CARDAMOM PLANTERS' ASSOCIATION COLLEGE
(Re-Accredited With 'A' Grade By NAAC)
Pankajam Nagar, Bodinayakanur - 625 582.

DEPARTMENT OF CS & IT

Unit - 4

Sequential Logic: RS, JK, D, and T Flip-Flops–Master-Slave Flip- Flops. Registers: Shift Registers – Types of Shift Registers.

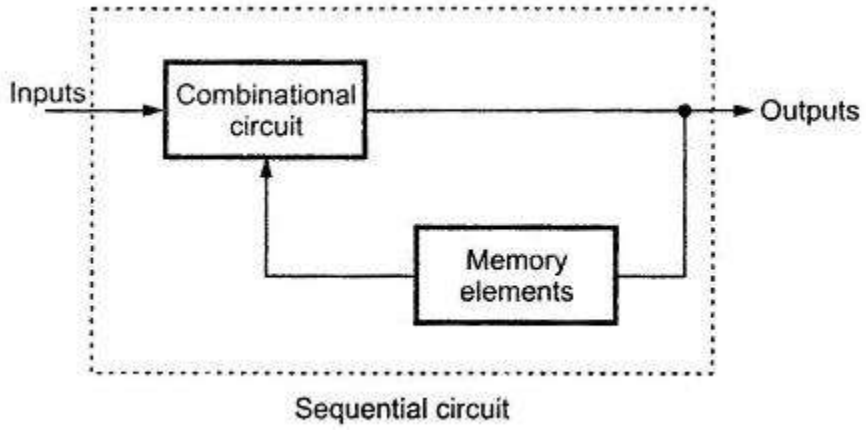
4.1. Sequential logic circuit

A **sequential logic circuit** is a type of digital circuit whose output depends not only on the current inputs but also on the **previous history of inputs** (i.e., it has memory).

Basic Components

Sequential circuits are built using:

- **Logic gates** (AND, OR, NOT, etc.)
- **Memory elements** (like flip-flops or latches)
- **Feedback paths** (to store past information)



A typical sequential circuit includes:

- Inputs
- Combinational logic
- Memory (state)
- Outputs

It can be represented as:

Next State = f(Current State, Input)
Output = g(Current State, Input)

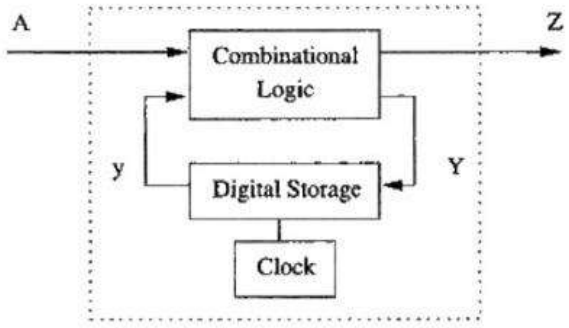
Examples of Sequential Circuits

- Counters
- Shift registers
- Digital clocks
- Finite State Machines (FSM)

4.1.1. Types of Sequential Circuits

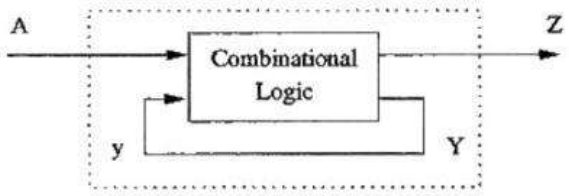
1. Synchronous Sequential Circuits

- Operate with a **clock signal**
- State changes occur at specific times (clock pulses)
- Example: Registers, Counters



2. Asynchronous Sequential Circuits

- No clock signal
- Output changes immediately when input changes
- Faster but harder to design



4.2. Flip-Flops

Flip-flops are the basic memory elements used in **sequential logic circuits**. Each flip-flop can store **1 bit** of information (0 or 1). It is a **bistable device** (has two stable states) and changes its state based on input signals, usually controlled by a **clock pulse**.

Common types:

- SR Flip-Flop
- JK Flip-Flop

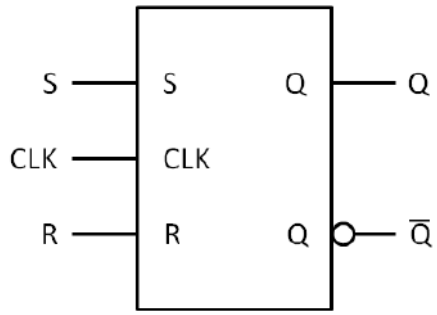
- D Flip-Flop
- T Flip-Flop

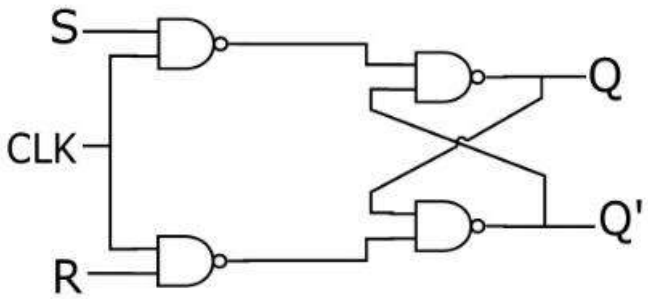
These store **1 bit of data**.

4.2.1. SR Flip-Flop

The **SR flip-flop** is the simplest type of flip-flop, built from **NOR** or **NAND gates**, and it stores **one bit of data**. It has two inputs:

- **S (Set)** → forces output $Q = 1$
- **R (Reset)** → forces output $Q = 0$





Truth Table

S	R	Q(n+1)	Meaning
0	0	Qn	No change (Memory)
0	1	0	Reset
1	0	1	Set
1	1	X	Invalid ❌

Working Principle

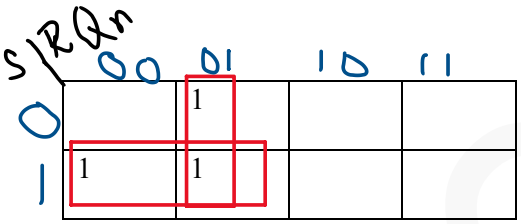
- When **S = 1, R = 0** the circuit is **set**, so Q becomes 1

- When **S = 0, R = 1** the circuit is **reset**, so Q becomes 0
- When both are 0**(S = 0, R = 0),** it **remembers previous state**
- When both are 1 → **unstable/invalid**

Characteristic Table (Output based on inputs)

Clock	S	R	Qn	Qn+1	Description
↑	0	0	0	0	No change
↑	0	0	1	1	No change
↑	0	1	0	0	Reset
↑	0	1	1	0	Reset
↑	1	0	0	1	Set
↑	1	0	1	1	Set
↑	1	1	0	X	Invalid ❌
↑	1	1	1	X	Invalid ❌

Characteristic Equation



$Q_{n+1} = S + \bar{R}\, Q_n$

Q_{n+1} → Next state

Q_n → Present state

This equation defines the **next state** in terms of inputs and the present state.

Excitation Table (Inputs required for desired output)

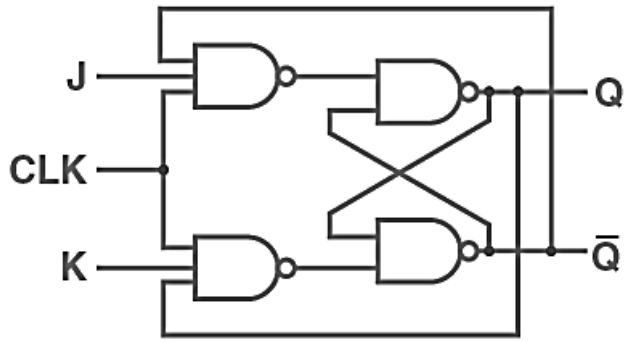
Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

4.2.2. JK Flip Flop

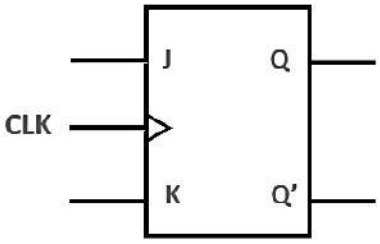
The **JK flip-flop** is an improvement over the SR flip-flop. It eliminates the **invalid state** (when S=R=1) by introducing two inputs:

- **J (Set)**
- **K (Reset)**

It is **edge-triggered** and controlled by a clock signal.



Symbol



Truth Table

CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	Q_n'

Working Principle

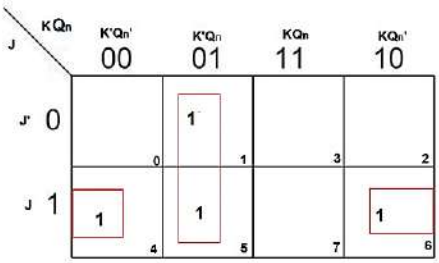
- When **J = 1, K = 0**, the circuit is set, so Q becomes 1
- When **J = 0, K = 1**, the circuit is reset, so Q becomes 0
- When **both are 0 (J = 0, K = 0),** it remembers **previous state**
- When **both are 1 (J = 1, K = 1),** the **output toggles (**Q changes to opposite)
    - If Q = 0 → becomes 1
    - If Q = 1 → becomes 0

Characteristic Table (Output based on inputs)

Clock	J	K	Qn	Q(n+1)	Operation
↑	0	0	0	0	No change
↑	0	0	1	1	No change
↑	0	1	0	0	Reset
↑	0	1	1	0	Reset
↑	1	0	0	1	Set

↑	1	0	1	1	Set
↑	1	1	0	1	Toggle
↑	1	1	1	0	Toggle

Characteristic Equation



$$Q_{n+1} = J\overline{Q_n} + \overline{K}Q_n$$

This shows how the next state depends on inputs J, K and the present state Q_n.

Excitation Table (Inputs required for desired output)

Qn	Q(n+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

This table is used when designing sequential circuits to determine required inputs for a desired transition.

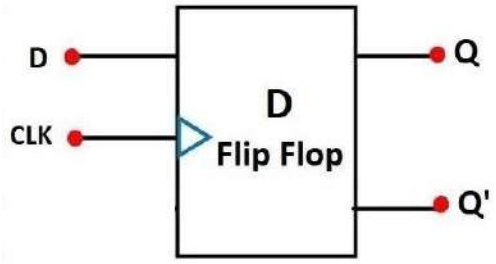
Advantages

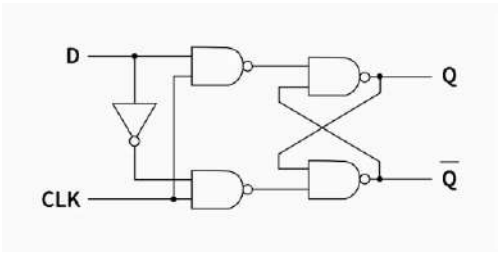
- No invalid condition
- More flexible than SR flip-flop
- Used in counters and registers

4.2.3. D Flip-Flop

The **D Flip-Flop** (Data or Delay Flip-Flop) is a memory device used in sequential circuits to store **1 bit of data**.

It has only **one input (D)**, which avoids the invalid state found in SR flip-flop.





Truth Table

D	Q(n+1)	Operation
0	0	Reset
1	1	Set

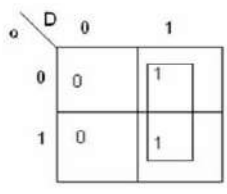
Working Principle

- **When D = 0**, the circuit is reset, so **Q becomes 0**
- **When D = 1**, the circuit is set, so **Q becomes 1**
- The output changes only at the **clock pulse**

Characteristic Table (Output based on inputs)

D	Qn	Q(n+1)
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic Equation



$Q_{n+1} = D$

Next state is always equal to input D.

Excitation Table (Inputs required for desired output)

Qn	Q(n+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Advantages

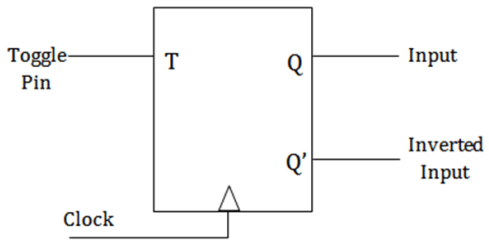
✓ Simple design
✓ No invalid state
✓ Reliable operation

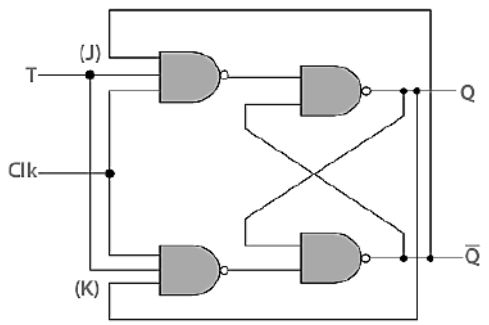
Applications

- Registers
- Shift registers
- Memory storage
- Data transfer circuits

4.2.4. T Flip-Flop (Toggle Flip-Flop)

The **T flip-flop** is a simple flip-flop used to **toggle** its output. It is derived from the JK flip-flop by connecting **J = K = T**





Truth Table

T	Q(n+1)	Operation
0	Qn	No change
1	$\bar{Q}n$	Toggle

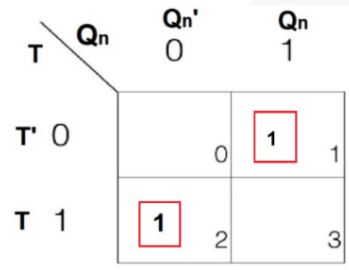
Working Principle

- **When T = 0**, the circuit **remembers previous state**
- **When T = 1**, the output **toggles**
  - If Q = 0 → becomes 1
  - If Q = 1 → becomes 0

Characteristic Table (Output based on inputs)

T	Qn	Q(n+1)	Operation
0	0	0	No change
0	1	1	No change
1	0	1	Toggle
1	1	0	Toggle

Characteristic Equation



$Qn+1=T \oplus Qn$

👉 ($\oplus$ = XOR operation)

Excitation Table (Inputs required for desired output)

Qn	Q(n+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Advantages

✓ Simple design
✓ Used for toggling operations
✓ No invalid state

Applications

- Counters (very important)
- Frequency division
- Toggle switches

4.3. Registers

A **register** is a collection of **flip-flops** grouped together to store **multiple bits of data**. Since each flip-flop stores **1 bit**, an *n-bit register* contains *n flip-flops*. Registers are fundamental in digital systems for **data storage, transfer, and manipulation**.

Structure

- **Basic Unit**: Flip-flop (usually D flip-flops).
- **Size**: Number of flip-flops = number of bits stored.
- **Control Signals**: Clock, Load/Enable, Clear/Reset.

Basic Idea

- Used for **temporary data storage**
- Works with a **common clock signal**
- Data is stored and transferred in **binary form**

Types of Registers

Type	Function	Example Use
Buffer Register	Temporary storage of data	I/O operations
Shift Register	Shifts data left/right	Serial-to-parallel conversion
Parallel Register	Loads all bits simultaneously	CPU registers
Counter Register	Counts clock pulses	Timers, frequency division
Universal Shift Register	Can shift both directions and load parallel data	Complex data manipulation

Operations

- **Load**: Store new data into the register.

- **Shift**: Move data left or right (serial transfer).
- **Clear**: Reset all bits to 0.
- **Hold**: Maintain current state.

Applications

- **CPU Registers**: Accumulator, instruction register, program counter.
- **Data Transfer**: Between memory and processor.
- **Counters & Timers**: Event counting, clock division.
- **Serial Communication**: Converting between serial and parallel data.

Example: 4-bit Register

- Built using **4 D flip-flops**.
- Each flip-flop stores one bit: Q_0, Q_1, Q_2, Q_3.
- On a clock edge, all flip-flops update simultaneously → parallel load.

4.3.1. Shift Register

A **shift register** is a type of register that can **store and move data** either to the left or right. It is built using **D flip-flops** connected in series, with the output of one feeding the input of the next.

Working Principle

- Data is entered into the register
- On each **clock pulse**, the data shifts to the next flip-flop
- Output is taken either **serially or in parallel**

Types of Shift Registers

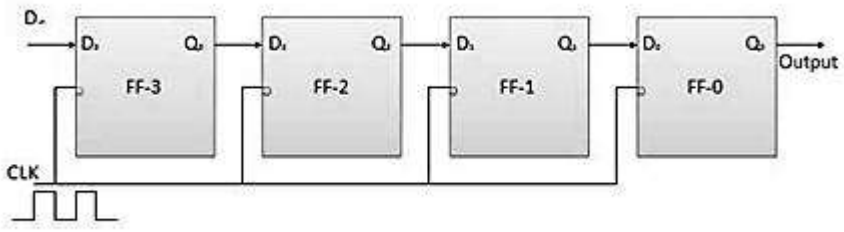
Type	Operation	Example Use
Serial-In Serial-Out (SISO)	Data enters one bit at a time, shifts out serially (Output also comes **one bit at a time**)	Communication lines
Serial-In Parallel-Out (SIPO)	Data enters serially, available in parallel (Output)	Serial-to-parallel conversion
Parallel-In Serial-Out (PISO)	Data loaded in parallel, shifted out serially (Output comes **bit by bit**)	Parallel-to-serial conversion
Parallel-In Parallel-Out (PIPO)	Data loaded and read in parallel	Fast data transfer
Universal Shift Register	Can shift both directions and load parallel data	Complex data manipulation

4.3.1.1. Serial-In Serial-Out

The **Serial-In Serial-Out (SISO)** shift register is the simplest form of a shift register. It accepts data **one bit at a time** (serial input) and shifts it through a chain of flip-flops until it appears at the **serial output (Data is retrieved one bit at a time)**.

Structure

- Built using **D flip-flops** connected in series.
- **Serial Input (SI)** → feeds the first flip-flop.
- **Clock** → synchronizes shifting; each pulse moves data one stage forward.
- **Serial Output (SO)** → comes from the last flip-flop.



Operation

1. On each clock pulse, the bit at the input moves into the first flip-flop.
2. Existing bits shift one position to the right.
3. After *n* clock pulses, the first input bit emerges at the output.

Example: 4-bit SISO Register

- Suppose input sequence = 1 0 1 1.
- After 1st clock → Q3 = 1.
- After 2nd clock → Q2 = 1, Q3 = 0.
- After 3rd clock → Q1 = 1, Q2 = 0, Q3 = 1.
- After 4th clock → Q0 = 1, Q1 = 0, Q2 = 1, Q3 = 1.
- Output (SO) = 1 (first bit emerges).

Load Operation (4 bit = 4 Clock Pulse)

Clock Pulse	Serial Input (SI)	Q3	Q2	Q1	Q0	Serial Output (SO)
0 (initial)	–	0	0	0	0	0
1	**1**	**1**	0	0	0	0
2	**0**	**0**	**1**	0	0	0
3	**1**	**1**	**0**	**1**	0	0
4	**1**	**1**	**1**	**0**	**1**	1

Read Operation (4 bit = 3 Clock Pulse)

Clock Pulse	Serial Output (SO)
4	1
5	0
6	1
7	1

Applications

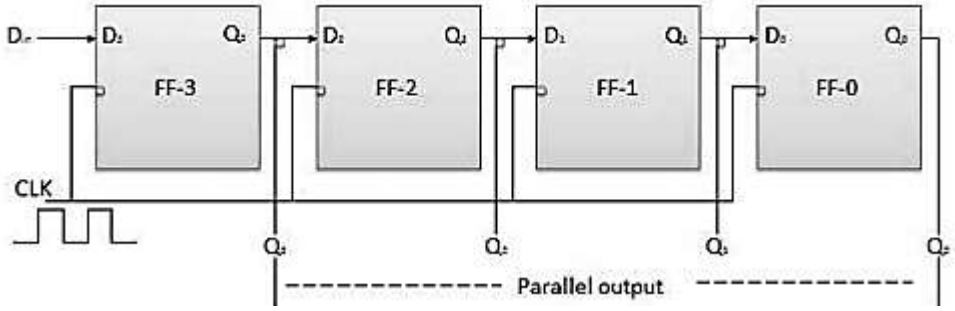
- **Serial communication**: Transmitting data bit by bit.
- **Delay elements**: Introduce controlled delays in digital circuits.
- **Data buffering**: Temporary storage in communication systems.

4.3.1.2. SIPO (Serial In Parallel Out)

The **Serial-In Parallel-Out (SIPO)** register accepts data **one bit at a time** (serial input) but makes the stored bits available **simultaneously at parallel outputs** after a series of clock pulses.

Structure

- Built using **D flip-flops** connected in series.
- **Serial Input (SI)** → enters the first flip-flop.
- **Clock** → shifts data through the chain.
- **Parallel Outputs (Q0, Q1, Q2, Q3 …)** → show the contents of all flip-flops at once.



Working Principle

- Data is applied **one bit at a time** to the first flip-flop
- On each **clock pulse**, data shifts to the next stage
- After **n clock pulses**, all bits are available at the outputs simultaneously

Truth Table (4-bit SIPO, Input Sequence = 1 0 1 1)

Clock Pulse	Serial Input (SI)	Q3	Q2	Q1	Q0	Parallel Output

0 (initial)	–	0	0	0	0	0000
1	1	1	0	0	0	0001
2	0	0	1	0	0	0010
3	1	1	0	1	0	0101
4	1	1	1	0	1	1011

Step-by-Step Operation (4-bit SIPO Example)

1. **Initial State**
    - All flip-flops are cleared: Q0 = Q1 = Q2 = Q3 = 0.
2. **Clock Pulse 1**
    - First input bit enters Q3.
    - Parallel output: 0001 (if input = 1).
3. **Clock Pulse 2**
    - Second input bit shifts into Q3, previous bit moves to Q2.
    - Parallel output: 0010 (if input = 0).
4. **Clock Pulse 3**
    - Third input bit enters Q3, earlier bits shift right.
    - Parallel output: 0101 (if input = 1).
5. **Clock Pulse 4**
    - Fourth input bit enters Q3, sequence fills all flip-flops.
    - Parallel output: 1011.

Applications

- **Serial-to-Parallel Conversion**: Receiving serial data and converting it for parallel processing.
- **Microprocessor Interfaces**: Collecting serial input from sensors or communication lines.
- **Data Storage**: Temporary holding of incoming serial streams.

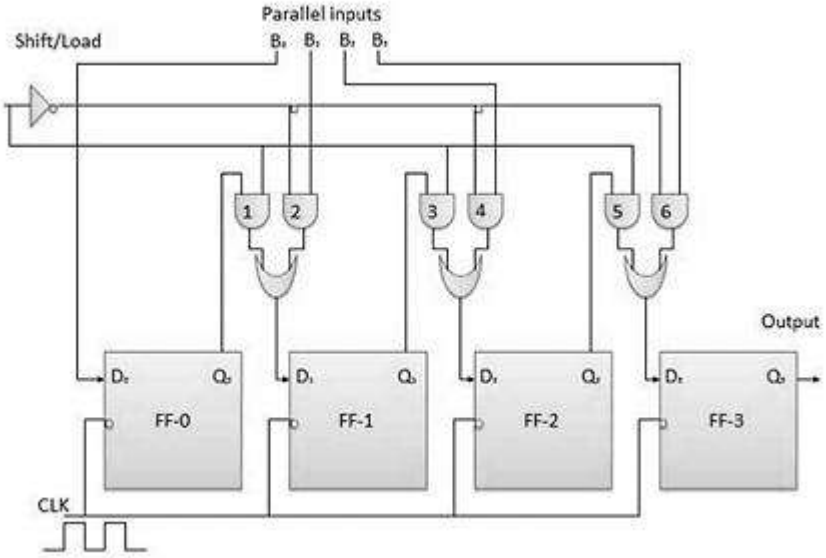
4.3.1.3. PISO (Parallel In Serial Out)

The **PISO register** allows data to be **loaded in parallel** (all bits at once) and then shifted out **serially** (one bit per clock pulse). It's essentially the reverse of the SIPO register.

Structure

- Built using **D flip-flops** with parallel load capability.

- **Parallel Inputs (P0, P1, P2, P3 …)** → load data simultaneously.
- **Clock** → controls shifting.
- **Serial Output (SO)** → data bits emerge one by one.
- **Control Signals**:
    - **Load/Enable** → determines whether to load parallel data or shift serially.



Working Principle

- Data is applied **simultaneously** to all flip-flops (parallel load)
- A **control signal (Load/Shift)** is used:
    - **Load = 1 → Data is loaded in parallel**
    - **Load = 0 → Data is shifted out serially**
- On each **clock pulse**, one bit is shifted out

Truth Table: 4-bit PISO Register (Data = 1011)

Clock Pulse	Load	Parallel Inputs (P3 P2 P1 P0)	Q3	Q2	Q1	Q0	Serial Output (SO)
0 (Load)	1	1011	1	0	1	1	–
1	0	–	0	1	1	–	1
2	0	–	1	1	–	–	0
3	0	–	1	–	–	–	1
4	0	–	–	–	–	–	1

Step-by-Step Operation (4-bit PISO Example)

1. **Parallel Load (Load = 1)**

- All inputs (P3, P2, P1, P0) are loaded simultaneously into the flip-flops.
- Example: Load 1011 → Q3=1, Q2=0, Q1=1, Q0=1.

2. **Shift Operation (Load = 0)**
   - On each clock pulse, the bits shift right.
   - The **serial output (SO)** produces one bit at a time.

Applications

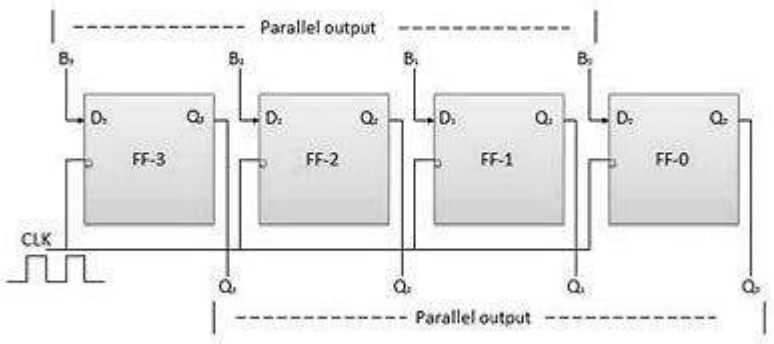
- **Parallel-to-Serial Conversion** for communication systems.
- **Data Transmission** over single-line channels.
- **Microprocessor Interfaces** where internal parallel data must be sent serially.

4.3.1.4. Parallel-In Parallel-Out (PIPO) Shift Register

The **PIPO register** is the most straightforward type of shift register. It allows data to be **loaded in parallel** (all bits at once) and also read out **in parallel**. Unlike SISO, SIPO, or PISO, there's no shifting involved — it's essentially a multi-bit storage element.

Structure

- Built using **D flip-flops** connected in parallel.
- **Parallel Inputs (P0, P1, P2, P3 …)** → load simultaneously into flip-flops.
- **Clock** → synchronizes loading.
- **Parallel Outputs (Q0, Q1, Q2, Q3 …)** → available immediately after loading.



Working Principle

- Data is applied **at all inputs at the same time**
- On the **clock pulse**, all bits are stored simultaneously
- Output is available **at the same time from all flip-flops**

Truth Table: 4-bit PIPO Register

Let's load parallel data = 1011.

Clock Pulse	Load Signal	Parallel Inputs (P3 P2 P1 P0)	Q3	Q2	Q1	Q0	Parallel Output
0 (initial)	–	–	0	0	0	0	0000
1 (Load=1)	1	1011	1	0	1	1	1011
2 (Hold)	0	–	1	0	1	1	1011

Operation

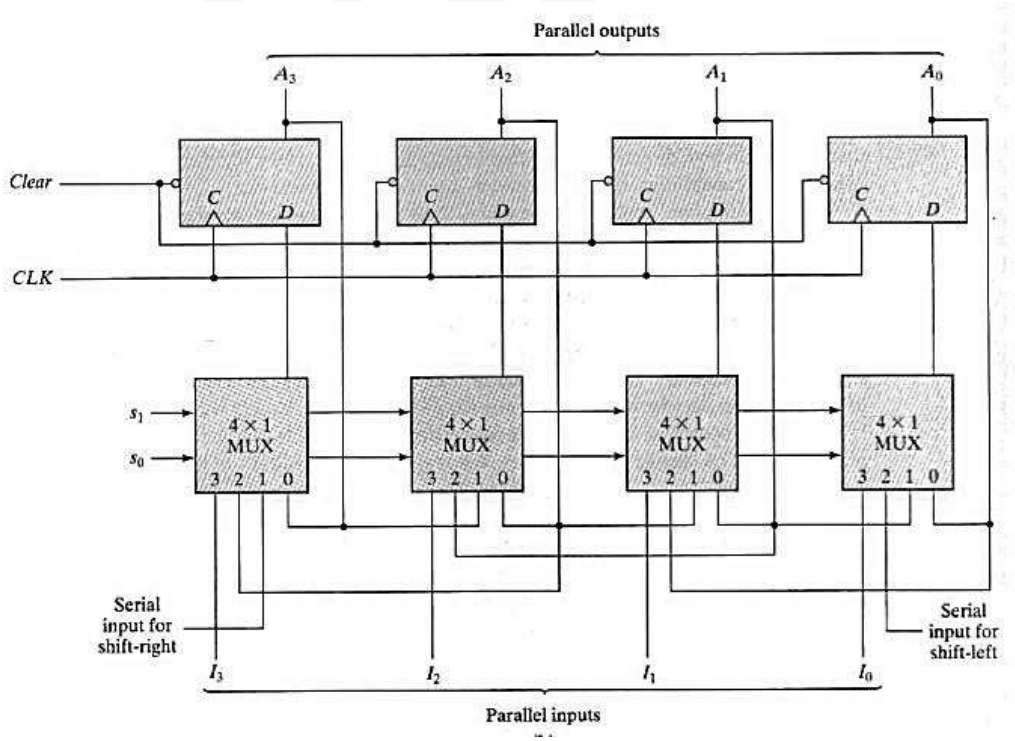
- **Load = 1** → parallel inputs are stored in flip-flops simultaneously.
- **Load = 0** → register holds the data (no shifting).
- Outputs are available **immediately in parallel**.

Applications

- **Fast data transfer** between processor and memory.
- **Temporary storage** of multi-bit data.
- **Registers in CPUs** (accumulator, instruction register, program counter).
- **Digital systems** where parallel data processing is required.

4.3.1.5. Universal Shift Register

The **Universal Shift Register** is the most versatile type of register. Unlike SISO, SIPO, PISO, or PIPO which have fixed modes, the universal shift register can perform **all four operations** depending on control inputs.



Structure

- Built using **D flip-flops** with multiplexers at their inputs.
- **Control signals** determine the mode of operation:
    - **Hold** (no change)
    - **Shift Left**
    - **Shift Right**
    - **Parallel Load**
- **Clock** synchronizes all operations.

Modes of Operation

Control Inputs	Operation	Description
00	Hold	Retains current data (no change).
01	Shift Right	Data moves one bit to the right; MSB replaced by serial input.
10	Shift Left	Data moves one bit to the left; LSB replaced by serial input.
11	Parallel Load	All flip-flops load data simultaneously from parallel inputs.

Working Principle

- Based on control inputs, the register selects operation
- Data can be:
    - Shifted **left or right**
    - Loaded **simultaneously**
    - Or kept unchanged
- Works with a **clock pulse**

Example: 4-bit Universal Shift Register

Suppose parallel input = 1011.

- **Parallel Load (11)** → Register stores 1011.
- **Shift Right (01)** → After one clock, contents become 0101.
- **Shift Left (10)** → After one clock, contents become 0110.
- **Hold (00)** → Contents remain unchanged.

Applications

- **Data Conversion**: Serial ↔ Parallel.
- **Bidirectional Communication**: Can shift left or right depending on protocol.
- **Microprocessor Systems**: Flexible data handling in ALUs and control units.

- **Digital Signal Processing**: Delay, buffering, and manipulation of bit streams.