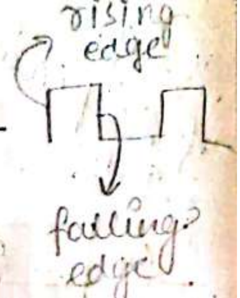
11/09/19

Chapter - 02

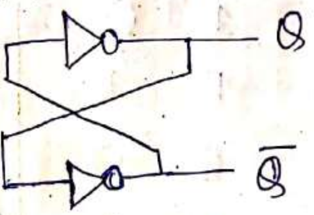
SEQUENTIAL CIRCUITS



- part of memory system.
- It does not only depend on the i/p at that instant but it will also depend on the history of i/p or o/p (previous i/p or o/p).
- It is capable to store or hold bits.
- Two types : 1). Synchronous → (wait for rising edge to respond)
  11). Asynchronous ↳ (respond fast)

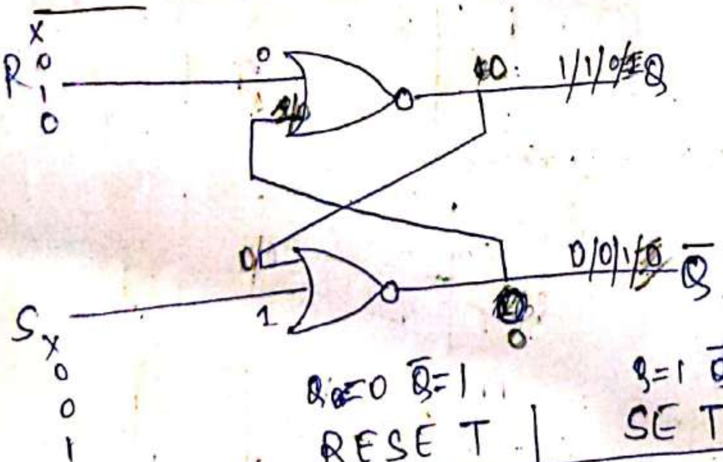
★ BISTABLE

Every bistable element with a stable state is called STATE.



1 – Set.
Q – Reset.

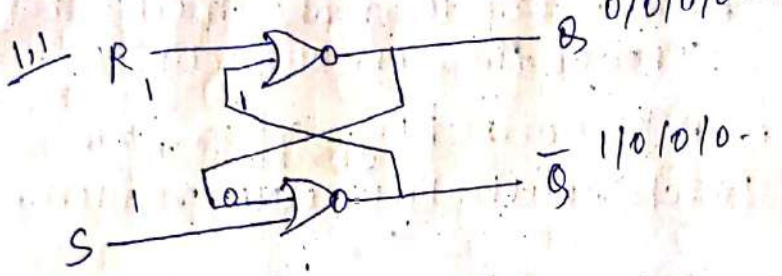
S-R LATCH



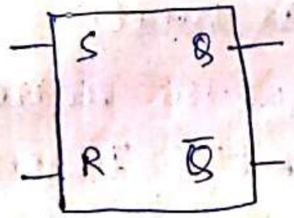
S	R	
x	x	SET
0	0	SET
0	1	RESET
1	0	SET

	Q=0 $\overline{Q}$=1 RESET	$Q=1$ $\overline{Q}=0$ SET ← let i/p be
10	SET	~~RE~~SET
00	RESET	SET

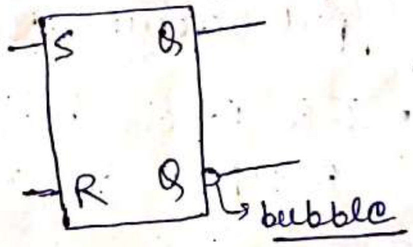
S	R	Q^+	$\overline{Q}^+$	
0	0	Q	$\overline{Q}$	← retains its previous state
0	1	0	1	reset
1	0	1	0	sets
1	1	0*	0*	undefined state → limitation of S-R laches.



Block Representation of SR latch:

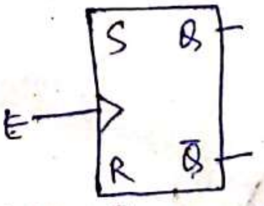


or

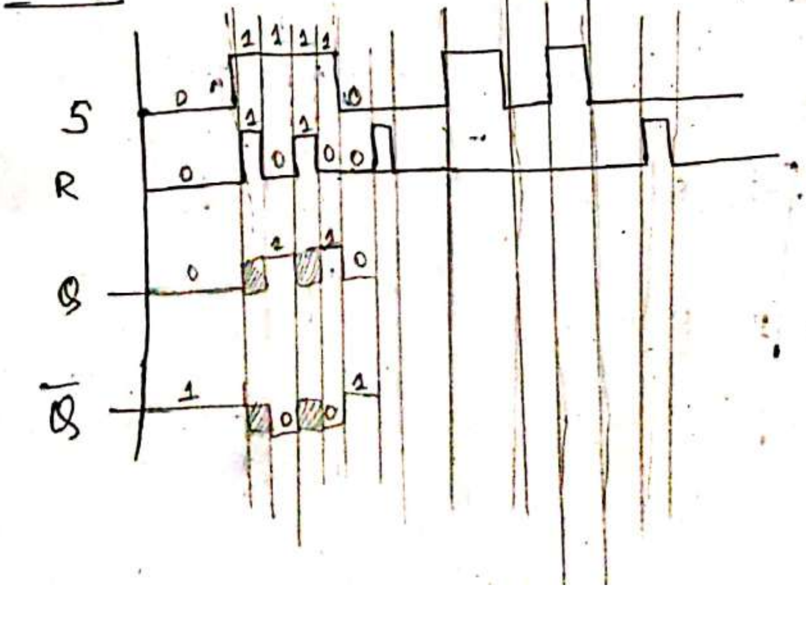


- Only when E is enabled i.e. high, ~~then~~ ~~only~~ S-R latch will show the o/p. If E is disabled then irrespective of i/p, o/p will be retained only."

↳ Gated → SR-latch

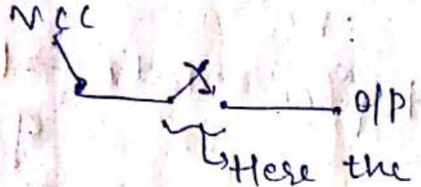


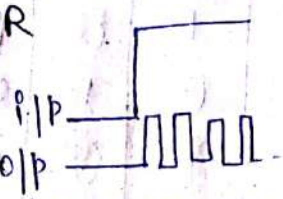
Ques



* Applications of SR Latch

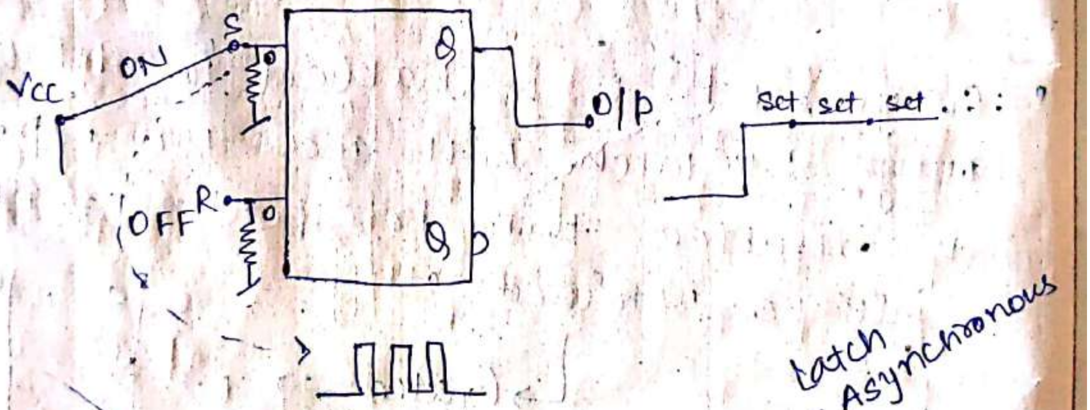
1). SWITCH ~~BE~~ DEBOUNCER





↳Here the switch bounces whenever we ~~we~~ try to make mechanical contact (i.e. on switch).

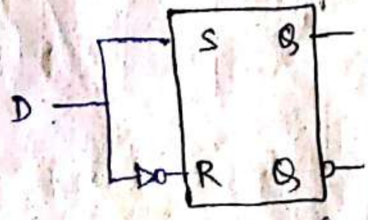
- To eliminate this bouncing effect, we use SR latch with Pulldown resistors.
- Pulldown resistor will pull the voltage of that point and make it 0 & will pull down the bounce as well.
- S-R latch is connected b/w the switch & o/p & is known as Switch Debouncer.



Latch = Asynchronous

* D-LATCH :- (Data latch)

Gated D-latch



D	Q^+	$\bar{Q}^+$
0	0	1
1	1	0

if D=0 → $\left.\begin{matrix}S=0\\R=1\end{matrix}\right\}$ ∴ $Q=0$, $\bar{Q}^+=1$

D=1 → $\left.\begin{matrix}S=1\\R=0\end{matrix}\right\}$ ∴ $Q=1$, $\bar{Q}^+=0$

* FLIP FLOPS :

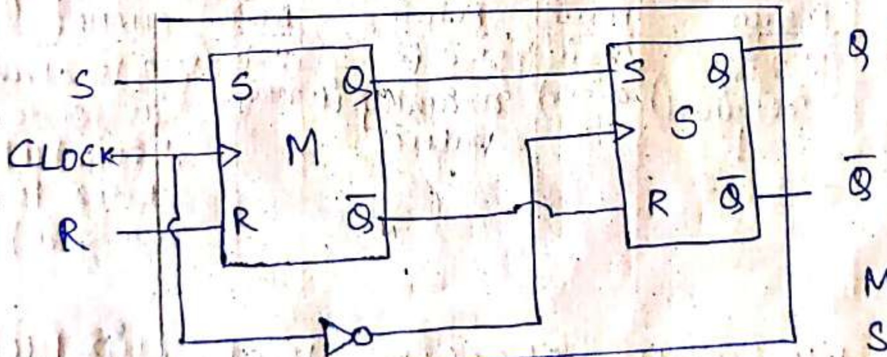
- Transparency is the property exhibited by latch. It refers to the immediate change in o/p with change in i/p.
- while, there is no transparency in flip flop
- Flip Flops are synchronous in nature.
- clock is not present in latch whereas flip flops have clock.
- latch is a high speed device whereas flip flops are not high speed.

SR Flip Flop : It consists of two GATED latches.

- Characterised into :

1). PULSE TRIGGERED (Master Slave flip flops)

2). EDGE TRIGGERED

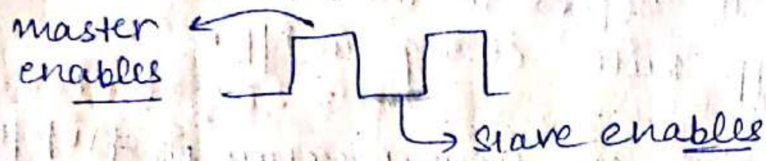


M=master
S=SLAVE

- 1st SR latch is master & 2nd is SLAVE.
- Flip Flops have capability to store the value. ∴ Slave ~~to~~ stores the state & master is responsible to the generation of new states.
- The state of 1st SR latch is the i/p to 2nd SR latch.

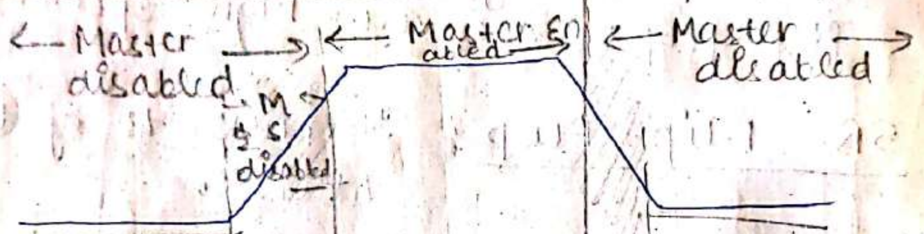
P.T.O

- In this master - slave set up, both will not be enabled at the same time.



CLOCK

theoretical
practical



	← SLAVE → ENABLED	← Slave	disabled	→	← Slave → Enabled
State of M	Retain (∵ M is disabled)	Retain (∵ M is disa.)	CHANGE (∵ M is enabled & it take new set of	Retain (∵ M is disab)	Retain ∵ (M is disabled)
State of S	Retain (∵ M is disabled & not giving o/p)	Retain (∵ S is disab)	Retain (Even if M is changing, S is disabled)	Retain (∵ S is disab)	CHANGE (∵ M will give its state to S & will be ready for new set of I/PS)
State of FF	Retain (∵ S is retained)	Retain (∵ S is disab.)	Retain (∵ S is retaining its value)	Retain (∵ S is retain)	CHANGE (∵ S is changing its state. ∴ flip flops states is also changed)

- Here, we can observe that flip flop takes one compeete cycle of clock to change its state.

Thus, flip flops are ~~smaller~~ slower as compared to SR latches.

Disabled & Power off is different.

Master is holding some value at o/p.

- As soon as slave is Enabled, it changes its state.

Function Table : of SR Flip Flop

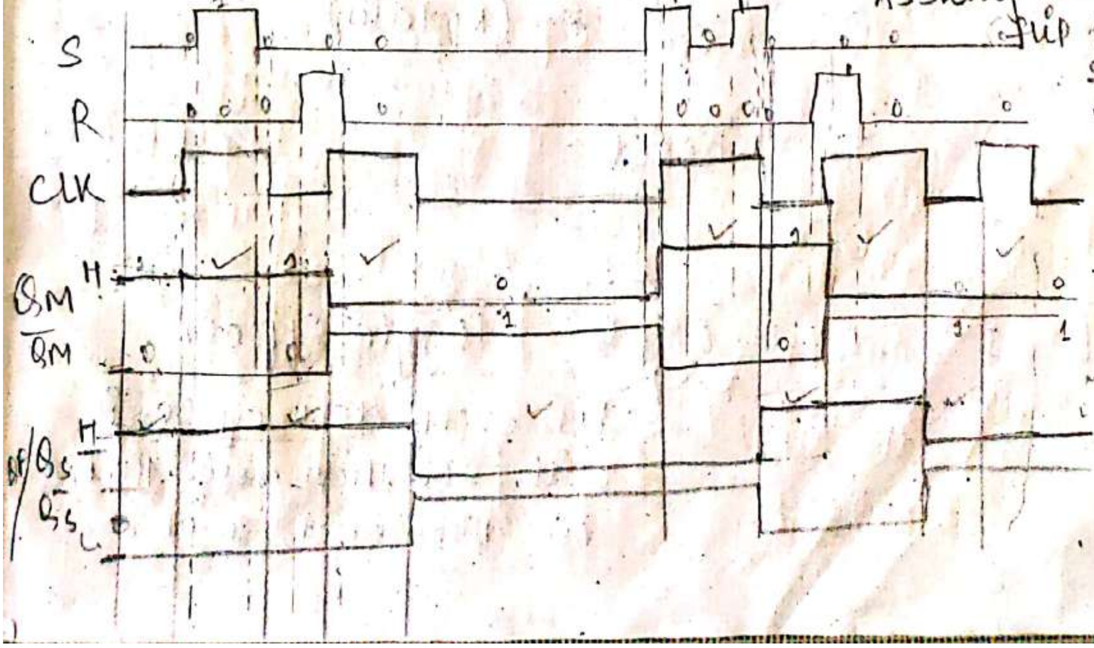
— : no clock
⎍ : with clock

CLK	S	R	Q^+	$\bar{Q}^+$	
(0) —	X	X	Q	$\bar{Q}$	
⎍	0	0	Q	$\bar{Q}$	(retain)
⎍	0	1	0	1	(reset)
⎍	1	0	1	0	(set)
⎍	1	1	0*	0*	(undefined)

Assuming that initial stage off flip flop is reset.

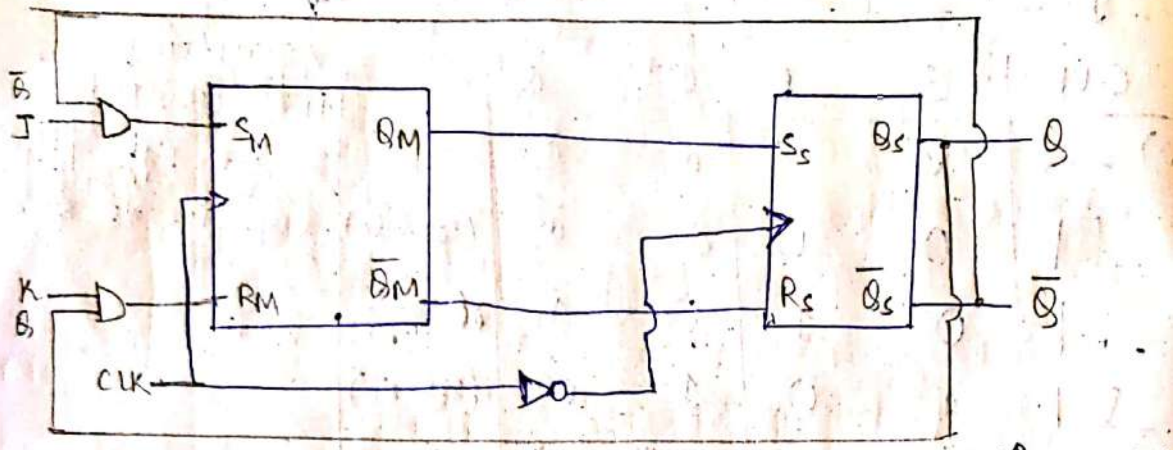
* Comparision b/w Latch & flip flop.

Latch	Flip Flop
① Transparency	NO transparency.
② Asynchronous	Synchronous
③ clock is not present.	clock is present.
④ High speed	Low speed
⑤ loose prev. data	can store prev. data
⑥ Building blocks are Gates.	Building blocks are Latches.

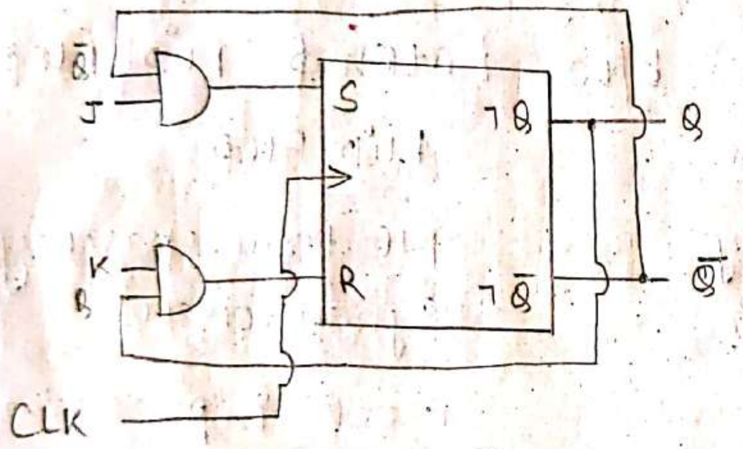


Assumig state of flip flop is set
∴ slave is also set
∴ M is also set

* JK FLIP FLOP



↳ internal diagram



Function Table

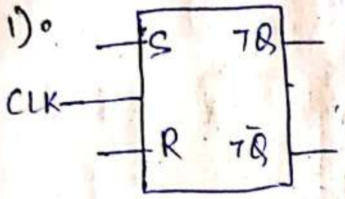
CLK	J	K	Q^+	$\bar{Q}^+$
0	X	X	Q	$\bar{Q}$
⎍	0	0	Q	$\bar{Q}$ (Retain)
⎍	0	1	Reset	
⎍	1	0	Set	
⎍	1	1	$\bar{Q}$	Q (toggle Effect)

↳ For every clock cycle, it will set then reset then set then reset so on.

*How JK flip flop overcomes the limitation of SR ~~Latch?~~ FF

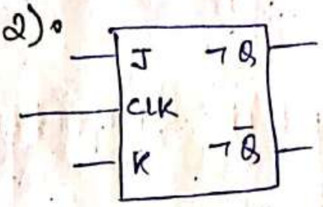
- Describe the limitations of SR latch.
- Introduce the design of JK flip flop.
- Explain its working & toggle Effect, states of both.
- This is how, it overcomes.

1).



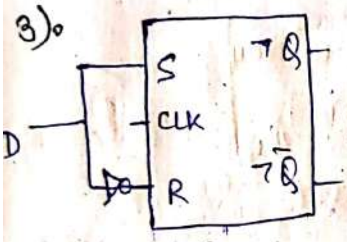
S-R flip flop

2).



J-K flip flop

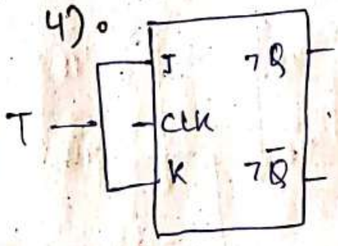
3).



D-flip flop

(If External i/p in SR FF is shorted)

4).



T-flip flop

(If external i/p of JK FF is sorted)

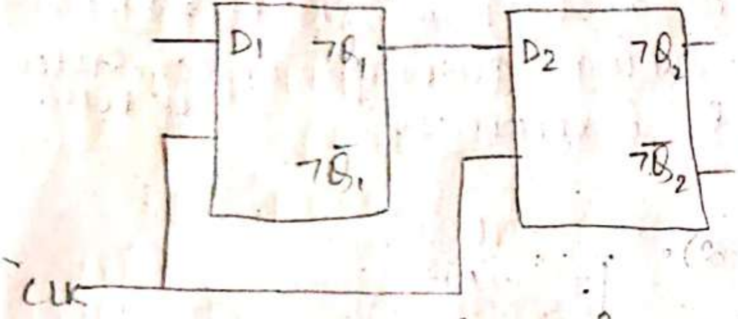
Fuction table

CLK	D	Q^+	$\bar{Q}^+$
0	X	Q	$\bar{Q}$
⎍	0	0	1 (reset)
⎍	1	1	0 (set)

Function table

CLK	T	Q^+	$\bar{Q}^+$
0	X	Q	$\bar{Q}$
⎍	0	Q	$\bar{Q}$ (retain)
⎍	1	$\bar{Q}$	Q (toggle)

* SHIFT REGISTER :

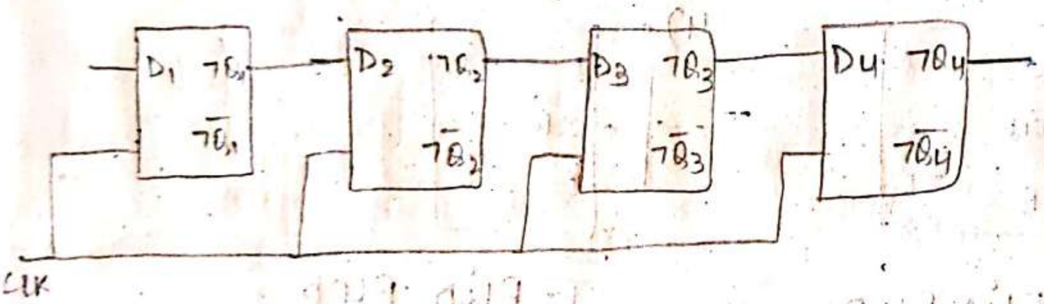


SISO (serial in serial Out)

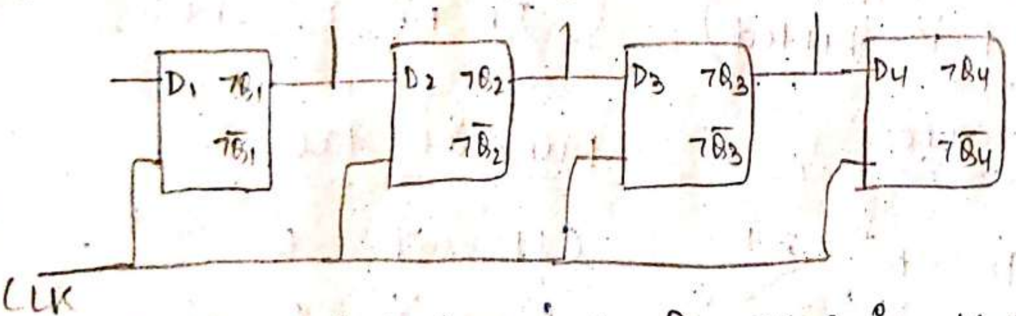
SIPO (serial in parallel out)

2 Bit Shift Register

* SISO Shift Register for 04-bit :



* SIPO Shift Register for 04-bit :



* PISO

(also called Parallel-in Unidirectional Registers) {Explain the operation with these steps:- Fig Next Page

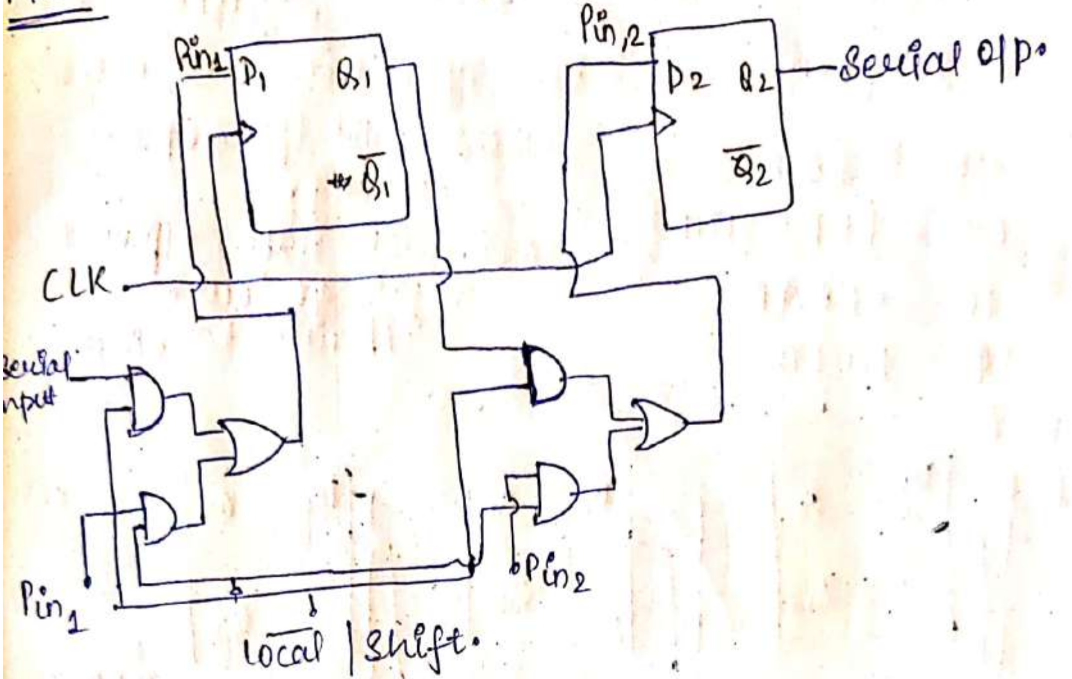
Steps

(i) L|S=0
(ii) Parallel load (Pin, Pin a)] To load Parallel data to flip flops.
(iii) CLK ⎍

Parallel I/p is loaded ←

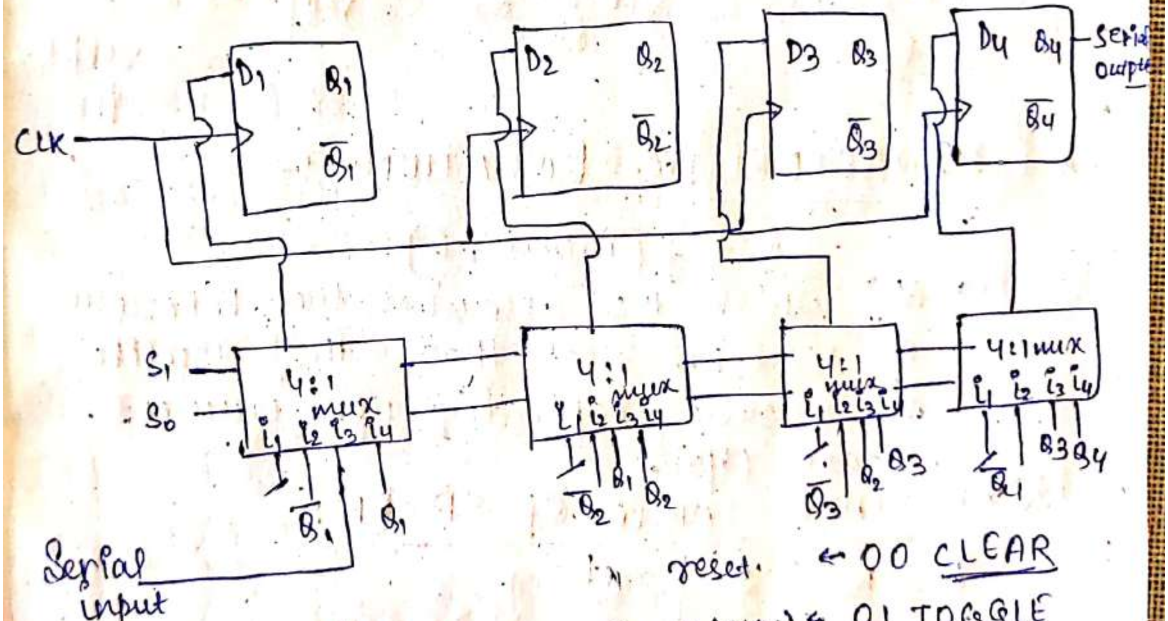
(iv) L|S=1] To perform shift of data.
(v) ⎍ → As many shifts is req., that many clock cycle is taken.

PISO



★ UNIVERSAL REGISTER :- (Universal Shift Register)

↳ It is a register which such a design that multiple operations can be performed.



reset. ← 00 CLEAR

change (opp.) ← 01 TOGGLE

10 SHIFT

11 HOLD.

Right - shift

Ques Design a ~~aw~~ universal register to perform :- 3-Bit operations as below

Imp

00 - SET

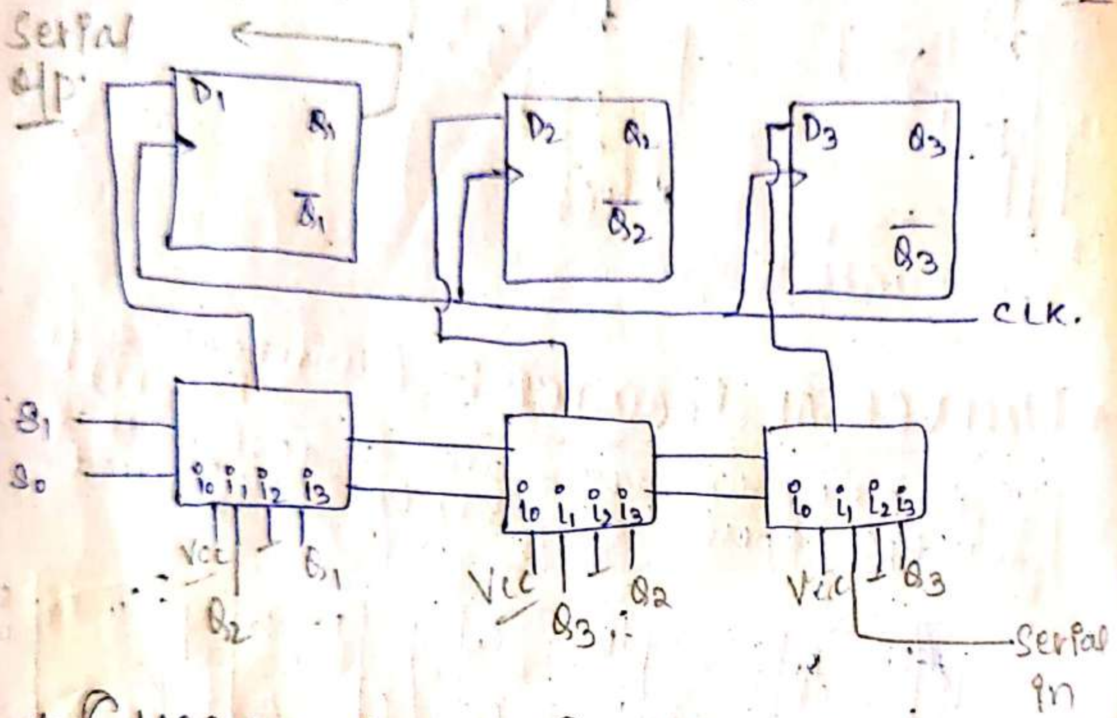
01 - LEFT SHIFT

10 - CLEAR

11 - HOLD

/Show serial i/p & o/p lines correctly.

// Circular shift, o/p will never go out. i/p will be circular.



★ CHARACTERISTIC EQUATIONS :-

$$Q^+ = f(\text{inputs}; Q)$$

→ eqn:- which the describes the behaviour of each FF such that they describe next state depending upon current state (i/p).

NEXT STATE TABLE Of SR FF :

S	R	Q	Q^+	
0	0	0	0	} retains
0	0	1	1	
0	1	0	0	} reset
0	1	1	0	
1	0	0	1	} set
1	0	1	1	
1	1	0	X	} don't care.
1	1	1	X	

K-map

S \ RQ	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$\boxed{Q^+ = S + \bar{R}Q}$ → Characteristic eqn of S-R Flip Flop.

* J-K Flip Flop:

Next state Table

J~~S~~	~~R~~K	Q	Q^+
0	0	0	0 } retain
0	0	1	1
0	1	0	0 } reset.
0	1	1	0
1	0	0	1 } set.
1	0	1	1
1	1	0	1 } toggle Effect.
1	1	1	0

J \ KQ	00	01	11	10
0	0	1	0	0
1	1	1	0	1

∴ ~~$Q^+ = \bar{R}Q + S\bar{Q}$~~

$\boxed{Q^+ = \bar{K}Q + J\bar{Q}}$

* D-Flip Flop:

Next State Table

D	Q	Q^+
0	0	0 } reset.
0	1	0
1	0	1 } set.
1	1	1

D \ Q	0	1
0	0	0
1	1	1

$\boxed{Q^+ = D}$

* T-Flip Flop:

Next State Table

T	Q	Q^+
0	0	0 } retains
0	1	1
1	0	1 } toggles
1	1	0

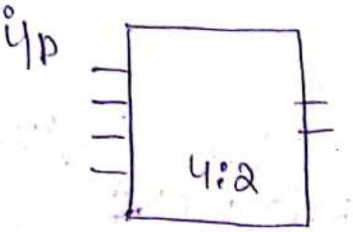
K-map

T \ Q	0	1
0	0	1
1	1	0

$\boxed{Q^+ = \bar{T}Q + T\bar{Q}}$

$= T \oplus Q$

* Encoders :



i_0	i_1	i_2	i_3	y_1	y_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1
If 1	0	1	0	$\boxed{1 \quad 0}$ → ? limitation	

$y_1 = i_2 + i_3$

$y_0 = i_1 + i_3$

$\therefore y_1 = i_2 + i_3 = 1+0 = 1$

$y_0 = i_1 + i_3 = 0+0 = 0$

* Priority Encoder gives ~~the~~ priority to the highest bit.

i_0	i_1	i_2	i_3	y_1	y_0
1	0	0	0	0	0
X	$\textcircled{1}$	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1
1	0	$\textcircled{1}$	0	1	0 ?

It will give priority any to 1 value ←

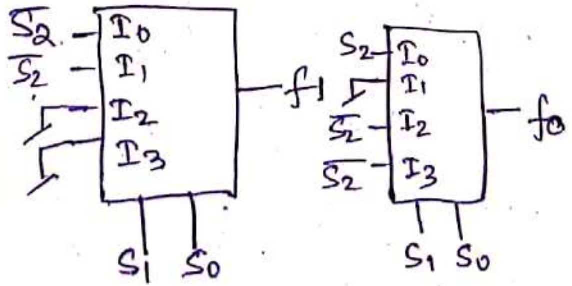
↳ gives priority to this highest bit.

LP ISA-1 (Sample)

6. Assumption let shop No. be 1–8.

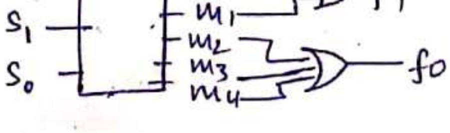
S_3	S_2	S_1	S_0	f_1	f_0
0	0	0	0	X	X
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	X	X
1	0	1	0	X	X
1	0	1	1	X	X
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

If we solve using MUX:



S_2	S_1	S_0	f_1	f_0
0	0	0	1	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

Using DECODER:


S_2
S_1
S_0
m_0
m_1
m_2
m_3
m_4
f_1
f_0