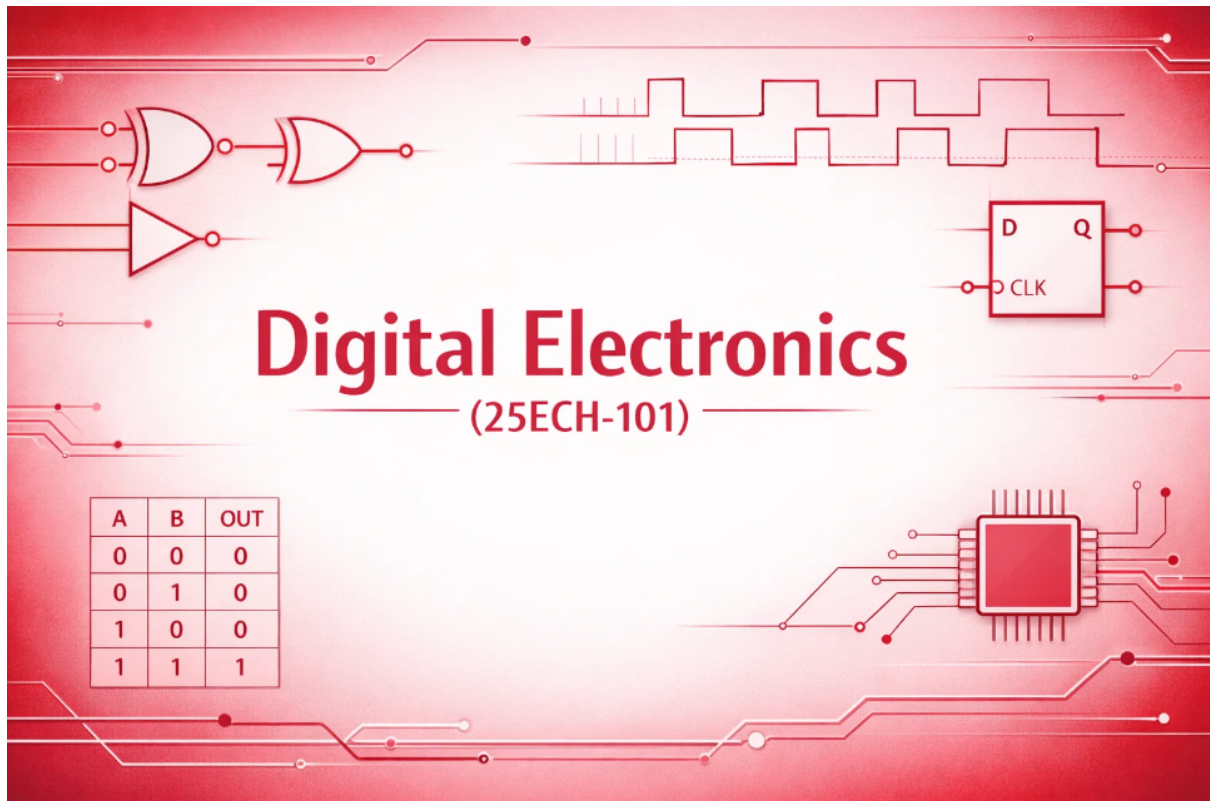




Unit 2 - Resources & Questions

Compiled by : Subhayu



Syllabus for Unit 2

Digital Electronics (25ECH-101)

Chapter 1 : Combinational Circuits – Introduction to Combinational Circuit Design, Half Adder and Full Adder, Half Subtractor and Full Subtractor, Multiplexer and Demultiplexer, Encoder and Decoder, Magnitude Comparator, Designing of Circuits using Combinational Building Blocks.

Chapter 2 : Sequential Circuits – Flip Flops – Introduction to Sequential Circuits, Latches and Flip-Flops, SR Flip-Flop, D Flip-Flop, JK Flip-Flop, T Flip-Flop, Race Around Condition, Master-Slave Flip-Flop.

Chapter 3 : Sequential Circuits – Counters and Registers – Introduction to Counters, Types of Counters, Shift Registers, Types of Shift Registers and Applications.

Chapter 1

Combinational Circuits

1.1 Introduction to Combinational Circuit Design

1.1.1 Definition

A **Combinational Circuit** is a digital logic circuit whose output depends only on the *present input values* and not on any previous state (i.e., no memory elements are involved).

Key Insight

Combinational circuits are **stateless systems**. Output at any instant = Function of current inputs only.

1.1.2 Characteristics

- No memory elements
- No feedback paths
- Output is a direct function of inputs
- Described using Boolean expressions or truth tables

1.1.3 Design Procedure

1. Define the problem clearly
2. Identify input and output variables
3. Construct the truth table
4. Simplify Boolean expressions (using K-map or algebra)



5. Implement using logic gates

1.1.4 Example

Design a circuit where output is 1 only when both inputs are 1.

Truth Table:

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Result: $Y = A \cdot B$ (AND gate)

1.2 Half Adder and Full Adder

1.2.1 Half Adder

Definition

A **Half Adder** adds two single-bit binary numbers and produces a **Sum** and a **Carry**.

Truth Table

A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean Expressions

$$S = A \oplus B$$

$$C = A \cdot B$$

Important

Half Adder cannot handle carry from previous stage $\rightarrow$ limitation in multi-bit addition.



1.2.2 Full Adder

Definition

A **Full Adder** adds three inputs: A , B , and Carry-in (C_{in}), producing Sum and Carry-out.

Truth Table

A	B	C_{in}	Sum	C_{out}
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

Boolean Expressions

$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + BC_{in} + AC_{in}$$

Solved Example

Add $A = 1$, $B = 1$, $C_{in} = 1$

$$S = 1 \oplus 1 \oplus 1 = 1$$
$$C_{out} = 1$$

Result: Sum = 1, Carry = 1

1.3 Half Subtractor and Full Subtractor

1.3.1 Half Subtractor

Definition

A **Half Subtractor** subtracts one bit from another and gives **Difference** and **Borrow**.



Truth Table

A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Expressions

$$D = A \oplus B$$

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

Conceptual Clarity

Borrow occurs only when $A < B$.

1.3.2 Full Subtractor

Definition

A **Full Subtractor** subtracts two bits along with a borrow from previous stage.

Inputs

- A (Minuend)
- B (Subtrahend)
- B_{in} (Borrow-in)

Expressions

$$D = A \oplus B \oplus B_{in}$$

$$B_{out} = \bar{A}B + \bar{A}B_{in} + BB_{in}$$

Solved Example

For $A = 0$, $B = 1$, $B_{in} = 1$

$$D = 0 \oplus 1 \oplus 1 = 0$$

$$B_{out} = 1$$



Interpretation: Borrow is required $\rightarrow$ result valid.

1.4 Multiplexer and Demultiplexer

1.4.1 Multiplexer (MUX)

Definition

A **Multiplexer** selects one of many input lines and forwards it to a single output line based on selection inputs.

Shortcut Understanding

MUX = Many inputs $\rightarrow$ One output

4:1 Multiplexer

- Inputs: I_0, I_1, I_2, I_3
- Select lines: S_0, S_1
- Output: Y

Expression

$$Y = I_0 \overline{S_1} \overline{S_0} + I_1 \overline{S_1} S_0 + I_2 S_1 \overline{S_0} + I_3 S_1 S_0$$

Example

If $S_1 S_0 = 10$, output is I_2

1.4.2 Demultiplexer (DEMUX)

Definition

A **Demultiplexer** takes a single input and routes it to one of many outputs.

Shortcut Understanding

DEMUX = One input $\rightarrow$ Many outputs



1:4 Demultiplexer

- Input: D
- Select lines: S_0, S_1
- Outputs: Y_0, Y_1, Y_2, Y_3

Expressions

$$Y_0 = D\overline{S_1}\overline{S_0}$$

$$Y_1 = D\overline{S_1}S_0$$

$$Y_2 = DS_1\overline{S_0}$$

$$Y_3 = DS_1S_0$$

Solved Example

If $D = 1$, $S_1S_0 = 01$

$$Y_1 = 1, \text{ others} = 0$$

1.5 Summary

- Combinational circuits depend only on current inputs
- Adders and subtractors perform arithmetic operations
- Multiplexers select data, Demultiplexers distribute data
- Boolean expressions are the foundation of all designs

Final Takeaway

Mastering combinational circuits builds the foundation for advanced digital systems like processors, memory units, and control logic.



1.6 Encoder and Decoder

1.6.1 Encoder

Definition

An **Encoder** is a combinational circuit that converts 2^n input lines into n output lines (binary representation).

Core Idea

Encoder performs **many-to-fewer conversion** (inputs $\rightarrow$ compact binary code).

4-to-2 Encoder

- Inputs: D_0, D_1, D_2, D_3
- Outputs: Y_1, Y_0

Truth Table:

D_3	D_2	D_1	D_0	Output (Y_1Y_0)
0	0	0	1	00
0	0	1	0	01
0	1	0	0	10
1	0	0	0	11

Expressions

$$Y_1 = D_2 + D_3$$

$$Y_0 = D_1 + D_3$$

Limitation

If multiple inputs are active simultaneously, output becomes ambiguous.

Important

Basic encoders assume **only one input is active at a time**.

1.6.2 Priority Encoder

Definition

A **Priority Encoder** resolves ambiguity by assigning priority to inputs.

Example: If D_3 and D_1 are both 1, output corresponds to D_3 (higher priority).



1.6.3 Decoder

Definition

A **Decoder** converts n input lines into 2^n output lines.

Core Idea

Decoder performs **binary-to-unique output mapping**.

2-to-4 Decoder

- Inputs: A, B
- Outputs: Y_0, Y_1, Y_2, Y_3

Truth Table:

A	B	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Expressions

$$Y_0 = \overline{A}\overline{B}$$

$$Y_1 = \overline{A}B$$

$$Y_2 = A\overline{B}$$

$$Y_3 = AB$$

Application Insight

Decoders are widely used in:

- Memory address decoding
- Instruction decoding in CPUs

1.7 Magnitude Comparator

1.7.1 Definition

A **Magnitude Comparator** compares two binary numbers and determines their relation:



- $A > B$
- $A = B$
- $A < B$

Conceptual Understanding

Comparator answers the question: "*Which number is bigger?*"

1.7.2 1-bit Comparator

Inputs

- A, B

Outputs

- G ($A > B$)
- E ($A = B$)
- L ($A < B$)

Truth Table

A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Expressions

$$G = A\bar{B}$$

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

$$L = \bar{A}B$$

1.7.3 Multi-bit Comparator Insight

- Compare MSB first
- If equal, move to next bit
- Continue until difference is found

**Important Strategy**

Higher-order bits dominate the comparison outcome.

1.8 Designing Circuits using Combinational Building Blocks

1.8.1 Concept

Complex digital systems are built using simpler combinational blocks like:

- Adders
- Multiplexers
- Decoders
- Encoders

Key Idea

Instead of designing from scratch, reuse **building blocks** to simplify circuit design.

1.8.2 Design using Multiplexer

Example Problem

Implement function:

$$F(A, B, C) = \sum m(1, 2, 6, 7)$$

Steps

1. Use a 4:1 MUX
2. Select lines: A, B
3. Express output in terms of C

MUX Inputs:

- $I_0 = 0$
- $I_1 = 1$
- $I_2 = C$
- $I_3 = C$



1.8.3 Design using Decoder

Example

Implement:

$$F(A, B, C) = \sum m(1, 3, 5, 7)$$

Approach

- Use 3-to-8 decoder
- OR outputs corresponding to minterms

$$F = Y_1 + Y_3 + Y_5 + Y_7$$

1.8.4 Design Insight

- MUX-based design is flexible
- Decoder-based design is systematic

Exam Tip

If asked to design using MUX → focus on select lines and input mapping.
If asked to design using Decoder → focus on minterms and OR combination.

1.9 Final Wrap-Up of Chapter

- Encoders compress information, Decoders expand it
- Comparators enable decision-making in circuits
- Building blocks simplify complex logic design
- Real-world systems combine multiple combinational units

Big Picture

Combinational circuits form the **foundation of digital electronics**, enabling arithmetic operations, data routing, and logical decision-making in modern computing systems.

Chapter 2

Sequential Circuits – Flip Flops

2.1 Introduction to Sequential Circuits

2.1.1 Definition

A **Sequential Circuit** is a digital circuit whose output depends on both *present inputs* and *past history (stored state)*.

Key Difference

Combinational Circuits → No memory

Sequential Circuits → Have memory (state-dependent)

2.1.2 Key Components

- **Combinational Logic** (AND, OR, NOT gates)
- **Memory Elements** (Latches, Flip-Flops)
- **Clock Signal** (in synchronous systems)

2.1.3 Types of Sequential Circuits

- **Synchronous** – Controlled by a clock
- **Asynchronous** – No clock, changes occur immediately

2.1.4 Example Insight

Consider a digital clock:

- Output depends on previous time → memory required



2.2 Latches and Flip-Flops

2.2.1 Latches

Definition

A **Latch** is a basic memory element that stores one bit of data and operates based on input levels.

Characteristics

- Level-triggered (responds to input levels)
- Simpler than flip-flops
- Can be unstable if not designed properly

2.2.2 Flip-Flops

Definition

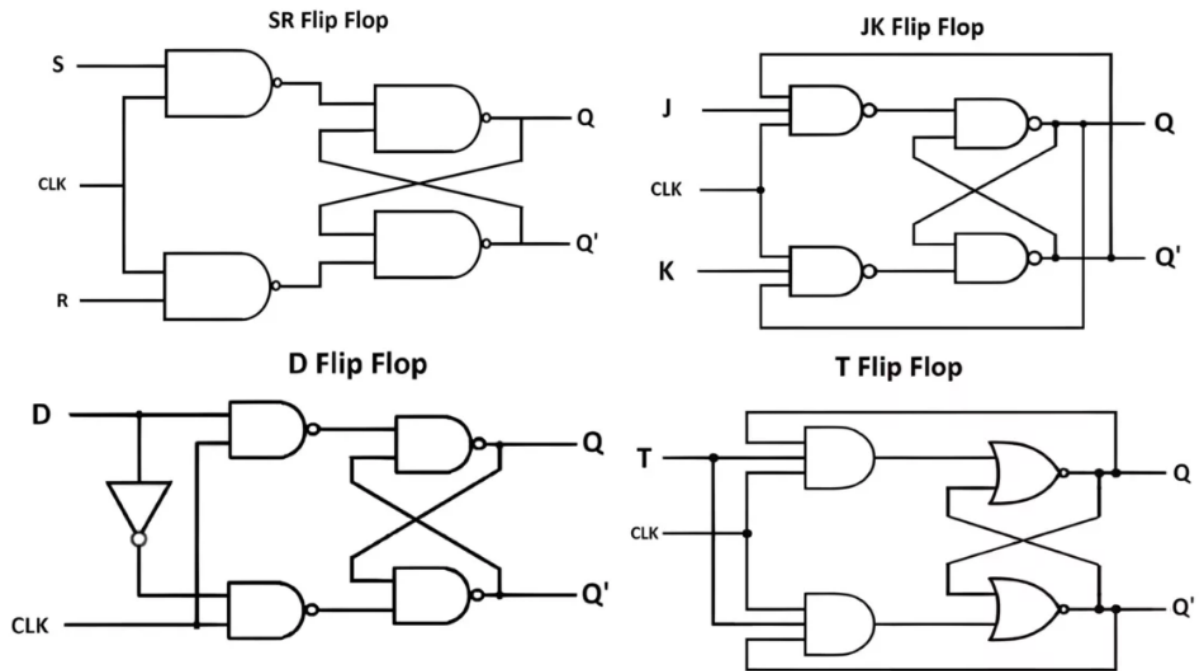
A **Flip-Flop** is a clock-controlled bistable device used to store one bit of information.

Important Concept

Flip-Flops are **edge-triggered** → state changes only at clock transitions.

2.2.3 Latch vs Flip-Flop

Feature	Latch	Flip-Flop
Triggering	Level	Edge
Control	No clock required	Clock required
Stability	Less stable	More stable



2.3 SR Flip-Flop

2.3.1 Definition

The **SR (Set-Reset) Flip-Flop** is the simplest flip-flop used to store one bit.

2.3.2 Inputs and Outputs

- Inputs: S (Set), R (Reset)
- Outputs: Q , $\overline{Q}$

2.3.3 Truth Table

S	R	Q_{next}
0	0	No Change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Invalid

Critical Point

The condition $S = 1$ and $R = 1$ is **invalid** because it leads to an undefined state.

2.3.4 Working Explanation

- $S = 1 \rightarrow$ Output becomes 1 (Set)



- $R = 1 \rightarrow$ Output becomes 0 (Reset)
- $S = R = 0 \rightarrow$ Memory state (holds previous value)

2.3.5 Solved Example

If previous $Q = 0$ and inputs are $S = 1, R = 0$

$$Q_{next} = 1$$

Interpretation: Flip-flop is set.

2.4 D Flip-Flop

2.4.1 Definition

A **D (Data or Delay) Flip-Flop** eliminates the invalid state of SR flip-flop by ensuring only one input controls the output.

2.4.2 Inputs and Outputs

- Input: D
- Output: Q

2.4.3 Truth Table

D	Q_{next}
0	0
1	1

2.4.4 Characteristic Equation

$$Q_{next} = D$$

Key Insight

D Flip-Flop simply **copies input to output** at the clock edge.



2.4.5 Working Principle

- On clock edge:
 - If $D = 1 \rightarrow Q = 1$
 - If $D = 0 \rightarrow Q = 0$
- Between clock edges $\rightarrow$ output remains constant

2.4.6 Solved Example

Given $D = 1$ at clock edge:

$$Q_{next} = 1$$

Interpretation: Value is stored in memory.

2.5 Applications of Flip-Flops

- Memory storage units (Registers)
- Counters
- Shift registers
- Digital systems (CPU, RAM)

Final Takeaway

Flip-Flops are the **building blocks of memory** in digital electronics. Without them, storing and processing sequential data would not be possible.

2.6 JK Flip-Flop

2.6.1 Definition

The **JK Flip-Flop** is an improved version of the SR Flip-Flop that eliminates the invalid condition.

2.6.2 Inputs and Outputs

- Inputs: J, K
- Outputs: $Q, \overline{Q}$



2.6.3 Truth Table

J	K	Q_{next}
0	0	No Change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Toggle

2.6.4 Characteristic Equation

$$Q_{next} = J\bar{Q} + \bar{K}Q$$

Key Insight

When $J = K = 1$, the JK Flip-Flop **toggles** its state instead of becoming invalid.

2.6.5 Solved Example

If current $Q = 0$ and $J = 1$, $K = 1$

$$Q_{next} = 1$$

Next clock cycle:

$$Q_{next} = 0$$

Interpretation: Output keeps toggling.

2.7 T Flip-Flop

2.7.1 Definition

The **T (Toggle) Flip-Flop** is a simplified version of the JK Flip-Flop where:

$$T = J = K$$

2.7.2 Operation

- $T = 0 \rightarrow$ No change
- $T = 1 \rightarrow$ Toggle output



2.7.3 Truth Table

T	Q_{next}
0	Q (No change)
1	$\overline{Q}$ (Toggle)

2.7.4 Characteristic Equation

$$Q_{next} = T \oplus Q$$

Important

T Flip-Flops are widely used in **counters** because of their toggling behavior.

2.7.5 Solved Example

If $Q = 1$ and $T = 1$:

$$Q_{next} = 0$$

2.8 Race Around Condition

2.8.1 Definition

The **Race Around Condition** occurs in a JK Flip-Flop when:

- $J = K = 1$
- Clock pulse duration is high

2.8.2 What Happens?

- Output toggles continuously during the clock pulse
- Final output becomes unpredictable

Critical Concept

Race Around Condition leads to **unstable and undefined output**.

2.8.3 Reason

- Propagation delay allows multiple toggles within one clock cycle



2.8.4 Solutions

- Use edge-triggered flip-flops
- Use Master-Slave configuration
- Reduce clock pulse width

2.9 Master-Slave Flip-Flop

2.9.1 Definition

A **Master-Slave Flip-Flop** consists of two flip-flops connected in series:

- Master (first stage)
- Slave (second stage)

2.9.2 Working Principle

- Master is active during clock = 1
- Slave is active during clock = 0

Core Idea

Only one stage is active at a time → prevents multiple toggling.

2.9.3 Operation Flow

1. Input is captured by Master during clock HIGH
2. Output is transferred to Slave during clock LOW

2.9.4 Advantage

- Eliminates race around condition
- Provides stable output

2.9.5 Example Insight

If $J = K = 1$:

- Master toggles once
- Slave updates only once → controlled output



2.10 Final Summary of Flip-Flops

- JK Flip-Flop removes invalid state of SR Flip-Flop
- T Flip-Flop simplifies toggling behavior
- Race Around Condition causes instability in JK Flip-Flop
- Master-Slave configuration resolves this issue

Big Picture

Sequential circuits rely on Flip-Flops to store and manipulate data over time, forming the backbone of memory and timing systems in digital electronics.

Chapter 3

Sequential Circuits – Counters and Registers

3.1 Introduction to Counters

3.1.1 Definition

A **Counter** is a sequential circuit that **counts the number of clock pulses** applied to its input.

Core Idea

Counters are **sequential circuits** that change state in a predefined sequence with each clock pulse.

3.1.2 Basic Concept

- Built using flip-flops
- Each flip-flop represents one bit
- Number of states = 2^n (for n flip-flops)

3.1.3 Example

A 3-bit counter:

- Number of states = $2^3 = 8$
- Counts from 000 to 111 (0 to 7)



3.1.4 Working Principle

- Each clock pulse triggers state transition
- Output sequence follows a specific pattern

Important Insight

Counters are widely used in **timing circuits, digital clocks, frequency division, and event counting.**

3.2 Types of Counters

3.2.1 Classification Overview

Counters can be classified based on:

- Clocking method
- Counting direction
- Modulus (number of states)

3.2.2 Asynchronous (Ripple) Counters

Definition

An **Asynchronous Counter** is one where the clock input is applied only to the first flip-flop, and subsequent flip-flops are triggered by the output of the previous one.

Characteristics

- Simple design
- Propagation delay accumulates
- Slower compared to synchronous counters

Key Limitation

Ripple effect causes delay → not suitable for high-speed applications.

Example

3-bit ripple counter:

- First flip-flop receives clock
- Each next flip-flop toggles based on previous output



3.2.3 Synchronous Counters

Definition

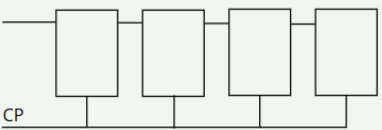
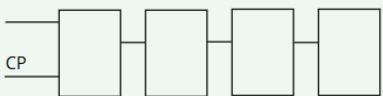
A **Synchronous Counter** is one where all flip-flops are triggered **simultaneously** by a common clock signal.

Characteristics

- Faster operation
- No ripple delay
- More complex design

Important Advantage

All flip-flops change state at the same time → **high-speed performance**.

Synchronous counter	Asynchronous counter
 Basic diagram	 Basic diagram
It have a non-sequential mode of counting (different state of counting 1 - 4 - 5- 7)	It have a sequential mode of counting (different state of counting 1 - 2 - 3 - 4 OR 4 - 3 - 2 - 1)
Ripple and Johnson counter	Ripple counter
Faster	Slower
Complex circuit	Simple circuit
Cost is high	Cost is low

3.2.4 Up Counter

Definition

An **Up Counter** counts in increasing order.

Example Sequence (3-bit)

000 → 001 → 010 → 011 → 100 → 101 → 110 → 111



3.2.5 Down Counter

Definition

A **Down Counter** counts in decreasing order.

Example Sequence (3-bit)

$111 \rightarrow 110 \rightarrow 101 \rightarrow 100 \rightarrow 011 \rightarrow 010 \rightarrow 001 \rightarrow 000$

3.2.6 Up-Down Counter

Definition

An **Up-Down Counter** can count both upward and downward based on a control input.

- Control = 1 $\rightarrow$ Count Up
- Control = 0 $\rightarrow$ Count Down

3.2.7 Modulus Counter

Definition

A **Modulus (MOD) Counter** is a counter that cycles through a fixed number of states.

- MOD- $n \rightarrow n$ states

Example

MOD-10 counter (Decade Counter):

- Counts from 0 to 9
- Resets after reaching 9

Exam Tip

MOD value determines the **range of counting**. Always check reset condition.

3.2.8 Ring Counter

Definition

A **Ring Counter** is a circular shift register where only one flip-flop is set at a time.



Example (4-bit)

0001 $\rightarrow$ 0010 $\rightarrow$ 0100 $\rightarrow$ 1000 $\rightarrow$ 0001

3.2.9 Johnson Counter

Definition

A **Johnson Counter** (Twisted Ring Counter) is a modified ring counter where inverted output is fed back.

Example (4-bit)

0000 $\rightarrow$ 1000 $\rightarrow$ 1100 $\rightarrow$ 1110 $\rightarrow$ 1111 $\rightarrow$ 0111 $\rightarrow$ 0011 $\rightarrow$ 0001 $\rightarrow$ 0000

Important

An n -bit Johnson Counter has **$2n$ states**, unlike ring counter which has n states.

3.3 Summary

- Counters track number of clock pulses
- Asynchronous counters are simple but slow
- Synchronous counters are fast and reliable
- Different counters serve different applications

Final Takeaway

Counters are essential in digital systems for **counting, timing, sequencing, and control operations**.

3.4 Shift Registers

3.4.1 Definition

A **Shift Register** is a sequential circuit used to **store and shift binary data** using flip-flops.

Core Idea

Shift Registers move (shift) data **left or right** with each clock pulse.



3.4.2 Basic Structure

- Consists of a series of flip-flops
- Each flip-flop stores one bit
- Output of one flip-flop is connected to input of next

3.4.3 Working Principle

- On each clock pulse, data shifts one position
- Direction depends on circuit design

3.4.4 Example

Consider a 4-bit shift register with input 1011:

Initial: 0000 → 0001 → 0010 → 0101 → 1011

3.5 Types of Shift Registers

3.5.1 Classification Based on Input/Output

3.5.2 Serial-In Serial-Out (SISO)

Definition

Data is entered **serially** (one bit at a time) and output is also obtained serially.

Characteristics

- Simple design
- Slow data transfer

Example

Input: 1011 → Output appears after 4 clock pulses

3.5.3 Serial-In Parallel-Out (SIPO)

Definition

Data is entered serially but output is available **simultaneously** at all flip-flops.



Use Case

- Serial to parallel data conversion

3.5.4 Parallel-In Serial-Out (PISO)

Definition

Data is loaded in parallel and shifted out serially.

Use Case

- Parallel to serial data transmission

3.5.5 Parallel-In Parallel-Out (PIPO)

Definition

Data is loaded and retrieved in parallel.

Characteristic

- Fastest type
- No shifting required for output

Quick Comparison

SISO → Serial in, Serial out (slow)

SIPO → Serial in, Parallel out

PISO → Parallel in, Serial out

PIPO → Parallel in, Parallel out (fastest)

3.5.6 Bidirectional Shift Register

Definition

A **Bidirectional Shift Register** can shift data both left and right.

Control

- Control signal determines direction



3.5.7 Universal Shift Register

Definition

A **Universal Shift Register** can perform multiple operations:

- Shift left
- Shift right
- Parallel load
- Hold (no change)

Important

Universal Shift Register = **most flexible and widely used**.

3.6 Applications of Shift Registers

3.6.1 Data Storage

- Temporary storage of binary data

3.6.2 Data Transfer

- Serial-to-parallel and parallel-to-serial conversion

3.6.3 Counters

- Used in ring and Johnson counters

3.6.4 Arithmetic Operations

- Shifting left → multiplication by 2
- Shifting right → division by 2

3.6.5 Digital Systems

- Used in CPUs, communication systems, and memory units

Real-World Insight

Shift registers are essential in **data communication**, especially in devices like keyboards, displays, and microcontrollers.



3.7 Final Summary of Counters and Registers

- Counters keep track of events using clock pulses
- Shift registers store and move data efficiently
- Different types serve different system requirements
- Together, they form the backbone of sequential data processing

Big Picture

Registers and Counters enable **data storage, transfer, and sequencing**, making them fundamental components in all modern digital systems.



Mid Semester Test 2 – Sample Paper

Total Marks : 20

Time Allotted : 1 hour

Instructions:

- The question paper consists of 2 sections: Section A and Section B.
- Section A has 5 questions of 2 marks each and Section B has 2 questions of 5 marks each.

Section A ($5 \times 2 = 10$ marks)

1. Differentiate between synchronous and asynchronous counters. Which one is faster and why?
2. Define a combinational circuit. How is it different from a sequential circuit? Give two practical examples.
3. A 4:1 Multiplexer has inputs $I_0 = 0$, $I_1 = 1$, $I_2 = A$, $I_3 = \bar{A}$. Determine the output expression.
4. What is Race Around Condition? Explain its cause and methods to eliminate it.
5. Derive the Boolean expression for a Full Adder and explain how it overcomes the limitations of a Half Adder.

Section B ($2 \times 5 = 10$ marks)

6. Explain the JK Flip-Flop and derive its characteristic equation. Why is it more versatile than SR Flip-Flop?
7. Differentiate between SISO, SIPO, PISO, and PIPO shift registers with applications.

Answer Key

Section A ($5 \times 2 = 10$ marks)

1. Difference between Synchronous and Asynchronous Counters

Feature	Synchronous Counter	Asynchronous Counter
Clock Input	Common clock to all flip-flops	Clock to first flip-flop only
Operation	All flip-flops change simultaneously	Changes occur sequentially (ripple effect)
Speed	Faster	Slower
Delay	Minimal	Accumulates with each stage

Conclusion: Synchronous counters are faster because all flip-flops are triggered at the same time, eliminating propagation delay.

2. Combinational Circuit and its Difference from Sequential Circuit

Definition: A combinational circuit is a digital circuit whose output depends only on present inputs.

Difference:

- Combinational: No memory, output depends only on current input
- Sequential: Has memory, output depends on past and present inputs

Examples:

- Combinational: Adder, Multiplexer
- Sequential: Flip-Flop, Counter

3. Output of 4:1 Multiplexer

General expression:

$$Y = I_0 \overline{S_1} \overline{S_0} + I_1 \overline{S_1} S_0 + I_2 S_1 \overline{S_0} + I_3 S_1 S_0$$



Substituting:

$$I_0 = 0, I_1 = 1, I_2 = A, I_3 = \bar{A}$$

$$Y = 0 + \bar{S}_1 S_0 + A S_1 \bar{S}_0 + \bar{A} S_1 S_0$$

Final Expression:

$$Y = \bar{S}_1 S_0 + A S_1 \bar{S}_0 + \bar{A} S_1 S_0$$

4. Race Around Condition

Definition: Race around condition occurs in JK flip-flop when $J = K = 1$ and the clock pulse is high for a long duration.

Cause:

- Output toggles continuously within one clock pulse
- Due to propagation delay

Elimination Methods:

- Use edge-triggered flip-flops
- Use master-slave flip-flop
- Reduce clock pulse width

5. Full Adder Expression and Advantage

Expressions:

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + BC_{in} + AC_{in}$$

Explanation:

- Full Adder adds three bits (A, B, Carry-in)
- It handles carry from previous stage

Advantage over Half Adder:

- Half Adder cannot process carry input
- Full Adder enables multi-bit binary addition



Section B ($2 \times 5 = 10$ marks)

6. JK Flip-Flop

Definition: A JK Flip-Flop is a modified SR Flip-Flop that eliminates the invalid state.

Truth Table:

J	K	Q_{next}
0	0	No change
0	1	0 (Reset)
1	0	1 (Set)
1	1	Toggle

Derivation of Characteristic Equation:

From truth table:

- Set condition $\rightarrow J = 1, K = 0$
- Reset condition $\rightarrow J = 0, K = 1$
- Toggle $\rightarrow$ depends on previous Q

$$Q_{next} = J\bar{Q} + \bar{K}Q$$

Explanation:

- $J\bar{Q}$ sets output
- $\bar{K}Q$ maintains/reset condition

Why More Versatile than SR Flip-Flop:

- No invalid condition
- Can perform toggle operation
- Used in counters and control circuits

Key Point

JK Flip-Flop combines functionality of SR and T Flip-Flops, making it highly flexible.

7. Types of Shift Registers

Comparison Table:



Type	Input	Output	Application
SISO	Serial	Serial	Delay circuits
SIPO	Serial	Parallel	Serial to parallel conversion
PISO	Parallel	Serial	Data transmission
PIPO	Parallel	Parallel	Data storage

Explanation:

- **SISO:** Data enters and leaves one bit at a time (slowest)
- **SIPO:** Converts serial input into parallel output (used in displays)
- **PISO:** Loads data simultaneously, outputs serially (used in communication)
- **PIPO:** Fastest, used for temporary storage

Conclusion:

- Choice depends on data format and application
- Widely used in digital communication and processing systems

Closing Note

I sincerely hope that these notes and the preparatory questions included alongside them prove helpful in strengthening your understanding and boosting your confidence for the examination you are preparing for.

As you revise, try not to rush through the content—pause at examples, work out the numericals on your own, and revisit sections that feel slightly uncomfortable. That discomfort usually signals learning in progress. Use these notes as a guide and a companion, not as something to be memorized line by line.

Wishing you the very best for your exams—may your preparation feel clear, structured, and calm when it matters most.

For future updates, refinements, and additional resources, keep checking **Noteify**:

<https://subhayu18.github.io/Noteify/>

If you have any doubts, suggestions, or simply wish to connect, feel free to reach out to me on LinkedIn:

<https://www.linkedin.com/in/subhayu-mukherjee-146421294/>

You can also use the direct contact options provided on the Noteify website itself.

Regards,
Subhayu