Register No. | | | | | | | | | | | | |

SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

AN AUTONOMOUS INSTITUTION

Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

INTERNAL ASSESSMENT EXAMINATION – I

III Semester

B.E-COMPUTER SCIENCE AND ENGINEERING

(COMMON TO B.E-COMPUTER SCIENCE AND TECHNOLOGY)

23ECB206- DIGITAL ELECTRONICS AND MICROPROCESSOR

Answer Key

PART A - (5 X 2 = 10 marks)

Q.No	Question	M	CO	BL
1.	**What is a Karnaugh Map and how is it used?** A Karnaugh Map (K-map) is a graphical tool used to simplify Boolean expressions by minimizing the number of logic gates required in a digital circuit. It is a grid-based method that represents truth tables, where adjacent cells differ by only one variable. It helps to eliminate redundant terms and reduce complex expressions into simpler forms.	2	CO-1	L -1
2.	**How does an encoder work?** An encoder is a combinational logic circuit that converts an active input signal into a coded output format. For instance, in a binary encoder, multiple input lines are represented in a binary code at the output. Encoders are used in applications where fewer output bits are required to represent multiple input lines.	2	CO-1	L -2
3.	**State the purpose of a magnitude comparator.** A magnitude comparator is a digital circuit used to compare two binary numbers and determine their relative magnitude. It produces three outputs to indicate whether the first number is greater than, less than, or equal to the second number.	2	CO-1	L -2
4.	**Define the term "race condition" in sequential circuits.** A race condition occurs in sequential circuits when the order or timing of signal transitions affects the final output. It happens when two or more signals change simultaneously and the final output depends on which signal changes first, leading to unpredictable behavior in the circuit.	2	CO-2	L -2
5.	**Name the different types of flip-flops.** 1. SR flip-flop (Set-Reset) 2. D flip-flop (Data or Delay) 3. JK flip-flop 4. T flip-flop (Toggle)	2	CO-2	L -1

PART B - (2 X 13 = 26 marks)

6.	**(a)**	**Design a full adder and full subtractor using basic logic gates and explain its working.**	13	CO-1	L-3

Design of Full Adder using Basic Logic Gates

A **full adder** is a combinational logic circuit that adds three input bits (two significant bits and one carry bit) and produces a sum and a carry output.

1. Full Adder Truth Table

Input A	**Input B**	**Carry In (Cin)**	**Sum (S)**	**Carry Out (Cout)**
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

2. Logic Expressions

- **Sum (S):** S=A⨁B⨁CinS = A \oplus B \oplus \text{Cin}S=A⨁B⨁Cin
- **Carry Out (Cout):** Cout=(A·B)+(B·Cin)+(A·Cin)\text{Cout} = (A \cdot B) + (B \cdot \text{Cin}) + (A \cdot \text{Cin})Cout=(A·B)+(B·Cin)+(A·Cin)

3. Logic Gate Implementation

1. **XOR Gate** for Sum: Connect the three inputs AAA, BBB, and Cin\text{Cin}Cin to form the sum using the XOR gate:

   S=A⨁B⨁CinS = A \oplus B \oplus \text{Cin}S=A⨁B⨁Cin

2. **AND Gates** and **OR Gate** for Carry Out:
   - Use three **AND gates** to generate intermediate products: A·BA \cdot BA·B, B·CinB \cdot \text{Cin}B·Cin, and A·CinA \cdot \text{Cin}A·Cin.
   - Combine the outputs of the three AND gates using an **OR gate** to form the final carry out:

   Cout=(A·B)+(B·Cin)+(A·Cin)\text{Cout} = (A \cdot B) + (B \cdot \text{Cin}) + (A \cdot \text{Cin})Cout=(A·B)+(B·Cin)+(A·Cin)

Full Adder Circuit Diagram

- The circuit uses **two XOR gates** for the sum and **three AND gates** along with an **OR gate** for the carry.

Design of Full Subtractor using Basic Logic Gates

A **full subtractor** is a combinational circuit that subtracts three input bits (two significant bits and a borrow bit) and produces a difference and a borrow output.

1. Full Subtractor Truth Table

Input A	**Input B**	**Borrow In (Bin)**	**Difference (D)**	**Borrow Out (Bout)**
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

2. Logic Expressions

- **Difference (D):** D=A⨁B⨁BinD = A \oplus B \oplus \text{Bin}D=A⨁B⨁Bin
- **Borrow Out (Bout):** Bout=(A¯·B)+(B·Bin)+(A¯·Bin)\text{Bout} = (\overline{A} \cdot B) + (B \cdot \text{Bin}) + (\overline{A} \cdot \text{Bin})Bout=(A·B)+(B·Bin)+(A·Bin)

3. Logic Gate Implementation

1. **XOR Gate** for Difference: The XOR gate can be used to generate the difference of the three inputs AAA, BBB, and Bin\text{Bin}Bin:

   D=A⨁B⨁BinD = A \oplus B \oplus \text{Bin}D=A⨁B⨁Bin

2. **AND, OR, and NOT Gates** for Borrow Out:
   - Use **NOT gates** to invert AAA wherever needed.
   - Combine A¯·B\overline{A} \cdot BA·B, B·BinB \cdot \text{Bin}B·Bin, and A¯·Bin\overline{A} \cdot \text{Bin}A·Bin using **AND gates**.
   - Use an **OR gate** to combine these terms:

Bout=(A‾·B)+(B·Bin)+(A‾·Bin)\text{Bout} = (\overline{A} \cdot B) + (B \cdot \text{Bin}) + (\overline{A} \cdot \text{Bin})Bout=(A·B)+(B·Bin)+(A·Bin)

Full Subtractor Circuit Diagram

- The circuit uses **XOR gates** for the difference calculation and a combination of **NOT, AND, and OR gates** for the borrow output.

OR

(b) Simplify the following Boolean expression using Karnaugh Map: F(A,B,C,D)=Σm(0,1,4,8,9,10)+d(2,11). 13 CO-1 L-3

To simplify the Boolean expression using a Karnaugh Map, follow these steps:

Given:

- **Function:** F(A,B,C,D)=Σm(0,1,4,8,9,10)F(A,B,C,D) = \Sigma m(0, 1, 4, 8, 9, 10)F(A,B,C,D)=Σm(0,1,4,8,9,10)
- **Don't care terms:** d(2,11)d(2, 11)d(2,11)

1. Karnaugh Map Layout:

Construct a 4-variable (16-cell) Karnaugh Map with variables A,B,C,DA, B, C, DA,B,C,D.

AB\CD	**00**	**01**	**11**	**10**
00	0	1	d(2)	0
01	1	0	0	0
11	0	d(11)	0	1
10	1	1	0	0

- Σm(0,1,4,8,9,10)\Sigma m(0, 1, 4, 8, 9, 10)Σm(0,1,4,8,9,10) represent minterms where the function is 1.
- d(2,11)d(2, 11)d(2,11) are don't care conditions.

2. Grouping of 1s and don't care terms:

Identify the largest possible groups of 1s and don't cares (which can be grouped as powers of 2, like 1, 2, 4, or 8 cells):

- **Group 1:** Cells (0, 1, 8, 9) → This is a 2x2 block (can include both minterms and don't cares).
- **Group 2:** Cells (8, 9, 10) → This is a horizontal group of 3 cells, covering another portion.

3. Simplified Boolean Expression:

- **Group 1 (0, 1, 8, 9):** This group covers the minterms when A′B′A'B'A′B′. So, the simplified expression is A′B′A'B'A′B′.

- **Group 2 (8, 9, 10):** This group covers when AAA and C′D′C'D'C′D′, resulting in AC′D′A C'D'AC′D′.

Final Simplified Expression:

F(A,B,C,D)=A′B′+AC′D′F(A,B,C,D) = A'B' + AC'D'F(A,B,C,D)=A′B′+AC′D′

This is the simplified Boolean expression using the Karnaugh Map.

Explain the working of a 1:16 Demultiplexer with the help of a block diagram and truth table.

7. (a) 13 CO-1 L-2

1:16 Demultiplexer

A **1:16 demultiplexer** takes a single input signal and routes it to one of the 16 outputs based on the value of a 4-bit select input. It essentially works as a decoder that selects one of 16 outputs to pass through the input data. The remaining 15 outputs will be set to logic LOW.

Block Diagram

Below is a conceptual block diagram of a **1:16 Demultiplexer**:

```
lua
Copy code
                  +-----------+
                  |   1:16     |
        Input ---|   DEMUX    |---- Y0
                  |             |---- Y1
     S3           |             |---- Y2
     S2  ----+----|             |---- Y3
     S1  ----|----|             |---- Y4
     S0  ----|----|             |---- Y5
                  |             |---- Y6
                  |             |---- Y7
                  |             |---- Y8
                  |             |---- Y9
                  |             |---- Y10
                  |             |---- Y11
                  |             |---- Y12
                  |             |---- Y13
                  |             |---- Y14
                  |             |---- Y15
                  +------------+
```

- **Input**: The single data input.
- **S3, S2, S1, S0**: The 4 selection lines (select inputs) that determine which output is activated.
- **Y0 to Y15**: The 16 possible output lines.

Truth Table

The select lines S3,S2,S1,S0S_3, S_2, S_1, S_0S3,S2,S1,S0 determine which output is active. The truth table for the **1:16 demultiplexer** is shown below:

S3	S2	S1	S0	Active Output (Yn)
0	0	0	0	Y0
0	0	0	1	Y1
0	0	1	0	Y2
0	0	1	1	Y3
0	1	0	0	Y4
0	1	0	1	Y5
0	1	1	0	Y6
0	1	1	1	Y7
1	0	0	0	Y8
1	0	0	1	Y9
1	0	1	0	Y10
1	0	1	1	Y11
1	1	0	0	Y12
1	1	0	1	Y13
1	1	1	0	Y14
1	1	1	1	Y15

Working

1. **Input**: The demultiplexer has a single input that can carry either a HIGH (1) or LOW (0) logic level.
2. **Selection Lines**: The four selection lines (S3, S2, S1, S0) control which output will receive the input signal. These lines are used to create a binary number that ranges from 0 to 15. Each combination of these selection lines corresponds to a unique output.
3. **Output**:
   - The input signal will be routed to one of the outputs (Y0 to Y15), depending on the value of the select lines.
   - All other outputs will remain LOW (0), while the selected output will match the value of the input (either HIGH or LOW).

Example

- **Input = 1**
- **Select Lines: S3 = 0, S2 = 1, S1 = 0, S0 = 1 (Selects Y5)**

In this case, the input (which is 1) will be routed to **Y5**, and all other outputs (Y0 to Y4, Y6 to Y15) will remain LOW (0).

Applications

- Used in communication systems where a single line is switched to multiple receivers.
- Can be used for data routing in CPUs, memory devices, and other circuits.

OR

(b) Explain the operation of a D flip-flop using timing diagrams. 13 CO-2 L-2

A **D flip-flop** (Data or Delay flip-flop) is a type of digital memory circuit used to store one bit of data. It operates based on a clock signal and outputs the value of the input (D) at the moment of a clock edge (typically the rising edge). Let’s break down its operation using a timing diagram.

Components of the D flip-flop:

1. **D (Data Input)**: The value that will be sampled and stored.
2. **CLK (Clock Signal)**: The flip-flop captures the value of the data input at a specific clock edge.
3. **Q (Output)**: The stored value after the clock edge.
4. **Q' (Complement Output)**: The inverse of the stored value (optional).

Basic Operation:

- The D flip-flop captures the value of the data input (**D**) only at the rising edge of the clock (if it’s a rising-edge-triggered flip-flop). At that moment, the output **Q** changes to the value of **D**.
- Between clock edges, **Q** remains stable (constant).

Timing Diagram:

Let’s look at the sequence of signals:

```
markdown
Copy code
Clock:  __|‾‾|___|‾‾|___|‾‾|___|‾‾|___
D:      __‾‾‾__|‾‾__|______|‾‾‾
Q:      ________|‾‾‾‾______|‾‾‾‾
```

- **Clock Signal**: The flip-flop operates on the rising edge of the clock signal, represented by the transitions from low to high (`0` to `1`).
- **D Input**: The value of **D** changes independently of the clock. The flip-flop captures **D** at the rising clock edge.
- **Q Output**: The output follows the input value at the clock edge and holds that value until the next clock edge.

Timing Diagram Explained:

1. **Initial state**: At the start, the clock is low, and the value of **D** is `0`. The output **Q** is also `0`.
2. **First clock edge**: When the clock rises (at the first rising edge), **D** is `1`. The flip-flop samples **D** at this moment, so **Q** changes from `0` to `1`.
3. **Between edges**: **Q** holds its value (`1`) while the clock is low or during the falling edge, regardless of changes in **D**.
4. **Second clock edge**: At the next rising clock edge, **D** is `0`. The flip-flop samples this value, and **Q** changes from `1` to `0`.
5. **Third clock edge**: At the third rising edge, **D** is `1` again. The flip-flop captures this, and **Q** switches back to `1`.

Key Features:

- **Edge-triggered**: The flip-flop responds only on the clock edge (typically rising, but there are falling-edge-triggered versions).

- **Data storage**: The value of **Q** updates only at the clock edge and remains constant between clock edges.

Summary:

- On the rising edge of the clock, the output **Q** takes on the value of **D**.
- **Q** holds its value until the next clock edge, regardless of changes in **D**.

This edge-triggered behavior makes the D flip-flop useful in synchronous circuits like registers and memory elements, where precise timing and synchronization are essential.

PART C –(1 x 14 = 14 Marks)

8. (a) **Design a combinational circuit for a 3-to-8 line decoder and evaluate its performance in terms of delay and power consumption.** 14 CO-1 L-3

Designing a 3-to-8 Line Decoder

A **3-to-8 line decoder** is a combinational circuit that takes 3 input bits and generates 8 output lines, where each output represents a unique combination of the input. Only one output will be high (logic 1) for each input combination, while the others will be low (logic 0). The function of the decoder is to "decode" the binary input into one active output.

Truth Table for a 3-to-8 Line Decoder:

Inputs	**Outputs**
A B C	Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
0 0 0	1 0 0 0 0 0 0 0
0 0 1	0 1 0 0 0 0 0 0
0 1 0	0 0 1 0 0 0 0 0
0 1 1	0 0 0 1 0 0 0 0
1 0 0	0 0 0 0 1 0 0 0
1 0 1	0 0 0 0 0 1 0 0
1 1 0	0 0 0 0 0 0 1 0
1 1 1	0 0 0 0 0 0 0 1

Boolean Expressions for Each Output:

Each output is based on the combination of inputs AAA, BBB, and CCC, and the outputs are expressed in terms of ANDed and NOTed inputs:

- Y0=A¯·B¯·C¯Y_0 = \overline{A} \cdot \overline{B} \cdot \overline{C}Y0=A·B·C
- Y1=A¯·B¯·CY_1 = \overline{A} \cdot \overline{B} \cdot CY1 =A·B·C
- Y2=A¯·B·C¯Y_2 = \overline{A} \cdot B \cdot \overline{C}Y2 =A·B·C
- Y3=A¯·B·CY_3 = \overline{A} \cdot B \cdot CY3=A·B·C
- Y4=A·B¯·C¯Y_4 = A \cdot \overline{B} \cdot \overline{C}Y4 =A·B·C
- Y5=A·B¯·CY_5 = A \cdot \overline{B} \cdot CY5=A·B·C
- Y6=A·B·C¯Y_6 = A \cdot B \cdot \overline{C}Y6=A·B·C

- Y7=A·B·CY_7 = A \cdot B \cdot CY7=A·B·C

Circuit Diagram:

Each output YiY_iYi can be generated using a combination of NOT gates and AND gates:

- **NOT gates** for generating A¯\overline{A}A, B¯\overline{B}B, and C¯\overline{C}C.
- **AND gates** for combining inputs as per the Boolean expressions.

Thus, the 3-to-8 line decoder requires:

- 3 NOT gates for the input inversion.
- 8 3-input AND gates, each taking a specific combination of AAA, BBB, CCC, and their complements to produce the corresponding output.

Performance Evaluation:

1. Delay Analysis:

The delay in the decoder is typically dependent on the number of logic gates that the signal must propagate through. There are two types of gates: NOT and AND gates.

- **NOT gate delay**: For inverting each input, there is a delay of 1 gate level.
- **AND gate delay**: Each 3-input AND gate requires multiple gate levels (e.g., for combining three inputs), which could be considered as one additional gate delay.

Total delay:

- The input signal first passes through a NOT gate (1 gate delay) if needed.
- Then it goes through a 3-input AND gate (1 gate delay for the AND operation).

So, the overall delay can be approximated as **2 gate delays** (one from the NOT gate and one from the AND gate) for each output signal.

2. Power Consumption:

Power consumption in a combinational circuit depends on factors like:

- **Static power consumption**: Due to leakage currents in the logic gates (typically small in modern CMOS technology).
- **Dynamic power consumption**: The primary source, which depends on the switching activity, capacitance, and supply voltage.

The **dynamic power** is given by the formula: P=CL·Vdd2·f·αP = C_L \cdot V_{dd}^2 \cdot f \cdot \alphaP=CL·Vdd2·f·α where:

- CLC_LCL = Load capacitance
- VddV_{dd}Vdd = Supply voltage
- fff = Frequency of operation
- α\alphaα = Switching activity

For the 3-to-8 decoder:

- Power is consumed whenever the inputs AAA, BBB, or CCC change, causing switching in the NOT and AND gates.
- The number of gates directly affects power consumption. Since there are 8 AND gates and 3 NOT gates, the switching activity across these gates will contribute to the overall power consumption.

Optimization for power:

- Using low-power logic families like CMOS can help reduce static power consumption.
- Reducing the switching activity (e.g., by ensuring inputs don't change frequently) can help lower dynamic power consumption.

Summary of Performance:

- **Delay**: Approximately 2 gate delays (one NOT gate and one AND gate for each output).
- **Power consumption**: Depends on switching activity, load capacitance, and supply voltage; modern CMOS logic typically offers low static and dynamic power consumption.

Further optimizations can be made by using advanced logic synthesis techniques or transistor-level design improvements.

OR

(b) **Design a clocked sequential circuit using SR flip-flops for a specific state transition and explain the step-by-step procedure.** 14 CO-2 L-3

Designing a **clocked sequential circuit** using **SR flip-flops** involves defining a desired state transition and then mapping it to a circuit. Here's a step-by-step guide to designing the circuit.

Problem Statement

Design a sequential circuit with two states AAA and BBB, where:

- The circuit starts in state AAA.
- If the input X=0X = 0X=0, the state remains unchanged.
- If the input X=1X = 1X=1, the circuit moves to the other state.

This means the state transitions are as follows:

- If in state AAA and X=1X = 1X=1, go to state BBB.
- If in state BBB and X=1X = 1X=1, go back to state AAA.

Step-by-Step Design Procedure

1. State Diagram

First, we create a **state diagram** to visualize the transitions:

- State AAA is represented as S0S_0S0.
- State BBB is represented as S1S_1S1.

State Transitions:\text{State Transitions:}State Transitions:
S0→X=1S1andS1→X=1S0S_0 \xrightarrow{X=1} S_1 \quad \text{and} \quad S_1 \xrightarrow{X=1} S_0S0X=1S1andS1X=1S0
S0→X=0S0andS1→X=0S1S_0 \xrightarrow{X=0} S_0 \quad \text{and} \quad S_1 \xrightarrow{X=0} S_1S0X=0S0andS1X=0S1

2. State Table

Next, we develop a **state table**. We assign binary values to the states:

- S0=0S_0 = 0S0=0
- S1=1S_1 = 1S1=1

Present State (Q)	Input (X)	Next State (Q')
0 (A)	0	0 (A)
0 (A)	1	1 (B)
1 (B)	0	1 (B)
1 (B)	1	0 (A)

3. SR Flip-Flop Excitation Table

The next step is to use the **excitation table for the SR flip-flop** to determine the flip-flop inputs based on state transitions.

Present State (Q)	Next State (Q')	SR Flip-Flop Inputs (S, R)
0	0	S = 0, R = 0
0	1	S = 1, R = 0
1	0	S = 0, R = 1
1	1	S = 0, R = 0

4. Input Combinations Based on State Table

Now, from the state table and the excitation table, we determine the values of the SR flip-flop inputs (S and R) for the transitions. We express S and R as a function of XXX and the current state QQQ.

Q (Present State)	X (Input)	S (Set)	R (Reset)
0	0	0	0
0	1	1	0

1	0	0	0
1	1	0	1

5. Derive Flip-Flop Input Equations

From the table, we can derive the equations for SSS and RRR.

- **For SSS:** S=Q‾·XS = \overline{Q} \cdot XS=Q·X
- **For RRR:** R=Q·XR = Q \cdot XR=Q·X

These Boolean equations represent the inputs to the SR flip-flop based on the current state QQQ and input XXX.

6. Circuit Diagram

Now, we can draw the circuit. The components will be:

- **SR Flip-Flop**: with inputs SSS and RRR.
- **Logic Gates**: to generate S=Q‾·XS = \overline{Q} \cdot XS=Q·X and R=Q·XR = Q \cdot XR=Q·X.
- **Clock**: to synchronize state transitions.

The circuit structure is as follows:

- The output QQQ of the flip-flop is fed into:
  - A **NOT gate** to generate Q‾\overline{Q}Q.
  - A **AND gate** with Q‾\overline{Q}Q and XXX to generate SSS.
  - A **AND gate** with QQQ and XXX to generate RRR.
- The SR flip-flop inputs SSS and RRR are connected as derived.

7. Circuit Operation

- If X=0X = 0X=0, both SSS and RRR are 0, meaning no change in state.
- If X=1X = 1X=1 and the present state is Q=0Q = 0Q=0, S=1S = 1S=1 and R=0R = 0R=0, meaning the flip-flop sets to state 1 (next state is S1S_1S1).
- If X=1X = 1X=1 and the present state is Q=1Q = 1Q=1, S=0S = 0S=0 and R=1R = 1R=1, meaning the flip-flop resets to state 0 (next state is S0S_0S0).

Conclusion

The final circuit is a clocked sequential circuit that toggles between two states S0S_0S0 and S1S_1S1 based on the input XXX. The SR flip-flop is controlled by the logic derived from the state transitions, ensuring that the circuit follows the desired behavior.