UNIT – I

1.	What is meant by an Integrated circuit? Briefly explain about different types of integrated circuits.	8	Section-1	1

Ans.

- Digital circuits are constructed with integrated circuits. An integrated circuit (abbreviated IC) is a small silicon semiconductor crystal called a chip, containing the electronic components for the digital gates.
- **The various gates** are interconnected inside the chip to form the required circuit
- **The chip is mounted in a ceramic or plastic container,** and connections are welded by thin gold wires to external pins to form the integrated circuit.
- **The number of pins** may range from 14 in a small IC package to 100 or more in a larger package. Each IC has a numeric designation printed on the surface of the package for identification.

Scale of Integration or Classification of integrated circuits:

- **Small-scale integration (SSI):** devices contain several independent gates in a single package. The inputs and outputs of the gates are connected directly to the pins in the package. The number of gates is **usually less than 10** and is limited by the number of pins available in the IC.
- **Medium-scale integration (MSI):** devices have a complexity of **approximately 10 to 200 gates** in a single package. They usually perform specific elementary digital functions such as decoders, adders, and registers.
- **Large-scale integration (LSI):** devices contain **between 200 and a few thousand gates** in a single package. They include digital systems, such as processors, memory chips, and programmable modules.
- **Very-large-scale integration (VLSI):** devices contain **thousands of gates** within a single package. Examples are large memory arrays and complex microcomputer chips.
- **Super large scale integration (SLSI):** devices contain between **10,000 and 1,00,000 gates** within a single package and perform computational operations such as microprocessor chips, micro-controllers, basic PICs (Peripheral Interface Controller) and calculator.
- **Ultra large scale integration (ULSI):** devices contain **More than 1 million gates** and it is used in computers CPUs, GPUs, Video processors.

Logic Families:

The IC technology usually depends on the following factors:

1) Speed.

2) Power description.

3) Noise immunity.

4) Input/output interface compatibility.

5) Cost.

Digital Integrated Circuits:

- The advantages if ICs being used in digital systems are

1) Small size.

2) High reliability.

3) Low cost.

4) Low power consumption.

Categories of Digital Integrated Circuits:

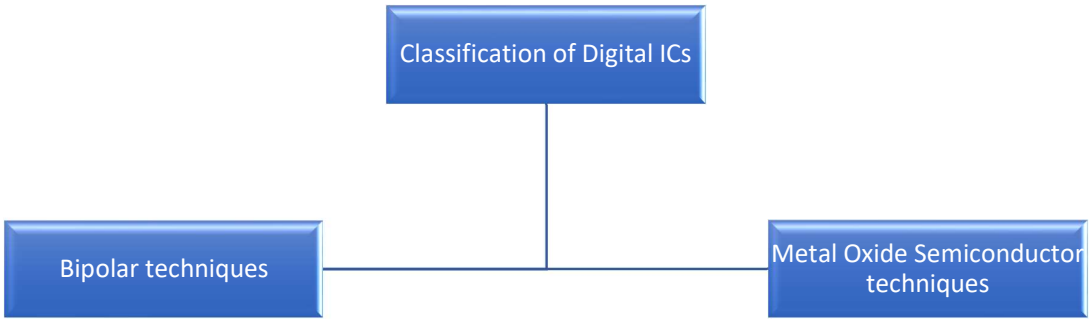
There are two broad categories of digital ICs. They are

1. Fixed function logic
2. Programmable logic

- **Fixed function logic**: The logic function of IC are set by the manufacturer and cannot be altered.
- **Programmable logic**: The logic function of IC can be altered.

The two basic techniques for manufacturing ICs are

1. Bipolar techniques
2. Metal Oxide Semiconductor (**MOS**) techniques



Classification of digital ICs families are

1. Bipolar techniques

- Resistor transistor logic (**RTL**)

- Diode transistor logic (**DTL**)
- Transistor transistor logic (**TTL**)
- Emitter coupled logic (**ECL**)

2. Metal Oxide Semiconductor techniques

- P channel MOSFET (**PMOS**)
- N channel MOSFET (**NMOS**)
- Complementary MOSFET (**CMOS**)

2.	Briefly explain bi-directional shift register with parallel load.	8	Section-1	1

Ans.

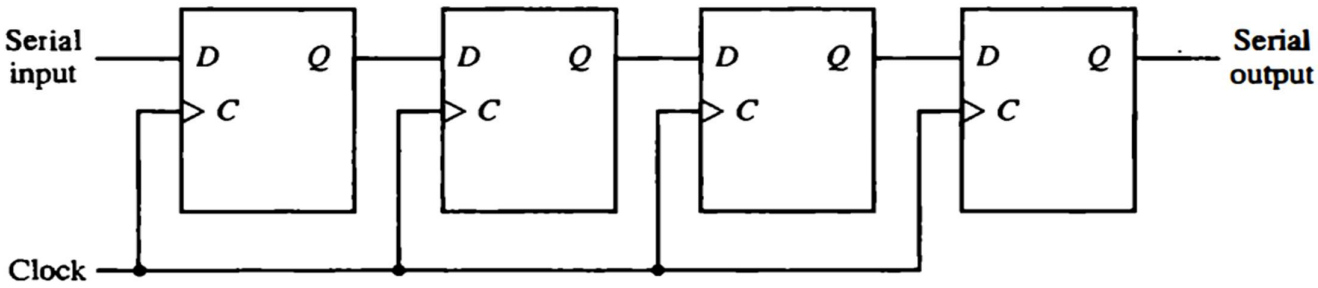
Register: A register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing any binary information of n bits.

Shift Register: A register capable of shifting its binary information in one or both directions is called a shift register.

Bidirectional Shift Register with Parallel Load:

A register capable of shifting in one direction only is called a unldirectional shift register. A register that can shift in both directions is called a bidirectional shift register. Some shift registers provide the necessary input and output terminals for parallel transfer. The most general shift register has all the capabilities listed below. Others may have some of these capabilities, with at least one shift operation.

1. An input for clock pulses to synchronize all operations.
2. A shift-right operation and a serial input line associated with the shift right.
3. A shift-left operation and a serial input line associated with the shift-left.
4. A parallel load operation and n input lines associated with the parallel transfer.
5. n parallel output lines.
6. A control state that leaves the information in the register unchanged even though clock pulses are applied continuously.



4 – bit Shift Register

In the described 4-bit bidirectional shift register with parallel load, each stage consists of a D flip-flop and a 4 x 1 multiplexer. The behavior of the register is controlled by two selection inputs, denoted as S_1 and S_0.

Here's a breakdown of the operation based on the selection inputs:

1. **Mode Control S_1S_0 = 00 (No Operation - Hold State):**
   - In this mode, both selection inputs are set to 0.
   - Each multiplexer selects its first data input (0).
   - This configuration forms a loop, where the output of each flip-flop is fed back into its own D input.
   - Consequently, the binary value held in each flip-flop remains unchanged during the next clock transition, maintaining the current state of the register.

2. **Mode Control S_1S_0 = 01 (Shift Right Operation):**
   - When S_1 is set to 0 and S_0 is set to 1, indicating mode 01.
   - Each multiplexer selects its second data input (1).
   - This configuration allows the serial input data to be transferred into the first flip-flop (A0).
   - Meanwhile, the content of each flip-flop except the last one (Ai) is transferred into the next flip-flop (Ai+1), effectively shifting the data to the right.
3. **Mode Control S_1S_0 = 10 (Shift Left Operation):**
   - With S_1 set to 1 and S_0 set to 0, representing mode 10.
   - Each multiplexer selects its third data input.
   - This setup enables the serial input data to enter the last flip-flop (A3).
   - Simultaneously, the content of each flip-flop except the first one (Ai) is transferred into the previous flip-flop(Ai-1), causing the data to shift to the left.
4. **Mode Control S_1S_0 = 11 (Parallel Load Operation):**
   - When both S_1 and S_0 are set to 1, indicating mode 11.
   - Each multiplexer selects its fourth data input, corresponding to the parallel input data.
   - In this mode, the binary information from each input (I0 through I3) is directly transferred into the corresponding flip-flop, resulting in a parallel load operation.

Mode control		
S_1	S_0	Register operation
0	0	No change
0	1	Shift right (down)
1	0	Shift left (up)
1	1	Parallel load

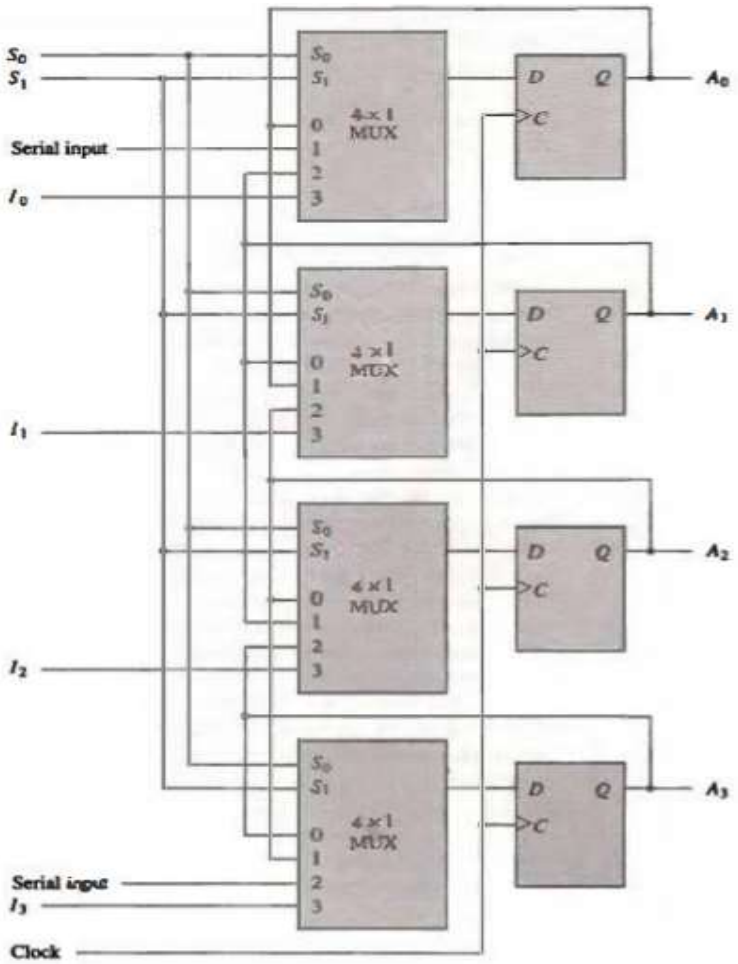


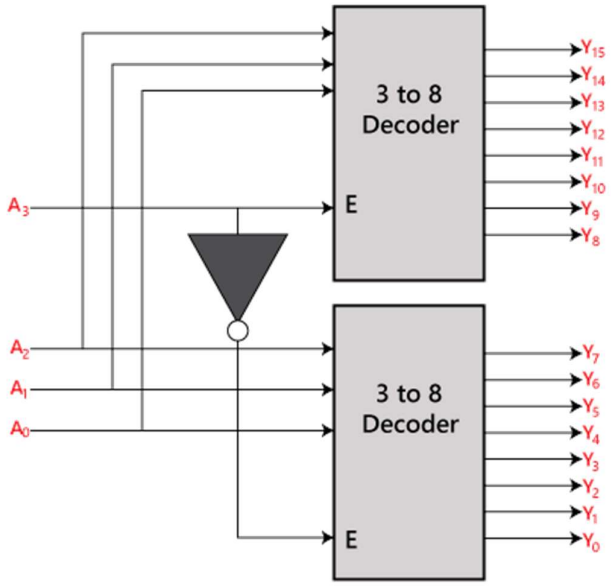
Figure 2-9 Bidirectional shift register with parallel load.

Shift registers enable efficient serial data transmission between distant digital systems. When transmitting an n-bit quantity over long distances, parallel transmission may be costly. Instead, data is loaded into a shift register in parallel, then transmitted serially. At the receiver, data is accepted serially into another shift register and outputted in parallel. This parallel-to-serial and serial-to-parallel conversion facilitates cost-effective data transmission over single lines.

3.	Construct 4 x16 line decoder using 3 x 8 line decoder.	8	Section-1	1

Ans.

Block Diagram:



Truth Table:

INPUTS				OUTPUTS														
A_3	A_2	A_1	A_0	Y_{15}	Y_{14}	Y_{13}	Y_{12}	Y_{11}	Y_{10}	Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Logical Expressions:

$Y_0=A_0'.A_1'.A_2'.A_3'$

$Y_1=A_0.A_1'.A_2'.A_3'$

$Y_2=A_0'.A_1.A_2'.A_3'$

$Y_3=A_0.A_1.A_2'.A_3'$

$Y_4=A_0'.A_1'.A_2.A_3'$

$Y_5=A_0.A_1'.A_2.A_3'$

$Y_6=A_0'.A_1.A_2.A_3'$

$Y_7=A_0.A_1.A_2.A_3'$

$Y_8=A_0'.A_1'.A_2'.A_3$

$Y_9=A_0.A_1'.A_2'.A_3$

$Y_{10}=A_0'.A_1.A_2'.A_3$

$Y_{11}=A_0.A_1.A_2'.A_3$

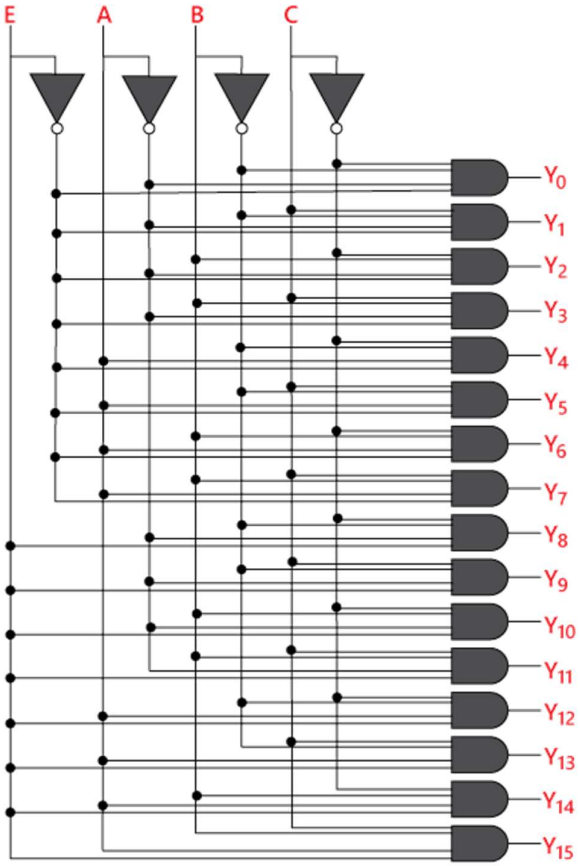
$Y_{12}=A_0'.A_1'.A_2.A_3$

$Y_{13}=A_0.A_1'.A_2.A_3$

$Y_{14}=A_0'.A_1.A_2.A_3$

$Y_{15}=A_0.A_1.A_2.A_3$

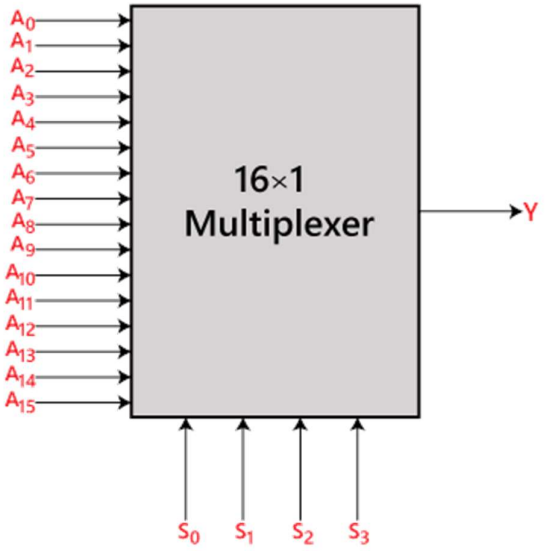
Logical Circuit:



4.	Construct 16x1 multiplexer with two 8x1 and one 2x1 multiplexers.	8	Section-1	1

Ans.

Block Diagram:



Truth Table:

INPUTS				Output
S_0	S_1	S_2	S_3	Y
0	0	0	0	A_0
0	0	0	1	A_1
0	0	1	0	A_2
0	0	1	1	A_3
0	1	0	0	A_4
0	1	0	1	A_5
0	1	1	0	A_6
0	1	1	1	A_7
1	0	0	0	A_8
1	0	0	1	A_9
1	0	1	0	A_{10}
1	0	1	1	A_{11}
1	1	0	0	A_{12}
1	1	0	1	A_{13}
1	1	1	0	A_{14}
1	1	1	1	A_{15}

Logical Expressions:

$Y=A_0.S_0'.S_1'.S_2'.S_3'+$

$A_1.S_0'.S_1'.S_2'.S_3+$

$A_2.S_0'.S_1'.S_2.S_3'+$

$A_3.S_0'.S_1'.S_2.S_3+$

$A_4.S_0'.S_1.S_2'.S_3'+$

$A_5.S_0'.S_1.S_2'.S_3+$

$A_6.S_0'.S_1.S_2.S_3'+$

$A_7.S_0'.S_1.S_2.S_3+$

$A_8.S_0.S_1'.S_2'.S_3'+$

$A_9.S_0.S_1'.S_2'.S_3+$

$A_{10}.S_0.S_1'.S_2.S_3'+$

$A_{11}.S_0.S_1'.S_2.S_3+$

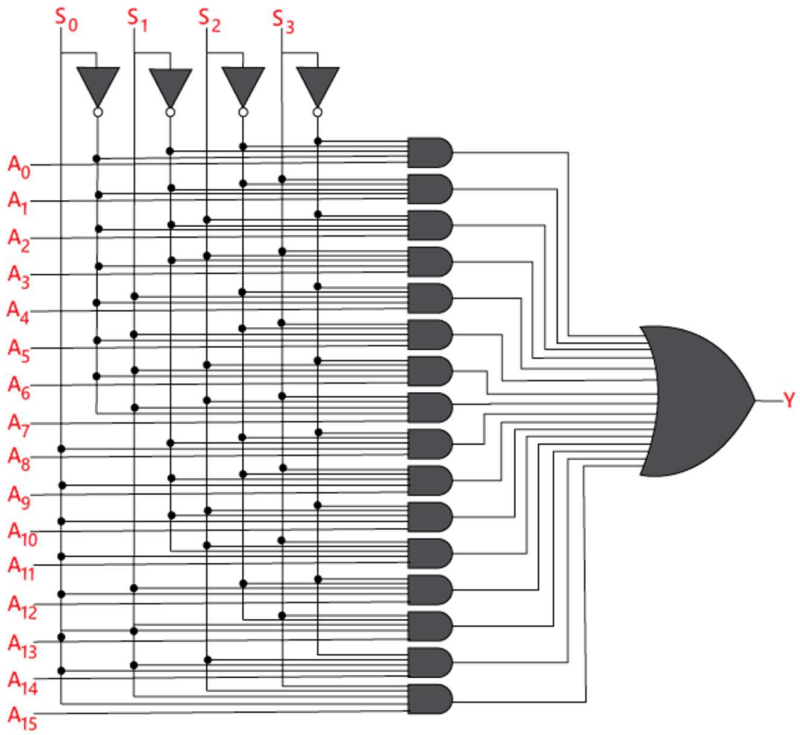
$A_{12}.S_0.S_1.S_2'.S_3'+$

$A_{13}.S_0.S_1.S_2'.S_3+$

$A_{14}.S_0.S_1.S_2.S_3'+$

$A_{15}.S_0.S_1.S_2.S_3$

Logical Circuit:



5.	a) Design a 4-bit binary-to-Gray code converter. (4M) b) Convert binary code to gray code. (2*2 = 4M) (i) $(1111)_B$ (ii) $(0101)_B$	8	Section-1	1

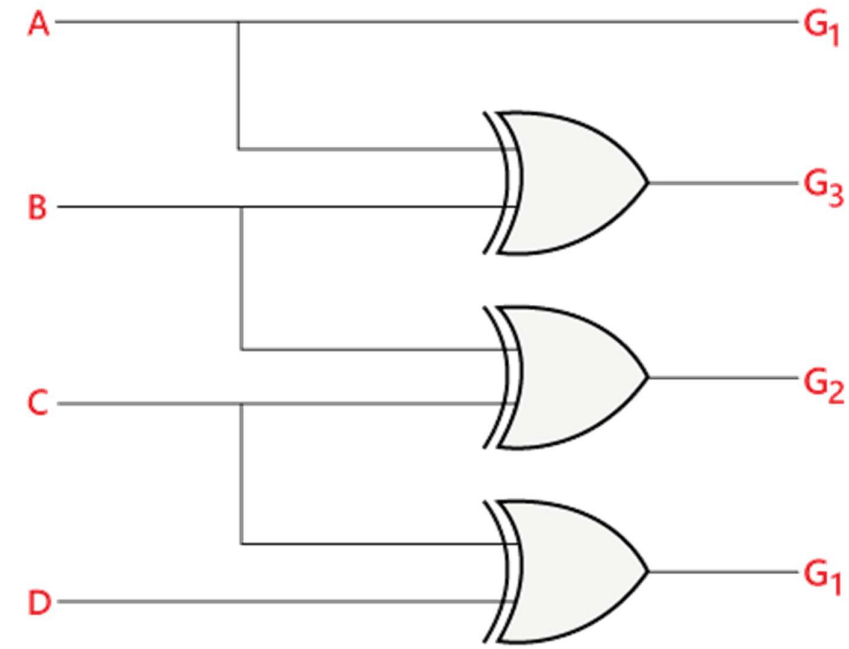
Ans.

a) Binary to Gray code converter.

- The Binary to Gray code converter is a logical circuit that is used to convert the binary code into its equivalent Gray code. By putting the MSB of 1 below the axis and the MSB of 1 above the axis and reflecting the (n-1) bit code about an axis after 2n-1 rows, we can obtain the n-bit gray code.
- The 4-bit binary to gray code conversion table is as follows:

Decimal Number	4-bit Binary Code	4-bit Gray Code
	ABCD	**$G_1G_2G_3G_4$**
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

- In 4-bit gray code, the 3-bit code is reflected against the axis drawn after the 2^{4-1}-1^{th} =8^{th} row.



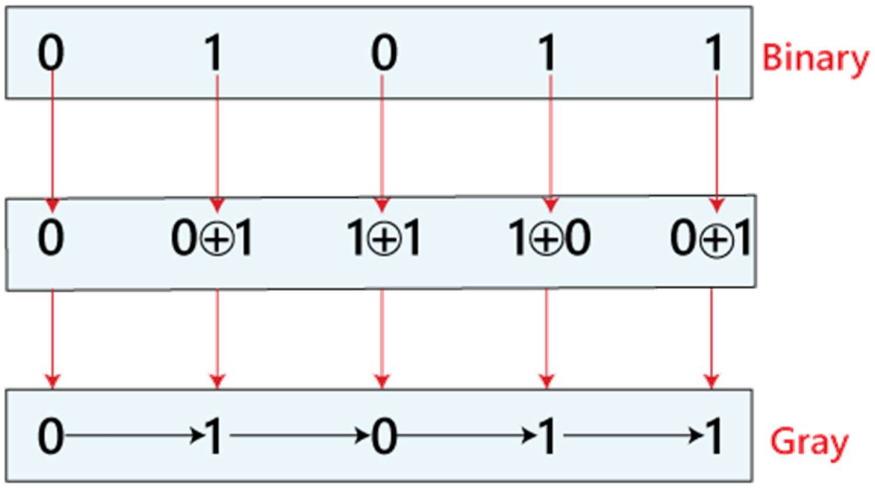
Logic Circuit for Binary to Gray Code Converter

- In the Gray code, the MSB will always be the same as the 1'st bit of the given binary number.
- In order to perform the 2nd bit of the gray code, we perform the exclusive-or (XOR) of the 1'st and 2nd bit of the binary number. It means that if both the bits are different, the result will be one else the result will be 0.
- In order to get the 3rd bit of the gray code, we need to perform the exclusive-or (XOR) of the 2nd and 3rd bit of the binary number. The process remains the same for the 4th bit of the Gray code. Let's take an example to understand these steps.

Example:

Suppose we have a binary number 01101, which we want to convert into Gray code. There are the following steps which need to perform this conversion:

- As we know that the 1st bit of the Gray code is the same as the MSB of the binary number. In our example, the MSB is 0, so the MSB or 1st bit of the gray code is 0.
- Next, we perform the XOR operation of the 1'st and the second binary number. The 1st bit is 0, and the 2nd bit is 1. Both the bits are different, so the 2nd bit of the Gray code is 1.
- Now, we perform the XOR of the 2nd bit and 3rd bit of the binary number. The 2nd bit is 1, and the 3rd bit is also 1. These bits are the same, so the 3rd bit of the Gray code is 0.
- Again perform the XOR operation of the 3rd and 4th bit of binary number. The 3rd bit is 1, and the 4th bit is 0. As these are different, the 4th bit of the Gray code is 1.
- Lastly, perform the XOR of the 4th bit and 5th bit of the binary number. The 4th bit is 0, and the 5th bit is 1. Both the bits are different, so that the 5th bit of the Gray code is 1.
- The gray code of the binary number 01101 is 01011.



b)

(i) Binary Code: $(1111)_2$

1. **G3** = B3 = 1
2. **G2** = B3 ⊕ B2 = 1 ⊕ 1 = 0
3. **G1** = B2 ⊕ B1 = 1 ⊕ 1 = 0
4. **G0** = B1 ⊕ B0 = 1 ⊕ 1 = 0

So, the Gray code for $(1111)_2$ is $(1000)_G$.

(ii) Binary Code: $(0101)_2$

1. **G3** = B3 = 0
2. **G2** = B3 ⊕ B2 = 0 ⊕ 1 = 1
3. **G1** = B2 ⊕ B1 = 1 ⊕ 0 = 1
4. **G0** = B1 ⊕ B0 = 0 ⊕ 1 = 1

So, the Gray code for $(0101)_2$ is $(0111)_G$.

6.	a) Convert the $(-5.96)_{10}$ to binary number using 8-bit binary representation with 4-digit integers and 4-digit fractional parts. (4M) b) Compute 0.95 + (-0.555) using fixed point numbers. (4M)	8	Section-1	1

Ans.

a)

Step 1: Convert the Integer Part to Binary

The integer part is 5. To convert 5 to binary:

$5_{10} = 101_2$

In a 4-bit representation:

$5 = 0101$

Step 2: Convert the Fractional Part to Binary

The fractional part is 0.96. To convert 0.96 to binary, multiply by 2 and take the integer part iteratively:

1. $0.96 \times 2 = 1.92$ (integer part: 1)
2. $0.92 \times 2 = 1.84$ (integer part: 1)
3. $0.84 \times 2 = 1.68$ (integer part: 1)
4. $0.68 \times 2 = 1.36$ (integer part: 1)

Taking the first 4 fractional bits, $0.96_{10} \approx 0.1111_2$.

Step 3: Combine the Integer and Fractional Binary Parts

Combine the integer part and the fractional part:

$5.96_{10} \approx 0101.1111_2$

Step 4: Convert to Two's Complement Representation

Since the number is negative (-5.96), we need to convert the combined binary number to its two's complement representation:

1. **Combine the parts:** 0101.1111_2
2. **Invert the bits:** 1010.0000_2
3. **Add 1 to the inverted bits:**
   $1010.0000_2 + 0.0001_2 = 1010.0001_2$

Therefore, the 8-bit two's complement representation of -5.96 with 4 integer and 4 fractional parts is:

1010.0001_2

b)

1. **Convert 0.95 to fixed-point binary representation:**
   - Separate into integer and fractional parts: $0.95 = 0.95$
   - Convert the integer part (0): $0_{10} = 0000_2$
   - Convert the fractional part (0.95):

$$0.95 \times 2 = 1.90 \quad (\text{integer part: } 1)$$

$$0.90 \times 2 = 1.80 \quad (\text{integer part: } 1)$$

$$0.80 \times 2 = 1.60 \quad (\text{integer part: } 1)$$

$$0.60 \times 2 = 1.20 \quad (\text{integer part: } 1)$$

Taking the first 4 bits of the fractional part: 0.1111_2

So, $0.95 \approx 0000.1111_2$

2. **Convert -0.555 to fixed-point binary representation:**

   - Separate into integer and fractional parts: $-0.555 = -0.555$
   - Convert the integer part (0): $0_{10} = 0000_2$
   - Convert the fractional part (0.555):

$$0.555 \times 2 = 1.11 \quad \text{(integer part: 1)}$$

$$0.11 \times 2 = 0.22 \quad \text{(integer part: 0)}$$

$$0.22 \times 2 = 0.44 \quad \text{(integer part: 0)}$$

$$0.44 \times 2 = 0.88 \quad \text{(integer part: 0)}$$

   Taking the first 4 bits of the fractional part: 0.1000_2

   So, $0.555 \approx 0000.1000_2$

   - To represent -0.555, find the two's complement of 0000.1000_2:
     - Invert the bits: 1111.0111_2
     - Add 1: $1111.0111_2 + 0.0001_2 = 1111.1000_2$

   So, $-0.555 \approx 1111.1000_2$

3. **Add the two binary numbers using fixed-point arithmetic:**

$$0000.1111_2 \quad \text{(for } 0.95\text{)}$$

$$+1111.1000_2 \quad \text{(for } -0.555\text{)}$$

   Performing the binary addition:

$$\begin{array}{r} 0000.1111 \\ +1111.1000 \\ \hline 1111.0111 \end{array}$$

   Since we're using 8-bit arithmetic, the overflow is ignored, so the result is:

$$1111.0111_2$$

Converting back to decimal:

- This is in two's complement, so invert the bits: 0000.1000_2
- Add 1: $0000.1000_2 + 0.0001_2 = 0000.1001_2$
- The result is 0000.1001_2, which is 0.1001_2.

Converting 0.1001_2 to decimal:

$$0.1001_2 = 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} = 0.5 + 0.0625 = 0.562$$

Therefore, the result of $0.95 + (-0.555)$ using fixed-point binary representation is approximately 0.395.

7.	Explain floating point representation of $(7924.622)_{10}$ in single precision and double precision format.	8	Section-1	1

Ans.

IEEE 754 Floating-Point Representation

Single Precision (32-bit):

- **Sign bit:** 1 bit
- **Exponent:** 8 bits
- **Mantissa (fraction):** 23 bits

Double Precision (64-bit):

- **Sign bit:** 1 bit
- **Exponent:** 11 bits
- **Mantissa (fraction):** 52 bits

Steps to Convert 7924.622_{10} to Floating-Point Representation

1. **Convert to Binary**
2. **Normalize the Binary Number**
3. **Determine the Sign Bit**
4. **Calculate the Exponent and Bias**
5. **Determine the Mantissa (Fraction)**

Step 1: Convert to Binary

First, convert the integer part and fractional part of 7924.622_{10} to binary.

Integer Part (7924):

$7924_{10} = 1111011110100_2$

Fractional Part (0.622):

Multiply by 2 and take the integer part iteratively:

1. $0.622 \times 2 = 1.244$ (integer part: 1)
2. $0.244 \times 2 = 0.488$ (integer part: 0)
3. $0.488 \times 2 = 0.976$ (integer part: 0)
4. $0.976 \times 2 = 1.952$ (integer part: 1)
5. $0.952 \times 2 = 1.904$ (integer part: 1)
6. $0.904 \times 2 = 1.808$ (integer part: 1)
7. $0.808 \times 2 = 1.616$ (integer part: 1)
8. $0.616 \times 2 = 1.232$ (integer part: 1)
9. $0.232 \times 2 = 0.464$ (integer part: 0)
10. $0.464 \times 2 = 0.928$ (integer part: 0)

Taking the first few bits for practical purposes, $0.622_{10} \approx 0.101000111101_2$.

So, the binary representation of 7924.622_{10} is approximately:
$7924.622_{10} \approx 1111011110100.101000111101_2$

Step 2: Normalize the Binary Number

Normalize $1111011110100.101000111101_2$:
$1.11101111010010100011110 1_2 \times 2^{12}$

Step 3: Determine the Sign Bit

Since 7924.622 is positive, the sign bit is 0.

Step 4: Calculate the Exponent and Bias

Single Precision:

- Bias: 127
- Exponent: $12 + 127 = 139_{10} = 10001011_2$

Double Precision:

- Bias: 1023
- Exponent: $12 + 1023 = 1035_{10} = 10000001011_2$

Step 5: Determine the Mantissa (Fraction)

Take the 23 (for single precision) or 52 (for double precision) bits after the binary point from the normalized binary number.

Normalized Binary: $1.11101111010010100011110 1...$

Single Precision:

Mantissa: 11101111010010100011110

Double Precision:

Mantissa: $1110111101001010001111010001111010001111010001111010001111 01...$

Putting it All Together

Single Precision:

- Sign bit: 0
- Exponent: 10001011
- Mantissa: 11101111010010100011110

Single precision representation:
0|10001011|11101111010010100011110

Double Precision:

- Sign bit: 0
- Exponent: 10000001011
- Mantissa: 11101111010010100011110100011110100011110100011110100011110100011101...

Double precision representation:
0|10000001011|11101111010010100011110100011110100011110100011110100011110100011101.

Summary:

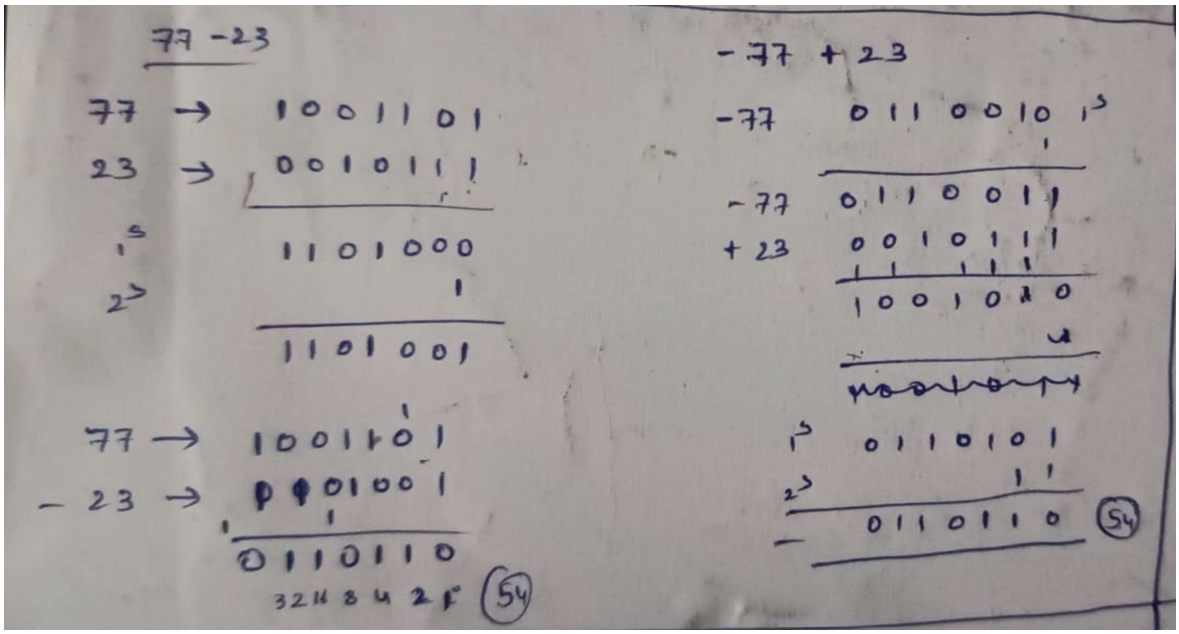
Single Precision (32-bit) Representation of 7924.622_{10}:
01000101111011110100100100011110

Double Precision (64-bit) Representation of 7924.622_{10}:
0100000010111110111010010100011110100011110100011110100011110100011101...

These representations accurately follow the IEEE 754 standard for floating-point numbers.

8.	Perform the binary arithmetic operation (+77) + (-23) and (-77) - (-23) in using signed 2's complement representation for negative numbers.	8	Section-1	1

Ans.



9.	Convert the following to binary to octal (4*2 = 8M) (a) $(2A7)_{16}$ (b) $(1AB)_{16}$ (c) $(10110111)_2$ (d) $(11011010)_2$	8	Section-1	1

Ans.

(a) Convert $(2A7)_{16}$ to octal

Step 1: Convert from hexadecimal to binary

Each hexadecimal digit corresponds to a 4-bit binary number:

$$2 = 0010$$
$$A = 1010$$
$$7 = 0111$$

Thus, $(2A7)_{16}$ converts to binary as:

$$2A7_{16} = 0010\ 1010\ 0111_2$$

Step 2: Convert from binary to octal

Group the binary digits into sets of 3, starting from the right (add leading zeros if necessary):

$$001\ 010\ 100\ 111$$

Now, convert each group to its octal equivalent:

$$001 = 1$$
$$010 = 2$$
$$100 = 4$$
$$111 = 7$$

Thus, $(2A7)_{16}$ converts to octal as:

$$(2A7)_{16} = 1247_8$$

(b) Convert $(1AB)_{16}$ to octal

Step 1: Convert from hexadecimal to binary

Each hexadecimal digit corresponds to a 4-bit binary number:

$$1 = 0001$$
$$A = 1010$$
$$B = 1011$$

Thus, $(1AB)_{16}$ converts to binary as:

$$1AB_{16} = 0001\ 1010\ 1011_2$$

Step 2: Convert from binary to octal

Group the binary digits into sets of 3, starting from the right (add leading zeros if necessary):

$$000\ 011\ 010\ 101\ 011$$

Now, convert each group to its octal equivalent:

$$000 = 0$$
$$001 = 1$$
$$101 = 5$$
$$010 = 2$$
$$011 = 3$$

Thus, $(1AB)_{16}$ converts to octal as:

$$(1AB)_{16} = 01553_8$$

(c) Convert $(10110111)_2$ to octal

Step 1: Group the binary digits into sets of 3, starting from the right (add leading zeros if necessary):

$$10110111_2 = 010\ 110\ 111$$

Step 2: Convert each group to its octal equivalent:

$$010 = 2$$
$$110 = 6$$
$$111 = 7$$

Thus, $(10110111)_2$ converts to octal as:

$$(10110111)_2 = 267_8$$

(d) Convert $(11011010)_2$ to octal

Step 1: Group the binary digits into sets of 3, starting from the right (add leading zeros if necessary):

$$11011010_2 = 011\ 011\ 010$$

Step 2: Convert each group to its octal equivalent:

$$011 = 3$$
$$011 = 3$$
$$010 = 2$$

Thus, $(11011010)_2$ converts to octal as:

$$(11011010)_2 = 332_8$$

10.	Briefly explain about the memory unit.	8	Section-1	1

Ans.

The memory unit in refers to the hardware component responsible for storing data and instructions that the CPU (central processing unit) needs to access quickly. It consists of various types of memory, such as RAM (random access memory) for temporary storage and cache memory for faster access to frequently used data. Memory units are crucial for the efficient operation of a computer system, providing storage for both program instructions and data being processed.

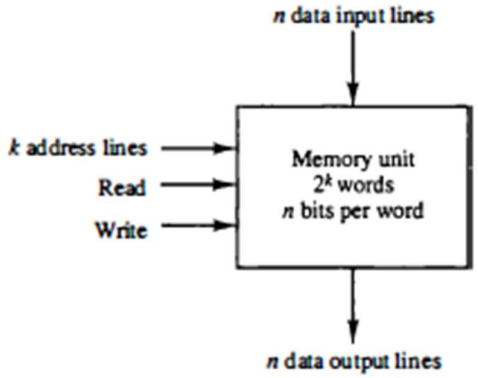
Word: The memory stores binary information in groups of bits called words.

Byte: A group of 8-bits is called a byte.

RAM: In random-access memory (RAM) the memory cells can be accessed for information transfer from any desired random location. It is a temporary memory bank where your computer stores data it needs to retrieve quickly.

Write and read operations:

- The two operations that a random-access memory can perform are the write and read operations.



Types of ROMs

ROM: Read-only memory (ROM) is a memory unit that performs the read operation only.

PROM: PROM is a type of read-only memory that you can program once.

EEPROM: (Electrically Erasable Programmable Read-Only Memory) is a user modifiable ROM. It can be erased and reprogrammed (written to) repeatedly by applying an electrical voltage.