1111 1001 1010)2   * Binary – to Hexadecimal conversion   Binary numbers can be converted into the equivalent hexadecimal numbers by making groups of four bits starting from LSB and moving towards MSB for integer’s part and then replacing each group of four bits by its hexadecimal representation.  For the fractional part, the above procedure is repeated starting from the bit next to the binary point and moving towards the right.  **Example.** Convert the following binary numbers to their equivalent hexadecimal numbers.  a) 10100110101111  b) 0.00011110101101(for student)  **Solution:**   1. (10100110101111)2= 0010 1001 1010 1111   **2 9 A F**  **= (**29AF)16 |

### **Theoretical learning** **Activity**

* Individual works about conversion between different numbering systems**.**

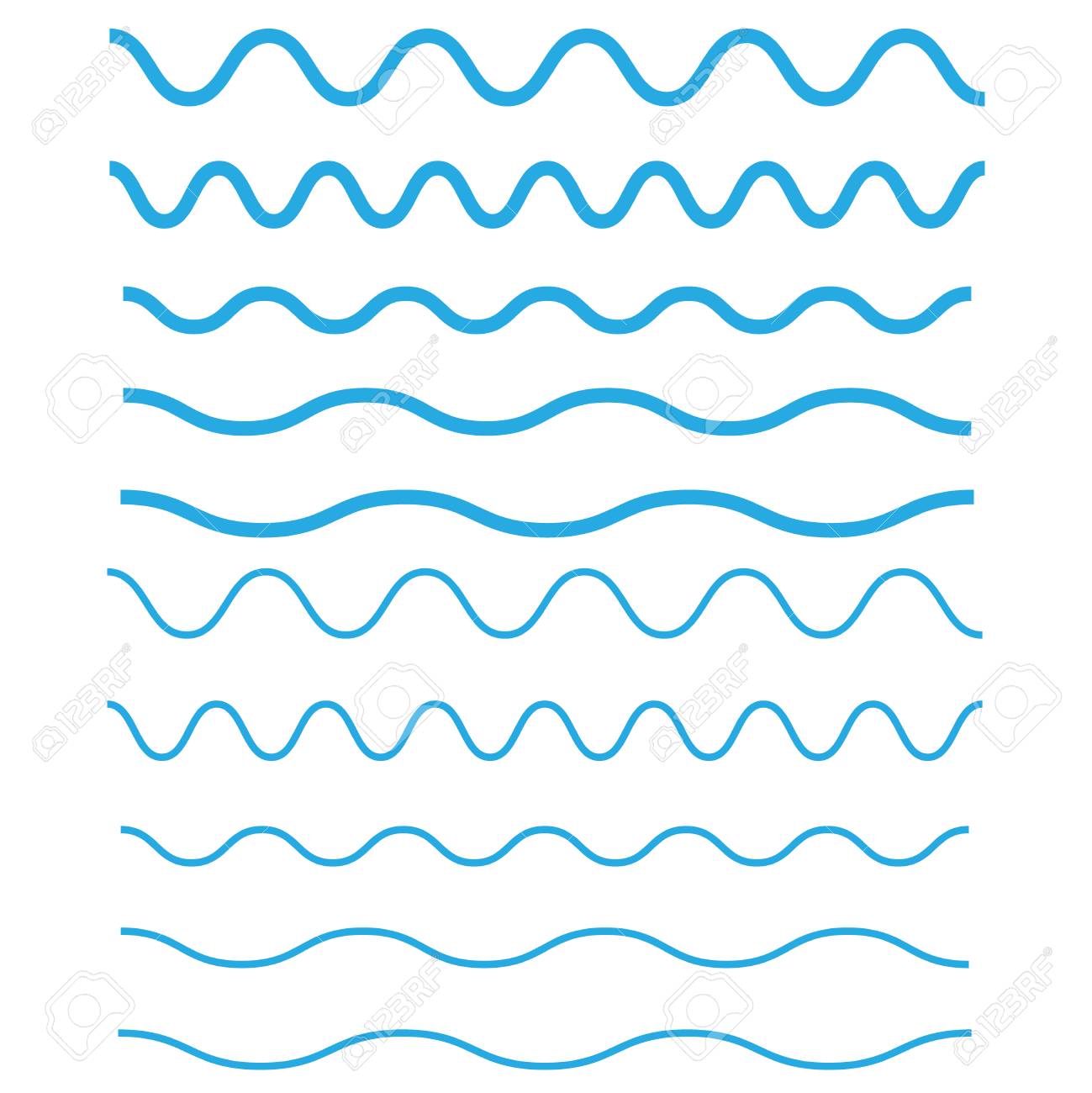
### **Practical learning** **Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * **Numbering system:** is defined as a system of writing to express numbers. It is the mathematical notation for representing numbers of a given set by using digits or other symbols in a consistent manner * **Types of numbering systems:** Decimal, Binary, Octal and hexadecimal |

## **Indicative content1****.3: Digital codes**



|  |
| --- |
| **Introduction to digital codes**  * **Alphanumeric codes**, also called character codes, are binary codes used to represent alphanumeric data. The codes write alphanumeric data, including letters of the alphabet, numbers, mathematical symbols and punctuation marks, in a form that is understandable and proccessable by a computer. These codes enable us to interface input–output devices such as keyboards, printers, etc ………..with the computer Take the letter “A.” when an “A” is typed into a computer, it is immediately translated into a string of 0's and 1's (called a digital code). The digital code for “A” is “01000001.” Every letter of the alphabet has its own digital code. The code for “C” is “01000011.” And the code for “T” is “01010100.”.      * **ASCII (American standard code for information interchange**) is the most common character encoding format for text data in computers and on the internet. In standard ASCII-encoded data, there are unique values for 128 alphabetic, numeric or special additional characters and control codes * **The binary coded decimal (BCD)** is a type of binary code used to represent a given decimal numbering an equivalent binary form. BCD-to-decimal and decimal-to-BCD conversions are very easy and straightforward. It is also far less cumbersome an exercise to represent a given decimal number in an equivalent BCD code than to represent it in the equivalent straight binary form discussed in the previous chapter. The BCD equivalent of a decimal number is written by replacing each decimal digit in the integer and fractional parts with its four-bit binary equivalent. As an example, the BCD equivalent of (23.15)10 is written as (0010 0011.0001 0101) BCD. The BCD code described above is more precisely known as the 8421 BCD code, with 8, 4, 2 and 1 representing the weights of different bits in the four-bit groups, starting from MSB and proceeding towards LSB.      * **The excess-3 code** is another important BCD code. It is particularly significant for arithmetic operations as it overcomes the shortcomings encountered while using the BCD code to add two decimal digits whose sum exceeds 9. The excess-3 code has no such limitation, and it considerably simplifies arithmetic operations * **Gray code** is an un-weighted binary code in which two successive values differ only by 1 bit. Owing to this feature, the maximum error that can creep into a system using the binary Gray code to encode data is much less than the worst-case error encountered in the case of straight binary encoding. Table below lists the binary and Gray code equivalents of decimal numbers 0–15.  **Digital code conversion**Binary–Gray Code Conversion A given binary number can be converted into its Gray code equivalent by going through the following steps:   1. Begin with the most significant bit (MSB) of the binary number. The MSB of the Gray code equivalent is the same as the MSB of the given binary number. 2. The second most significant bit, adjacent to the MSB, in the Gray code number is obtained by adding the MSB and the second MSB of the binary number and ignoring the carry, if any. That is, if the MSB and the bit adjacent to it are both ‘1’, then the corresponding Gray code bit would be a ‘0’. The third most significant bit, adjacent to the second MSB, in the Gray code number is obtained by adding the second MSB and the third MSB in the binary number and ignoring the carry, if any. 3. The process continues until we obtain the LSB of the Gray code number by the addition of the LSB and the next higher adjacent bit of the binary number.   The conversion process is further illustrated with the help of an example showing step-by-step .  conversion of (1011)2 into its Gray code equivalent:  Binary 1011  **Gray code** 1- - -  Binary 1011  **Gray code** 11- -  Binary 1011  **Gray code** 111-  Binary 1011  **Gray code** 1110   * Gray Code–Binary Conversion   A given Gray code number can be converted into its binary equivalent by going through the following steps:   1. Begin with the most significant bit (MSB). The MSB of the binary number is the same as the MSB of the Gray code number. 2. The bit next to the MSB (the second MSB) in the binary number is obtained by adding the MSB in the binary number to the second MSB in the Gray code number and disregarding the carry, if any. 3. The third MSB in the binary number is obtained by adding the second MSB in the binary number to the third MSB in the Gray code number. Again, carry, if any, is to be ignored. 4. The process continues until we obtain the LSB of the binary number.   The conversion process is further illustrated with the help of an example showing step-by-step  conversion of the Gray code number 1110 into its binary equivalent:  Gray code 1110  **Binary** 1- - -  Gray code 1110  **Binary** 10 - -  Gray code 1110  **Binary** 101  Gray code 1110  **Binary** 1011 |

### **Theoretical learning** **Activity**

* Individual work about the conversion between different digital codes

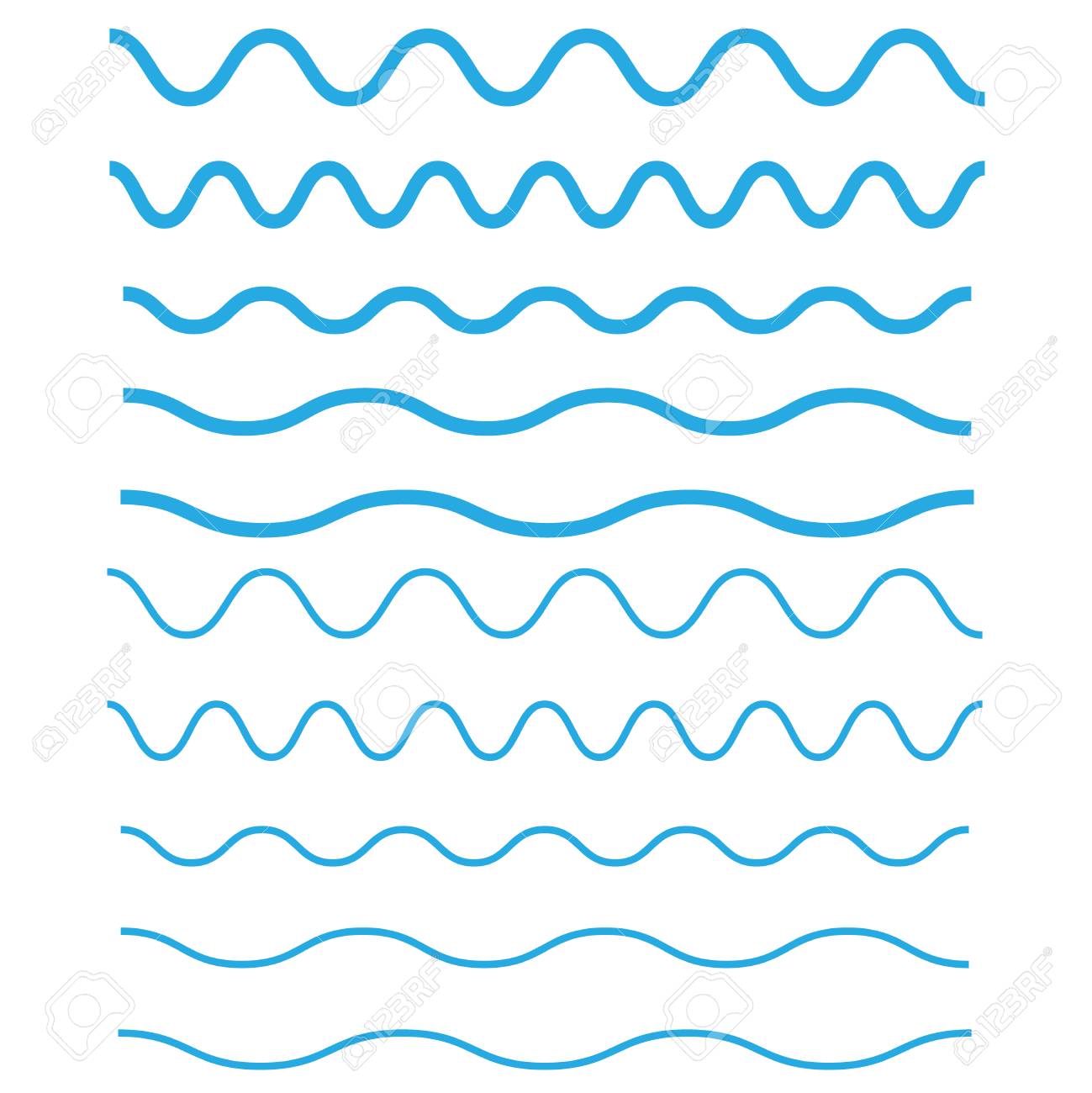
### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * **Alphanumeric codes**, also called character codes, are binary codes used to represent alphanumeric data * **ASCII (American standard code for information interchange**) is the most common character encoding format for text data in computers and on the internet. * **The binary coded decimal (BCD)** is a type of binary code used to represent a given decimal numbering an equivalent binary form * **The excess-3 code** is another important BCD code. It is particularly significant for arithmetic operations as it overcomes the shortcomings encountered while using the BCD code to add two decimal digits whose sum exceeds 9 * **Gray code** is an un-weighted binary code in which two successive values differ only by 1 bit. Owing to this feature, the maximum error that can creep into a system using the binary Gray code to encode data is much less than the worst-case error encountered in the case of straight binary encoding. Table below lists the binary and Gray code equivalents of decimal numbers 0–15. |

## **Indicative content1.3: Digital arithmetic**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Digital arithmetic operations** We all are familiar with the arithmetic operations such as additions, subtraction, multiplication and division of decimal numbers. Similar operations can be performed on binary numbers. Binary arithmetic is much simpler than decimal arithmetic because here only two digits, 0 and 1 are involved.   * Binary Addition  Rules of Binary Addition  * 0 + 0 = 0 * 0 + 1 = 1 * 1 + 0 = 1 * 1 + 1 = 0, and carry 1 to the next more significant bit .For example*,*  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 00011010 + 00001100 = 00100110 |  | *1  1* |  | *carries* | | 0  0  0  1  1  0  1  0 | = | 26(base 10) | | + 0  0  0  0  1  1  0  0 | = | 12(base 10) | | 0  0  1  0  0  1  1  0 | = | 38(base 10) | |  | | | | | | 00010011 + 00111110 = 01010001 |  | *1  1  1  1  1* |  | *carries* | | 0  0  0  1  0  0  1  1 | = | 19(base 10) | | + 0  0  1  1  1  1  1  0 | = | 62(base 10) | | 0  1  0  1  0  0  0  1 | = | 81(base 10) |  * **Binary Subtraction**  Rules of Binary Subtraction  * 0 - 0 = 0 * 0 - 1 = 1, and borrow 1 from the next more significant bit * 1 - 0 = 1 * 1 - 1 = 0   We can subtract one binary number from another by using the standard techniques adapted for decimal numbers (subtraction of each bit pair, right to left "borrowing" as needed from bits to the left).  However, if we can leverage the already familiar (and easier) technique of binary addition to subtract, that would be better. As we just learned, we can represent negative binary numbers by using the "two's complement" method and a negative place-weight bit. Here, we'll use those negative binary numbers to subtract through addition. Here's a sample problem:  Subtraction: 710 - 510 Addition equivalent: 710 + (-510)  If all we need to do is represent seven and negative five in binary (two's complemented) form, all we need is three bits plus the negative-weight bit:  Positive seven = 01112  Negative five = 10112  Now, let's add them together:  1111 <--- Carry bits  0111  + 1011  ------  10010  |  Discard extra bit  Answer = 00102  Since we've already defined our number bit field as three bits plus the negative-weight bit, the fifth bit in the answer (1) will be discarded to give us a result of 00102, or positive two, which is the correct answer.  Another way to understand why we discard that extra bit is to remember that the leftmost bit of the lower number possesses a negative weight, in this case equal to negative eight.  When we add these two binary numbers together, what we're actually doing with the MSBs is subtracting the lower number's MSB from the upper number's MSB. In subtraction, one never "carries" a digit or bit on to the next left place-weight.  Let's try another example, this time with larger numbers. If we want to add -2510 to 1810, we must first decide how large our binary bit field must be.  To represent the largest (absolute value) number in our problem, which is twenty-five, we need at least five bits, plus a sixth bit for the negative-weight bit.  Let's start by representing positive twenty-five, then finding the two's complement and putting it all together into one numeration:  +2510 = 0110012 (showing all six bits)  One's complement of 110012 = 1001102  One's complement + 1 = two's complement = 1001112  -2510 = 1001112  Essentially, we're representing negative twenty-five by using the negative-weight (sixth) bit with a value of negative thirty-two, plus positive seven (binary 1112).  Now, let's represent positive eighteen in binary form, showing all six bits:  1810 = 0100102  Now, let's add them together and see what we get:  11 <--- Carry bits  100111  + 010010  --------  111001  Since there were no "extra" bits on the left, there are no bits to discard. The leftmost bit on the answer is a 1, which means that the answer is negative, in two's complement form, as it should be.   * **Binary Multiplication**  Rules of Binary Multiplication  * 0 x 0 = 0 * 0 x 1 = 0 * 1 x 0 = 0 * 1 x 1 = 1, and no carry or borrow bits   *For example,*   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 00101001 × 00000110 = 11110110 |  | 0  0  1  0  1  0  0  1 | = | 41(base 10) | | × 0  0  0  0  0  1  1  0 | = | 6(base 10) | | 0  0  0  0  0  0  0  0 |  | | | 0  0  1  0  1  0  0  1 |  | | | 0  0  1  0  1  0  0  1 |  | | | 0  0  1  1  1  1  0  1  1  0 | = | 246(base 10) | |  | | | | | | 00010111 × 00000011 = 01000101 |  | 0  0  0  1  0  1  1  1 | = | 23(base 10) | | × 0  0  0  0  0  0  1  1 | = | 3(base 10) | | *1  1  1  1  1* |  | *carries* | | 0  0  0  1  0  1  1  1 |  | | | 0  0  0  1  0  1  1  1 |  | | | 0  0  1  0  0  0  1  0  1 | = | 69(base 10) |   **Note:**  The rules of binary multiplication are the same as the truths of the [**AND**](http://academic.evergreen.edu/projects/biophysics/technotes/program/logic.htm#gates)  **Another Method:**  Binary multiplication is the same as repeated binary addition; add the multiband to itself the multiplier number of times.  For example*,*   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 00001000 × 00000011 = 00011000 |  | *1* |  | *carries* | | 0  0  0  0  1  0  0  0 | = | 8(base 10) | | 0  0  0  0  1  0  0  0 | = | 8(base 10) | | + 0  0  0  0  1  0  0  0 | = | 8(base 10) | | 0  0  0  1  1  0  0  0 | = | 24(base 10) |  * **Binary Division**   Binary division is the repeated process of subtraction, just as in decimal division.  For example   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 00101010 ÷ 00000110 = 00000111 |  |  |  |  |  |  |  |  | 1 | 1 | 1 | = | 7(base 10) | |  | | | | | | | | | | 1  1  0 | ) | 0 | 0 | ~~1~~ | *1*0 | 1 | 0 | 1 | 0 | = | 42(base 10) | |  |  |  |  | - | 1 | 1 | 0 |  |  | = | 6(base 10) | |  | |  | | | |  | | |  |  |  |  | *1* |  |  |  |  | *Borrows* | |  |  |  | ~~1~~ | ~~0~~ | *1*0 | 1 |  | |  |  |  | - | 1 | 1 | 0 |  | |  | | |  | | | |  | |  |  |  |  |  | 1 | 1 | 0 | |  |  |  |  | - | 1 | 1 | 0 | |  | | | |  | | | | |  |  |  |  |  |  |  | 0 | | **(2)** | | | | | | | | | | | | | | | 10000111 ÷ 00000101 = 00011011 |  |  |  |  |  |  | 1 | 1 | 0 | 1 | 1 | = | 27(base 10) | |  | | | | | | | | | | 1  0  1 | ) | ~~1~~ | ~~0~~ | ~~0~~ | *1*0 | 0 | 1 | 1 | 1 | = | 135(base 10) | |  |  | - | 1 | 0 | 1 |  |  |  |  | = | 5(base 10) | |  | | | |  | | | | |  |  | 1 | ~~1~~ | *1*0 |  |  |  | |  | - | 1 | 0 | 1 |  |  |  | |  |  | | | |  | | | |  |  |  |  | 1 | 1 |  |  | |  |  |  | - |  | 0 |  |  | |  | | |  | | |  | | |  |  |  |  | 1 | 1 | 1 |  | |  |  |  | - | 1 | 0 | 1 |  | |  | | |  | | | |  | |  |  |  |  |  | 1 | 0 | 1 | |  |  |  |  | - | 1 | 0 | 1 | |  | | | |  | | | | |  |  |  |  |  |  |  | 0 |  **Complement of a Number**One’s complement representation Representing a signed number with 1's complement is done by changing all the bits that are 1 to 0 and all the bits that are 0 to 1. Reversing the digits in this way is also called complementing a number. Let's look at an example in 4-bit arithmetic. How can we represent the number -510 in 1's complement?   1. First, we write the positive value of the number in binary.   **0101 (+5)**   1. Next, we reverse each bit of the number so 1's become 0's and 0's become 1's   **1010 (-5)**   1. Last, we add 1 to the number.   **1011 (-5)** Two’s complement representation We see that both representations use the most significant bit to represent the sign. We also notice that we only have one way to represent 0 in 2's complement. This is an advantage because it simplifies representation of signed numbers. As with 1's complement, only negative values need to be complemented in 2's complement. The positive values are the same as the normal binary numbers. You should verify for yourself that the negative values are correct by using the steps below to produce this table.  Let's consider how we would solve our problem of subtracting 110 from 710 using 1's complement.   |  |  | | --- | --- | | 1. First, we need to convert 00012 to its negative equivalent in 1's complement. | **0111(7)**  **- 0001- (1)** | | 1. To do this we change all the 1's to 0's and 0's to 1's. Notice that the most-significant digit is now 1 since the number is negative. | **0001 -> 1110** | | 1. Next, we add the negative value we computed to 01112. This gives us a result of 101012. | **0111(7)**  **+ 1110+(-1)**  **10101 (?)** | | 1. Notice that our addition caused an overflow bit. Whenever we have an overflow bit in 1's complement, we add this bit to our sum to get the correct answer. If there is no overflow bit, then we leave the sum as it is. | **0101**  **+ 1**  **0110 (6)** | | 1. This gives us a final answer of 01102 (or 610). | **0111 (7)**  **- 0001- (1)**  **0110 (6)** |   Now let's look at an example where our problem does not generate an overflow bit. We will subtract 710 from 110 using 1's complement.   |  |  | | --- | --- | | 1. First, we state our problem in binary. | **0001 (1)**  **- 0111- (7)** | | 1. Next, we convert 01112 to its negative equivalent and add this to 00012. | **0001 (1)**  **+ 1000+(-7)**  **1001 (?)** | | 1. This time our result does not cause an overflow, so we do not need to adjust the sum. Notice that our final answer is a negative number since it begins with a 1. Remember that our answer is in 1's complement notation so the correct decimal value for our answer is -610 and not 910. | **0001 (1)**  **+ 1000+(-7)**  **1001 (-6)** | |

### **Theoretical learning** **Activity**

* Individual work about binary arithmetic operations
* Individual work about one’s and two’s complements representation

### **Practical learning** **Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * There is two types of number’s complement: One’s and two’s complements * Binary arithmetic operations: Addition, Multiplication, Subtraction and Division |

### **Learning outcome 1: formative assessment**

**Written assessment**

* + 1. **MULTIPLE CHOICE QUESTIONS**

For the following questions use the decimal to binary and hexadecimal to binary conversion techniques and choose the correct answer by circling the letter associated with it.

* 1. The binary code of (21.125)10 is

1. 10101.001
2. 10100.001
3. 10101.010
4. 10100.111
   1. The decimal equivalent of (F8E6)16 is:
5. 2479
6. 6171810
7. 6371810
8. None of the above

Read the following statements and answer by true or false

* + 1. Binary number system made of 0,1,2,3,4,5,6,7,8 and 9 digits
    2. In hexadecimal number system only 0’s and 1’s digits are used.
    3. **SHORT AND OPEN-ENDED QUESTIONS**

1. What is a numbering system?
2. Do the following conversions:

(a) eight-bit 2’s complement representation of (−23)10;

(b) The decimal equivalent of (00010111)2 represented in 2’s complement form

3. Give the next three numbers in each of the following hex sequences:

(a) 4A5, 4A6, 4A7, 4A8,

(b) B998, B999,

4. Determine the hexadecimal equivalent of (82.25)10

**ANSWERS**

**Choose the correct answer**:

1) The binary code of (21.125)10 is

* + 1. **10101.001**
    2. 10100.001
    3. 10100.010
    4. 10100.111

**Answer: A**

1. The equivalent of (F8E6)16 is:
   * 1. 2479
     2. 6171810
     3. **6371810**
     4. None of the above

**Answer: C**

Read the following statements and answer by true or false

* + 1. Binary number system made of 0,1,2,3,4,5,6,7,8 and 9 digits/ **False**
    2. In hexadecimal number system only 0’s and 1’s digits are used./ **False**
    3. **SHORT AND OPEN-ENDED QUESTIONS**

1. What is a numbering system?

**Answer**: The number systems are used quite frequently in the field of digital electronics and computers. An example of this is when a user key-in some data into the computer, s(he), will do it using decimal number system i.e. the system we all have used for several years for doing arithmetic problems

2. Do the following conversions:

(a) eight-bit 2’s complement representation of (−23)10; **Answer: 11101001**

(b) The decimal equivalent of (00010111)2 represented in 2’s complement form **Answer**: **+23**

3. Give the next three numbers in each of the following hex sequences:

(a) 4A5, 4A6, 4A7, 4A8, **Answer**: **4A9, 4AA, 4AB**

(b) B998, B999, **Answer**: **B99A, B99B, B99C**

4. Determine the hexadecimal equivalent of (82.25)10

**Answer: (52.4)16**

**References:**

* Floyd, T. L. (2015). Digital Fundamentals. England: 11th edition Pearson.
* A. (2007). Digital Electronics Principles, Devices and Applications. India: John Wiley.
* R.Tokheim. (2014). Digital Electronics Principles and Applications. USA: 8th ed.
* Theraja, B. (n.d.). A textbook of Electrical Technology Vol IV-Electronic Devices and circuits.

# 

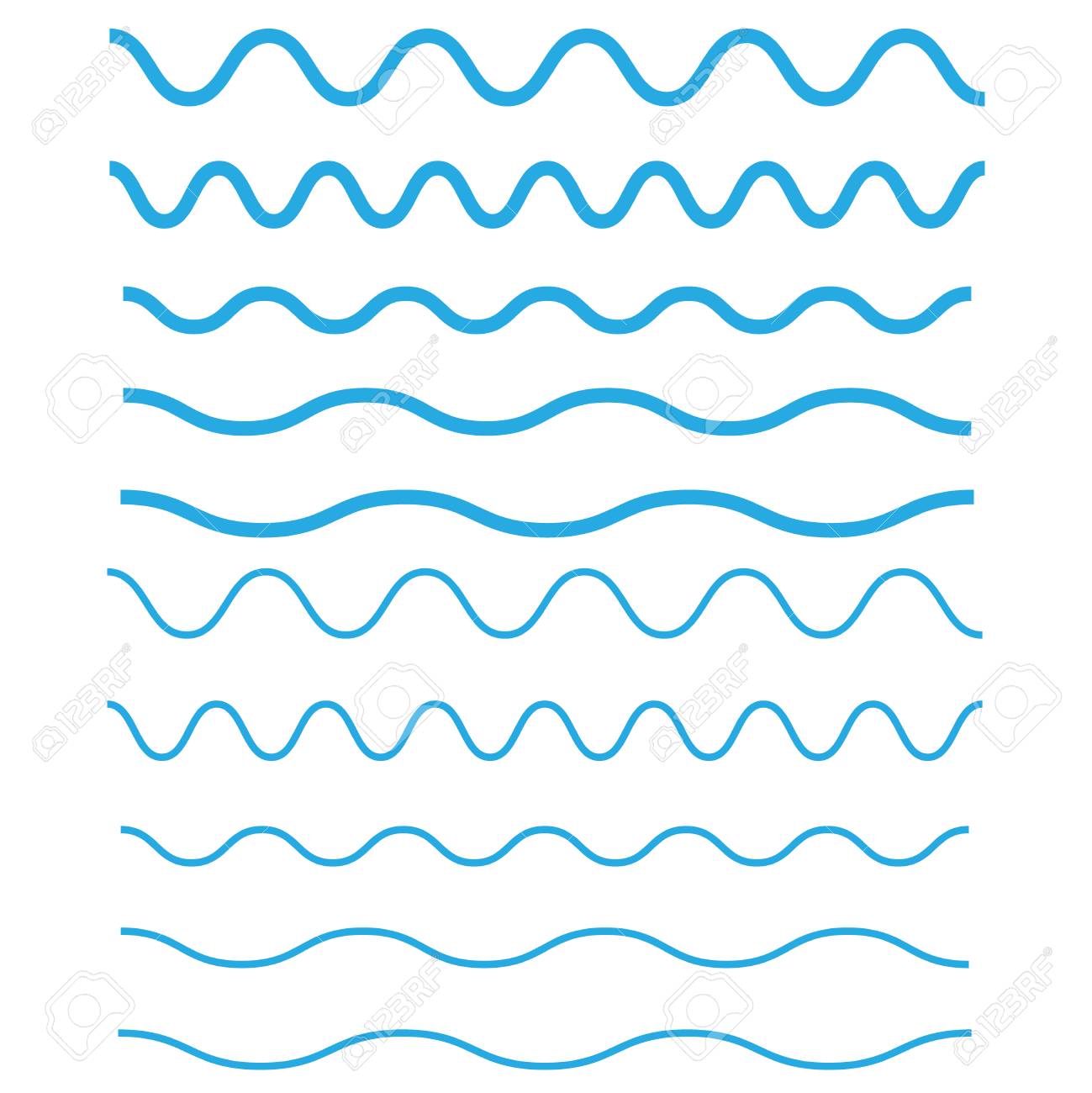
# **Learning outcome 2: Apply logic gates**

|  |
| --- |
|  |

|  |
| --- |
| **Indicative contents**  **2.1 : Introduction to logic gates**  **2.2:** **Types of logic gates**  **2.3:** **Apply Logic families**  **2.4:** **Apply Digital ICs** |

|  |  |  |
| --- | --- | --- |
| **Duration:15hrs** | | |
| **Learning outcome 2 objectives:**  By the end of the learning outcome, the trainees will be able to:   1. Apply properly logic gates according to their types 2. Identify clearly logic families according to their types 3. Classify properly digital ICs according to their size, technology & applications 4. Apply and simulate properly Logic circuits according to work to be done | | |
| **Resources** | | |
| **Equipment** | **Tools** | **Materials** |
| Computer  Projector  Multimeters  DC power supply | Books  Multisim software  Proteus software  universal pliers | Markers  Chalks  board  Internet  logic gates  ICs  breadboard  jumper wires  electronic components |
| **Advance preparation:**   * Read the definition of logic gates, logic families and digital ICs * Deeply research on the types of logic gates, logic families and digital ICs * Deeply research on the applications of logic gates, logic families and ICs | | |

## **Indicative content2.1: Types of logic gates**



|  |
| --- |
| * **Logic gates** are electronic circuits that can be used to implement the most elementary logic expressions, also known as Boolean expressions. The logic gate is the most basic building block of combinational logic. There are three basic logic gates, namely the OR gate, the AND gate and the NOT gate. Other logic gates that are derived from these basic gates are the NAND gate, the NOR gate, the EXCLUSIVE-OR gate and the EXCLUSIVE-NOR gate * **Truth table**A truth table lists all possible combinations of input binary variables and the corresponding outputs of a logic system. The logic system output can be found from the logic expression, often referred to as the Boolean expression that relates the output with the inputs of that very logic system.When the number of input binary variables is only one, then there are only two possible inputs, i.e. ‘0’ and ‘1’. If the number of inputs is two, there can be four possible input combinations, i.e. 00, 01, 10 and 11, similarly, for three input binary variables; the number of possible input combinations becomes eight, i.e. 000, 001, 010, 011, 100, 101, 110 and 111. This statement can be generalized to say that, if a logic circuit has n binary inputs, its truth table will have 2𝑛 possible input combinations. * **An OR gate** performs an ORing operation on two or more than two logic variables. The OR operation on two independent logic variables A and B is written as Y = A+B and reads as Y equals A OR B and not as A plus B. An OR gate is a logic circuit with two or more inputs and one output. The output of an OR gate is LOW only when all of its inputs are LOW. For all other possible input combinations, the output is HIGH.   The operation of a two-input OR gate is explained by the logic expression  Y = A+B  As an illustration, if we have four logic variables and we want to know the logical output of (A+ B+C +D), then it would be the output of a four-input OR gate with A, B, C and D as its inputs.     * **An AND gate** is a logic circuit having two or more inputs and one output. The output of an AND gate is HIGH only when all of its inputs are in the HIGH state. In all other cases, the output is LOW. Figures 4.4(a) and (b) show the logic symbols of three-input and four-input AND gates respectively. Figure 4.4(c) gives the truth table of a four-input AND gate.   The AND operation on two independent logic variables A and B is written as Y = A.B and reads as Y equals A AND B and not as A multiplied by B. Here, A and B are input logic variables and Y is the output. An AND gate performs an ANDing operation     * **A NOT gate** is a one-input, one-output logic circuit whose output is always the complement of the input. That is, a LOW input produces a HIGH output, and vice versa. When interpreted for a positive logic system, logic ‘0’ at the input produces a logic ‘1’ at the output, and vice versa. It is also known as a ‘complementing circuit’ or an ‘inverting circuit’. Figure 4.6 shows the circuit symbol and the truth table.   The NOT operation on a logic variable X is denoted as X or X’. That is, if X is the input to a NOT circuit, then its output Y is given by Y = X or X’. And reads as Y equals NOT X. Thus, if X = 0, Y = 1 and if X = 1, Y = 0.  The output will be permanently in logic ‘0’ state as the two inputs can never be in logic ‘1’ state together owing to the presence of the inverter.     **Universal logic gates** **NAND** stands for NOT AND. An AND gate followed by a NOT circuit makes it a NAND gate  The truth table of a NAND gate is obtained from the truth table of an AND gate by complementing the output entries  The output of a NAND gate is logic ‘0’ when all its inputs are logic  ‘1’. For all other input combinations, the output is logic ‘1’. NAND gate operation is logically expressed as: y=( A.B)’  In general, the Boolean expression for a NAND gate with more than two inputs can be written as Y= (A.B.C.D….)’    **NOR** stands for NOT OR.An OR gate followed by a NOT circuit makes it a NOR gate  The truth table of a NOR gate is obtained from the truth table of an OR gate by complementing the output entries. The output of a NOR gate is a logic ‘1’ when all its inputs are logic ‘0’. For all other input combinations, the output is logic ‘0’. The output of a two-input NOR gate is logically expressed as Y=(A+B)’     * **The EXCLUSIVE-OR gate**, commonly written as EX-OR gate, is a two-input, one-output gate. Figures 4.8(a) and (b) respectively show the logic symbol and truth table of a two-input EX-OR gate. As can be seen from the truth table, the output of an EX-OR gate is logic ‘1’ when the inputs are unlike and logic ‘0’ when the inputs are like. EX-OR logic functions can be implemented using more than one two-input gates. The truth table of a multiple-input EX-OR function can be expressed as follows. The output of a multiple-input EX-OR logic function is a logic ‘1’ when the number of 1s in the input sequence is odd and a logic ‘0’ when the number of 1s in the input sequence is even, including zero. That is, an all 0s input sequence also produces a logic ‘0’ at the output   The output of a two-input EX-OR gate is expressed by Y = (A⊕B) = A’B+AB’     * **EXCLUSIVE-NOR** (commonly written as EX-NOR) means NOT of EX-OR, i.e. the logic gate that we get by complementing the output of an EX-OR gate.   Figure shows its circuit symbol along with its truth table.  The truth table of an EX-NOR gate is obtained from the truth table of an EX-OR gate by complementing the output entries. Logically,     **Applications of logic gates** The range of usage of logic gates is quite extensive, however some of its applications are as follows:   1. In manufacturing more complex devices e.g. Binary counters 2. In decision making regarding automatic control of machine and different industrial process 3. In calculators and computers 4. In digital processing of communications 5. In musical instruments, games and different domestic appliances |

### **Theoretical learning Activity**

* Trainees within group of five discuss about different types of logic gates

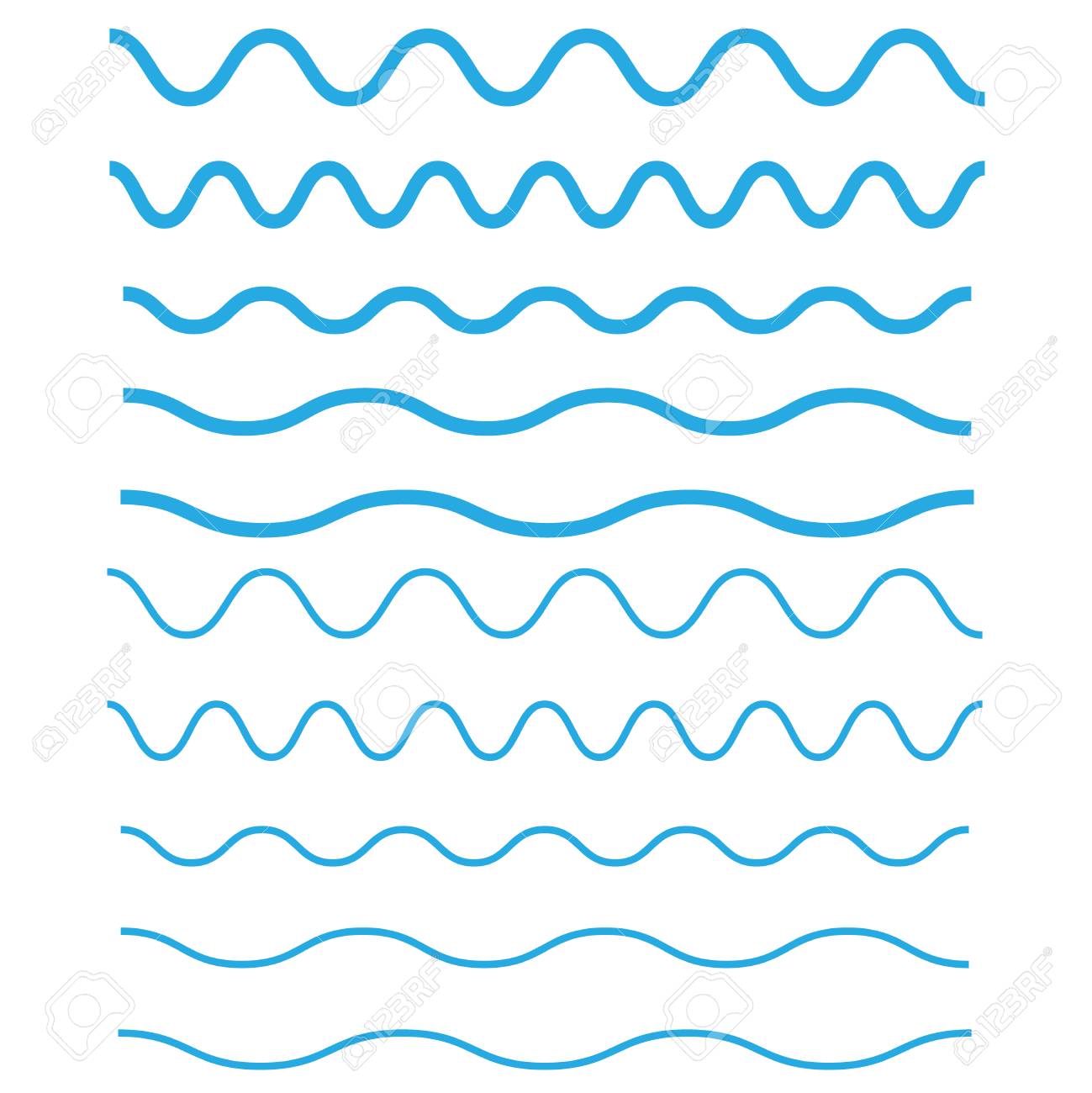
### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

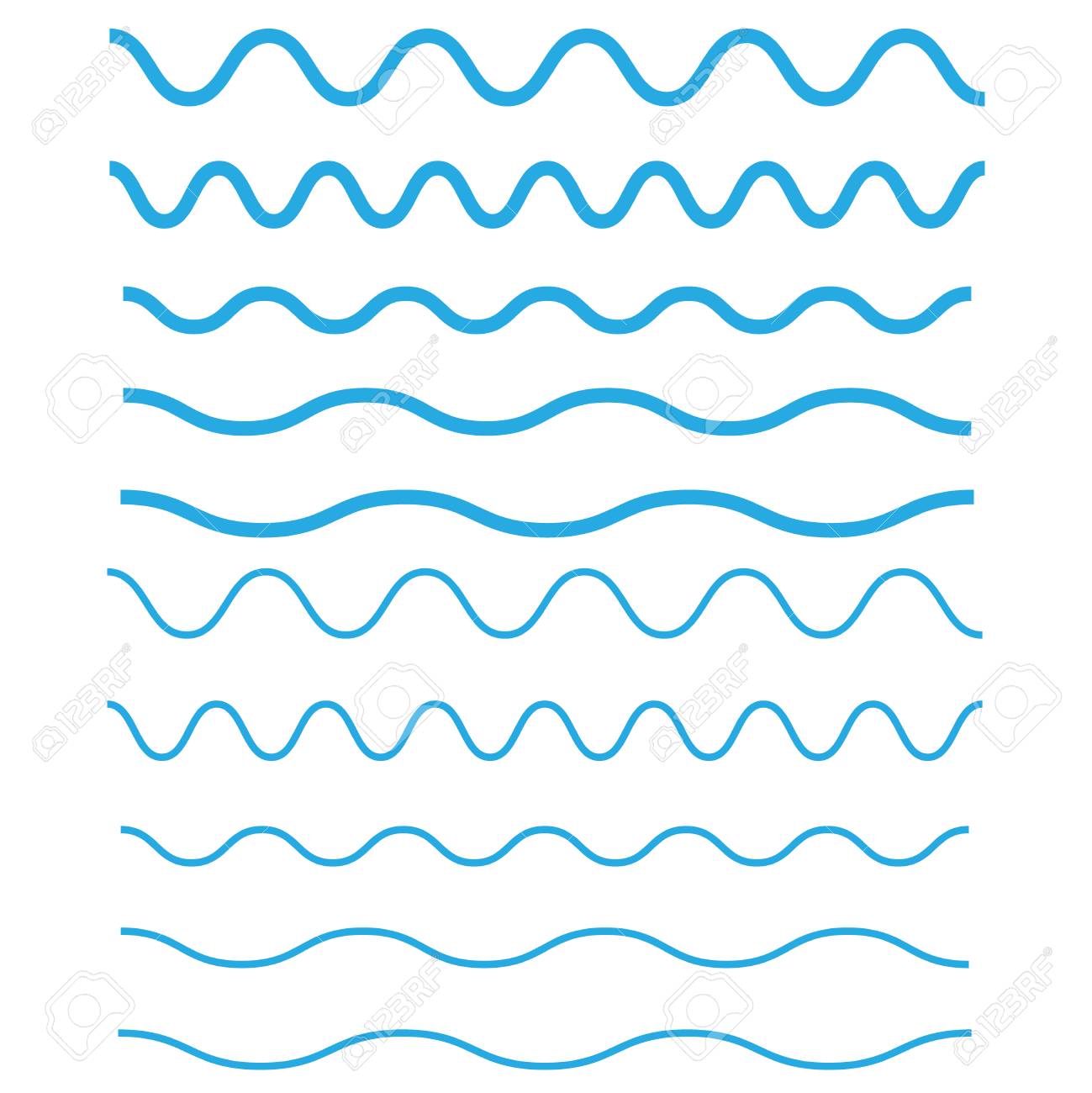
|  |
| --- |
| * Logic gate is a device that acts as a building block for digital circuits * There are three types of logic gates * Logic gates are used in many circuits like a push button lock, light activated burglar alarm, safety thermostat, an automatic watering system etc. |

## **Indicative content** **2.2: Apply Logic families**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Types of logic families** A logic family of monolithic digital integrated circuit devices is a group of electronic logic gates constructed using one of several different designs, usually with compatible logic levels and power supply characteristics within a family   * **TTL That stands for Transistor-Transistor Logic,** a specific family of digital logic circuits used in electronics. They were once widely popular but have largely been replaced by CMOS (Complementary Metal-Oxide-Semiconductor) logic due to their lower power consumption and higher operating speeds. A family of integrated circuits (ICs) built using bipolar junction transistors (BJTs).They perform basic logic operations like AND, OR, NOT, NAND, and NOR.  **Working principle of TTL** TTL circuits rely on the voltage levels of the transistors to represent logic states.  High voltage (around 5V) represents logic HIGH, while low voltage (around 0V) represents logic LOW.  Each logic gate acts as a voltage amplifier, boosting the signal strength and ensuring clean transitions between logic states. **Types of TTL** Standard TTL (74LS series): The most common type, offering good speed and noise immunity.  Low-power TTL (74H series): Consumes less power but slower than standard TTL.  Schottky TTL (74S series): Faster than standard TTL but has higher power consumption.  Advantages of TTL:  Simple design, making them easy to understand and troubleshoot.  Wide range of available ICs for various logic functions.  Good noise immunity, making them reliable in noisy environments.  **Disadvantages of TTL**:  Higher power consumption compared to CMOS.  Slower switching speeds than CMOS.  More susceptible to damage from electrostatic discharge (ESD)  ECL (Emitter coupled logic)  ECL is a high-performance logic family known for its ****exceptional speed****, making it a champion in applications requiring lightning-fast processing. Unlike TTL's reliance on bipolar junction transistors (BJTs), ECL utilizes a ****current-mode logic**** approach built around differential amplifiers  **Working principle**  Instead of voltage levels, ECL circuits encode logic values using differential current flowing through pairs of transistors.  High logic (logic 1) is represented by a small current difference between the two transistors in a pair.  Low logic (logic 0) corresponds to a larger current difference.  Differential amplifiers amplify these current differences, maintaining signal integrity and enabling rapid switching between logic states.  **Advantages of ECL**:  Blazing-fast speeds: ECL reigns supreme in speed, boasting switching times in the picoseconds range (billionths of a second) compared to TTL's nanosecond (millionth of a second) range. This makes it ideal for high-frequency applications like telecommunications, instrumentation, and high-speed data processing.  Superior noise immunity: The differential current approach grants ECL excellent noise immunity, making it less susceptible to electrical interference compared to voltage-based logic families.  Wide operating temperature range: ECL circuits can function reliably over a wider range of temperatures than TTL, making them suitable for demanding environments.  Disadvantages of ECL:  High power consumption: The constant current flow in ECL circuits translates to significantly higher power consumption compared to TTL.   This can be a major drawback in battery-powered devices or applications requiring thermal management.  Increased complexity: ECL circuits are more complex to design and fabricate than TTL, leading to higher costs and reduced component availability.  Sensitive to output loading: Improper loading of ECL outputs can affect signal integrity and circuit performance.  Current Use of ECL  High-speed networking equipment: Routers, switches, and other network infrastructure often rely on ECL for their lightning-fast data processing capabilities.  Test and measurement instruments: Oscilloscopes, logic analyzers, and other high-precision equipment utilize ECL for accurate and real-time signal capture and analysis.  Military and aerospace applications: ECL's speed and reliability make it suitable for demanding environments in radar systems, communication equipment, and avionics.  The MOS logic family, also known as Metal-Oxide-Semiconductor, is a broad category of digital integrated circuits (ICs) built using MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) technology. These circuits handle digital signals represented by voltage levels (typically 0 and 1), and they offer several advantages over other logic families like TTL (Transistor-Transistor Logic):  ****Advantages of MOS logic****  ****High Density:**** Thanks to the smaller size of MOSFETs compared to bipolar transistors, MOS circuits can pack more logic gates onto a single chip, leading to smaller and more compact devices.  ****Low Power Consumption:**** MOSFETs operate with lower currents than bipolar transistors, resulting in lower power consumption for MOS circuits. This makes them more suitable for battery-powered devices and portable electronics.  ****High Noise Immunity:**** MOS circuits are less susceptible to electrical noise interference compared to TTL circuits. This improves circuit stability and reliability.  ****Higher Operating Speeds:**** With advancements in fabrication technology, MOS circuits can now achieve faster switching speeds than some TTL families.  ****Scalability:**** MOSFETs can be easily scaled down in size, allowing for continuous advancements in miniaturization and increased transistor density. ****Types of MOS Logic Families**** The MOS logic family encompasses several subcategories with different characteristics and applications:  ****NMOS (N-channel MOS):**** The simplest and earliest MOS family, using only N-channel MOSFETs. Offers low power consumption but limited operating speeds.  ****PMOS (P-channel MOS):**** Uses P-channel MOSFETs, offering good noise immunity but consuming more power and having slower speeds than NMOS.  ****Applications of MOS logic:****  MOS circuits are used in a vast array of electronic devices and systems, including:  ****Microprocessors and microcontrollers:**** The core of computers and many other intelligent devices.  ****Memory chips:**** SRAM, DRAM, and flash memory chips for storing data.  ****Logic gates and logic ICs:**** Building blocks for implementing various digital functions.  ****Application-Specific Integrated Circuits (ASICs):**** Custom-designed circuits for specific needs.  ****Analogue-to-Digital Converters (ADCs) and Digital-to-Analogue Converters (DACs):**** Converting between analogue and digital signals.  ****Sensors and actuators:**** Transforming physical phenomena into digital signals and controlling physical devices. **Comparison of main logic families** 1. TTL (Transistor-Transistor Logic):  Advantages: Simple design, good noise immunity, wide range of ICs.  Disadvantages: High power consumption, slower speed, susceptible to ESD.  Use cases: Educational projects, retro-computing, and some specific industrial applications.  2. CMOS (Complementary Metal-Oxide-Semiconductor):  Advantages: Low power consumption, high speed, excellent noise immunity, high packing density.  Disadvantages: More complex design, higher component cost.  Use cases: Dominant logic family for most modern electronics, including computers, smart phones, embedded -systems.  3. ECL (Emitter-Coupled Logic)  Advantages: Blazing-fast speed, superior noise immunity, wide operating temperature range.  Disadvantages: High power consumption, complex design, sensitive to output loading.  Use cases: High-speed networking equipment, test and measurement instruments, military and aerospace applications.  4. NMOS (N-channel Metal-Oxide-Semiconductor):  Advantages: Simple design, lower cost than CMOS.  Disadvantages: Limited noise immunity, higher power consumption than CMOS.  Use cases: Mainly historical, used in early integrated circuits before the development of CMOS.  5. PMOS (P-channel Metal-Oxide-Semiconductor):  Advantages: Higher drive current capability than NMOS.  Disadvantages: Higher power consumption than NMOS, typically used in conjunction with NMOS in CMOS circuits.  Use cases: Primarily used for specific functions within CMOS circuits, not commonly used as a standalone logic family.   * **Charactestics of logic families**  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Feature | TTL | CMOS | ECL | NMOS | PMOS | | Logic type | Voltage-based | Voltage-based | Current-based | Voltage-based | Voltage-based | | Speed | Moderate | High | Very high | Low | Moderate | | Power consumption | High | Low | High | High | Low | | Noise immunity | Good | Excellent | Superior | Limited | Moderate | | Complexity | Moderate | High | High | Low | Moderate | | Cost | Moderate | High | High | Low | Moderate | |

## **Indicative content2.3: Apply Digital ICs**



|  |
| --- |
| **An integrated circuit**  It isa set of electronic circuits on one small flat piece of semiconductor material, usually silicon. In an IC, a large number of miniaturized transistors and other electronic components are integrated together on the chip. The following are some advantages of ICs  **Advantages of digital ICs**  Miniaturization: ICs pack thousands, even millions, of transistors onto a single chip, making them incredibly compact compared to discrete circuits built from individual components. This miniaturization enables:  Smaller and lighter devices: Smaller gadgets like smart-phones, laptops, and medical devices wouldn't be possible without ICs.  Increased functionality: More components can fit into a smaller space, allowing for more complex features and capabilities in devices.  Reduced material usage: Less material is needed for the electronics in a device, making it more environmentally friendly.  Faster operating speeds: Transistors on a chip can switch states much faster than discrete components, leading to quicker computations and processing.  Lower power consumption: Miniaturization and improved fabrication techniques result in more efficient circuits that use less power.  Increased reliability: Fewer components and soldered connections mean fewer points of failure and a longer lifespan for devices.  Cost-effectiveness: ICs are produced in mass quantities using automated processes, which significantly reduce their cost per unit. This makes them:  Affordable for consumers: Electronics become more accessible to a wider range of people due to lower prices.  Economically viable for manufacturers: Mass production leads to economies of scale, benefiting both manufacturers and consumers. **Limitations of Integrated Circuits** Complexity: Designing and manufacturing ICs is a complex and expensive process. This can lead to:  Higher initial development costs: Developing new ICs requires significant investment in research and development.  Potential for errors: Complex design processes can introduce subtle errors that might not be discovered until production.  Limitations of individual components: Although some ICs can operate at high frequencies and withstand high temperatures, some individual components might have limitations like:  Lower operating voltage: Some ICs may not be suitable for high-voltage applications.  Limited power handling: Certain ICs might not be able to handle large currents or dissipate high amounts of heat.  Sensitivity to electrostatic discharge (ESD): Some ICs can be damaged by static electricity, requiring careful handling **Classification of digital ICs** Digital ICs are categorized as logic ICs (such as microprocessors and microcontrollers), memory chips (such as MOS memory and floating-gate memory), interface ICs (level shifters, serializer/deserializer, etc.), power management ICs, and programmable devices.  **Classifying integrated circuits (ICs) based on the number of active components** is a common approach called Scale of Integration (SoI). It broadly categorizes ICs according to their complexity and capabilities. Here's a breakdown  **1. Small-Scale Integration (SSI):**  Contains 2 to 10 active components, typically basic logic gates like NOT, AND, OR, etc.  Examples: Inverters, NAND gates, Schmitt triggers, basic flip-flops.  Applications: Simple logic functions, glue logic for connecting larger components.  **2. Medium-Scale Integration (MSI):**  Features 10 to 100 active components, allowing for more complex logic functions.  Examples: Adders, multiplexers, de-multiplexers, counters, shift registers.  Applications: Building blocks for digital systems, control units, data manipulation.  **3. Large-Scale Integration (LSI):**  Includes 100 to 3,000 active components, enabling integration of complex circuits.  Examples: Microprocessors, memory chips, microcontrollers, basic functional blocks like CPUs, memory controllers.  Applications: Core components of complex electronic devices, computers, embedded systems.  **4. Very-Large-Scale Integration (VLSI):**  Features 3,000 to 100,000 active components, allowing for highly complex circuits.  Examples: Microprocessors with integrated cache, graphics processing units (GPUs), dedicated ASICs.  Applications: High-performance devices, complex signal processing, advanced computing, graphics rendering.  **5. Ultra-large-Scale Integration (ULSI):**  Contains over 100,000 active components, enabling extremely complex circuits.  Examples: Modern microprocessors, high-end GPUs, AI accelerators, complex ASICs.  Applications: Cutting-edge devices, demanding computational tasks, artificial intelligence, high-speed data processing.  **Classification integrated circuit Based on applications**  **1. Analogue ICs:**  Process continuous (analogue) signals like voltages and currents.  Examples: Operational amplifiers, power amplifiers, voltage regulators, sensors, audio codes.  Applications: Signal processing, amplification, power conversion, data acquisition, communication systems.  **2. Digital ICs:**  Process discrete (digital) signals represented by voltages as 0s and 1s.  Examples: Microprocessors, microcontrollers, memory chips, logic gates, flip-flops.  Applications: Computing, data storage, control systems, digital communication, signal processing.  **3. Mixed ICs:**  Combine both analogue and digital circuitry on a single chip.  Examples: Analogue-to-digital converters (ADCs), digital-to-analogue converters (DACs), mixed-signal front-end modules.  Applications: Interface between analogue and digital domains, data acquisition systems, medical devices, communication equipment.  **Classifying integrated circuits (ICs) based on their manufacturing methods** provides valuable insights into their characteristics, capabilities, and limitations. Here are some key classifications:  1. **Monolithic Integration:**  The most common approach, where all circuit elements are formed on a single semiconductor substrate using various fabrication techniques like photolithography and chemical etching.  Subcategories:  Bipolar Monolithic: Uses bipolar junction transistors (BJTs) as the primary active element. Examples: TTL (Transistor-Transistor Logic) family, ECL (Emitter-Coupled Logic).  CMOS Monolithic: Employs complementary metal-oxide-semiconductor (CMOS) transistors. Examples: Most modern digital ICs, microprocessors, memory chips.  Bi-CMOS Monolithic: Combines both BJT and CMOS transistors on a single chip for specific performance-power trade-offs.  **2. Hybrid Integration:**  Combines multiple pre-fabricated chips like transistors, diodes, and passive components onto a single substrate using techniques like wire bonding or flip-chip assembly.  Offers advantages like:  Combining diverse technologies from different fabrication processes.  Smaller circuit size compared to monolithic integration for certain functionalities.  Used in specialized applications like high-power circuits, microwave devices, and some medical electronics. **Applying digital ICs in electronic circuits**  Logic gates, like those in the 74HCXX and 74LSXX series, are the fundamental building blocks of digital circuits. They perform basic logic operations on binary signals (0s and 1s) represented by voltage levels. Understanding these gates is key to designing and analyzing digital systems. **74HCXX and 74LSXX logic gates** 74HCXX: Uses High-Speed CMOS (Complementary Metal-Oxide-Semiconductor) technology, offering low power consumption, high noise immunity, and good operating speeds. Popular for general-purpose digital applications.  74LSXX: Employs Low-Power Schottky TTL (Transistor-Transistor  Logic) technology, providing faster switching speeds than 74HCXX but consuming more power and having lower noise immunity. Used in situations where speed is critical.  Logic Functions:  Both series offer a wide range of logic gates performing various functions:  Basic Gates: NOT, AND, OR, NAND, NOR, XOR, XNOR  Combination Gates: Multiplexers, De-multiplexers, Adders, Flip-Flops, Shift Registers  Specialized Gates: Schmitt Trigger, Monostable Multivibrator  Pin Configuration:  Most gates in both series follow a standard pin configuration:  Vcc: Power supply input  GND: Ground  Inputs: Receive binary signals for the logic operation  Output: Provides the resulting binary signal after the operation  Choosing the Right Logic Gate:  Selecting the appropriate gate depends on several factors:  Required Logic Function: Determine the desired operation on the input signals.  Series Choice: Consider factors like power consumption, operating speed, and noise immunity.  Number of Inputs/Outputs: Select a gate with enough inputs and outputs for your circuit needs.  Package type: Choose a package (DIP, SOIC, etc.) suitable for your board layout and soldering capabilities     * The CD4017 is a decade counter integrated circuit (IC) belonging to the popular CMOS 4000 series. It features 10 outputs representing the numbers 0 to 9 and is commonly used in various electronic projects for * LED Chasers: Creating sequential lighting patterns with LEDs, simulating "knight rider" effects   ****Frequency Dividers:**** Dividing a clock signal by a factor of 10, useful for generating lower frequencies from a higher source  ****Music Sequencers:**** Controlling the sequence of notes in a simple electronic music circuit.   * **Alarms and Timers**: Implementing basic timing functions and generating alarm signals. * **Counting and Displaying**: Counting events or pulses and displaying the count on LEDs or other indicators.   key features of the CD4017:  Operating Voltage: Typically 3V to 15V, making it compatible with various power supplies.  Current Consumption: Low power consumption (around 1µA in standby mode), suitable for battery-powered applications.  Output Drive Capability: Can drive LEDs directly or interface with other circuits through transistors.  CMOS Technology: Offers high noise immunity and good operating speeds.  Decode Outputs: Provides individual outputs for each count state (0-9), simplifying connection and control.  Reset and Clock Inputs: Allows resetting the counter to zero and controlling the counting sequence.  CD4017 Pin-out Pins: from pin 1 to pin 7 and pin 9 to pin 11. Input Voltage pin: Pin 16 is for the applied voltage from 3V to 15V, Ground pin: pin 8 is for ground. Clock Enable Pin: Pin 13 is for the clock, when there's logic zero, the clock gets enabled and the counter increases one count for every clock pulse     * The xx555 timer (often simply called the 555 timer) is a popular and versatile analog integrated circuit (IC) used in various electronics projects. It can function as: * Monostable Multivibrator: Generates a single pulse of adjustable duration when triggered by a pulse on its control pin. This enables applications like timers, delay circuits, and pulse generators. * Astable Multivibrator: Creates a continuous square wave oscillation with adjustable frequency and duty cycle. This makes it useful for applications like audio tone generators, blinkers, and LED sequencers. * Schmitt Trigger: Acts as a voltage comparator with hysteresis, improving circuit noise immunity and stability. * key features of the xx555 timer: * Single Supply: Operates on a single voltage supply between 5V and 15V, making it versatile for various projects. * Adjustable Timing: Timing parameters like pulse duration and oscillation frequency can be easily adjusted using external resistors and capacitors. * Wide Range of Applications: From simple circuits like time timers and buzzer controllers to complex tasks like pulse width modulation (PWM) and voltage level control. * Cost-Effective and Easy to Use: Widely available and affordable, with many online resources and tutorials available for beginners. * common applications of the xx555 timer include: * Blinker circuits: Creating simple LED blinking patterns with adjustable frequency. * Tone generators: Generating square wave tones for audio applications. * Delay circuits: Creating timed delays in various electronic projects. * Pulse width modulation (PWM) circuits: Controlling the average voltage output to regulate power to devices like LEDs and motors. * Voltage level triggers: Detecting when a voltage crosses a certain threshold and triggering actions.   Pin configuration   **Logic circuits simulation in software** Simulation software those are the software used to build, design and simulate electronic circuit.  **Simulation software description**  Benefits of Using Simulation Software:  Reduce costs and risks: Test ideas and designs before investing in real-world implementation, minimizing potential failures and saving resources.  Explore multiple scenarios: Analyze various conditions and parameters to optimize outcomes, predict trends, and make informed decisions.  Improve safety and efficiency: Identify potential dangers or inefficiencies in complex systems before risking real-world operation.  Accelerate research and development: Conduct virtual experiments faster and more iteratively than traditional methods.  Examples of Simulation Software:  Solid Works: Mechanical engineering simulation for designing and testing parts and assemblies.  ANSYS: Computational fluid dynamics and structural analysis software for engineering applications.  COMSOL Multi-physics: Multipurpose software for modelling various physical phenomena in diverse fields.  SimCity: Classic city building simulation game, showcasing the core principles of simulating social and economic systems.  Considerations when choosing simulation software:  The type of simulation you need: Choose software designed for your specific field and research area.  Software complexity and user interface: Select a program suitable for your skill level and project requirements.  Computing power and cost: Consider the software's hardware requirements and licensing costs. |

### **Theoretical learning Activity**

* Four group discussion on advantages and limitations of ICs
* Individual work on classification of digital ICs

### **Practical learning Activity**

Design and implementation of the circuit that will cut the supply of the water pump and turn it off using 555timer.

Points to Remember (Take home message)

|  |
| --- |
| * Integrated circuit is an electronic circuit formed on a small piece of semiconducting material, which performs the same function as a larger circuit made from discrete components * Digital ICs are classified Based on Technology, number of active components, applications and on manufacturing method * Examples of logic circuit simulation software: Solid Works, ANSYS, COMSOL, Multi-physics SimCity...... |

******Learning outcome 2: formative assessment**

**Written assessment**

1. Logic gates are an important concept in the field of electronics and they are classified into three classes which are Basic logic gates, Universal logic gates and special logic gates. Among the logic gates list bellow choose one which is a special logic gate:
2. AND
3. OR
4. NAND
5. Both B and C are correct
6. None of the above
7. NAND and NOR gates are universal logic gates which are made by inverting some of basic logic gates. Among the list bellow choose one which was implemented to get a NAND gate.
8. Inversion followed by AND gates
9. AND gates followed by an inverter
10. AND gate followed by OR gate
11. None of these
12. **Explain why a two-input NAND gate is called universal gate?**
13. The following is a truth table of logic gate. Analyze it and say which type of logic gate is obtained from?

|  |  |  |
| --- | --- | --- |
| A | B | O/P |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

1. write in full the following abbreviations:
2. TTL
3. ECL
4. MOS
5. Differentiate between bistable multivibrator and monostable multivibrator?

**ANSWERS:**

1. Logic gates are an important concept in the field of electronics and they are classified into three classes which are Basic logic gates, Universal logic gates and special logic gates. Among the logic gates list bellow choose one which is a special logic gate:

* 1. AND
  2. OR
  3. NAND
  4. Both B and C are correct
  5. **None of the above**

2. NAND and NOR gates are universal logic gates which are made by inverting some of basic logic gates. Among the list bellow choose one which was implemented to get a NAND gate.

Inversion followed by AND gates

**AND gates followed by an inverter**

AND gate followed by OR gate

None of these

**3. Explain why a two-input NAND gate is called universal gate?**

NAND gate is called universal gate because any digital system can be implemented with the NAND gate. Sequential and combinational circuits can be constructed with these gates because element circuits like flip-flop can be constructed from two NAND gates connected back-to-back. NAND gates are common in hardware because they are easily available in the ICs form. A NAND gate is in fact a NOT-AND gate. It can be obtained by connecting a NOT gate in the output of an AND gate.

4. The truth table represents an **OX-OR**

5. Write in full the following abbreviations:

* + - 1. TTL: Transistor -Transistor Logic
      2. ECL : Emitter coupled Logic
      3. MOS: Metal-oxide semiconductor

6. Differentiate between bistable multivibrator and monostable multivibrator?

Monostable: *One-shot multivibrator* that has only ONE stable state and is triggered externally with it returning back to its first stable state.

Bistable: *Flip-flop* that has TWO stable states that produces a single pulse either positive or negative in value

**References:**

* 1948– Principles of modern digital design / by Parag K. Lala.
* Programmable Logic Devices (PLDs) in VLSI Design ( Dr. G. Senthil Kumar Associate Professor)
* Digital Fundamentals( E LEVENTH EDITION Thomas L. Floyd)
* Digital Electronics Principles, Devices and Applications( Anil K. Maini Defence Research and Development Organization (DRDO), India)

## 

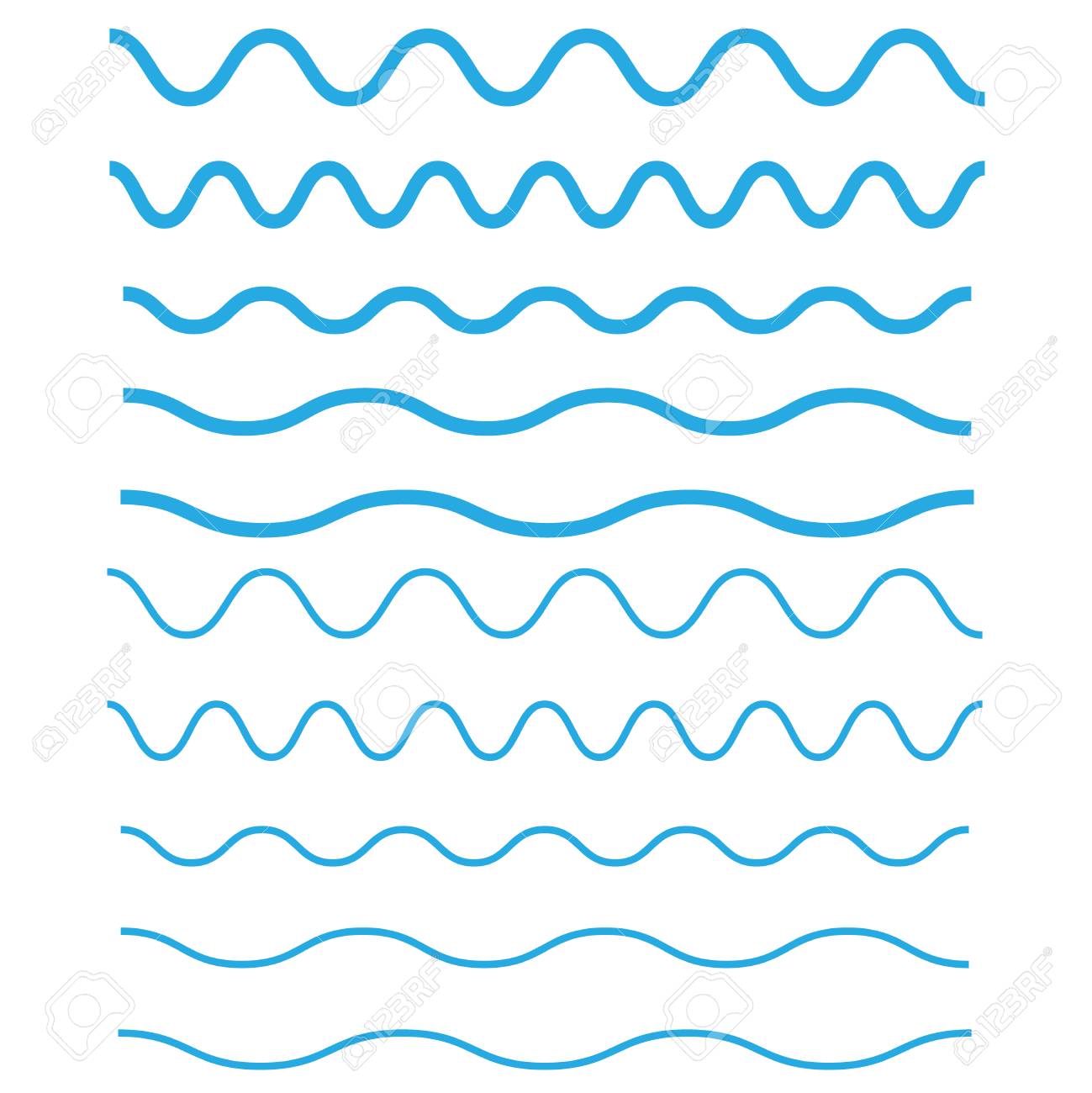
# **Learning outcome 3: Apply Boolean** **algebra**

|  |
| --- |
|  |

|  |
| --- |
| **Indicative contents**  **3.1 : Theorems of Boolean Algebra**  **3.2:** **Standard Forms of Boolean Expressions**  **3.3:** **Boolean expressions simplification techniques** |

|  |  |  |
| --- | --- | --- |
| **Duration:15hrs** | | |
| **Learning outcome 3 objectives:**  By the end of the learning outcome, the trainees will be able to:  1. To identify properly Theorems of Boolean Algebra according to their types  2. To form properly Boolean expressions based on the standard forms  3. To simplify properly Boolean expressions according to the simplification techniques | | |
| **Resources** | | |
| **Equipment** | **Tools** | **Materials** |
| Computer  projector | Books | Markers  chalks  board  Internet |
| **Advance preparation:**   * Deeply research on the Boolean Algebra laws and Demorgan’s theorems * Deeply research on Boolean algebra simplification techniques | | |

## **Indicative content3.1: Theorems of Boolean algebra**



|  |
| --- |
| **Description of Boolean algebra Laws** Boolean algebra is mathematics of logic with fundamental laws that are used to build a workable, cohesive framework upon which are placed the theorems proceeding from these laws.  It is one of the most basic tools available to the logic designer and thus can be effectively used for simplification of complex logic expressions. Other useful and widely used techniques based on Boolean theorems include the use of Karnaugh maps.   **Demorgan’s theorems** (AB)’=A’+B’  (A+B)’=A’B’ |

### **Theoretical learning Activity**

* Individual work on Boolean expressions simplification using Boolean algebra laws and k-map techniques

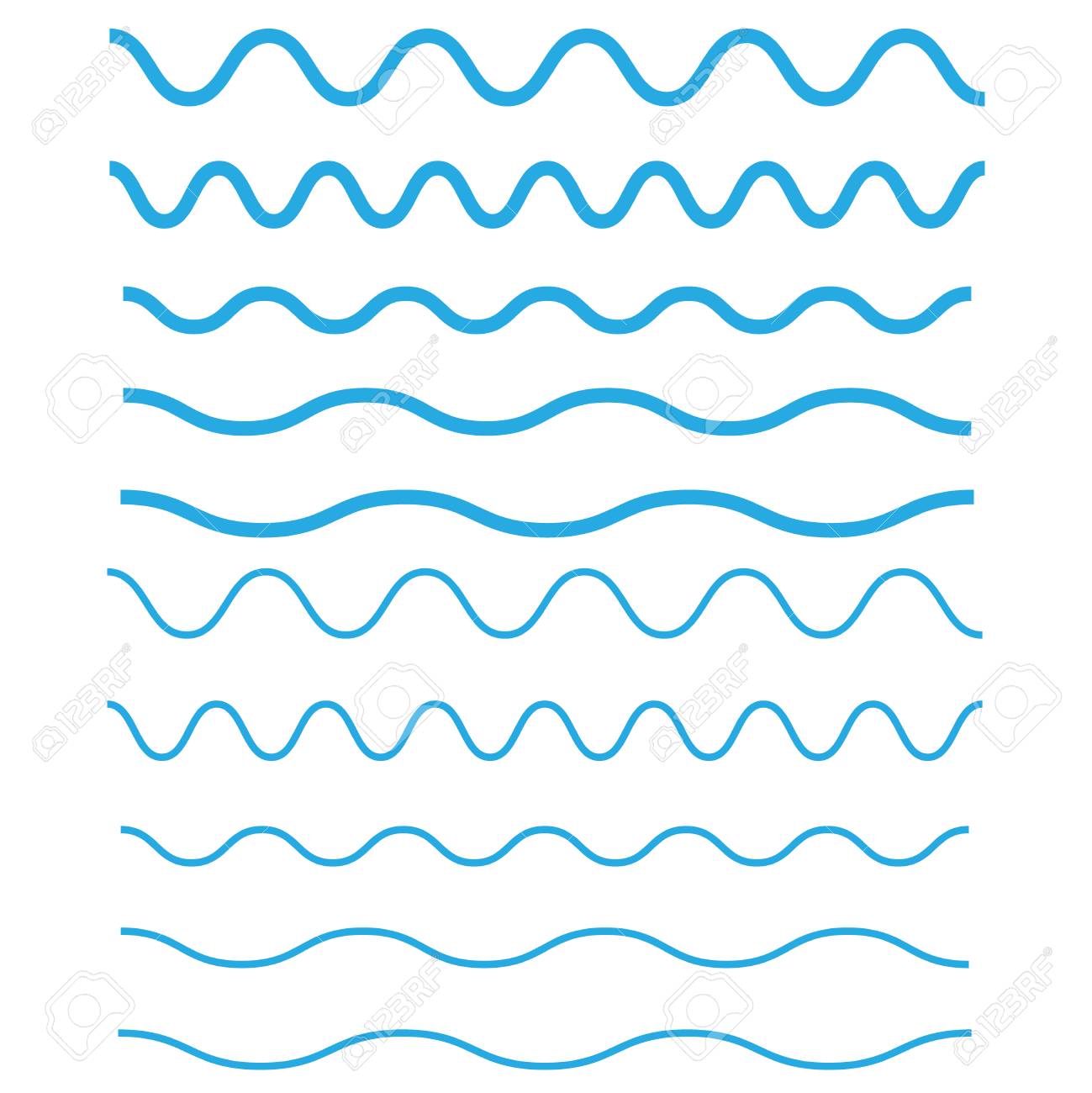
### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * There are two techniques used to simplify Boolean expression: * By using Boolean algebra Laws * By using k-map |

## **Indicative content3.2: Standard Forms of Boolean Expressions**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Boolean Expressions standard forms** **A sum-of-products** expression contains the sum of different terms, with each term being either a single literal or a product of more than one literal. It can be obtained from the truth table directly by considering those input combinations that produce logic ‘1’ at the output. Each such input combination produces a term. Different terms are given by the product of the corresponding literals.  The sum of all terms gives the expression. For example, the truth table in Table below can be represented by the Boolean expression Y=A’.B’.C’+A’.B.C+A.B.C’+A.B’.C   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | Y | | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 0 |   Considering the first term, the output is ‘1’ when A = 0, B = 0 and C = 0. This is possible only when A’, B’ and C’ are ANDed. Also, for the second term, the output is ‘1’ only when B, C and A’ are ANDed. Other terms can be explained similarly. A sum-of-products expression is also known as a ***minterm*** *expression*.  Example:  Find the sop form for the function f(A,B,C) which is represented by the truth table given below:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Decimal | A | B | C | F | | 0 | 0 | 0 | 0 | 0 | | 1 | 0 | 0 | 1 | 1 | | 2 | 0 | 1 | 0 | 1 | | 3 | 0 | 1 | 1 | 0 | | 4 | 1 | 0 | 0 | 1 | | 5 | 1 | 0 | 1 | 0 | | 6 | 1 | 1 | 0 | 0 | | 7 | 1 | 1 | 1 | 1 |   Sol: the combinations for which the function f assumes the value ‘1’ are 1,2,4,7.  Thus f (A, B, C) = ∑ (1, 2, 4, 7)  = 001+010+100+111  f (A, B, C) = A’B’C+A’BC’+AB’C’+ABC  To express a Boolean expression in its SOP form uses the following procedure:   1. Write down all the terms 2. Put X’s where letters must be provided to convert the term to a minterm 3. Use all the combinations of Xs in each term to generate minterms. 4. Drop out redundant terms Example:   Find them interms for A+BC  Sol:   * Write down all the terms A + BC * Insert X where letters are missing AXX +XBC * Different combinations of Xs AB’C’+AB’C+ABC’+*ABC*+A’BC+*ABC* * Drop out redundant terms AB’C’+AB’C+ABC’+*ABC*+A’BC   **A product-of-sums** expression contains the product of different terms, with each term being either a single literal or a sum of more than one literal. It can be obtained from the truth table by considering those input combinations that produce logic ‘0’ at the output. Each such input combination gives a term, and the product of all such terms gives the expression. Different terms are obtained by taking the sum of the corresponding literals. Here, ‘0’ and ‘1’ respectively mean the uncomplemented and complemented variables, unlike sum-of-products expressions where ‘0’ and ‘1’ respectively mean complemented and uncomplemented variables.  To illustrate this further, consider once again the truth table in Table below. Since each term in the case of the product-of-sums expression is going to be the sum of literals, this implies that it is going to be implemented using an OR operation. Now, an OR gate produces a logic ‘0’ only when all its inputs are in the logic ‘0’ state, which means that the first term corresponding to the second row of the truth table will be A+B+C’. The product-of-sums Boolean expression for this truth table is given by  (A+B+C’)(A+B’+C)(A’+B+C)(A’+B’+C’)   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | Y | | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 0 |   Transforming the given product-of-sums expression into an equivalent sum-of-products expression is a straightforward process. Multiplying out the given  (A+B+C’)(A+B’+C)(A’+B+C)(A’+B’+C’)=(AA+AB’+AC+BA+BB’+BC+C’A+C’B’+C’C)(A’A’+A’B’+A’C’+BA’+BB’+BC’+CA’+CB’+CC’)  expression and carrying out the obvious simplification provides the equivalent sum-of-products expression  A given sum-of-products expression can be transformed into an equivalent product-of-sums expression by (a) taking the dual of the given expression, (b) multiplying out different terms to get the sum-of-products form, (c) removing redundancy and (d) taking a dual to get the equivalent product-of-sums expression. As an illustration, let us find the equivalent product-of- sums expression of the sum-of- products expression AB+A’B’.  The dual of the given expression =(A+B)(A’+B’)=(A+B’)(A’+B’)  Therefore the expression is AB’+A’B+BB’, so AB+A’B’= (A+B’)( A’+B)C +A’B’B’  **,** |

### **Theoretical learning Activity**

* Individual work on Boolean expression simplification using Boolean algebra laws and k-map techniques

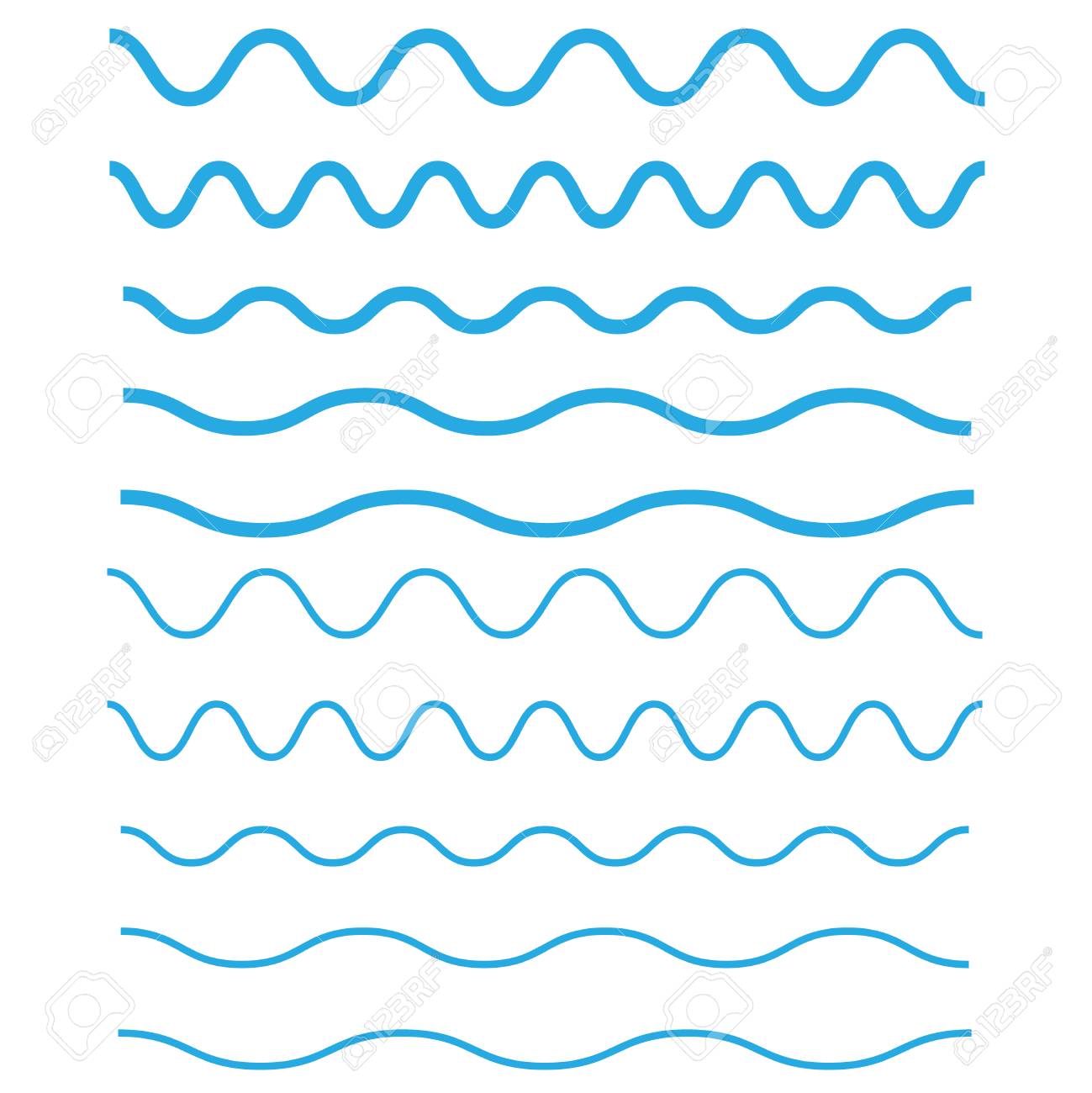
### **Practical learning Activity**

Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * Boolean expressions simplification techniques: Boolean algebra laws and k-maps * Boolean expression standard forms are: SOP and POS |

## **Indicative content3.3: Boolean expressions simplification techniques**



|  |
| --- |
| **Using Karnaugh map (K-map)** Generating Boolean equations to implement a desired logic function is a necessary step before a circuit can be implemented. Truth tables are a common means of describing logical relationships between Boolean inputs and outputs. Once a truth table has been created, it is not always easy to convert that truth table directly into a Boolean equation. This translation becomes more difficult as the number of variables in a function increases. A graphical means of translating a truth table into a logic equation was invented by Maurice Karnaugh in the early 1950s and today is called the *Karnaugh-map*, or *K-map*.  A K-map is a type of truth table drawn such that individual product terms can be picked out and summed with other product terms extracted from the map to yield an overall Boolean equation. Karnaugh maps allow us to convert a truth table to a simplified Boolean expression without using Boolean algebra.  A Karnaugh map is a graphical representation of the logic system. It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions.  **An n-variable Karnaugh** **map** has 2n squares, and each possible input is allotted a square. In the case of minterm ‘1’ is placed in all those squares for which the output is ‘1’. 0s are omitted for simplicity. An ‘X’ is placed in squares corresponding to ‘don’t care’ conditions. In the case of a maxterm Karnaugh map, a ‘1’ is placed in all those squares for which the output is ‘0’, and a ‘0’ is placed for input entries corresponding to a ‘1’ output. Again, 0s are omitted for simplicity, and an ‘X’ is placed in squares corresponding to ‘don’t care’ conditions.  The choice of terms identifying different rows and columns of a Karnaugh map is not unique for a given number of variables. The only condition to be satisfied is that the designation of adjacent rows and adjacent columns should be the same except for one of the literals being complemented. Also, the extreme rows and extreme columns are considered adjacent. Some of the possible designation styles for two-, three- and four-variable minterm Karnaughs maps are given in Figures 5.1, 5.2 and 5.3 respectively.  The style of row identification need not be the same as that of column identification as long as it meets the basic requirement with respect to adjacent terms. It is, however, accepted practice to adopt a uniform style of row and column identification.  Having drawn the Karnaugh map, the next step is to form groups of 1s as per the following guidelines:   1. Each square containing a ‘1’ must be considered at least once, although it can be considered as often as desired. 2. The objective should be to account for all the marked squares in the minimum number of groups. 3. The number of squares in a group must always be a power of 2, i.e. groups can have 1, 2, 4, 8, 16, …. squares. 4. Each group should be as large as possible, which means that a square should not be accounted for by itself if it can be accounted for by a group of two squares; a group of two squares should not be made if the involved squares can be included in a group of four squares and so on. 5. ‘Don’t care’ entries can be used in accounting for all of 1-squares to make optimum groups. They are marked ‘X’ in the corresponding squares. It is, however, not necessary to account for all ‘don’t care’ entries. Only such entries that can be used to advantage should be used.   Having accounted for groups with all 1s, the minimum ‘sum-of-products’ or ‘product-of-sums’ expressions can be written directly from the Karnaugh map.  Step1: Draw the truth table  Step 2: Divide the input variables up.  Step 3: Draw a Karnaugh map based on the input variables  Note: The inputs are arranged so that only one bit changes at a time for the Karnaugh map. Step 4: Look for patterns in the map  Step 5: Write the equation using the patterns  Figure 6.10 shows the truth table, minterm Karnaugh map and maxterm Karnaugh map of the Boolean function of a two-input OR gate. The minterm and maxterm Boolean expressions for the two-input OR gate are as follows:  Y = A+B (maxterm or product-of-sums)  Y= A’B+AB’+AB( Minterm or sum of product       1. Combine adjacent cells that have “1”s into groups of 2, 4 or 8 allowing the removal of 1, 2 or 3 variables from a term. 2. The maps are considered to wrap around so that top and bottom corresponding squares are adjacent and also left and right edge are also adjacent. 3. When squares can be chosen in several groups the object is to use the largest group. 4. Every cell containing 1 must be included in at least one group. 5. Write down the terms of the simplified expression by determining which variables are included within each groups   The following four variable Karnaugh maps illustrate reduction of Boolean expressions too tedious for Boolean algebra. Reductions could be done with Boolean algebra. However, the Karnaugh map is faster and easier, especially if there are many logic reductions to do.    The above Boolean expression has seven product terms. They are mapped top to bottom and left to right on the K-map above. For example, the first P-term A'B'CD is first row 3rd cell, corresponding to map location A=0, B=0, C=1, D=1. The other product terms are placed in a similar manner. Encircling the largest groups possible, two groups of four are shown above.  The dashed horizontal group corresponds the simplified product term AB. The vertical group corresponds to Boolean CD. Since there are two groups, there will be two product terms in the Sum-Of-Products result of Out=AB+CD.  OUT=A’B’CD+A’BCD+ABCD+AB’CD+ABC’D’+ABC’D+ABCD’  The four cells above are a group of four because they all have the Boolean variables B' and D' in common. In other words, B=0 for the four cells, and D=0 for the four cells.  The other variables (A, B) are 0 in some cases, 1 in other cases with respect to the four corner cells. Thus, these variables (A, B) are not involved with this group of four. This single group comes out of the map as one product term for the simplified result: Out=B'C'   **Using Boolean algebra laws** Boolean algebra is a useful tool for simplifying digital circuits. Why do it? Simpler can mean cheaper, smaller, faster. Example:  Simplify F = x’yz + x’yz’ + xz.  F= x’yz + x’yz’ + xz = x’y(z+z’) + xz  = x’y.1 + xz = x’y + xz  Example: Prove x’y’z’ + x’yz’ + xyz’ = x’z’ + yz’  Proof: x’y’z’+ x’yz’+ xyz’  = x’y’z’ + x’yz’ + x’yz’ + xyz’  = x’z’(y’+y) + yz’(x’+x)  = x’z’.1 + yz’.1  = x’z’ + yz’    The complement of a function is derived by interchanging (• and +), and (1 and 0), and complementing each variable. Otherwise, interchange 1s to 0s in the truth table column showing F. The complement of a function IS NOT THE SAME as the dual of a function.  Example :  Find G(x,y,z), the complement of F(x,y,z) = xy’z’ + x’yz  Ans: G = F’ = (xy’z’ + x’yz)’ = (xy’z’)’ . (x’yz)’  De-Morgan = (x’+y+z).(x+y’+z’)  De-Morgan again Note: The complement of a function can also be derived by finding the function’s dual, and then complementing all of the literals |

### **Theoretical learning Activity**

Individual work on simplification of Boolean expressions

### **Practical learning Activity**

Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * Boolean expressions simplification techniques: * Using Boolean algebra laws * Using k-map |

**Learning outcome 3: formative assessment**

**Written assessment**

* + - 1. Find the dual complement of

1. AB+BC+CD
2. AB+CD+EF
   * + 1. Simplify (AB+CD)[(A’+B’)(C’+D’)]
       2. Convert the following SOP expression to an equivalent POS expression.

ABC+AB’C’+AB’C+ABC’+A’B’C

* + - 1. **Using Boolean identities, reduce the given Boolean expression**:****

**F(X, Y, Z) = X′Y + YZ′ + YZ + XY′Z′**

**ANSWERS**

* 1. Find the dual complement of

1. AB+BC+CD
2. AB+CD+EF

Solution

i. The dual of AB+BC+D is given by A+BB+CC+D.

ii. The complement of AB+CD+EF is given by A+BC+DE+F.

* 1. Simplify (AB+CD)[(A’+B’)(C’+D’)]

Solution • Let AB+D=X.

• Then the given expression reduces to XX.

• Therefore,( AB+CD)[(A’+B’)(C’+D’)]= 0.

* 1. Convert the following SOP expression to an equivalent POS expression.

ABC+AB’C’+AB’C+ABC’+A’B’C

**Solution:** (++).(+B+C).(+B+).(++C).(A+B+)

* 1. ****Using Boolean identities, reduce the given Boolean expression:****

****F(X, Y, Z) = X′Y + YZ′ + YZ + XY′Z′****

****Solution:****

Given, F(X, Y, Z) = X′Y + YZ′ + YZ + XY′Z′

Using the idempotent law, we can write YZ’ = YZ’ + YZ’

⇒ F(X, Y, Z) = X′Y+(YZ′+YZ′)+YZ + XY′Z′

Now, interchange the second and third term, we get

⇒ F(X, Y, Z) = X′Y+(YZ′+YZ)+(YZ′+XY′Z′)

By using distributive law,

⇒ F(X, Y, Z) = X′Y+Y(Z′+Z)+Z′(Y+XY′)

Using Z’ + Z = 1 and absorption law (Y + XY’)= (Y + X),

⇒ F(X, Y, Z) = X′Y+Y.1+Z′(Y+X)

⇒ F(X, Y, Z) = X′Y+Y+Z′(Y+X) [ Since Y.1 = Y ]

⇒ F(X, Y, Z) = Y(X′+1)+Z′(Y+X)

⇒ F(X, Y, Z) = Y.1+Z′(Y+X) [ As (X’ + 1) = 1 ]

⇒ F(X, Y, Z) = Y +Z′(Y+X) [ As, Y.1 = Y ]

⇒ F(X, Y, Z) = Y+YZ’+XZ’

⇒ F(X, Y, Z) = Y(1+Z′)+XZ′

⇒ F(X, Y, Z) = Y.1+XZ′ [Since (1 + Z’) = 1]

⇒ F(X, Y, Z) = Y+XZ′ [Since Y.1 = Y]

Hence, the simplified form of the given Boolean expression is F(X, Y, Z) = Y+XZ′.

**References :**

* Programmable Logic Devices (PLDs) in VLSI Design ( Dr. G. Senthil Kumar Associate Professor)
* Digital Fundamentals( E LEVENTH EDITION Thomas L. Floyd)
* Digital Electronics Principles, Devices and Applications( Anil K. Maini Defence Research and Development Organization (DRDO), India)
* R.Tokheim. (2014). Digital Electronics Principles and Applications. USA: 8th ed.

# 

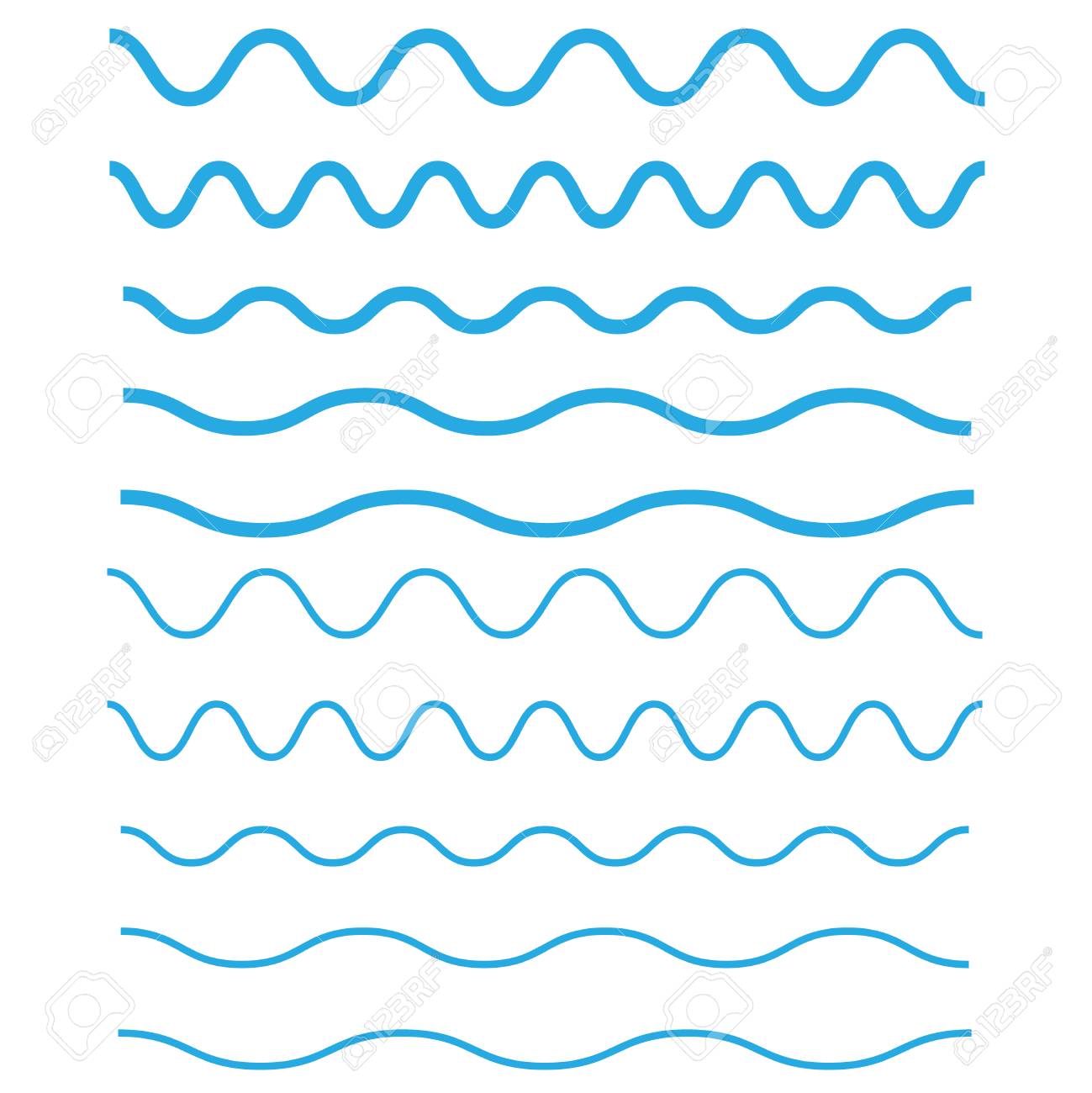
**Learning outcome 4: Apply Fixed Logic Devices**

|  |
| --- |
|  |

|  |
| --- |
| **Indicative contents**  **4.1:** **Apply Combinational logic circuits**  **4.2:** **Multivibrators using ICs**  **4.3:** **Apply Sequential logic circuits** |

|  |  |  |
| --- | --- | --- |
| **Duration:20hrs** | | |
| **Learning outcome 4 objectives:**  By the end of the learning outcome, the trainees will be able to:  1 .Identify properly different types of combinational circuit according to their working and applications  2. Identify properly the parts of Sequential logic circuits according to its design  3. Identify properly the types of Sequential logic circuits according to their working and applications | | |
| **Resources** | | |
| **Equipment** | **Tools** | **Materials** |
| Computer  projector  Multimeters  DC power supply | Books  Multisim software  Proteus software  universal plier | Markers  Chalks  board  Internet  logic ICs  breadboard  jumper wires  electronic components |
| **Advance preparation:**   * Research on combinational logic circuit Design ,types and Working principle * Research on sequential logic circuit Design ,types and Working principle | | |

## **Indicative content4.1: Apply Combinational logic circuits**



## 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Design and Working principle** Combinational circuit is a circuit in which we combine the different gates in the circuit, for example Adder and subtractor, encoder and decoder, multiplexer and de-multiplexer...  Some of the characteristics of combinational circuits are following:   * + The output of combinational circuit at any instant of time depends only on the levels present at input terminals.   + The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.   + A combinational circuit can have an ***n*** number of inputs and ***m*** number of outputs.   Block diagram  Block Diagram of combinational circuit    A combinational circuit is the digital logic circuit in which the output depends on the combination of inputs at that point of time with total disregard to the past state of the inputs. The digital logic gate is the building block of combinational circuits  practical applications of combinational logic circuits  Combinational logic circuits are used in digital applications such as electronic devices, telecommunications, and computer systems. They are used to convert digital signals into other forms of data or signals. For example, they are used to process digital audio signals and convert them into analogue audio signals **Types of combinational logic circuits**  * **Apply Adder and Subtractor**   **Half adder** is a combinational logic circuit with two inputs and two outputs (carry and sum), made of two logic gates (XOR, AND)  Block diagram  Block Diagram of Half Adder  Circuit Diagram    Truth Table   |  |  |  |  | | --- | --- | --- | --- | | A | B | S | C | | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 0 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 |   **Full Adder:** It can add two one-bit numbers A and B, and carry c. The full adder is a three input and two output combinational circuit, made of two half adders.  **Circuit Diagram**    **Truth Table**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **A** | **B** | **Cin** | **S** | **Co** | | 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 |   **Half Subtractor:** Half subtractor is a combination circuit with two inputs and two outputs (difference and borrow). It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.  **Block diagram**    **Circuit Diagram**    **Truth Table**   |  |  |  |  | | --- | --- | --- | --- | | **A** | **B** | **A-B** | **BORROW** | | **0** | **0** | **0** | **0** | | **0** | **1** | **1** | **1** | | **1** | **0** | **1** | **0** | | **1** | **1** | **0** | **0** |   **Full Subtractor:** The full subtractor is a combinational circuit with three inputs A,B,C and two output D and C'. A is the 'minuend', B is 'subtrahend', C is the 'borrow' produced by the previous stage, D is the difference output and C' is the borrow output.  Block diagram    **Circuit Diagram**    **Truth Table**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **A** | **B** | **C** | **A-B-C** | **C’** | | **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** |  **Apply Comparator** Comparator or magnitude comparator is a hardware electronic device that takes two numbers as input in binary form and determines whether one number is greater than, less than or equal to the other number. Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).    Circuit diagram    The Digital Comparator is another very useful combinational logic circuit used to compare the value of two binary digits.  The binary or digital comparator can be constructed using standard AND, NOR and NOT gates to compare the digital signals present at their input terminals and produce an output depending upon the condition of those inputs.  For example, along with being able to add and subtract binary numbers we need to be able to compare them and determine whether the value of input A is greater than, smaller than or equal to the value at input B etc. The digital comparator accomplishes this using several logic gates that operate on the principles of Boolean Algebra.  There are two main types of **Digital Comparator** available and these are.  1. Identity Comparator – an Identity Comparator is a digital comparator with only one output terminal for when A = B, either A = B = 1 (HIGH) or A = B = 0 (LOW)  2. Magnitude Comparator – a Magnitude Comparator is a digital comparator which has three output terminals, one each for equality, A = B  greater than, A > B  and less than A < B  **Truth table**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | B | A | A>B | A=B | A<B | | 0 | 0 | 0 | 1 | 0 | | 0 | 1 | 1 | 0 | 0 | | 1 | 0 | 0 | 0 | 1 | | 1 | 1 | 0 | 1 | 0 |  **Apply Multiplexer and De-multiplexer** The multiplexer is a combinational logic circuit designed to select one of several data inputs and forwards it to output.  A **Mux** has maximum of 2n data inputs, ‘n’ selection lines and single output line  **Mux Block diagram**  basic multiplexer switchWhat is a Multiplexer (MUX)? - ppt download    **Mux circuit using NAND**    **2 to 1 Multiplexer**  A 2-to-1 multiplexer is the digital multiplexer circuit that has two data inputs D0 and D1, one selects line S and one output Y  2 To 1 Multiplexer Circuit 1   |  |  |  |  | | --- | --- | --- | --- | | **Select input S** | **Data input Do** | **Data input D1** | **Output Y** | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 |   **Advantages of a Multiplexer**   * A multiplexer reduces the number of wires used. Hence it reduces the circuit complexity and overall cost. * A multiplexer improves the reliability of the digital systems because it reduces the number of external wired connections. * We can implement many combinational circuits using MUX. * Multiplexer simplifies the logic design.   **Applications of a Multiplexer**   * *Communication Systems*: To select one out of many data inputs in communication systems to transmit the various types of data (audio, video, etc.) at the same instant. * *Telephone Networks*: To transmit multiple audio signals into a single channel. * *Computer Memory*: To implement a large amount of data, at the same instant * Reduces the number of copper wires required to connect memory to other parts in the computer. * *Transmission from the computer system to a satellites*: To transmit the data from the computer system of a spacecraft or satellite to the earth by utilizing “GPS” (Global Positioning System) and “GSM” (Global System for Mobile Communication).  **Apply Encoder and Decoder** ****Combinational Logic**** is the concept in which two or more input states define one or more output states. The ****Encoder and Decoder**** are combinational logic circuits. In which we implement combinational logic with the help of Boolean algebra.  To encode something is to convert an unambiguous piece of information into a form of code that is not so clearly understood and the device which performs this operation is termed ad Encoder.   * **An Encoder** is a device that converts the active data signal into a coded message format or it is a device that converts analogue signal to digital signals. It is a combinational circuit,that converts binary information in the form of 2N input lines into N output lines which represent N bit code for the input. When an input signal is applied to an encoder the logic circuitry involved within it converts that particular input into coded binary output      * **A decoder** is also a combinational circuit as an encoder but its operation is exactly reverse as that of the encoder. A decoder is a device that generates the original signal as output from the coded input signal and converts n lines of input into 2n lines of output. An AND gate can be used as the basic decoding element because it produces a high output only when all inputs are high.      | **ENCODER** | **DECODER** | | --- | --- | | Encoder circuit basically converts the applied information signal into a coded digital bit stream. | Decoder performs reverse operation and recovers the original information signal from the coded bits. | | In case of encoder, the applied signal is the active signal input. | Decoder accepts coded binary data as its input. | | The number of inputs accepted by an encoder is 2n. | The number of input accepted by decoder is only n inputs. | | The output lines for an[encoder](https://www.geeksforgeeks.org/encoder-in-digital-logic/) | The output lines of an decoder is 2n. | | The encoder generates coded data bits as its output. | The decoder generates an active output signal in response to the coded data bits. | | The operation performed is simple. | The operation performed is complex. |  **Applications of combinational logic circuits** **Data Processing:**   * Adders and subtractors: Used in arithmetic units of CPUs and calculators to perform calculations. * Multiplexers and de-multiplexers: Control data flow and route signals, essential for communication and data transfer. * Encoders and decoders: Convert data between different formats, vital for digital communication protocols and display technologies.   Logic gates: Implement basic logic functions like NOT, AND, OR, NAND, etc., forming the foundation for more complex circuits.   * **Control and Automation:**   State machines: Implement sequential logic using clocked combinational circuits, controlling devices like traffic lights, elevators, and industrial robots.  Comparators: Compare voltage levels and trigger actions based on the comparison, used in threshold detectors, voltage regulators, and alarms.  Code converters: Convert between different logic families (e.g., TTL to CMOS) ensuring compatibility between various components.  Pulse width modulation (PWM) circuits: Generate variable-duty cycle signals for controlling motor speed, LED brightness, and other power applications   * **Signal Processing and Communication:**   Analog-to-digital converters (ADCs): Convert analog signals (e.g., audio, temperature) into digital data using comparators and encoders.  Digital-to-analog converters (DACs): Convert digital data back into analog signals for audio playback, sensor output, and control signals.  Filters: Remove unwanted frequencies or noise from signals using logic gates and timing circuits.  Error correction circuits: Detect and correct errors in data transmission using coding techniques and logic functions. |

### **Theoretical learning Activity**

* Three trainees in group discuss about types of combinational circuits

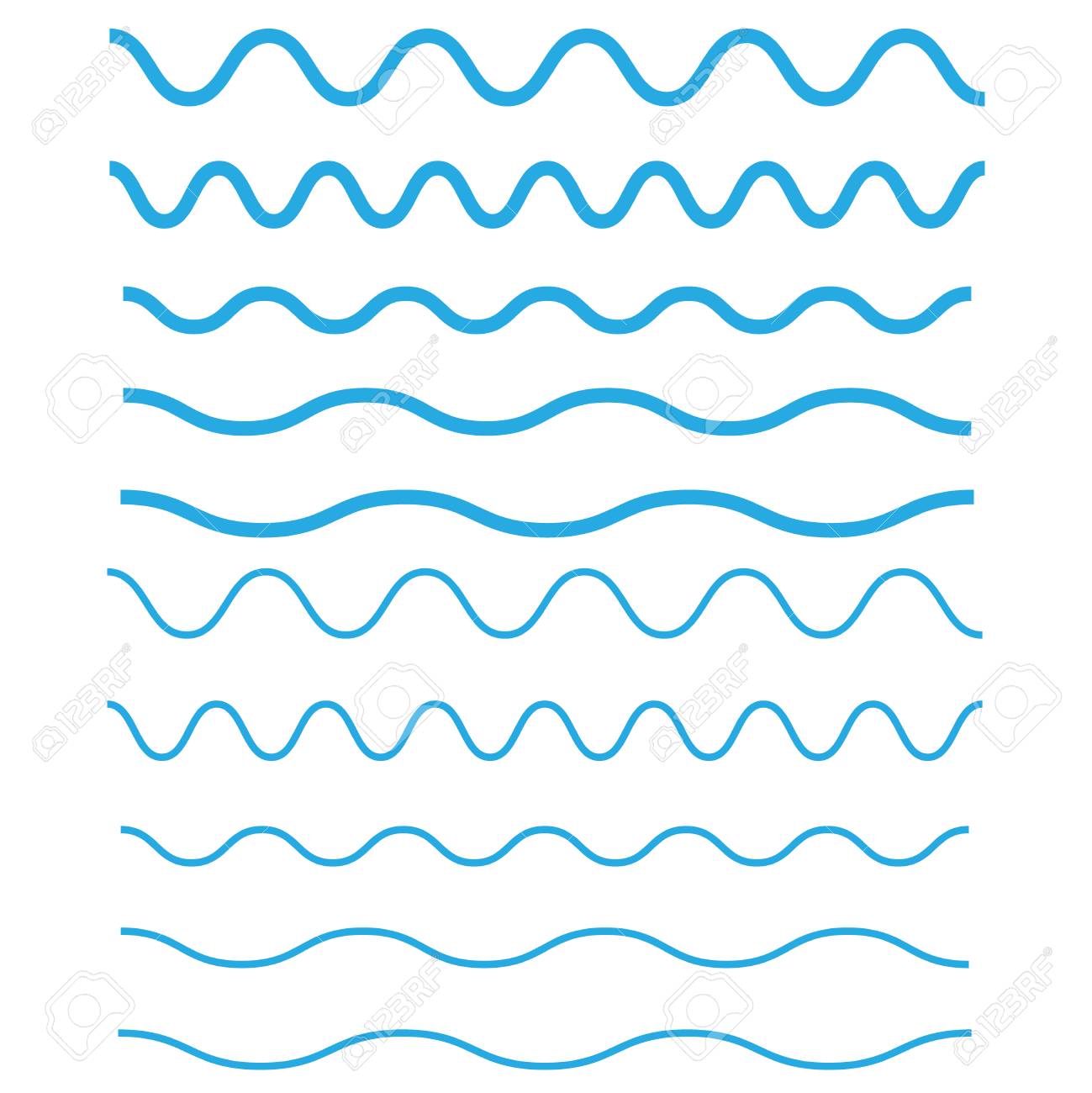
### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * A combinational logic circuit is a circuit in which we combine different gates in the circuit * Types of logic circuits : Adder and Subtractor, Comparator ,Multiplexer and De-multiplexer , Encoder and Decoder |

## **Indicative content4.****2: Multivibrators using ICs**



## 

|  |
| --- |
| **Multivibrators using ICs** Multivibrators are versatile analog circuits that generate periodic or non-periodic waveforms using integrated circuits (ICs). Depending on their configuration and the chosen IC, they can produce various types of waveforms like square waves, triangular waves, and sawtooth waves. Here's a breakdown of multivibrators using ICs:   * Types of Multivibrators:   Monostable Multivibrator: Generates a single pulse of adjustable duration when triggered by a pulse on its control pin. Useful for timers, delay circuits, and pulse generators.  Popular ICs: 555 timer (NE555, LM555), 74LS221, CD4047  Astable Multivibrator: Creates a continuous square wave oscillation with adjustable frequency and duty cycle. Ideal for audio tone generators, blinkers, and LED sequencers.  Popular ICs: 555 timer (NE555, LM555), 74HC106, CD4013  Bistable Multivibrator (Flip-Flop): Switches between two stable states (high and low voltage) based on input signals or clock pulses. Used for memory elements, data storage, and sequential logic circuits.  Popular ICs: 74LS74, 74HC75, CD4017  Common ICs for Multivibrators:  555 Timer: Highly versatile IC offering both monostable and astable functionalities with adjustable timing parameters using external resistors and capacitors.   * 74HC/LS Logic Gates:    Can be configured to create astable multivibrators by connecting specific gates (e.g., NAND gates) with feedback loops and timing capacitors.  Specialized Multivibrator ICs: Certain ICs like the CD4047 and CD4013 are specifically designed for monostable and astabl emultivibrator applications, simplifying circuit design.   * Applications of Multivibrators:   Timers and delays: Controlling the duration of events in various circuits.  Pulse generators: Creating square waves for clock signals, driving other circuits, and testing purposes.  Tone generators: Generating audio tones for simple sound applications.  Blinkers and LED sequencers: Controlling the flashing patterns of LEDs.  Data storage and logic circuits: Flip-flops are essential for building memory elements and complex digital systems.   * Astable Multivibrator Using 555 Timer   The designing and working of astable multivibrator using a 555 timer IC is done by using transistors and operational amplifiers. The 555 timer IC affords exact time delay from ms to hours. The oscillation frequency can be measured manually by small modification. 555 timer IC is a relatively cheap, stable and user-friendly integrated circuit and that is apt for circuit designers with for both astable and monostable applications.  The first 555 integrated circuit was designed in the year 1971 by the corporation of Signetics as the SE555 or NE555. Astable multivibrator using 555 IC is a simple oscillator circuit that generates continuous pulses. The frequency of the circuit can be controlled by shifting the values of resistors R1, R2 ad capacitor C1. |

### **Theoretical learning Activity**

* Three trainees in group discuss about applications of multivibrator using ICs

### **Practical learning Activity**

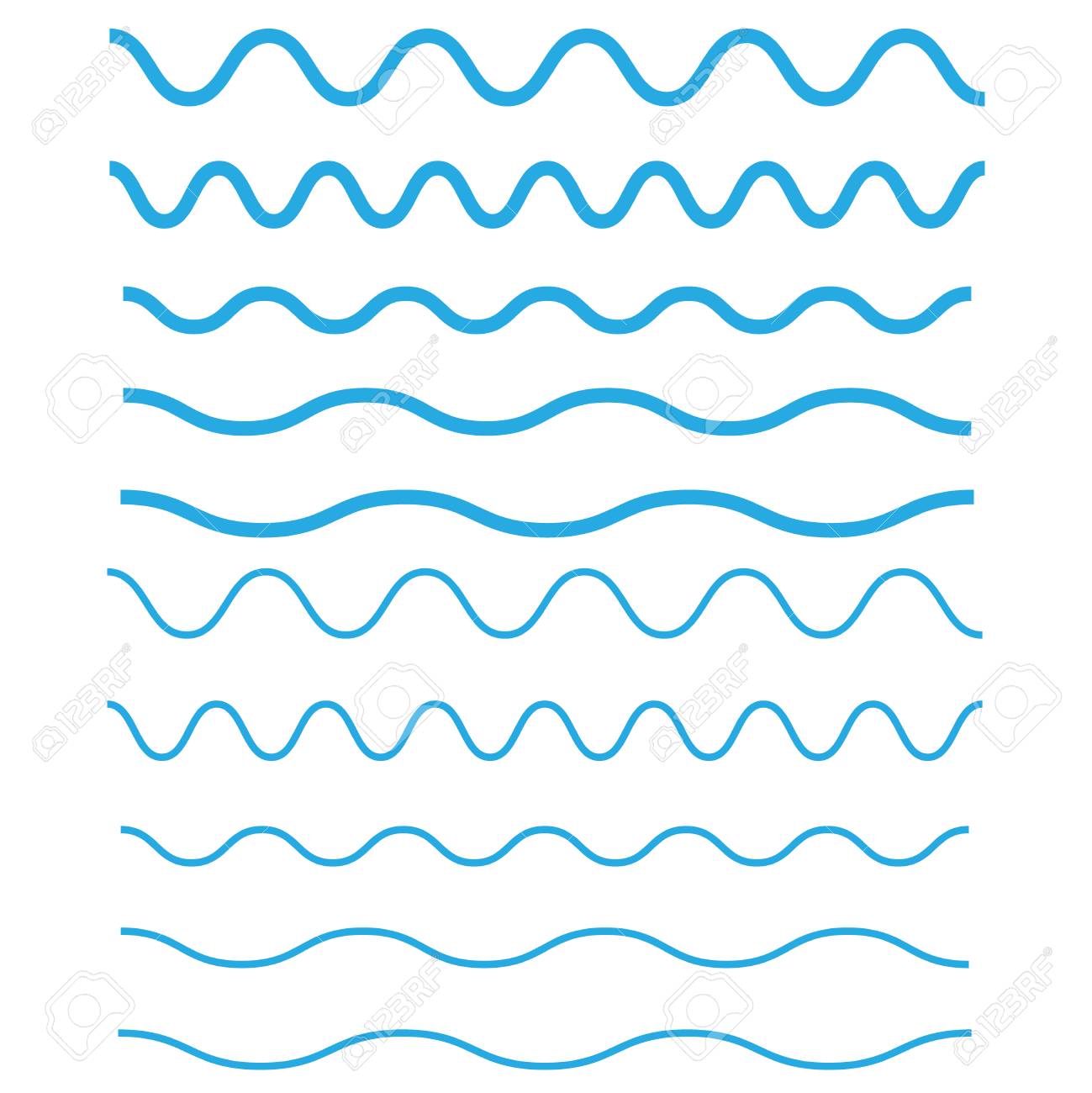
* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * Types of multivibrators are Astable, monostbale and Bistable * Applications of multivibrators:   Controlling the duration of events in various circuits.  Creating square waves for clock signals, driving other circuits, and testing purposes.   Generating audio tones for simple sound applications.  Controlling the flashing patterns of LEDs. |

.

## **Indicative content4.3: Apply Sequential logic circuits**



## 

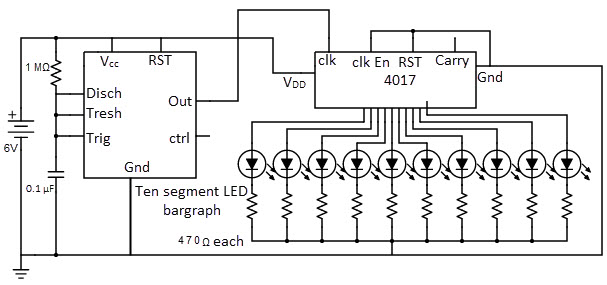
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Design and Working principle**  * **Introduction**   Sequential circuit has memory so output can vary based on both current and previous inputs.  This type of circuits uses previous input, output, clock and a memory element.  The word “Sequential” means that things happen in a “sequence”, one after another and in Sequential Logic circuits, the actual clock signal determines when things will happen next.  Sequential circuits are based on flip flops.  **Block diagram**  sequential logic representation  In automata theory, sequential logic is a type of logic circuit whose output depends on the present value of its input signals and on the sequence of past inputs, the input history. This is in contrast to combinational logic, whose output is a function of only the present input. **Types of Sequential logic circuits**  * Inputs to the sequential circuit can only affect the storage element at given instance of time. * *Edge-triggere*d *circuits* change state on the *rising edge* or *falling edge* of the clock pulse.change state when the *Level-triggered circuits*  clock voltage reaches its *highest* or *lowest*     Asynchronous circuit (clockless or self-timed circuit) is a sequential digital logic circuit that does not use a global clock circuit or signal generator to synchronize its components. Instead, the components are driven by a handshaking circuit which indicates a completion of a set of instructions. **Apply Flip-Flops** Flip-flops (FFs) are fundamental building blocks in digital electronics, acting as memory elements that store binary information (0 or 1) and change state based on specific input signals. Understanding the different types of flip-flops helps you choose the right one for your digital circuits. Here's a breakdown of the most common types:   * **SR Flip-Flop:**   This is the most common flip-flop among all. This simple flip-flop circuit has a set input (S) and a reset input (R). In this system, when you Set “S” as active, the output “Q” would be high, and “Q**‘**” would be low. Once the outputs are established, the wiring of the circuit is maintained until “S” or “R” go high, or power is turned off.  As shown above, it is the simplest and easiest to understand. The two outputs, as shown above, are the inverse of each other. The **truth table of SR Flip-Flop** is highlighted below.   |  |  |  |  | | --- | --- | --- | --- | | **S** | **R** | **Q** | **Q’** | | 0 | 0 | 0 | 1 | | 0 | 1 | 0 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | ∞ | ∞ |  * Simplest FF with two inputs (Set and Reset). * Setting both inputs high outputs a 1, while resetting both outputs a 0.   Only one input can be high at a time, leading to undefined state if both are high or low simultaneously.  Used for data transfer and synchronization functions.   * **JK Flip-Flop:**   Due to the undefined state in the SR flip-flops, another flip-flop is required in electronics. The JK flip-flop is an improvement on the SR flip-flop where S=R=1 is not a problem.    The input condition of J=K=1 gives an output inverting the output state. However, the outputs are the same when one tests the circuit practically.  In simple words, If J and K data input are different (i.e. high and low), then the output Q takes the value of J at the next clock edge. If J and K are both low, then no change occurs. If J and K are both high at the clock edge, then the output will toggle from one state to the other. JK Flip-Flops can function as Set or Reset Flip-flops.  ****JK FF****Truth Table:   |  |  |  |  | | --- | --- | --- | --- | | **J** | **K** | **Q** | **Q’** | | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | | 1 | 0 | 0 | 1 | | 1 | 1 | 0 | 1 | | 0 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 1 | 0 |  * More versatile than SR FF, incorporating two inputs (J and K) and a clock input (CLK). * Setting both J and K high toggles the output (0 to 1 or 1 to 0). * Holding J and K low maintains the current state. * Clock edge triggers the state change based on J and K inputs. * Ideal for counters, shift registers, and frequency dividers. * **D Flip-Flop:**   D flip-flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift registers and input synchronization.   ***Truth Table:***  |  |  |  |  | | --- | --- | --- | --- | | **Clock** | **D** | **Q** | **Q’** | | ↓ » 0 | 0 | 0 | 1 | | ↑ » 1 | 0 | 0 | 1 | | ↓ » 0 | 1 | 0 | 1 | | ↑ » 1 | 1 | 1 | 0 |  * Edge-triggered FF with a single data input (D) and a clock input (CLK). * When CLK rises, the output copies the value of D. * Simple and reliable for data storage and transfer applications. * **T Flip-Flop:**   A T flip-flop is like a JK flip-flop. These are basically single-input versions of JK flip-flops. This modified form of the JK is obtained by connecting inputs J and K together. It has only one input along with the clock input.    Truth Table:   |  |  |  | | --- | --- | --- | | **T** | **Q** | **Q (t+1)** | | 0 | 0 | 0 | | 1 | 0 | 1 | | 0 | 1 | 1 | | 1 | 1 | 0 |  **Apply Counters** A counter is a device in computing and digital logic,that is used to store and display the particular event so many times. The most common type of a counter is a sequential digital logic circuit. This circuit consists of one i/p line, namely clock and number of o/p lines.  The values of the o/p lines denote a number in the BCD or binary number system.  Generally, these circuits are designed with flip-flops which are connected in cascade.  These devices are widely used in digital circuits and are designed as separate ICs and also combined as parts larger integrated circuits. This article discusses about what is an electronic counter and its types. Please follow the below link to know more about:  **Types of counters**    * **Up/Down counter**   A bidirectional counter is a synchronous up/down binary counter that has the ability to count in both directions either to or from some preset value as well as zero.  As well as counting “up” from zero and increasing or incrementing to some preset value, it is sometimes necessary to count “down” from a predetermined value to zero allowing us to produce an output that activates when the zero count or some other pre-set value is reached.  This type of counter is normally referred to as a **Down Counter**, (**CTD**). In a binary or BCD down counter, the count decreases by one for each external clock pulse from some preset value. Special dual purpose IC’s such as the TTL 74LS193 or CMOS CD4510 are 4-bit binary Up or Down counters which have an additional input pin to select either the up or down count mode.   Decade Counter A decade counter is used to count decimal digits rather than binary and it may have each or other binary codings. A normal 4-stage counter can be easily changed to a decade counter by adding a NAND gate as shown in the below figure. You can observe that Flip Flop2 & Flip Flop4 provide the I/PS to the NAND gate. The o/ps of this gate are connected to the CLR i/p of each of the Flip Flops. A decade counters counts from 0-9 and then changes to 0. The o/p of the counter can be set to ‘0’ by pulsing the reset line low. The count of the counter increases on each CLK pulse until it reaches to 1001. When it increases to 1010 both I/PS of the NAND gate go to high. The result of the NAND gate output goes low, and changes the counter to ‘0’. D going low can be a CARRY OUT signal, indicating that there has been a count of ten.     * **A ripple counter**   It is a type of asynchronous counter circuit in which the clock pulse propagates through the flip-flops one at a time, like a ripple in water. This means that the output of each flip-flop triggers the clock input of the next flip-flop in the chain     * Ripple counter works:   The first flip-flop (FF1) is clocked by the external clock signal. This causes the output of FF1 to toggle (change state from 0 to 1 or vice versa).  The output of FF1 then becomes the clock input for the second flip-flop (FF2). However, due to the delay in the flip-flop, the output of FF2 will not change until the next clock pulse arrives at FF1.  On the next clock pulse, the output of FF1 changes again, which triggers the clock input of FF2. This time, the output of FF2 will toggle.  This process continues down the chain of flip-flops, with each flip-flop's output triggering the clock input of the next flip-flop. As a result, the counter "ripples" through its sequence of states.   * Ring counter   A ring counter is a type of sequential logic circuit built using flip-flops connected in a closed loop, with the output of the last flip-flop fed back to the input of the first. This creates a circulating data pattern that shifts around the ring with each clock pulse  The circuit diagram of the ring counter    ***Operation of Ring Counter*** Initially, all the flip flops in ring counter are reset to 0 by applying CLEAR signal. Before applying the clock pulse, we apply the PRESET pulse to the flip flops which assigns the value ‘1’ to the ring counter circuit. For each clock signal, the data circulates among all the 4 flip flop stages of ring counter.This 4 staged ring counter is called Mod 4 ring counter or 4 bit ring counter. To circulate the data correctly in the ring counter, we must load the counter with required values like all 0’s or all 1’s.  *Truth table of ring counter*   |  |  |  |  | | --- | --- | --- | --- | | Q0 | Q1 | Q2 | Q3 | | 1 | 0 | 0 | 0 | | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 0 | 0 | 1 |   When CLEAR input CLR = 0, then all flip flops are set to 1. When CLEAR input CLR = 1, the ring counter starts its operation. For one clock signal, the counter starts its operation. On next clock signal, the counter again resets to 0000. Ring counter has 4 sequences: 0001, 0010, 0100, 1000, 000. ***Johnson Counter*** A Johnson counter is an altered ring counter, where the o/p of the last stage is reversed and fed back as i/p to the first stage. The register cycles through an arrangement of bit pattern length is equal to double the length of the shift register. The applications of these counters involve similar to the decade counter, DAC, etc. They can be easily designed using JK-FF.  It is also termed as a twisted ring counter    Thus, this is all about what is counter, electronic counter, circuit diagram and its types. We hope that you have got a better understanding of this concept. Furthermore, any queries regarding this topic, please give your valuable suggestions by commenting in the comment section below. Here is a question for you, what is the function of counter? **Apply Shift Registers** A shift register is a type of digital circuit that stores and manipulates data by shifting it one bit at a time. It's essentially a chain of connected flip-flops, where each flip-flop acts as a single-bit storage element. Imagine a bucket brigade, where data is passed from one "bucket" (flip-flop) to the next with each beat of a drum (clock pulse).    **Components:**   * Flip-flops: These are the basic building blocks of a shift register. Each flip-flop can hold a single bit of data (0 or 1). * Clock signal: This is a periodic signal that synchronizes the shifting process. With each clock pulse, the data in each flip-flop moves to the next one. * Input and output pins: Data is entered into the register through the input pin, and it exits through the output pin. * **Types of Shift Registers:** * Serial-in/serial-out (SISO): Data enters and exits the register one bit at a time. This is the most common type and is used for serial data transmission and storage. * Serial-in/parallel-out (SIPO): Data enters the register one bit at a time, but exits in parallel (all bits simultaneously). This type is used for converting serial data to parallel data for processing. * Parallel-in/serial-out (PISO): Data enters the register in parallel, but exits one bit at a time. This type is used for converting parallel data to serial data for transmission. * Parallel-in/parallel-out (PIPO): Data enters and exits the register in parallel. This type is used for temporary data storage and buffering. * **Applications of Shift Registers:** * Serial data communication: Shift registers are used to convert parallel data into a serial stream for transmission over a single wire. This is used in many communication protocols, such as USB and SPI. * Data storage and delay: Shift registers can be used to temporarily store data or delay a signal by a specific number of clock cycles. * Digital signal processing: Shift registers can be used to implement various digital signal processing algorithms, such as filtering and modulation. * Control logic: Shift registers can be used to implement simple control logic circuits, such as counters and state machines |

### **Theoretical learning Activity**

* Trainees in group discuss on types of counters, registers, comparators and flip-flop
* Trainees within groups brainstorm about applications of counters, comparators, comparators and flip-flop

### **Practical learning Activity**

XYZ electronics Ltd is electronic company located in KIGALI-CITY. This company providing counters, It wants a specialized technician for implementing counters by using CD 4017 and timer as one of them using the bellow circuit diagram.



This activity performed in two hours (2hours).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Indicators and Elements of verification** | | | **Score** | |  |
| Indicator 1: materials and equipment’s are selected | | | Yes (Y) | No (N) |  |
| *Element 1* | | PCB is well selected |  |  |  |
| *Element 2* | | Resistors are well selected based on their values |  |  |  |
| *Element 3* | | Capacitor is well selected based on their values |  |  |  |
| *Element 4* | | Diode are well selected |  |  |  |
| *Element 5* | | Integrated circuit are well selected |  |  |  |
| *Element 5* | | Multmeter is well selected |  |  |  |
| Indicator 2: material and equipment’s are arranged in the working area | | |  |  |  |
| *Element 1* | | Materials are well arranged |  |  |  |
| *Element 2* | | Equipment is well arranged |  |  |  |
| Indicator 3:Implementation | | |  |  |  |
| *Element 1* | | Circuit is well soldered |  |  |  |
| *Element 2* | | Circuit is well supplied |  |  |  |
| *Element 3* | | Circuit is well functioned |  |  |  |
| *Element 4* | | The circuit is provided the output expected |  |  |  |
| *Element 5* | | The layout of the circuit is well respected |  |  |  |
|  |

Points to Remember (Take home message)

|  |
| --- |
| * Types of counters : Up/Down counter , Decade counter, Ripple counter, Ring counter Johnson counter * Types of shift registers: Serial In − Serial Out , Serial In − Parallel Out , Parallel In − Serial Out Parallel In − Parallel Out , Universal shift Register * Types of flip-flop: Set-Reset (SR), Toggle (T),Data (D), Jack Kilby (JK) |

**Learning o****utcome 4: formative assessment**

**Written assessment**

A. 1) Among the internal parts of a 555 timer multivibrator, there is a comparator which is used to compare between input and output voltage divider

Choose among the following statements which one is correct referring to the function of the comparators in the 555 timer circuit

a) To compare the output voltages to the internal voltage divider

b)To compare the input voltages to the internal voltage divider

c) To compare the output voltages to the external voltage divider

d) To compare the input voltages to the external voltage divider

2)A ripple counter It is a type of asynchronous counter circuit in which the clock pulse propagates through the flip-flops one at a time, like a ripple in water.

This means that the output of each flip-flop triggers the clock input of the next flip-flop in the chain. This counter has the speed which is limited by the propagation delay of the flip-flop. Referring to the above statement, answer the following questions by circling the right answer.

i) A ripple counter’s speed is limited by the propagation delay of............

a)Each flip-flip

b) All flip-flips and gates

c) The flip-flops only with gates

d) Only circuit gates

ii) How many flip-flips are required to construct a decade counter?

A**)** 4 B) 8 C) 5 D) 10

**3. Answer by using true or false**

1)A universal shift register has both serial and parallel input and output capacity

2) A comparator simply outputs a HIGH or LOW based on a comparison of the voltage levels at its output

**4.** Define the following terms:

1. Counter

2. Shift register

**5.** Enumerate different modes of operation of 555timer multivibrator?

**6**. Differentiate between synchronous and asynchronous counter?

**ANSWERS**

1) What is the function of the comparators in the 555 timer circuit?

a) To compare the output voltages to the internal voltage divider

**b) To compare the input voltages to the internal voltage divider**

c) To compare the output voltages to the external voltage divider

d) To compare the input voltages to the external voltage divider

2.

ii) A ripple counter’s speed is limited by the propagation delay of............

**a) Each flip-flip**

b) All flip-flips and gates

c) The flip-flops only with gates

d) Only circuit gates

iii) How many flip-flips are required to construct a decade counter?

A**) 4** B) 8 C) 5 D) 10

**3. Answer b y true or false**

1)A universal shift register has both serial and parallel input and output capacity **/T**

2) A comparator simply outputs a HIGH or LOW based on a comparison of the voltage levels at its output **/T**

**4. Define the following terms:**

1**. Counter** is a device which stores (and sometimes displays) the number of times particular event or process has occurred, often in relationship to a clock signal

2. **Shift register** is a cascade of flip-flops sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it, shifting in the data present at its input and shifting out the last bit in the array, at each transition of the clock input

**5.** Enumerate different modes of operation of 555timer multivibrator?

Stable, Astable and Bistable

**6**. Differentiate between synchronous and asynchronous counter?

Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops

Synchronous counter – all state bits change under control of a single clock

**References**

* Digital Fundamentals( E LEVENTH EDITION Thomas L. Floyd)
* Digital Electronics Principles, Devices and Applications( Anil K. Maini Defence Research and Development Organization (DRDO), India)
* R.Tokheim. (2014). Digital Electronics Principles and Applications. USA: 8th ed.
* Theraja, B. (n.d.). A textbook of Electrical Technology Vol IV-Electronic Devices and circuits.

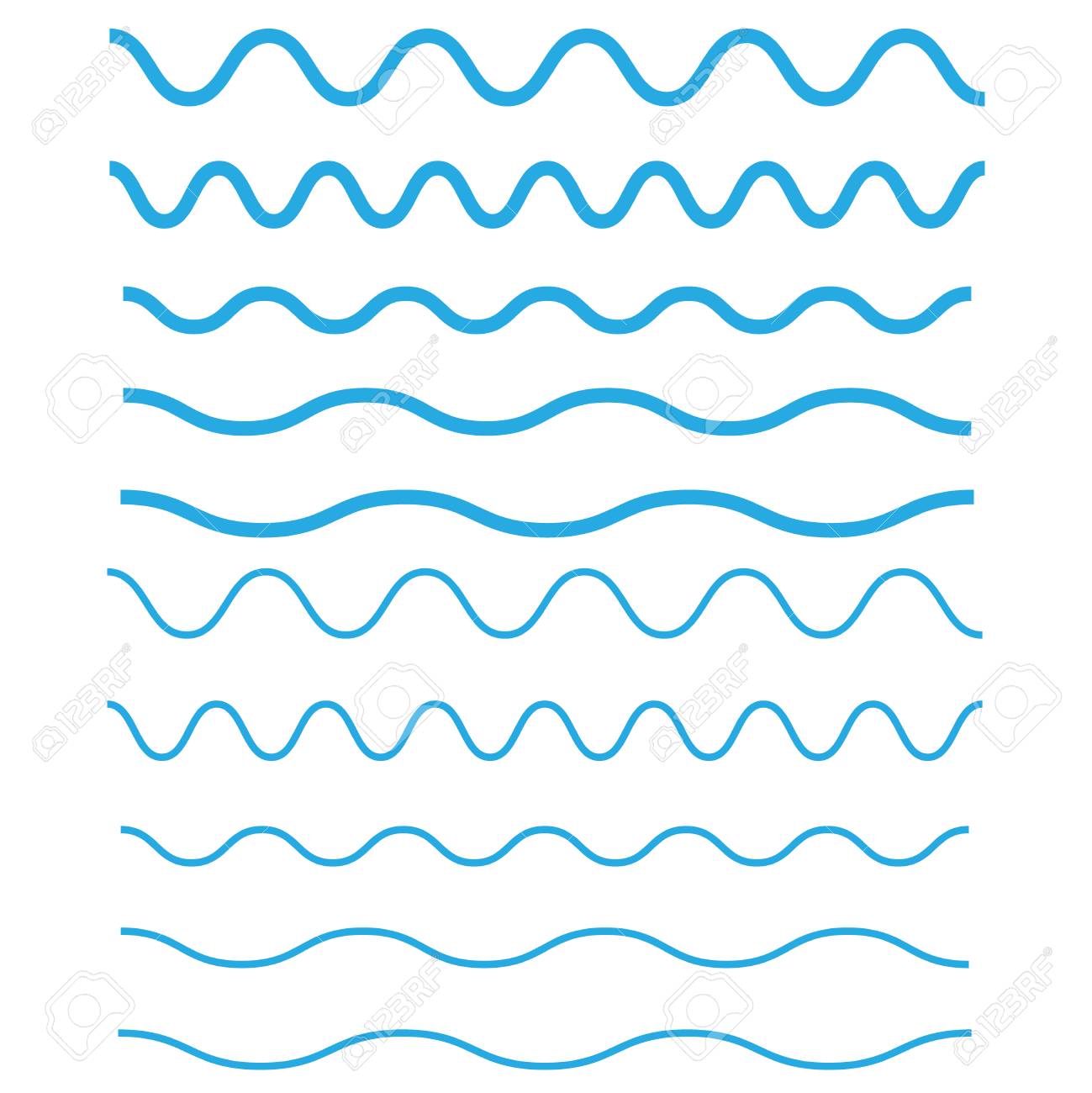
# **Learning outcome 5: Apply Programmable Logic Devices**

|  |
| --- |
|  |

|  |
| --- |
| **Indicative contents**  **5.1: Introduction to Programmable Logic Devices (PLD)**  **5.2: Types of PLDs**  **5.3: Programming languages of PLDs** |

|  |  |  |
| --- | --- | --- |
| **Duration:10hrs** | | |
| **Learning outcome 5 objectives:**  By the end of the learning outcome, the trainees will be able to:  1. Describe properly PLD’s working principle is according to its applications  2. Classify clearly PLD types are according to their architecture, logic capacity and programmability  3.identify properly PLD’s Programming Languages according to their types | | |
| **Resources** | | |
| **Equipment** | **Tools** | **Materials** |
| Computer  projector  Multimeters  DC power supply | Books  PLD programming software | Markers  chalks  board  Internet  logic ICs  PLDs  breadboard  jumper wires  electronic components |
| **Advance preparation:**   * Research on PLD ,types and Working principle * Research on Programmable logic languages | | |

## **Indicative content 5.1: Introduction to Programmable Logic Devices (PLD)**



## 

|  |
| --- |
| An IC that contains large numbers of gates, flip-flops, etc. that can be configured by the user to perform different functions is called a Programmable Logic Device (PLD). It permits elaborate digital logic designs to be implemented by the user on a single device. The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.  The fixed logic system has circuits whose configurations are permanent. Their instructions perform only a fixed set of operations repeatedly. Once manufactured and programmed, the logic cannot be changed. This system is a fantastic asset for repeated tasks.  But one tiny mistake in the manufacturing process like uploading the wrong code in the device, and the entire system is discarded, and a new design is developed. That’s quite some risk that companies aren’t willing to take unless necessary. Additionally, fixed logic does not allow the users to expand or build on their existing functionalities.  Thus, we need something more flexible, easy to work, and more cost-efficient. Thus, programmable logic comes to the rescue. It is easy-to-program, affordable and equipped with better features. Inexpensive software is used to develop, code and test the required design. This design is then programmed into a device and tested in a live electronic circuit.  The corresponding performance then decides if the logic needs to be altered, or if the prototype is fit to be determined as the final design itself. The fixed logic system thus offers limited usability; a programmable logic seems more feasible and beneficial. **PLD Working principle** Programmable logic devices (PLDs) are a class of integrated circuits that can be configured by the user to perform specific logic functions. Unlike fixed-function logic devices, such as ASICs (application-specific integrated circuits), PLDs offer the flexibility to be programmed and reprogrammed multiple times, making them ideal for prototyping, low-volume production, and applications where the logic requirements may change.  **8**   * ****Basic Architecture:****   PLDs typically consist of an array of logic blocks, programmable interconnection resources, and input/output (I/O) cells. The logic blocks contain configurable elements like logic gates (AND, OR, NAND, NOR, etc.) and flip-flops, which can be interconnected using the programming resources to implement the desired logic function.   * **Benefits of using PLDs:** * Flexibility: PLDs can be programmed to implement a wide range of logic functions, making them suitable for various applications. * Reduced development time: Prototyping and design iterations are faster with PLDs as they can be reprogrammed without requiring physical changes to the circuit. * Lower cost: Compared to ASICs, PLDs offer a more cost-effective solution for low-volume production and prototyping. * Reduced time-to-market: The flexibility and faster development cycle of PLDs enable quicker product launches. * **Applications of PLDs:**   PLDs are used in various applications across different industries, including:  Digital signal processing: Implementing filters, codecs, and other signal processing algorithms.  Telecommunications: Building data communication interfaces, network equipment, and mobile phone base stations.  Industrial control: Designing control systems for robots, machines, and other industrial equipment.  Medical devices: Implementing diagnostic and therapeutic equipment functions.  Consumer electronics: Developing features for gaming consoles, televisions, and other devices.  **PLD Working principle**  The working principle of a programmable logic device (PLD) involves three key aspects:  1**. Programmable Configurability:**  Unlike fixed-function logic chips, PLDs contain arrays of basic logic elements like logic gates, flip-flops, and memory cells.  These elements are not pre-wired, but their connections and functions can be programmed by the user.  Programming typically involves applying specific voltages or pulses to configuration fuses inside the chip. These fuses act as tiny switches, permanently altering the internal connections and defining the desired logic function.  **2. Logic Implementation:**   * Once programmed, the internal connections of the PLD implement a custom logic circuit based on the user's program. This program can be written in various ways, such as Hardware Description Languages (HDLs) or graphical tools. * The programmed logic can perform complex operations like:   + Combinational logic: Using AND, OR, NOT gates to determine output based on a combination of input signals.   + Sequential logic: Utilizing flip-flops to store and update data based on previous inputs and outputs.   + State machines: Defining sequences of operations based on current state and input conditions.   **3. Application in Various Devices:**   * PLDs find application in diverse electronic systems, including:   + Programmable Logic Controllers (PLCs): Used in industrial automation for controlling machinery and processes based on sensor inputs and program logic.   + Field-Programmable Gate Arrays (FPGAs): More complex PLDs with larger logic capacity, providing flexible hardware platforms for digital signal processing, embedded systems, and custom chip design.   + Complex Programmable Logic Devices (CPLDs): Offer a mix of logic gates and dedicated logic blocks for specific functions like I/O interfaces and state machines |

### **Theoretical learning Activity**

* Learners in group brainstorm about working principle and applications of PLDs

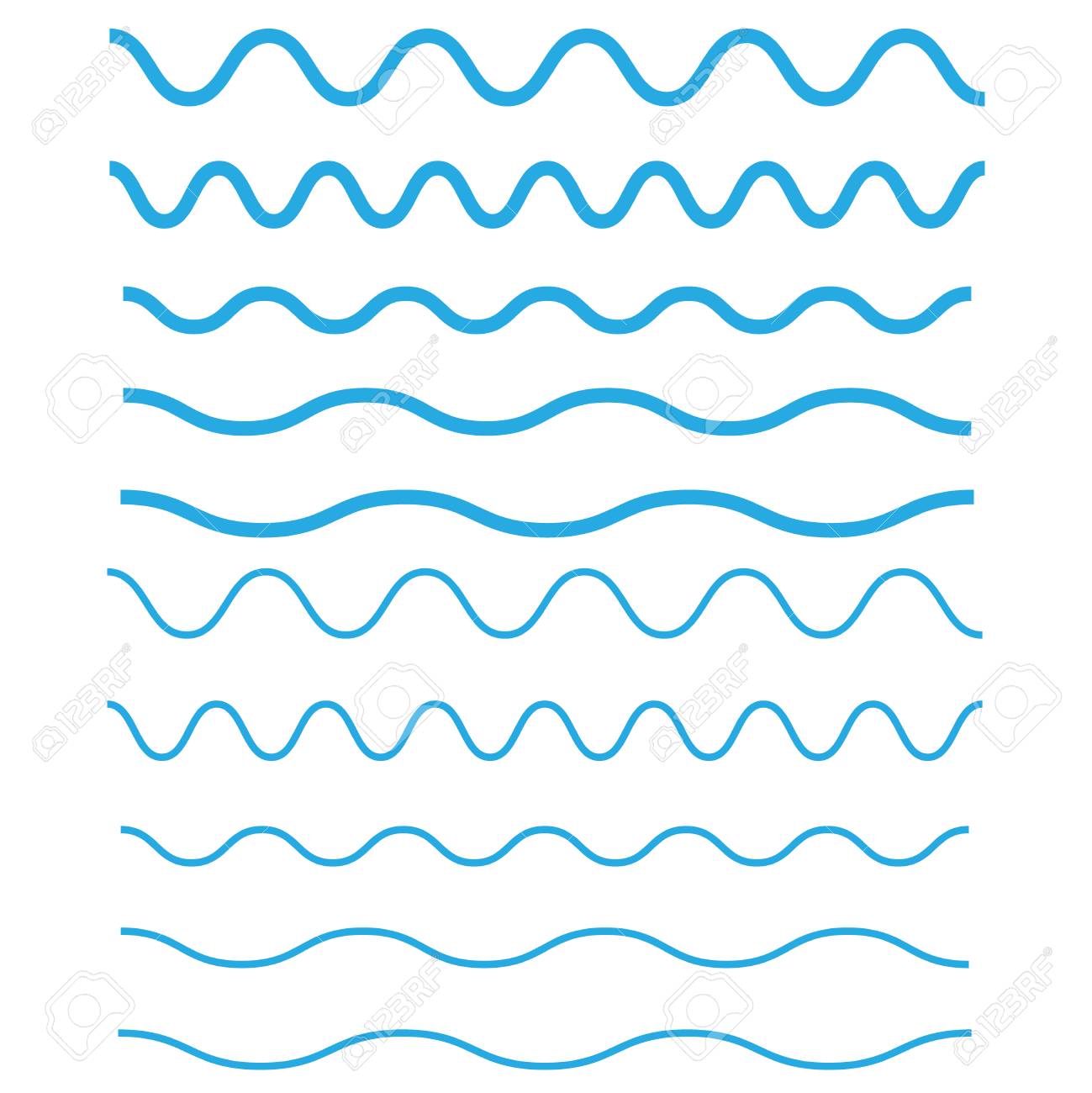
### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * **Comparison: programmable logic Vs fixed logic**   The fixed logic system has circuits whose configurations are permanent. Their instructions perform only a fixed set of operations repeatedly. Once manufactured and programmed, the logic cannot be changed. This system is a fantastic asset for repeated tasks.  But one tiny mistake in the manufacturing process like uploading the wrong code in the device, and the entire system is discarded, and a new design is developed. That’s quite some risk that companies aren’t willing to take unless necessary. Additionally, fixed logic does not allow the users to expand or build on their existing functionalities. |

## **Indicative content 5.2: Programming languages of PLDs**



|  |
| --- |
| **Description of Programming languages**  Programming languages is methodology for formally defining the syntax and semantics of programming languages.  Programming languages combine: Mathematical notation, Natural language  Diagrams.  Key Components of PLD:   * Lexical Structure:   Defines the basic building blocks of a language, like keywords, identifiers, operators, and punctuation.  Example: Lexical structure of Python, showing tokens and rules   * Syntax:   Specifies the rules for combining these blocks into valid program structures.  Typically represented using grammars, such as Backus-Naur Form (BNF).  Example: BNF grammar for a simple expression language   * Semantics:   Defines the meaning of these structures, explaining how they are interpreted and executed by a computer.  Often expressed using operational semantics, de-notational semantics, or axiomatic semantics.  Example: Operational semantics for a factorial function **Programming process**   * Language Design:   + Define the intended purpose and target domain of the language.   + Determine the features and constructs needed.   + Use PLD to formally specify the language's syntax and semantics. * Interpreter or Compiler Development:   + Develop a tool to translate programs written in the language into machine-executable code (compiler) or execute them directly (interpreter).   + PLD provides a precise blueprint for language implementation. * Programming:   + Developers write programs using the language's constructs and rules, guided by its PLD. * Verification and Validation:   + PLD can aid in verifying the correctness of programs and the language implementation itself.   + It can support formal proofs of program properties and consistency checks. * Testing and Debugging:   + PLD can help identify potential errors and guide testing strategies.   + It can clarify expected behaviour and pinpoint deviations. * Documentation:   + PLD serves as a comprehensive reference for language features and usage.   + Clear and precise documentation enhances understanding and adoption. * Maintenance and Evolution:   + PLD facilitates language updates and extensions by providing a clear understanding of its structure and behaviour.   + It supports consistent modifications and preservation of language properties.  **Functional Simulation**  **Functional Simulation of a PLD (Programmable Logic Device) is a crucial step in the design process that involves verifying the intended logic behavior of a design before it's physically programmed onto the device**.**** It's a software-based simulation that doesn't consider actual device delays, but rather focuses on ensuring the logic functions as intended. The following are key steps for functional simulation:   * Generate a Functional Simulation Net list: * This net list represents the logical connections of the design, independent of the target PLD device.It's often generated from HDL (Hardware Description Language) code or a schematic design. * Functional Simulation Net list Generation: * Create a Test bench: * This is a separate file containing input stimuli (test vectors) to exercise different logic paths in the design. * It also includes expected output values for verification. Test bench Creation: * Run the Simulation: * The simulator applies the test vectors from the test bench to the functional simulation net list. * It calculates and displays output values in a waveform viewer. * You compare these outputs with the expected values to verify the design's correctness. Simulation Waveform Viewer: * Analyze Results and Debug: * If discrepancies are found, you'll need to debug the design and modify either the HDL code or schematic. * Repeat the simulation process until the design behaves as expected  **Timing Simulation**  The simulator utilizes the timing model to account for delays during simulation.  It verifies that signals meet setup and hold timing requirements for flip-flops and registers.  It identifies potential timing violations that could lead to functional errors. Timing Simulation Waveform Viewer  Device Programming, often referred to as "downloading," is the process of transferring a PLC program from the programming environment to the PLC's memory, enabling the PLC to execute the control logic and manage field devices. The following are steps for timing simulation:  The simulator utilizes the timing model to account for delays during simulation.  It verifies that signals meet setup and hold timing requirements for flip-flops and registers.  It identifies potential timing violations that could lead to functional errors. Timing Simulation Waveform Viewer  Device Programming, often referred to as "downloading," is the process of transferring a PLC program from the programming environment to the PLC's memory, enabling the PLC to execute the control logic and manage field devices.  Establish Communication:  Connect the PLC to the programming device (typically a computer) using a suitable communication interface, such as:   * + - Ethernet cable     - USB cable     - Serial cable     - Wireless connection   + Configure the communication settings in both the PLC and programming software to match. * Verify PLC Compatibility:   + Ensure the programming software is compatible with the specific PLC model and firmware version.   + Update software or firmware if necessary. * Open PLC Program:   + Load the PLC program you've created or edited in the programming software. * Initiate Download:   + Use the software's download function to transfer the program to the PLC.     - The software typically compiles the program into machine code understandable by the PLC.     - It then sends the code to the PLC's memory, often overwriting any existing program. * Monitor Progress:   + The software usually displays a progress bar or status messages indicating the download status.   + Ensure successful completion to avoid potential errors. * Verify Download:   + Once complete, check the PLC's status indicators or software feedback to confirm successful transfer.   + Some PLCs may require a restart for the new program to take effect  **Device Programming (Downloading)** PLD programming devices are essential tools used to transfer the desired logic configuration onto a PLD (Programmable Logic Device). They connect to a computer and the PLD, enabling the programming software to write the configuration data into the PLD's internal memory.  Common types of PLD programming devices include:   * Dedicated Programmers: * Standalone devices specifically designed for programming PLDs. * Vary in size, features, and supported PLD types. * Typically connect to a computer via USB or serial port. * Offer user-friendly interfaces and advanced features like:   + Automatic device identification   + Programming verification   + Security options * Development Boards: * Integrated circuits that combine a PLD with other components for prototyping and development. * Often include a built-in programmer for direct programming. * Provide a convenient testing platform for PLD designs. * In-System Programmers (ISPs): * Programming devices that can program a PLD while it's already installed in a system. * Used for updating firmware or making design changes without removing the device.   Typically connect to a computer via USB or JTAG interface   * Universal Programmers: * Versatile devices that can program a wide range of PLDs as well as other programmable devices like microcontrollers and memories.   Common in production environments where multiple device types are used |

### **Theoretical learning Activity**

* Individual work about different types of programming languages

### **Practical learning Activity**

* Not applicable

Points to Remember (Take home message)

|  |
| --- |
| * Programming languages is methodology for formally defining the syntax and semantics of programming languages. * Programming languages combine: Mathematical notation, Natural language * **Function of PLD:** A programmable logic device (PLD) is an electronic component used to build reconfigurable digital circuits. Unlike digital logic constructed using discrete logic gates with fixed functions, a PLD has an undefined function at the time of manufacture |

**Learning** **outcome 5: formative assessment**

**Written assessment**

* 1. A combinational PDL with a programmable AND array and a programmable OR array is called a

PLD

PROM

PAL

PLA

* 1. In a read-only memory, information can be stored
     + - 1. At the time of fabrication
         2. By the user only once during its life time
         3. By the user a number of times
         4. In any of the above ways depending upon the type of memory
  2. Answer by True or False
     + - 1. Both the AND and OR arrays in a PAL are programmable
         2. CAD software can be used to create a detailed logic circuit that will be perform like an actual PLD
  3. What is the function of programmable logic devices?

A programmable logic device (PLD) is an electronic component used to build reconfigurable digital circuits. Unlike digital logic constructed using discrete logic gates with fixed functions, a PLD has an undefined function at the time of manufacture

* 1. Differentiate between functional and timing simulations of PLD?

**ANSWERS**

* 1. A combinational PDL with a programmable AND array and a programmable OR array is called a

PLD

PROM

PAL

**PLA**

* 1. In a read-only memory, information can be stored
     + - 1. At the time of fabrication
         2. **By the user only once during its life time**
         3. By the user a number of times
         4. In any of the above ways depending upon the type of memory
  2. Answer by True or False
     + - 1. Both the AND and OR arrays in a PAL are programmable/ **F**
         2. CAD software can be used to create a detailed logic circuit that will be perform like an actual PLD /**T**
  3. What is the function of programmable logic devices?

A programmable logic device (PLD) is an electronic component used to build reconfigurable digital circuits. Unlike digital logic constructed using discrete logic gates with fixed functions, a PLD has an undefined function at the time of manufacture

* 1. Differentiate between functional and timing simulations of PLD?
  2. **Functional Simulation of a PLD (Programmable Logic Device) is a crucial step in the design process that involves verifying the intended logic behaviour of a design before it's physically programmed onto the device.** It's a software-based simulation that doesn't consider actual device delays, but rather focuses on ensuring the logic functions as intended. The following are key steps for functional simulation

In timing simulation the simulator utilizes the timing model to account for delays during simulation. It verifies that signals meet setup and hold timing requirements for flip-flops and registers. It identifies potential timing violations that could lead to functional errors. Timing Simulation Waveform Viewer

### **References**

* 1948– Principles of modern digital design / by Parag K. Lala.
* Programmable Logic Devices (PLDs) in VLSI Design ( Dr. G. Senthil Kumar Associate Professor)
* Digital Fundamentals( E LEVENTH EDITION Thomas L. Floyd)
* Digital Electronics Principles, Devices and Applications( Anil K. Maini Defence Research and Development Organization (DRDO), India)
* R.Tokheim. (2014). Digital Electronics Principles and Applications. USA: 8th ed.
* Theraja, B. (n.d.). A textbook of Electrical Technology Vol IV-Electronic Devices and circuits.