TRIBHUVAN UNIVERSITY
Faculty of Management
Shanker Dev Campus

A lab report on Digital Logic

Submitted By:

Name: Ashesh Neupane

TU Exam Roll No: 16295/23

Level: Bachelor

Semester: Second

Batch: 2080

Subject: Digital Logic

Submitted To:

Narayan GC

Faculty Member of BIM

Verified By :

Table of Contents

Lab-1: Logic Gates

Objective: To investigate the input and output of AND gate, OR gate, NOT gate, NAND gate, NOR gate, XOR gate and X-NOR gate.

Lab Requirements: a) Bread Board
b) Integrated Circuits (IC)
c) Power Supply
d) Wire and Wire cutter
e) LED

ICs:

7400 → NAND
7402 → NOR
7404 → NOT
7408 → AND
7432 → OR
7486 → X-OR

Basic Theory

(1) **AND Gate:** The AND gate is a logic gate that has two or more inputs but only one type of output. The output of AND gate is high when all inputs are high and output is low if any of the input is low.

Boolean Expression: $X = A \cdot B$

Circuit Diagram:

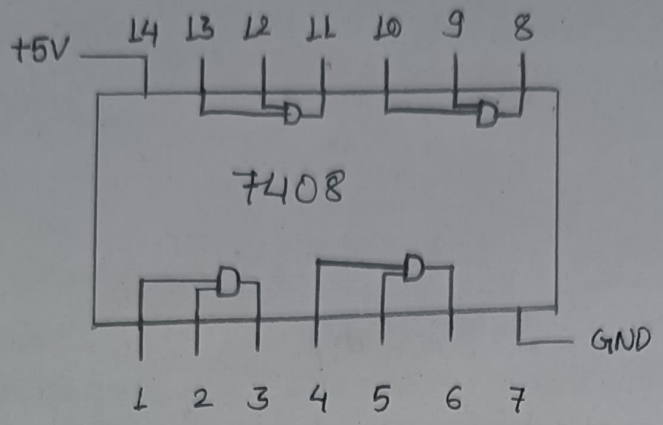
Fig: AND gate

Observation Table:

Inputs		Outputs
A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

Fig: Truth table

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of the AND gate, LED is used which indicates true(1) or false(0). True refers the LED is lighted and false refers LED is not lighted. To prove AND, bread board is used. when inputs are connected to the bread board through connecting wire, the LED lights when both inputs are high and doesnot light in every other condition.

Conclusion:

→ Here, the logic operation of AND gate was verified using the lab equipments.

(2) OR Gate: An OR gate is a logic gate that has two or more inputs but only one output. Output of OR gate is low when all the inputs are low otherwise output is high.

Boolean Expression: $X = A+B$

Circuit Diagram:

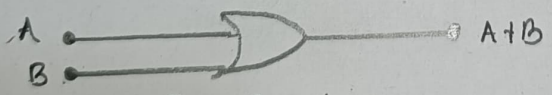
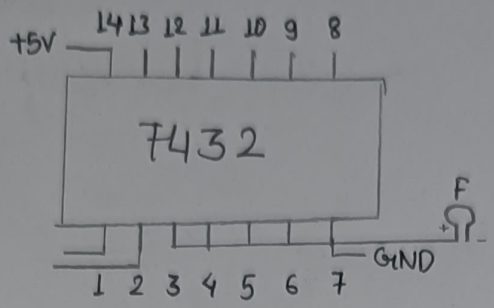


Fig: OR gate

Observation Table:

Inputs		Output
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of OR gate, LED is used which indicates true (1) or false (0). True refers the LED is lighted and false refers the LED is not lighted. To prove OR gate, bread board is used. When both inputs are connected to board using connecting wire, the LED doesnot light when both inputs are low and lights when all or one inputs are high.

Conclusion:

→ Hence, the logic operation of OR gate was verified using the lab equipment.

(3) NOT Gate: The NOT gate is the simplest of all logic gates. It has only one input and one output where the output is the opposite of input.

Boolean Expression: $X = \bar{A}$

Circuit Diagram:

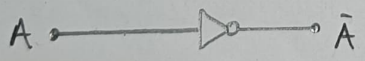


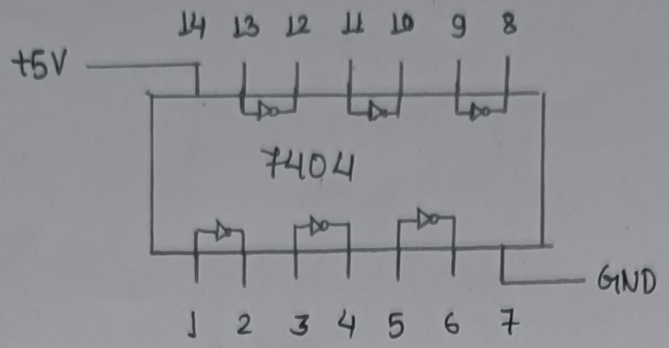
Fig: NOT gate

Observation Table:

Input (A)	Output ($\bar{A}$)
0	1
1	0

Fig: Truth table

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of NOT gate. LED is used which indicates true (1) and false (0). True refers the LED is lighted and false refers the LED is not lighted. To prove NOT gate, bread board is used. When inputs are connected to board using connecting wire, the LED lights when input is low and doesnot light when the input is high.

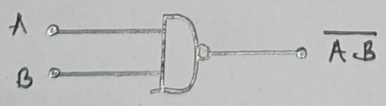
Conclusion:

→ Hence, the logic operation of NOT gate was verified using lab equipments.

(4) NAND Gate: Combination of AND gate and NOT gate results in the formation of NAND gate. It is universal gate. NAND gate is a gate where output is oppositive of AND gate. In NAND gate output is high if any one of the input is low and output is high when all the inputs are high.

Boolean Expression: $X = \overline{A.B}$

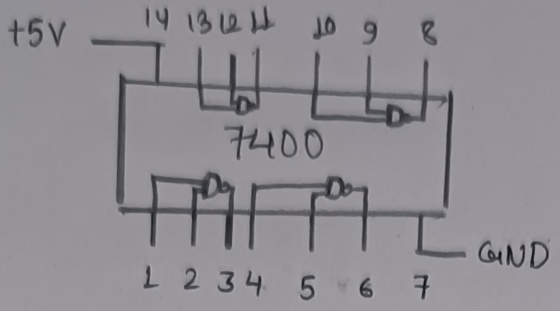
Circuit Diagram:



Observation Table:

Inputs		Output
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of NAND gate, LED is used which indicates true (1) or false (0). True refers the LED is lighted and false refers the LED is not lighted. To prove bread board is connected to input using connecting wires, the LED doesnot light when both inputs are high and lights in every other cases.

Conclusion:

→ Hence, the logic operation of NAND gate was verified using lab equipment.

(5) NOR Gate: NOR gate can be defined as the combination of OR gate and NOT gate. This is also called universal gate. The NOR gate is designed in such a way that the output of OR gate is the input of NOT gate.

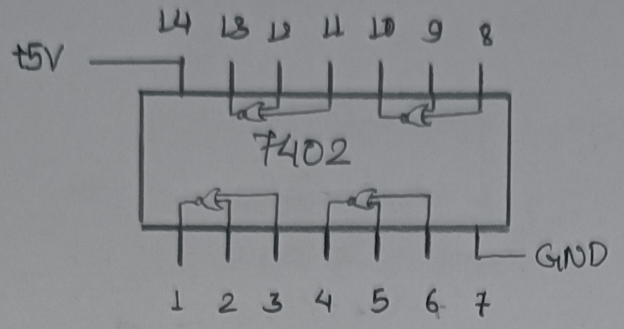
Boolean expression: $X = \overline{A+B}$

Circuit Diagram:

Observation Table:

Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of NOR gate, LED is used which indicates true (1) or false (0). True is represented by lighting of LED and false is represented when the LED is not lighted. To prove NOR gate, bread board is used. When both the inputs are connected to board through connecting wires, the LED is lighted when both inputs are low and is not lighted in every other condition.

Conclusion:

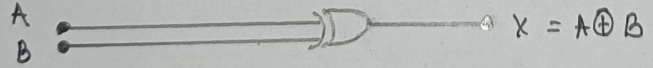
→ Hence, the logic operation of NOR gate was verified using the lab equipment.

(6) X-OR Gate: It is a type of gate which gives low output when both the inputs are same (i.e. 0&0 or 1&1) and gives high output in other condition. It has maximum two inputs only.

Boolean Expression: $X = \bar{A}B + A\bar{B}$

$= A \oplus B$

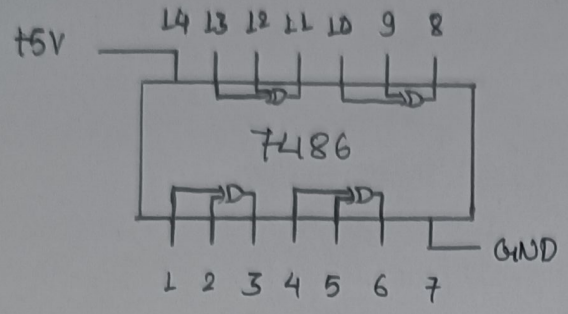
Circuit Diagram:



Observation Table:

Inputs		Output
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Experiment Diagram:



Discussion:

→ In lab, to prove the statement of XOR gate, LED is used which indicates true (1) or false (0). True is represented by lighting of the LED and false is represented when the LED is not lighted. To prove, X-OR gate, bread board is used when both inputs are connected to board through connecting wire. The LED is not lighted when both inputs are same and is lighted in every other condition.

Conclusion:

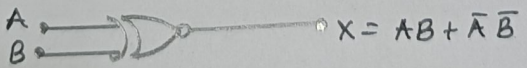
→ Hence, the logic operation of X-OR gate was verified using the lab equipment.

(7) X-NOR Gate: It is a type of gate which gives high output when both inputs are same and gives low output in every other condition.

Boolean Expression: $X = A.B + \bar{A}\bar{B}$

$= A \odot B$

Circuit Diagram:



Observation Table:

Inputs		Output
A	B	$A.B + \bar{A}\bar{B}$
0	0	1
0	1	0
1	0	0
1	1	1

Lab-2: Adders and Subtractors

Objective: To verify the operations of Half Adder, Full Adder, Half Subtractor and Full Subtractor.

Lab Requirements:
a) Bread Board
b) Integrated Circuit (IC)
c) Power Supply
d) Wire and wire cutter
e) LED

Basic Theory

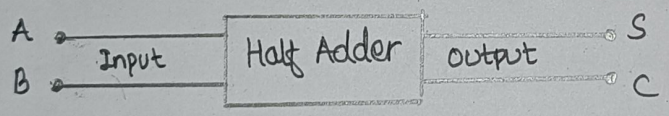
(i) **Half Adder:** The half adder is a combinational logic circuit which is used to perform addition of two input bits and hence provide the output. It has two inputs A and B and two outputs Sum (S) and Carry (C).

Boolean Expression: $Sum = A \oplus B$

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

$carry = AB$

Block Diagram:

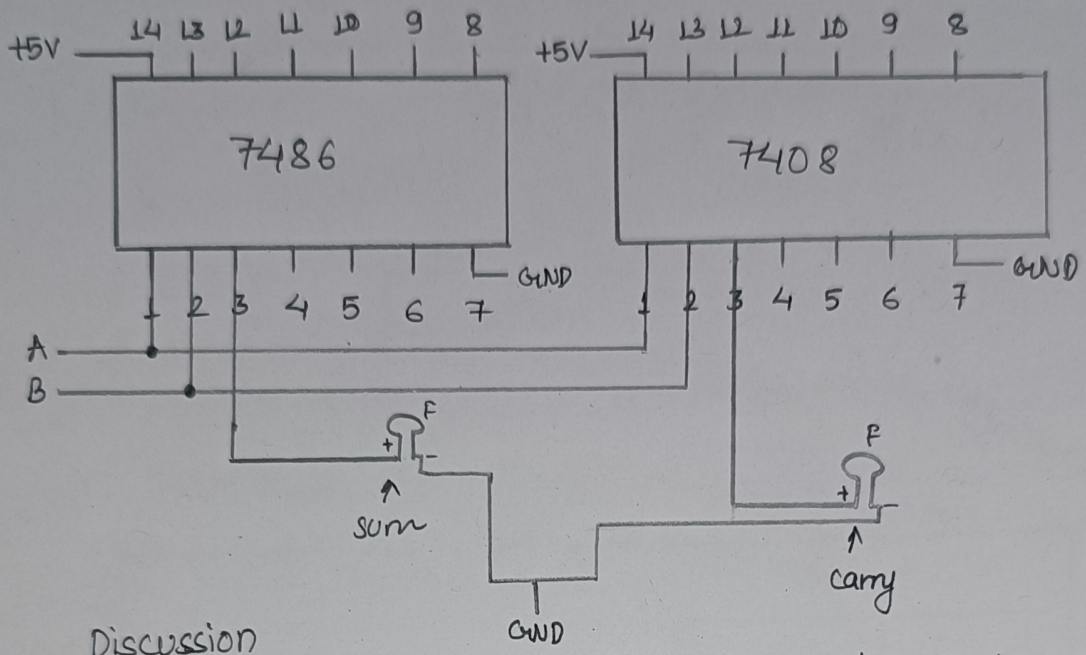


Observation Table:

Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig: Observation Table

Experiment Diagram



Discussion

→ In lab, to prove the statement of Half Adder, LED is used which indicates true (1) and false (0). True refers the LED is lighted and false refers the LED is not lighted. To prove, the operation of half adder, when both inputs are connected to bread board in XOR and AND gates by the help of connecting wires, the combination follows the table as in the truth table where '1' represents the lighting of LED and 0 refers to when LED is not lighted.

Conclusion

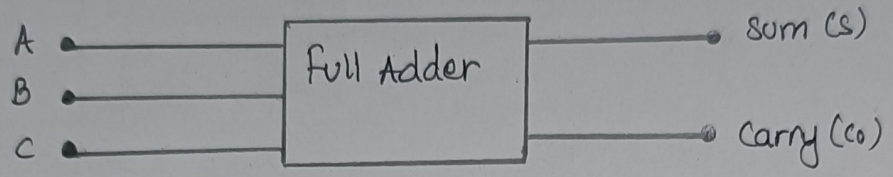
→ Hence, the operation of half adder was verified using lab equipments.

(2) Full Adder: Full Adder is a combinational circuit It has three input and two output. One is sum and another is carry. It performs arithmetic addition with previous carry.

Boolean Expression: Sum = $A \oplus B \oplus C$

Carry = $(A \oplus B) + A \cdot B$

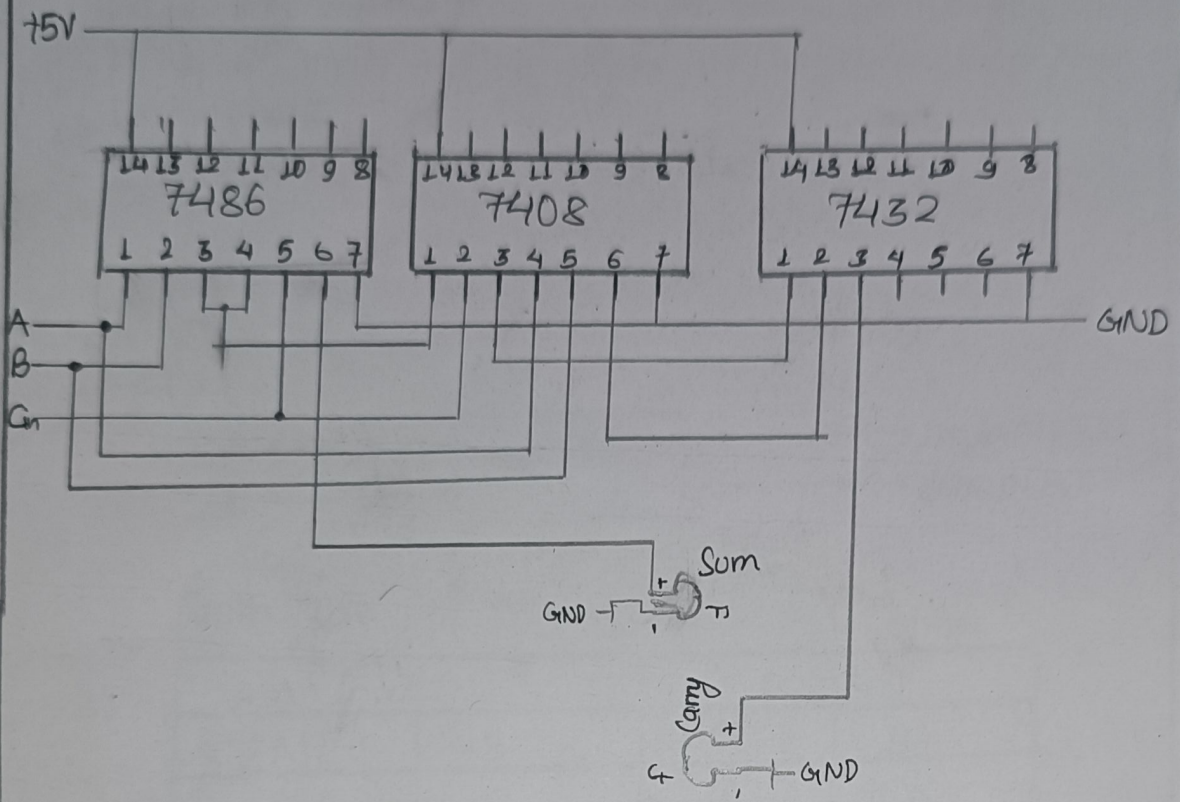
Block Diagram:



Truth Table:

Input			Output	
A	B	C	S	C_0
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

Experiment Diagram:



Discussion:

→ In lab, a full adder can be demonstrated using LEDs to represent the outputs. A full adder takes three inputs: A, B and carry-in (Cin), and produces two outputs: Sum (S) and Carry. The output indicates whether the total sum of the inputs is 1 or 0, while the carry-out indicates if there is an overflow (1) or not (0).

Conclusion:

→ Hence, the operation of full adder was verified using lab equipments.

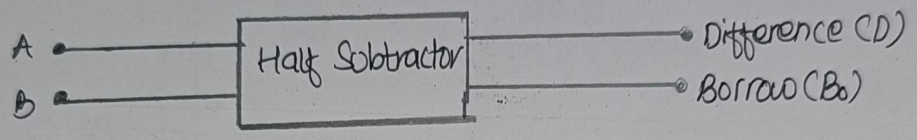
(3) Half Subtractor: A half subtractor is a combinational circuit that performs subtraction of two single-bit binary numbers. It has two inputs and two outputs: the Difference (D) and the Borrow (B_o).

Boolean Expressions: $D = \bar{A} \cdot B + A \cdot \bar{B}$

$= A \oplus B$

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

Block Diagram:



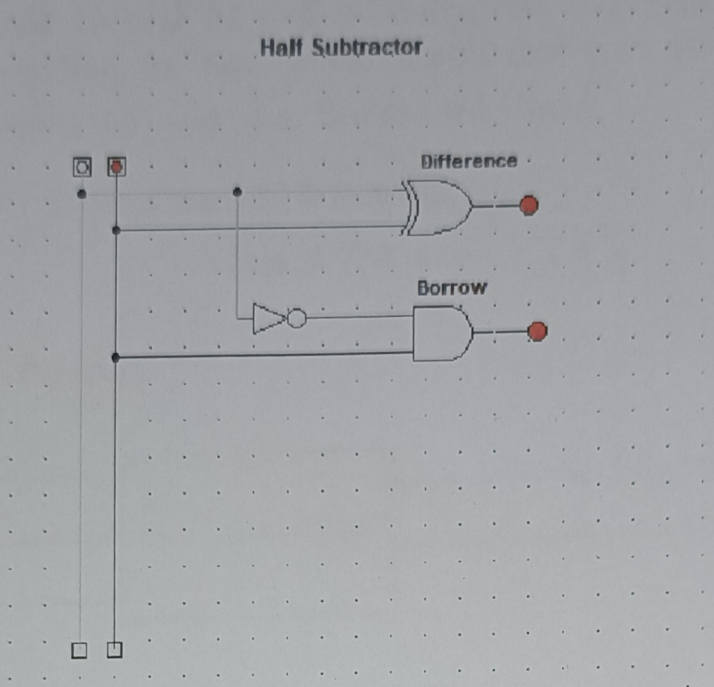
Truth Table:

Input		Output	
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

$= A \oplus B$

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



In the above experiment, a half subtractor circuit is demonstrated. A half subtractor is a combinational logic circuit that is used to subtract two single-bit binary numbers. It produces two outputs: the "Difference" and the "Borrow".

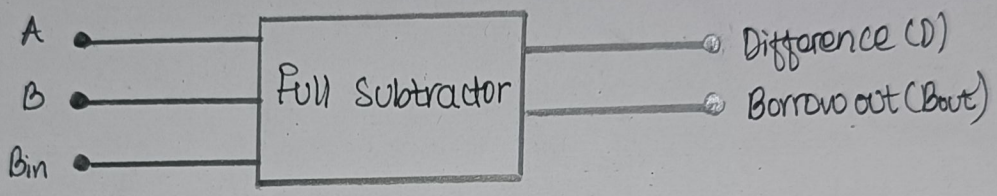
- Inputs: The half subtractor takes two inputs, typically labeled as A and B.
- Difference: The XOR gate is used to compute the difference between the two inputs. It outputs a '1' if the inputs are different and '0' if they are the same.
- Borrow: The AND gate and NOT gate are used to determine if a borrow is needed. It outputs a '1' when A is '0' and B is '1', indicating that a borrow is required from the next higher bit.

(4) Full subtractor: A full subtractor is a combinational circuit that performs subtraction of three single-bit binary numbers: two minuends (A and B) and a borrow bit (Bin) from a previous operation. It has three inputs and two outputs: the Difference (D) and the Borrow out (Bout).

Boolean Expressions: $D = A \oplus B \oplus Bin$

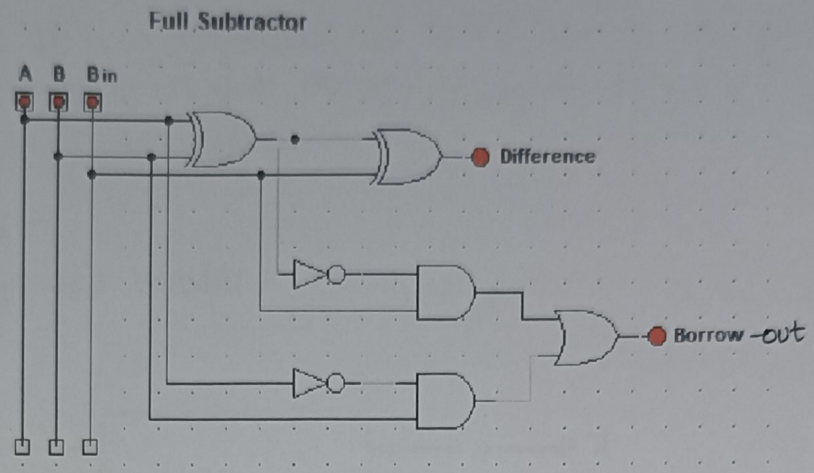
$$Bout = \bar{A}B + Bin \cdot \overline{A \oplus B}$$

Block Diagram:



Truth Table:

Input			Output	
A	B	Bin	D	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



In the above experiment, a full subtractor circuit is demonstrated. A full subtractor is a combinational logic circuit used to subtract three single-bit binary numbers: A, B and a borrow-in (Bin). It produces two outputs: the "Difference" and the "Borrow-out."

- Inputs: The full subtractor takes three inputs:
  - A - The minuend
  - B - The subtrahend
  - Bin - The borrow-in from the previous lower significant bit.
- Difference: The XOR gates compute the difference between the inputs A, B, and Bin. It outputs '1' if the total number of '1' inputs is odd and '0' if the number is even.
- Borrow-out: The AND gates and OR gate determine if a borrow is required. The borrow-out is '1' when either A is '0' and B or Bin is '1', indicating that a borrow is needed from the next higher bit.

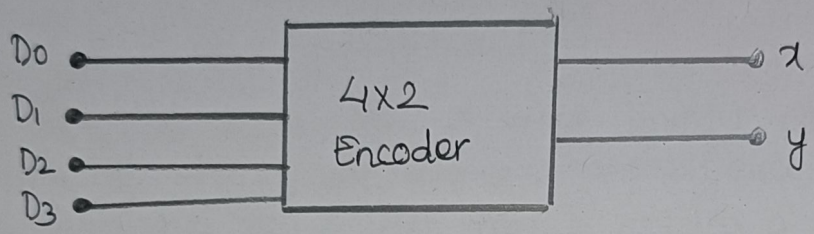
Lab-3: Decoder and multiplexers

Basic Theory

(1) Encoder: It is a combinational circuit and is used for data compression in data transmission system. It has '2^n' input line and 'n' output line.

Example:

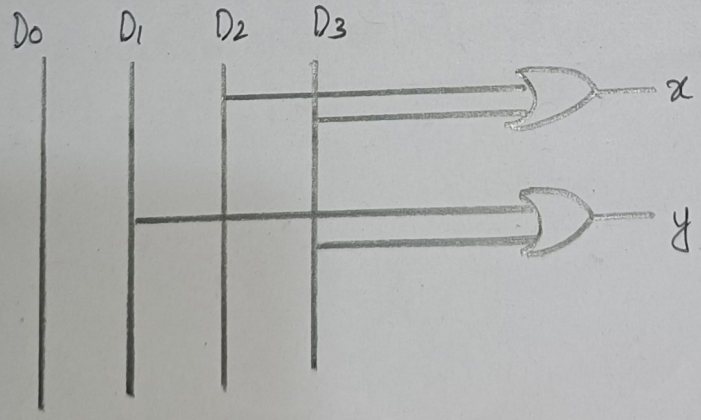
- Design 4x2 Encoder



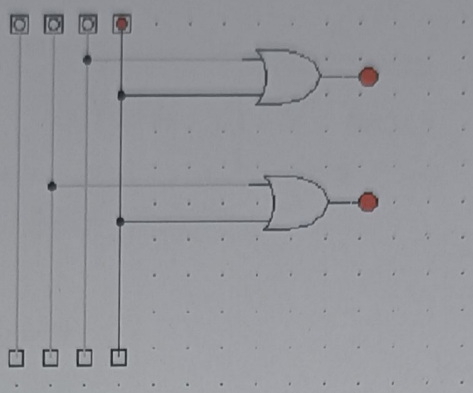
Operational Table:

Input				Output	
D_0	D_1	D_2	D_3	x	y
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Logic Circuit Diagram:



Experiment Diagram of 4*2 Encoder



In the above experiment, a 4*2 encoder circuit is shown. A 4*2 encoder is a combinational circuit that compresses 4 input lines into 2 output lines. It assigns a unique binary code to each active input signal. The circuit diagram illustrates the use of logic gates to determine the output based on the active input signal.

- The encoder has 4 input signals (let's assume D_0, D_1, D_2 and D_3)
- Based on which input is high, the circuit will produce a 2-bit binary output.

The OR gates and their connections determine how the input signals influence the output lines, ensuring that only the active input is encoded into the correct binary representation at the output.

(2) Decoder: It is a combinational circuit and it perform reverse operation of encoder. It has 'n' input line and maximum of '2^n' output line.

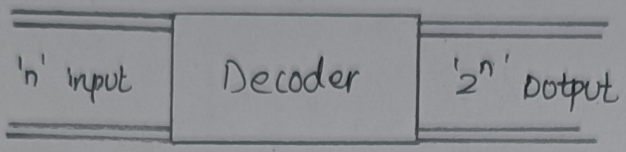
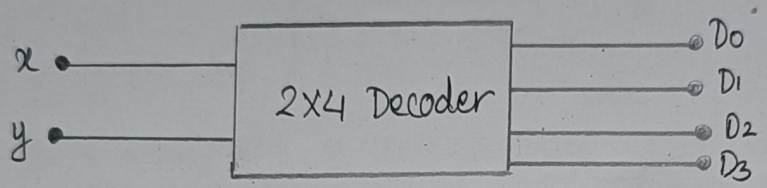


Fig: Block Diagram

Example:

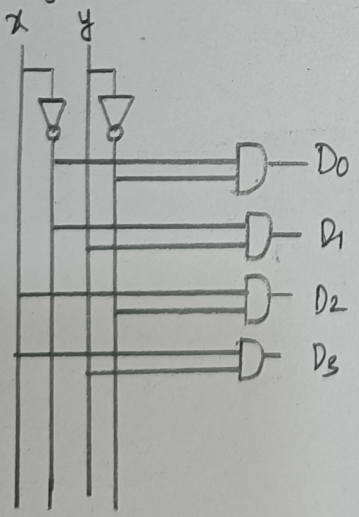
· Design 2x4 Decoder.



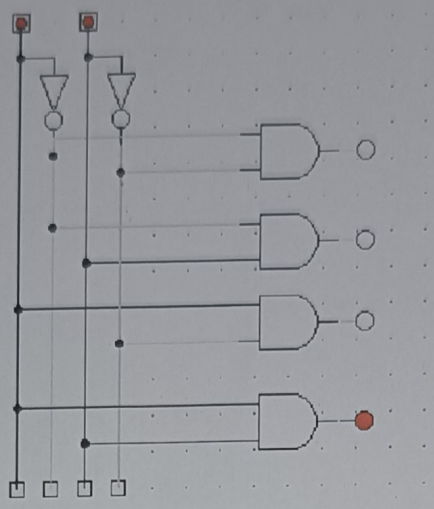
Operational Table

Input		Output			
x	y	D_0	D_1	D_2	D_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

Logic Circuit Diagram



Experiment Diagram of 2*4 Decoder

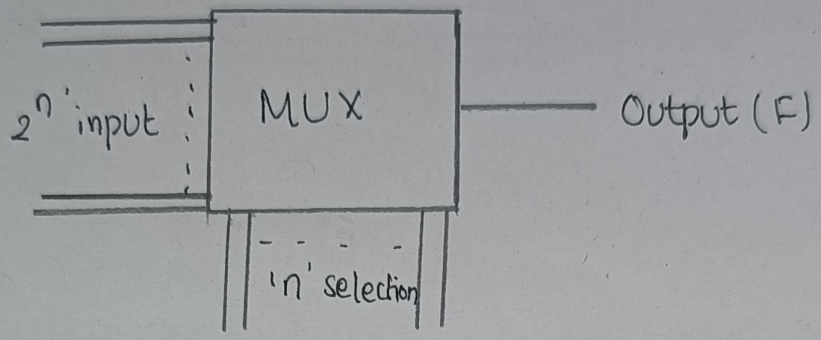


In the above experiment, a 2*4 decoder circuit is demonstrated. A 2*4 decoder is a combinational logic circuit that decodes 2 input lines into 4 unique output lines. Each output line corresponds to one of the possible combinations of the two inputs.

- Inputs: The circuit has two input lines. Based on their binary values, one of the four outputs is activated.
- Outputs: The decoder produces four outputs. Only one of these outputs will be '1' at a time, corresponding to the binary input value, while the others remain '0'.
- Logic: The circuit uses a combination of AND gates and NOT gates to determine which output to activate. The NOT gates invert the input signals where needed, allowing each AND gate to represent a specific binary input combination.

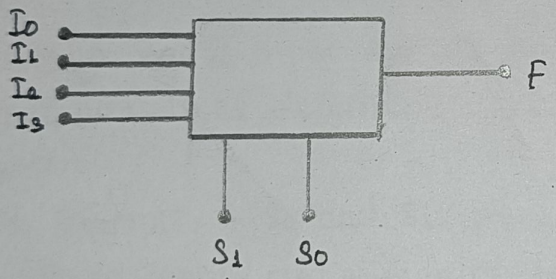
(3) Multiplexer (MUX) : Multiplexing means combined multiple channel into single composite channel. MUX is the combinational circuit. It has '2^n' input line, 'n' selection line and single output line. The inputs are selected on the basis of selection line. This is also called selector.

Block Diagram:



Example:

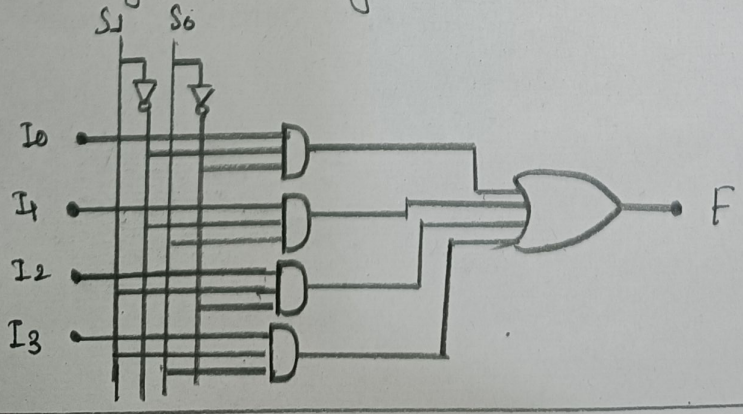
- Design 4x1 MUX



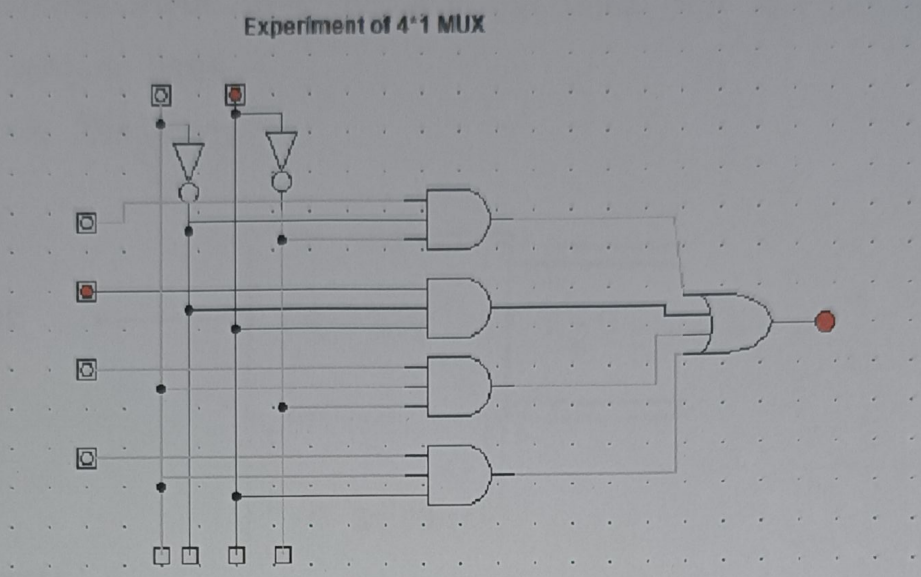
Operational Table

Selection		output
S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Logic Circuit Diagram



Name: ASHESH NEUPANE | TU Exam Roll No: 16295/23



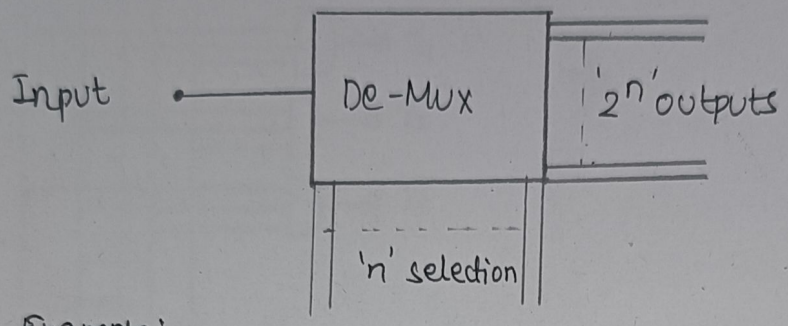
In the above experiment, a 4*1 multiplexer circuit is shown. A 4*1 MUX is a combinational circuit that selects one of the 4 input data lines and forwards it to a single output line based on the select lines.

- The MUX has 4 input signals and 2 select lines.
- The select lines determine which input is connected to the output.
- For example, if S_0 and S_1 are set to 0 and 1, the input I_1 will be selected and forwarded to the output.

The experiment diagram uses AND gates and OR gates to control which input line is routed to the output based on the select lines combination. This structure ensures that only one input reaches the output, making the MUX a fundamental part of digital selection logic.

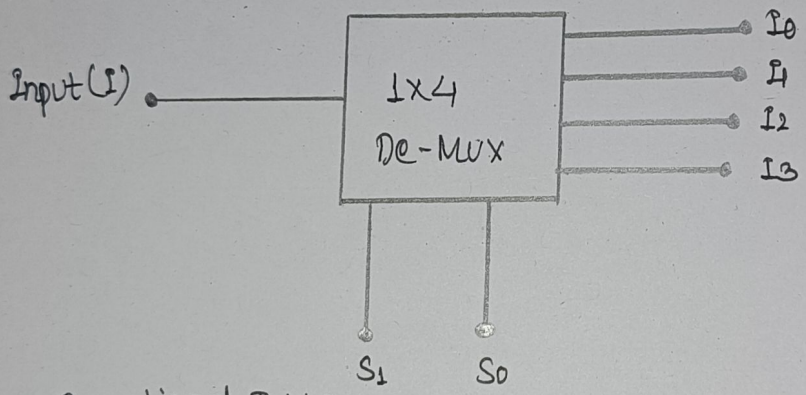
(4) De-Multiplexer (De-Mux) : De-Mux is a combinational circuit and it performs reverse operation of multiplexer it has single input line and selection lines and maximum of 2^n output lines.

Block Diagram :



Example:

- Design 1x4 De-Mux



Operational Table

Selection		Output
S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

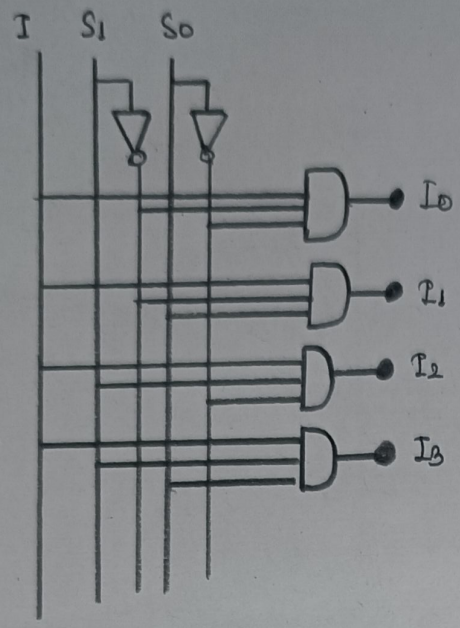
$I_0 = I . \overline{S_1} . \overline{S_0}$

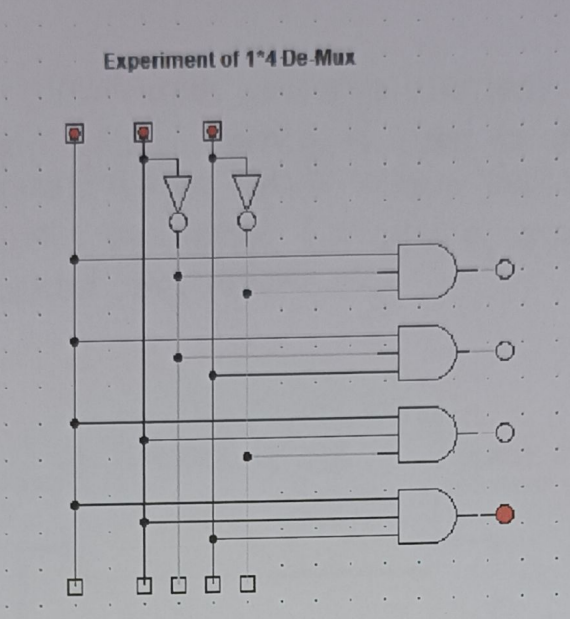
$I_1 = I . \overline{S_1} . S_0$

$I_2 = I . S_1 . \overline{S_0}$

$I_3 = I . S_1 . S_0$

Logic Circuit Diagram:





In the above experiment, a 1*4 demultiplexer circuit is illustrated. A 1*4 De-MUX is a combinational circuit that takes a single input and distributes it to one of the 4 outputslines based on the values of the selection lines.

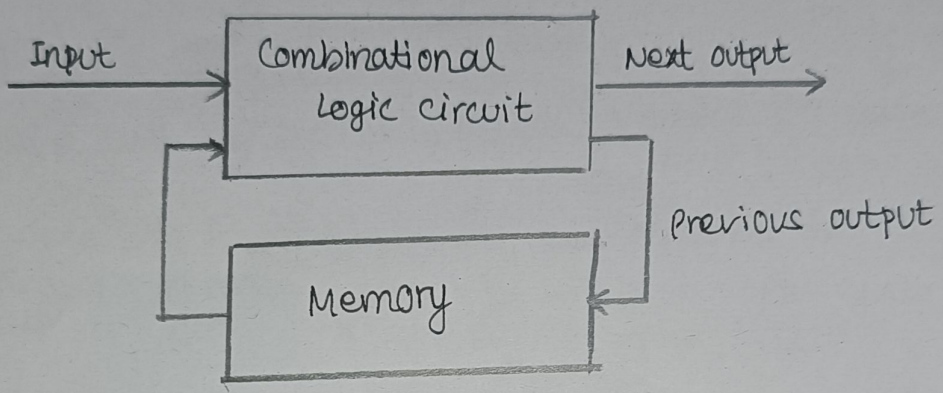
- The De-MUX has a single input and two selection lines.
- The selection lines determine which of the 4 output lines will carry the input signal.

The circuit diagram uses AND gates in conjunction with the selection lines to ensure that the input is directed to the appropriate output based on the select line configuration. This allows the De-MUX to perform the function of distributing data from a single source to multiple destinations.

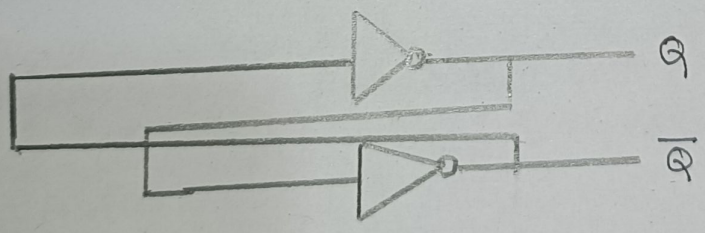
Lab-4 : Sequential circuits (Flip-Flop)

Basic Theory

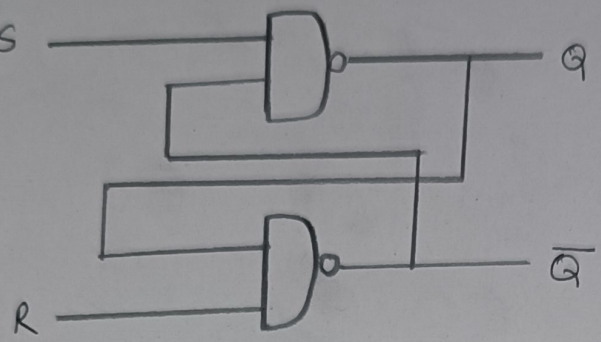
Combinational circuit with memory element is called sequential logic circuit. Memory is used to store previous output. In sequential logic circuit, output depends on present input and previous ~~inpu~~ output. Examples of sequential logic circuit are counter, shift register etc.



- Bi-stable Memory Device: It is a simplest sequential circuit that store 1 bit of binary data. It is constructed by using two NOT gates connected in a loop so that output of the one NOT gate becomes the input of another NOT gate.

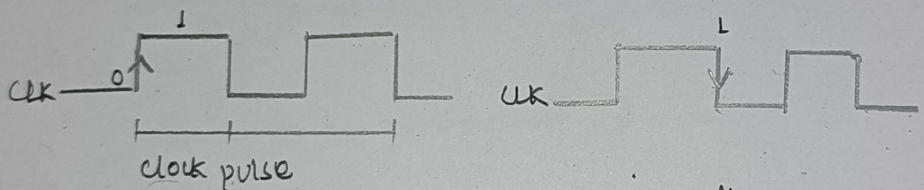


- Latch: In Bi-stable memory device, there is no any user inputs. To add input to the circuit, we simply replace NOT gate with universal gates and the constructed circuit is called Latch.



Types of Latch are S-R Latch, D-Latch, J-K Latch & T Latch

- Clock: Clock is a timing signal used to synchronize the operations of circuits and components within a digital system. Its types are: positive clock and negative clock.

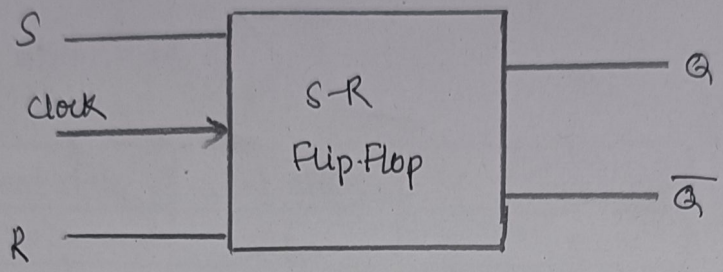


- positive clock
- Rising edge clock triggering
- Negative clock
- Falling edge clock triggering

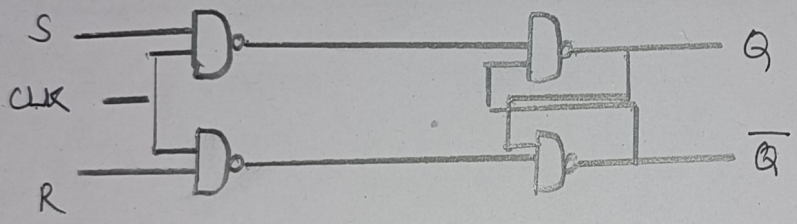
- Flip-Flop: Latch with clock enabled is called flip flop. It is a bi-stable memory device that store 1 bit of binary data. Flip-Flop is a basic building block of any sequential circuit. Types of flip-flop are S-R FlipFlop, D-FlipFlop, J-K FlipFlop & T-FlipFlop.

Explanation for each type of Flip-Flop

→ S-R Flip Flop: "S" stands for set and "R" stands for Reset. It is constructed cross coupling of NAND or NOR gates. When clock = 0, it doesnot function its operation and remain unchanged state. When clock = ?, it perform its function.



Logic Circuit Diagram



Characteristics Table

Clock	R	S	Qp	Qn
0	X	X	X	No change
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	Race condition
1	1	1	1	Race condition

Characteristics Equation

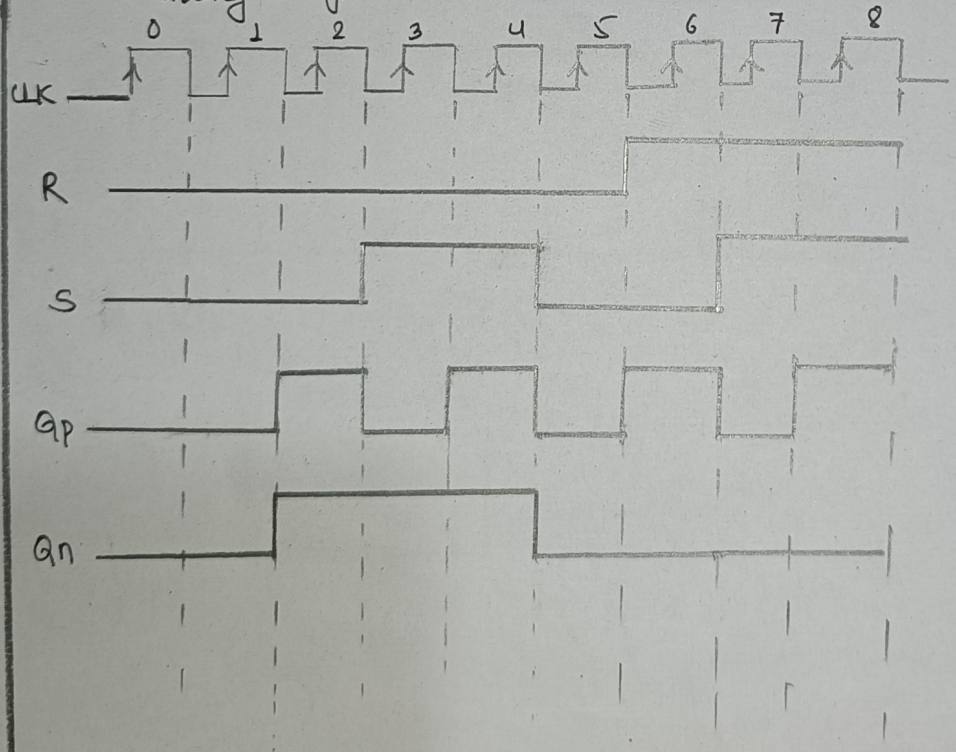
Qn \ S.Qp	$\bar{S}\bar{Q}p$	$\bar{S}.Qp$	$S.Qp$	$S.\bar{Q}p$
$\bar{R}$		1	1	1
R			X	X

$$Qn = S + \bar{R}Qp$$

Excitation Table

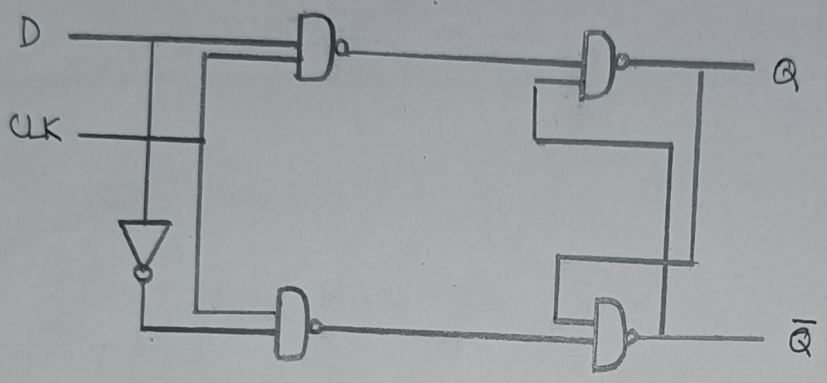
Present state	Next state	FF input	
Qp	Qn	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Timing Diagram



→ D-Flip Flop: D stands for data. A D flip flop is a digital memory circuit that captures and stores the value of the input (D) on the rising or falling edge of a clock signal, outputting it as Q until the next clock cycle. It is most commonly used for data storage and synchronization in digital circuits.

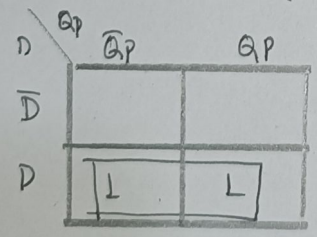
Logic Circuit Diagram



Characteristics Table

Clock	D	Q_P	Q_n
0	X	X	No change
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Characteristics Equation

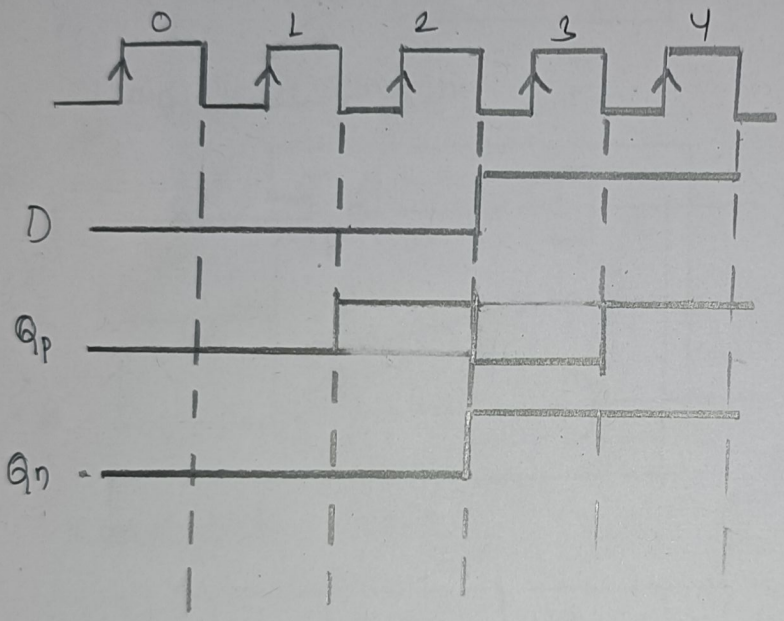


$Q_n = D$

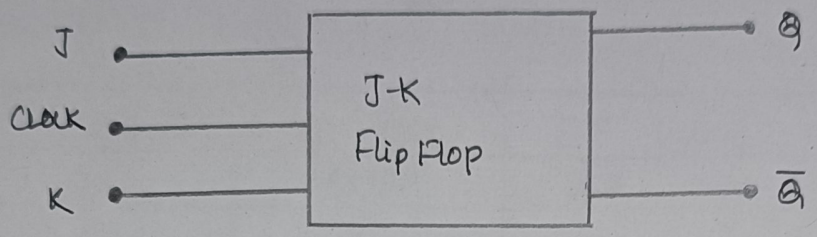
Excitation Table

Present state	Next state	f.f input
Q_p	Q_n	D
0	0	0
0	1	1
1	0	0
1	1	1

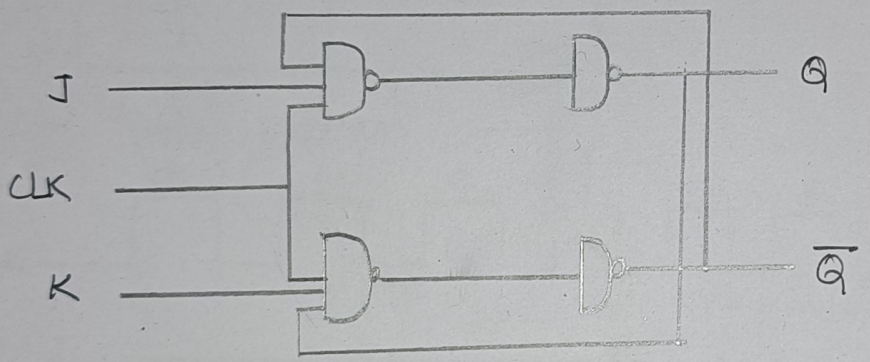
Timing Diagram



→ J-K Flip-Flop: A J-K flipflop is a versatile digital memory circuit that functions as a bistable multivibrator, capable of toggling between two states based on input signals. It has two inputs, J and K, allowing for various operations like setting, resetting, and toggling, making it widely used in sequential circuits. J & K stands for indirect set and indirect reset.



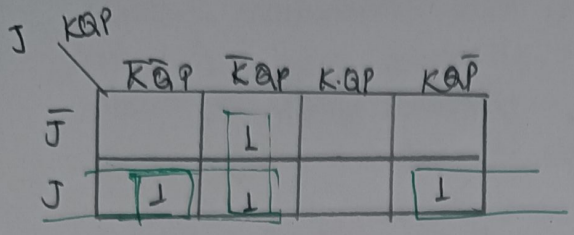
Logic Circuit Diagram



Characteristics Table

Clock	J	K	Qp	Qn
0	x	x	x	No change
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Characteristics Equation

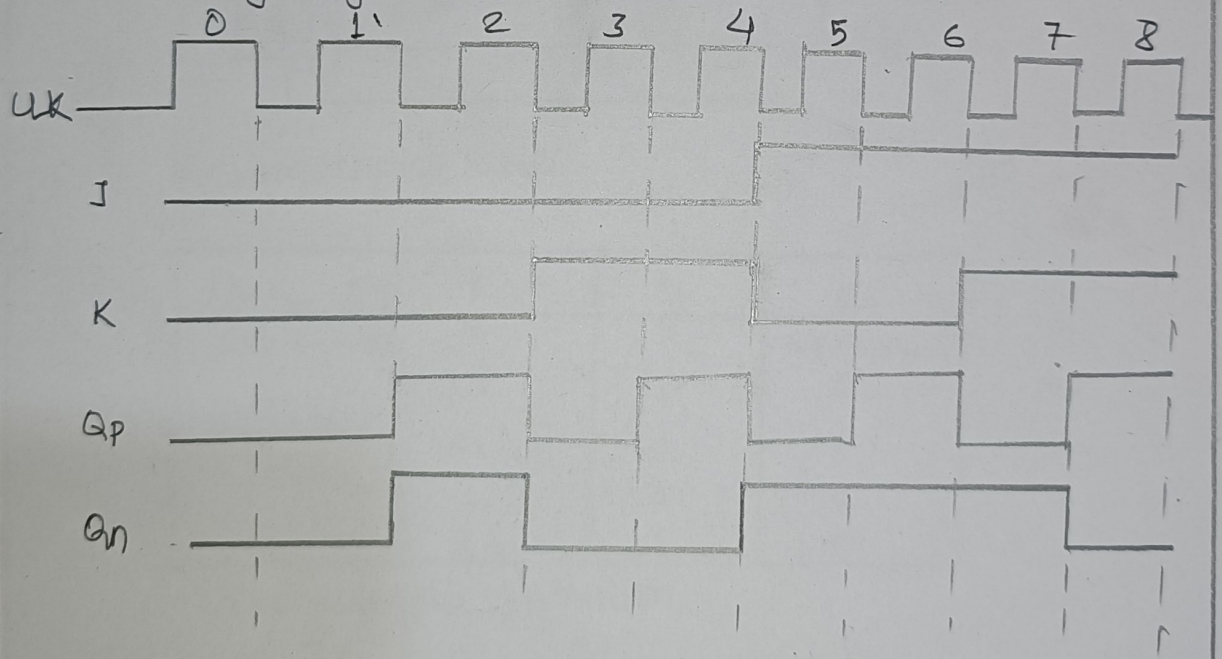


$$Qn = J\bar{Q}P + \bar{K}QP$$

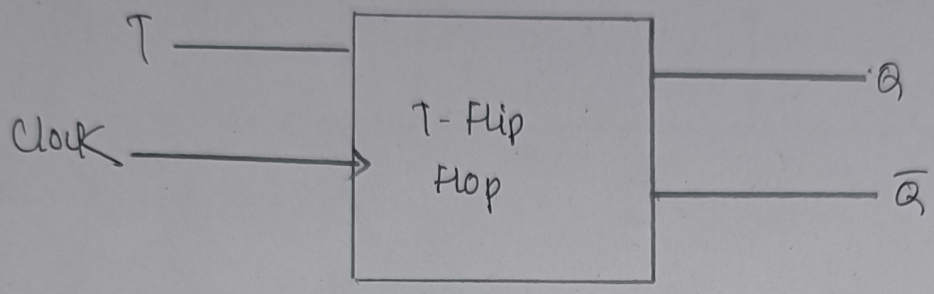
Excitation Table

Present state	Next state	F-F Input	
Qp	Qn	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

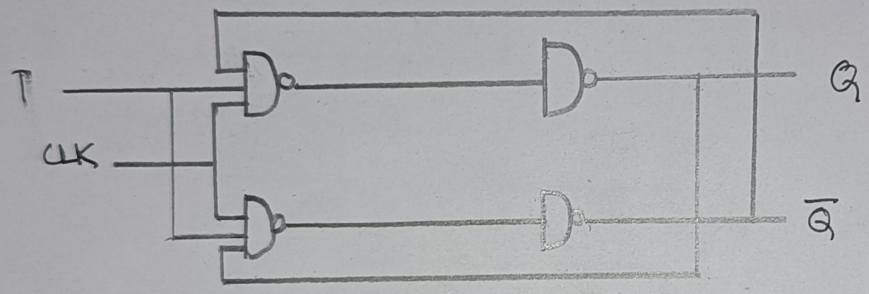
Timing Diagram



→ T - Flip Flop: A T flip-flop, or toggle flip-flop, is a type of bistable multivibrator that changes its output state with each clock pulse when its T (toggle) input is high. It simplifies circuit design by providing a single input for toggling between states, making it ideal for counters and frequency dividers.



Logic Circuit Diagram



Characteristics Table

Clock	T	Q_P	Q_n
0	X	X	No change
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Characteristics Equation

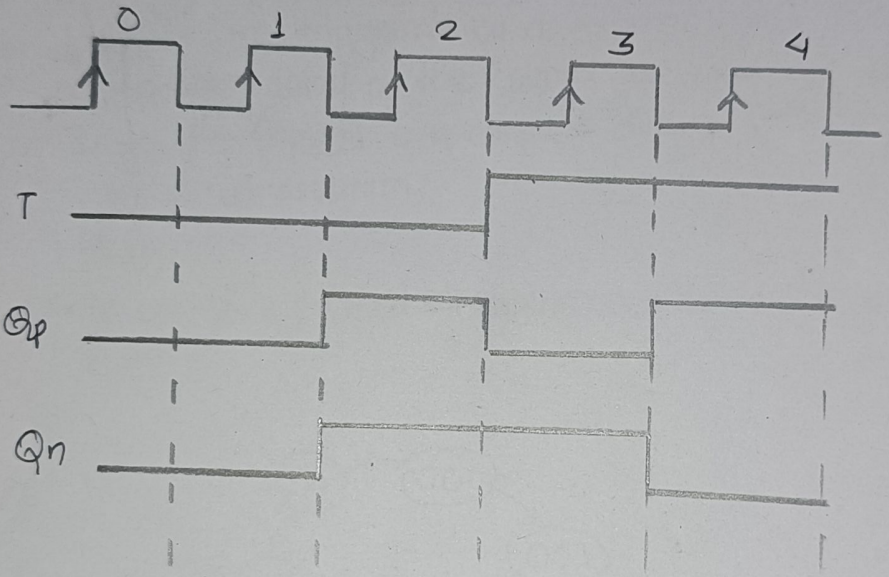
	$\bar{Q}_P$	Q_P
$\bar{T}$		1
T	1	

$$Q_n = T.\bar{Q}_P + \bar{T}.Q_P$$

Excitation Table

Present State	Next State	FF inputs
Q_p	Q_n	T
0	0	0
0	1	1
1	0	1
1	1	0

Timing Diagram



Lab-5: Counters

Basic Theory

Counter is a sequential circuit and it is used to count predefined sequence of the state on the application of clock pulse.

It is constructed by using flip flop and it is the basic elements of control unit for generating timing sequences. Counter are classified into synchronous counter & asynchronous counter.

(1) Synchronous Counter: In synchronous counter, all of the flip-flop are triggered by a common clock pulse. The propagation delay of the counter is equal to delay of single flip flop. It is a complex design procedure and logic circuit diagram.

Example:

- Design 3-bit (mod 8) synchronous up counter by using T-flip flop.

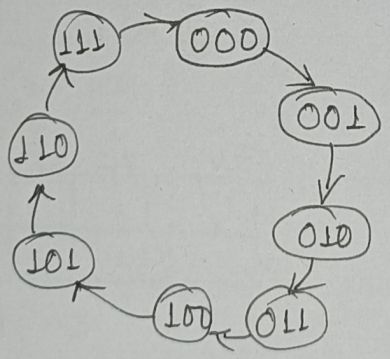
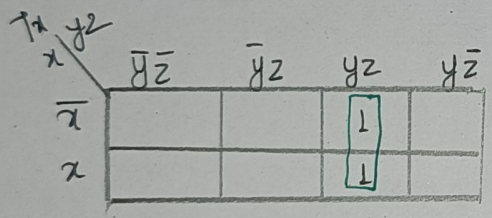


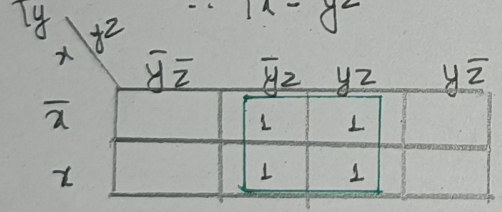
Fig: State Diagram

State Table:

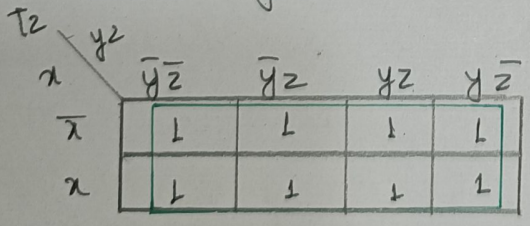
Present state			Next state			Flip-Flop input		
x	y	z	x	y	z	Tx	Ty	Tz
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$\therefore Tx = yz$

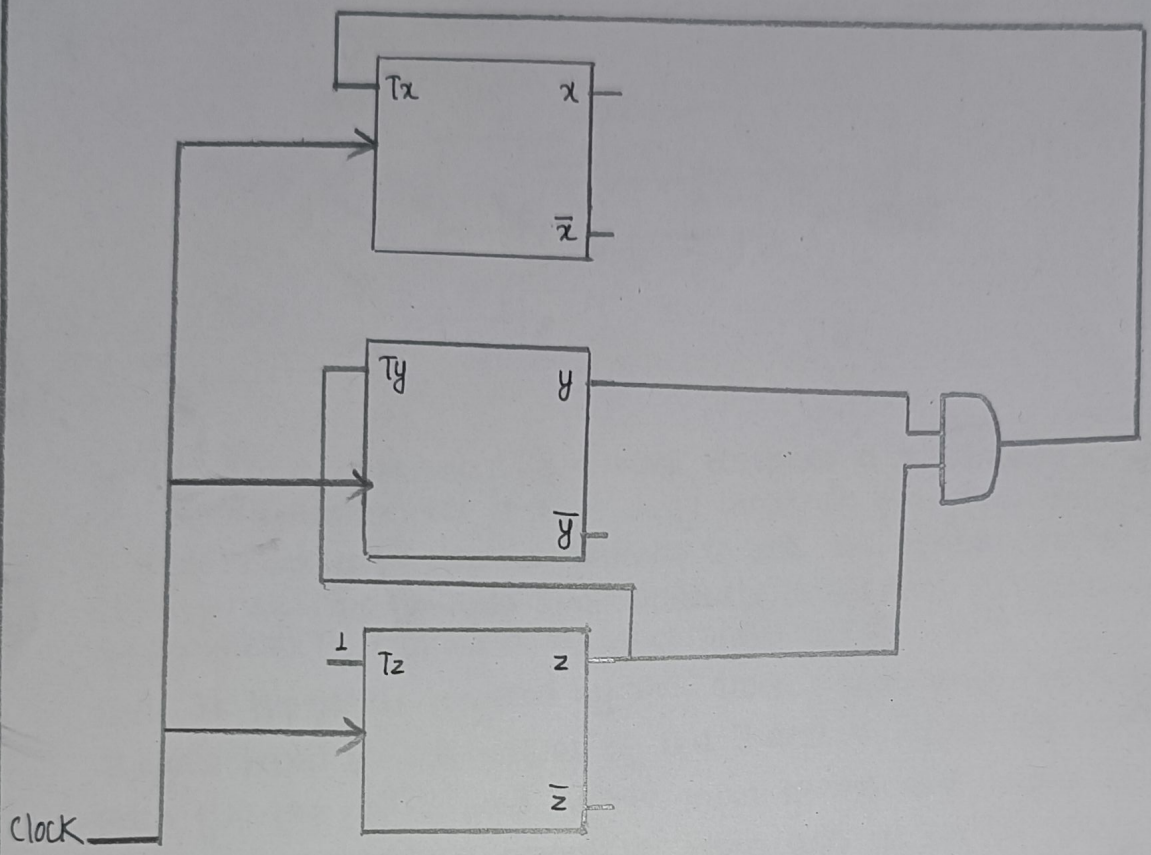


$\therefore Ty = z$

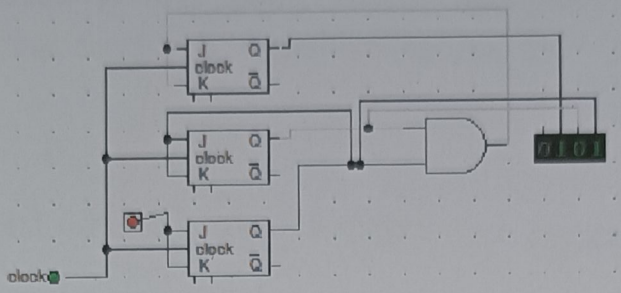


$\therefore Tz = 1$

Logic Circuit Diagram



Experiment of 3 bit (mod 8) synchronous up counter



In the above experiment, we have designed a 3-bit (mod 8) synchronous up counter. This counter is built using three JK flip-flops connected in cascade. Each flip-flop corresponds to one bit in the counter. The clock pulses are provided simultaneously to all the flip-flops to ensure synchronization, making it a synchronous counter.

Each JK flip-flop is triggered by the clock input, and its output toggles based on the values of the J and K inputs. The output of each flip-flop (Q) is used as the input for the next higher bit, ensuring the counting sequence progresses from 000 to 111 (in binary). The AND gate logic is used to reset the counter once it reaches the count of 8 (1000 in binary). When the reset condition is met, the counter is set back to 000, maintaining the modulo 8 behavior.

The counting process goes as follows:

- Initially, the counter starts at 000.
- With each clock pulse, the counter increments its binary value.
- When the output reaches 111, the next pulse resets the counter to 000.

This experiment demonstrates the basic working of a synchronous up counter, where the output progresses through a predefined number of states in a synchronized manner with the clock input.

(2) Asynchronous Counter: In asynchronous counter, first flip flop is clocked by external clock after that output of the previous flipflop clock to the next flipflop. It is a simple design procedure and logic circuit diagram. The propagation delay of the counter is equal to accumulate delay of all flipflops.

Example:

- Design 3-bit (mod 8) asynchronous up counter with its operational table.

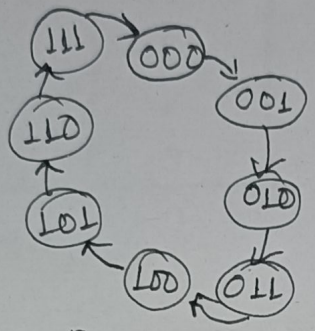
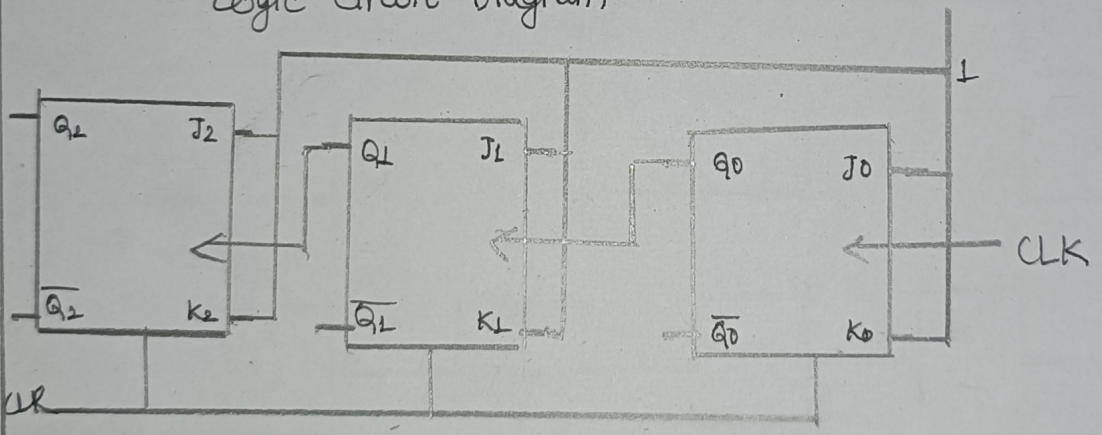


Fig: State Diagram

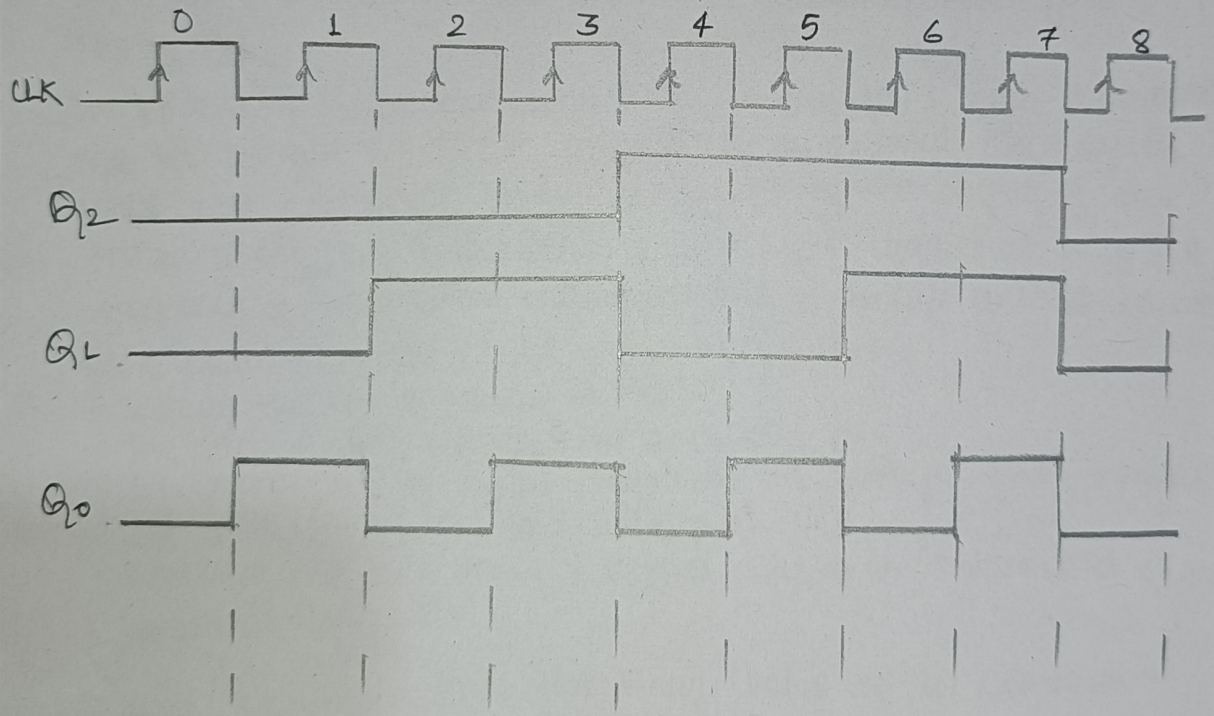
Logic Circuit Diagram



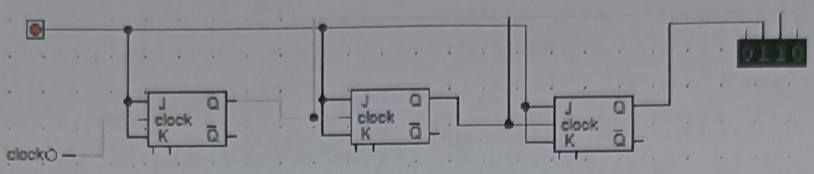
Operational Table

Clock	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1
1	0	0	0

Timing Diagram



Experiment of 3 bit (mod 8) asynchronous up counter



In the above experiment, we have designed a 3 bit (mod 8) asynchronous up counter. This counter, unlike the synchronous one, operates with a clock signal that is not fed to all flip-flops simultaneously. Instead, the output of one flip-flop serves as the clock input to the next flip-flop in the sequence, resulting in a ripple effect.

The counter is constructed using three JK flip-flops, each representing one bit. The first flip-flop is triggered by an external clock signal, while each subsequent flip-flop is triggered by the output of the preceding one. Due to this setup, each flip-flop toggles at half the frequency of the previous one, producing a binary counting sequence. The operation of the counter follows this pattern:

- Initially, the counter starts at 000.
- The first flip-flop toggles with every clock pulse.
- The second flip-flop toggles only when the first flip-flop completes a full cycle (from 0 to 1 and back to 0).
- Similarly, the third flip-flop toggles only when the second flip-flop completes its cycle.

Once the counter reaches the binary value of 111 (decimal 7), it resets back to 000 on the next clock pulse, ensuring that the counter follows a modulo-8 sequence.

Lab-6: Shift Registers

Basic Theory

Shift register is a sequential circuit constructed by using D Flip Flop. It has two capabilities: data storage and data movement. In shift register, all of the flip-flop are triggered by a common clock pulse and set and reset simultaneously.

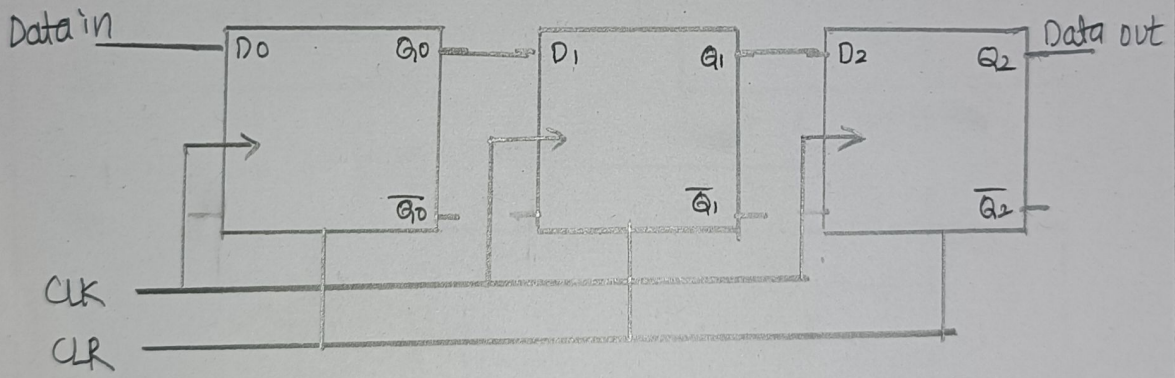
Uni-directional shift register

1) Serial In Serial Out (SISO) shift Register: In SISO shift register, data enter into a shift register serially. One bit at a time and data out of the shift register serially, one bit at a time. All of the flip flops are triggered by a common clock pulse and set and reset simultaneously.

Example:

- Design 3-bit SISO shift register with it's operational table & timing diagram.

Logic Circuit Diagram:

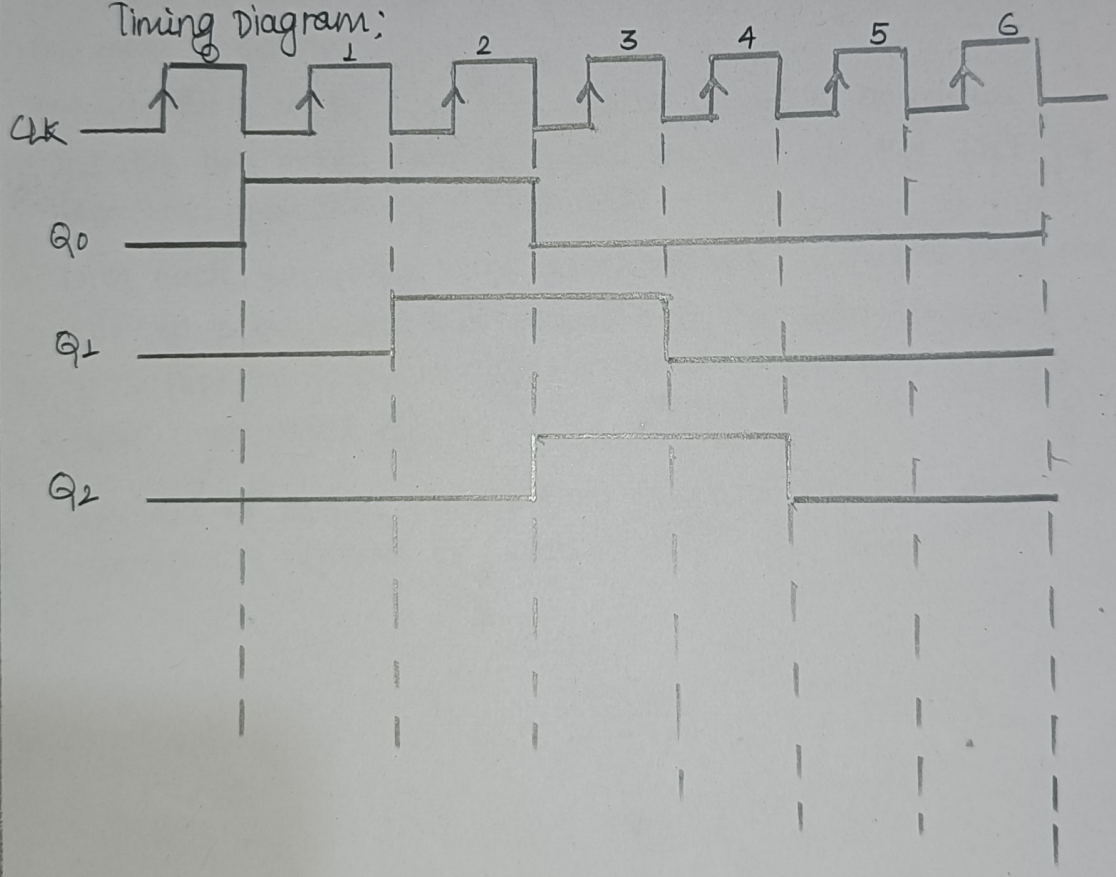


Operational Table:

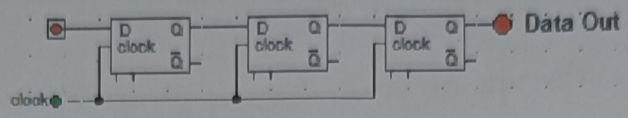
For Data 011

Clock	Q_0	Q_1	Q_2	
0	0	0	0	→ Initially clear
1	1	0	0	
1	1	1	0	
1	0	1	1	→ Data in serially
1	0	0	1	
1	0	0	0	
1	0	0	0	
			1	→ Data out serially
			1	
			0	

Timing Diagram:



Experiment of 3 bit SISO shift register



In the above experiment, we have designed a 3-bit Serial-In-Serial-Out (SISO) shift register. This shift register is used to store and shift binary data sequentially. The design consists of three D flip flops connected in series, where the output of each flip-flop is connected to the input of the next.

The operation of the SISO shift register proceeds as follows:

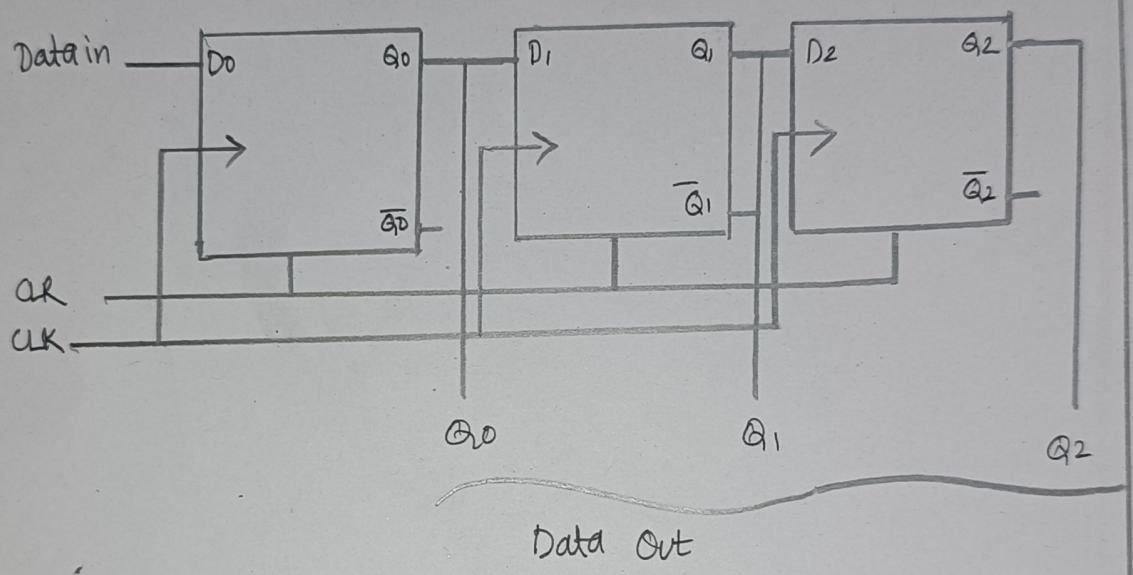
- Initially, the binary data is loaded serially into the first flip flop with the application of a clock pulse.
- With each subsequent clock pulse, the data is shifted from one flip-flop to the next. This means that the data input to the first flip-flop is eventually transferred to the last flip-flop after three clock cycles.
- The data is output serially from the last flip-flop after the appropriate number of shifts.

(2) Serial In Parallel Out (SIPO) shift register: In SIPO shift register, data enter into a shift register serially, one bit at a time and data out of the shift parallelly, all at one clock pulse. All of the flip flop are triggered by a common clock pulse and set and reset simultaneously.

Example:

- Design 3-bit SIPO shift register with its operational table & timing diagram.

Logic Circuit Diagram:

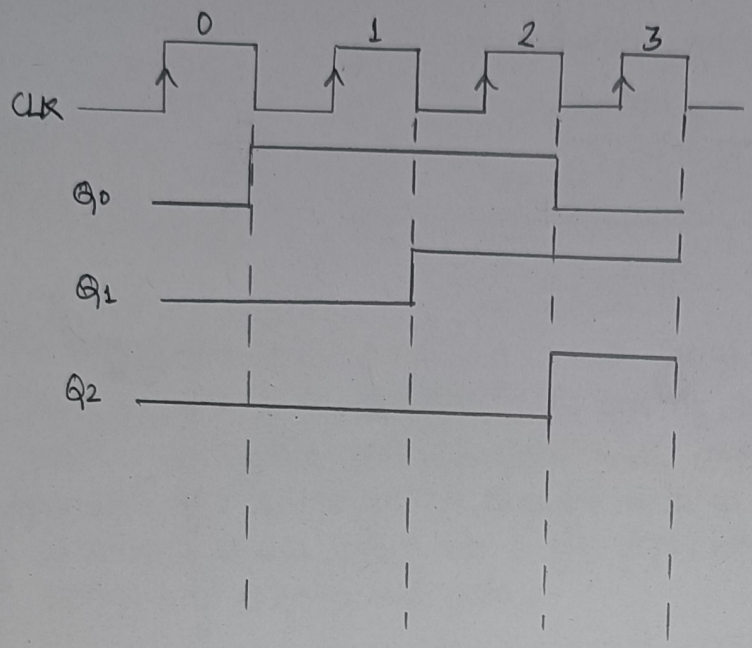


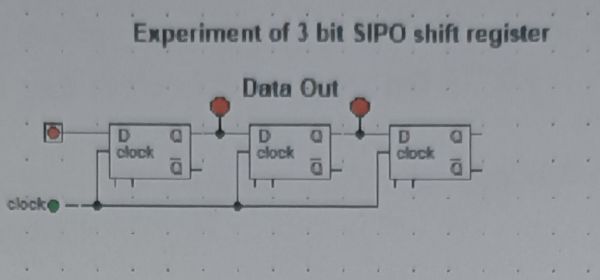
Operational Table:

For Data OLL

Clock	Q0	Q1	Q2	
0	0	0	0	→ Initially clear
1	1	0	0	
1	1	1	0	
1	0	1	1	→ Data in serially
	0	1	1	→ Data out parallelly

Timing Diagram:





In the above experiment, a 3-bit Serial In Parallel Out (SIPO) Shift Register is demonstrated. This type of shift register takes in data serially (bit by bit) on each clock pulse and outputs the bits in parallel after shifting them through a series of flip-flops. The main components in this circuit are D flip-flops, which are responsible for storing and shifting the data.

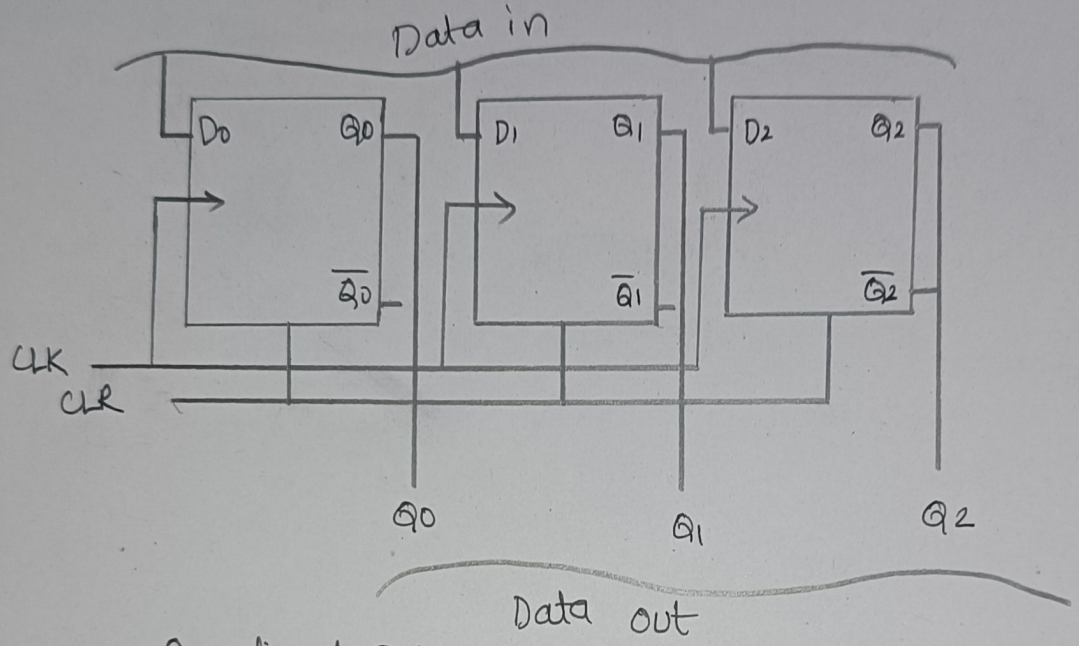
Here's a breakdown of how the circuit works:

- Data Input: A serial data input is provided, which is shifted into the shift register with each clock pulse. The switch shown in diagram is used to provide this serial input.
- Clock Input: A common clock signal is connected to all three D flip-flops. With each clock pulse, the data is shifted from one flip-flop to the next.
- D Flip-Flops: Each D flip-flop holds one bit of data. On each clock pulse the data at the input of the first flip-flop is stored and the data stored in the first flip-flop moves to the second, and so on.
- Data Output: The outputs from each flip-flop are shown at the top with the labeled "Data out" points, providing the parallel output. After three clock pulses, the serial data entered into the register will appear at the outputs of all three flip-flops in parallel.

(3) Parallel In Parallel out (PIPO) Shift Register: In PIPO shift register, data enter into a shift register parallely all at one clock pulse and data out of the shift register parallely all at one clock pulse. All of the flip flop are triggered by a common clock pulse and set and reset simultaneously.

Example:

- Design 3 bit PIPO shift register with its operational table & timing diagram.

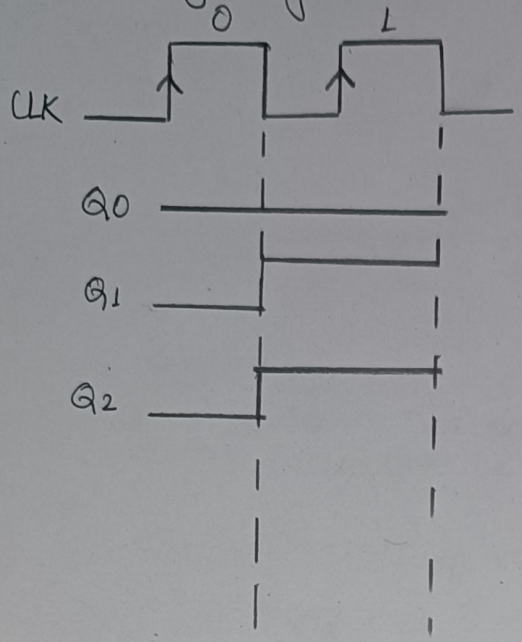


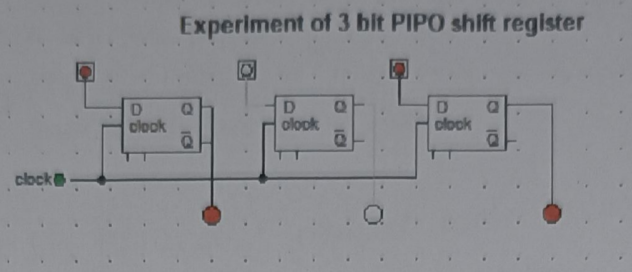
Operational Table:

For Data 011

Clock	Q0	Q1	Q2	
0	0	0	0	→ Initially clear
1	0	1	1	→ Data in parallely
	↓	↓	↓	
	0	1	1	→ Data out parallely

Timing Diagram:





In the above experiment, a 3 bit Parallel In Parallel Out (PIPO) Shift Register is shown. This type of shift register accepts input data in parallel form and outputs the data in parallel as well, after a clock pulse.

Here's how this PIPO shift register works:

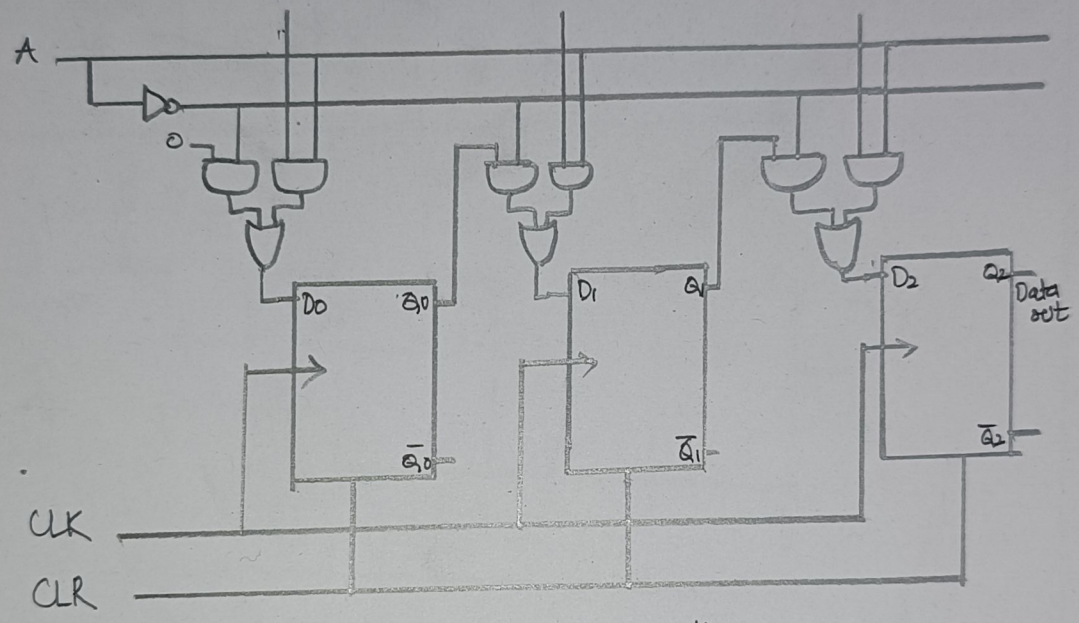
- Parallel Data Input: Each D flip-flop has a separate data input. This allows us to input multiple bits of data at the same time. In this 3-bit register, three inputs are provided for each of the D flip flops.
- Clock Input: All three D flip-flops share a common clock signal. When the clock pulse occurs, the data from the parallel inputs is simultaneously loaded into each flip-flop.
- D Flip-flops: Each D flip-flop holds one bit of data. When a clock pulse is applied the data present at the D input of each flip-flop is stored in the flip flop's internal memory.
- Parallel Data Output: The output from each flip-flop is taken in parallel form, allowing all three bits to be read at once. These outputs are labeled as the "Data out" points.

(4) Parallel In Serial out (PISO) Shift Register: In PISO shift register, data enter into a shift register parallelly all in one clock pulse and data out of the shift register serially, one bit at a time. All of the flip flop are triggered by a common clock pulse & set and reset simultaneously.

Example:

- Design 3 bit PISO shift register with its operational table & timing diagram.

Logic Circuit Diagram:



A = 1, Data in parallelly

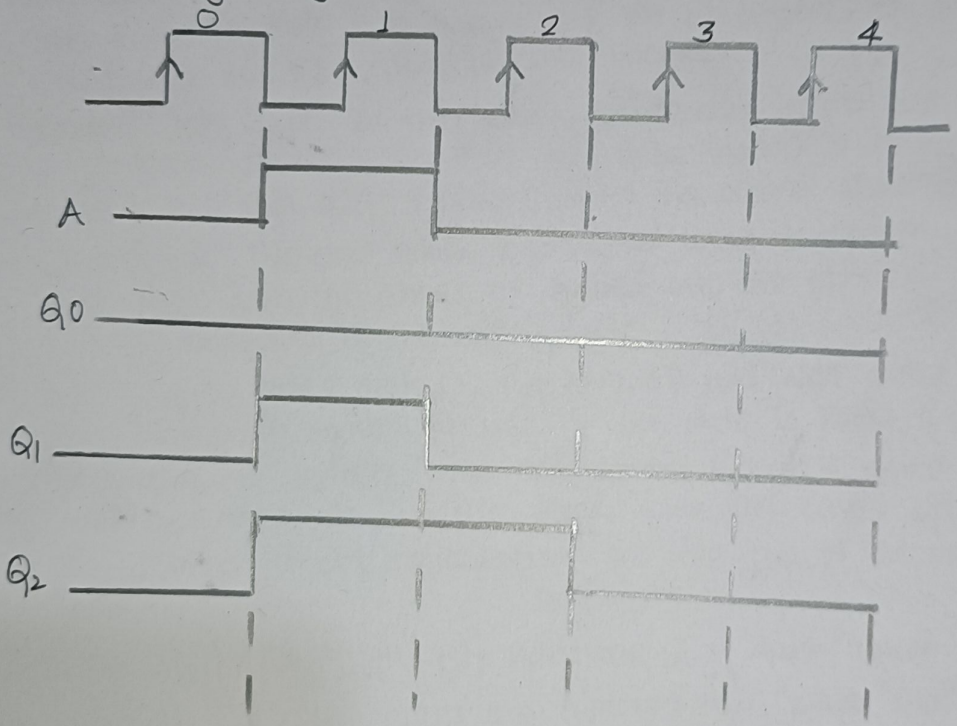
A = 0, Data in serially

Operational Table:

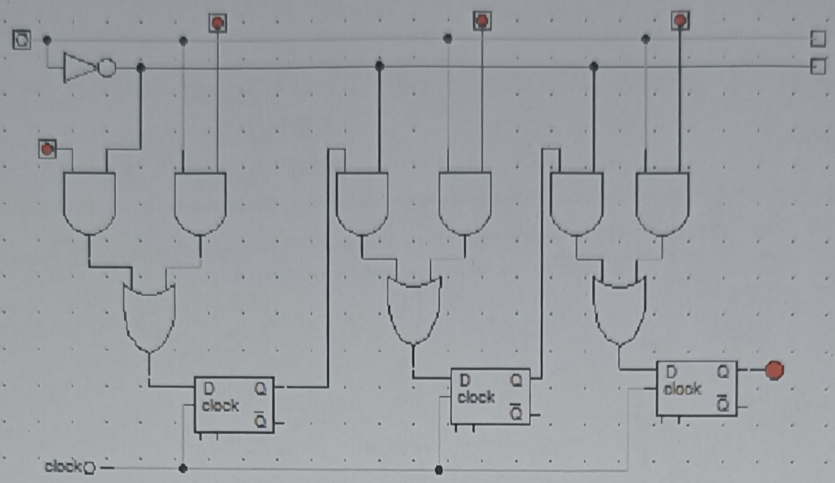
For data 011

Clock	A	Q_0	Q_1	Q_2	
0	X	0	0	0	→ Initially clear
1	1	0	1	1	→ Data in parallely
1	0	0	0	1	1
1	0	0	0	0	1 → Data out serially
1	0	0	0	0	0

Timing Diagram



Experiment of 3 bit PISO shift register



In the above experiment, we see a 3-bit Parallel In Serial Out (PISO) Shift Register. This type of shift register accepts parallel input data and then shifts the data out serially, one bit at a time, on each clock pulse. Here's how the PISO shift register works:

- Parallel Data Input: In this setup, three data inputs are provided to three D flip-flops, allowing multiple bits to be loaded in parallel. The data is loaded into the shift register when the control signal enables parallel loading.
- Clock Input: The clock signal is connected to all D flipflops, and with each clock pulse, data can either be loaded into the flip-flops or shifted out serially, depending on the control logic.
- Control Logic (MUX & Gates): The AND, OR and NOT gates seen in the circuit helps in controlling whether the data is being loaded in parallel or shifted out serially. When the control signal is active, parallel data is loaded into the flip-flops; when the control signal is inactive, the data shifts out serially from one flip-flop to the next on each clock pulse.
- D Flip-Flops: Each D flip-flop holds one bit of data. When in "load" mode, the parallel inputs are captured. In "shift" mode, the data from the flip-flops is shifted to the next flip-flop and finally to the serial output.
- Serial Data Output: The output is taken serially from the last flip-flop after the data has been shifted through the register. This is seen as the final red output indicator in the circuit.

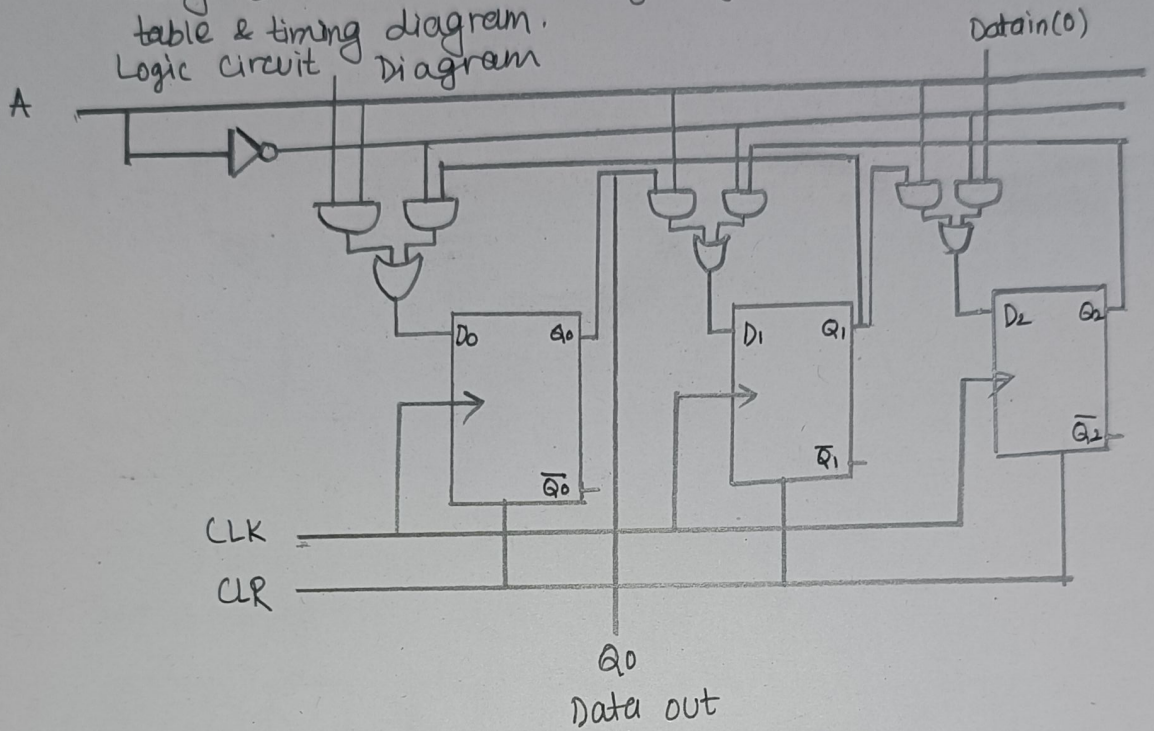
Bi-directional shift register

In bidirectional shift register, data can be shifted in both direction either left to right or right to left. All of the flipflop are triggered by a common clock pulse & set and reset simultaneously

Example:

- Design 3-bit bidirectional shift register with its operational table & timing diagram.

Logic circuit Diagram



A = 1, Data in by shifting right

A = 0, Data out by shifting left

Operational Table:

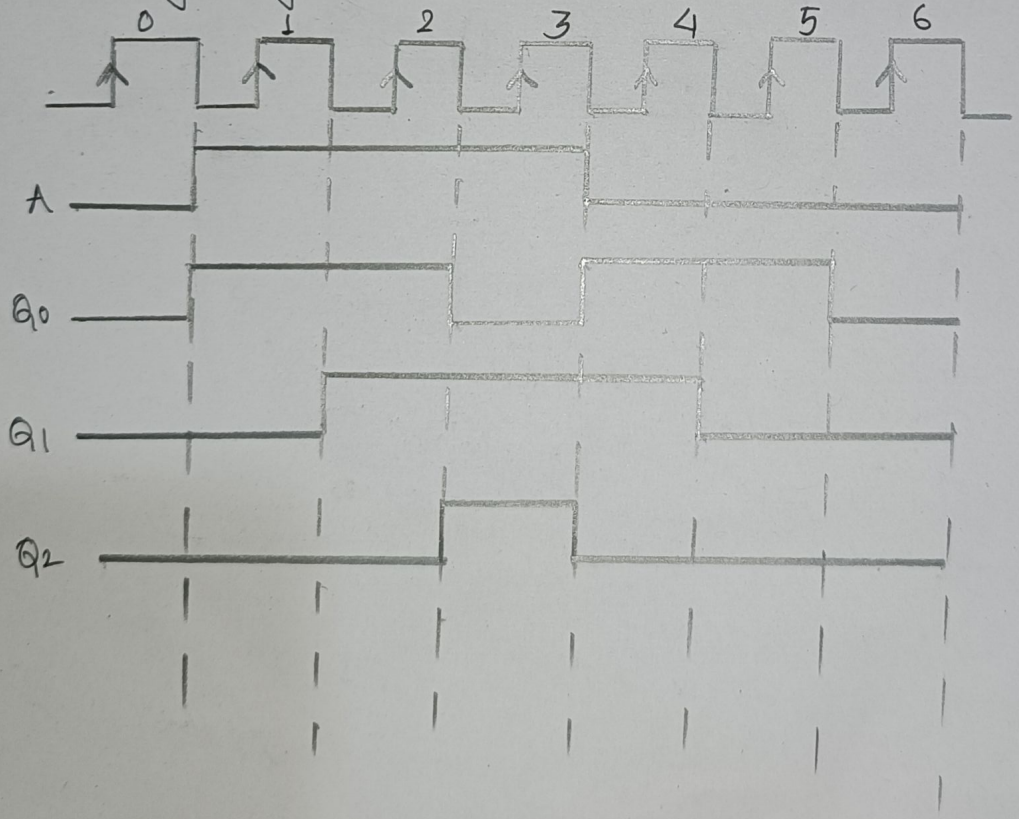
For Data 011

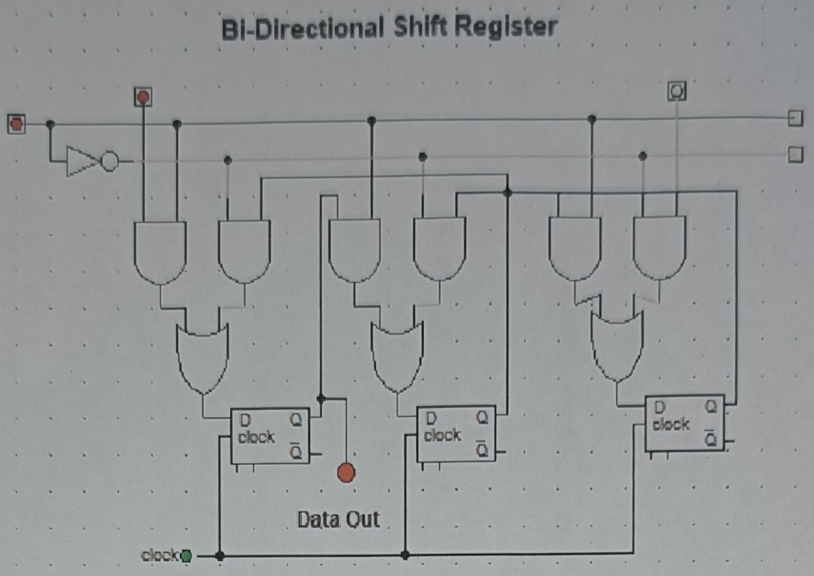
Clock	A	Q_0	Q_1	Q_2	
0	X	0	0	0	→ Initially clear
1	1	1	0	0	
1	1	1	1	0	
1	1	0	1	1	→ Data in by shifting right
1	0	1	1	0	
1	0	1	0	0	
1	0	0	0	0	

011

Data out by shifting left

Timing Diagram:





In the above experiment, a bi-directional shift register was designed using D flip-flops and logic gates. The circuit allows shifting data either left or right, depending on the control signals. D flip-flops store data, while the AND, OR and NOT gates control the shifting direction. The clock signal ensures synchronized operation.