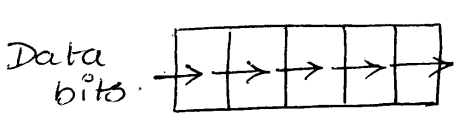
UNIT-5 MODULE-4
REGISTERS

* REGISTERS -

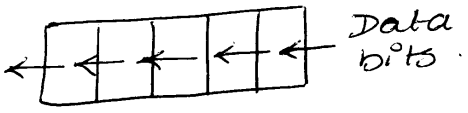
The group of flip flops can be used to store a word, which is called registers.
A flip flop can store 1-bit information. So an n-bit register has a group of n-flip flops and is capable of storing any binary information/number containing n-bits.

The binary information (data) in a register can be moved from stage to stage within the register or into or out the register upon the application of clock pulses. Hence they are also called as "Shift Registers".

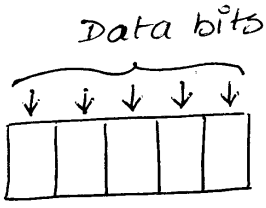
The figure below shows the symbolic representation of the different types of data movement in shift register operations.



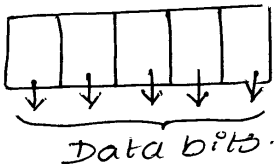
(a) Serial shift right, then out.



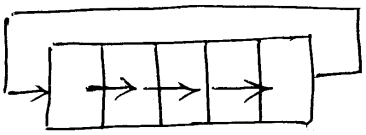
(b) Serial shift left, then out.



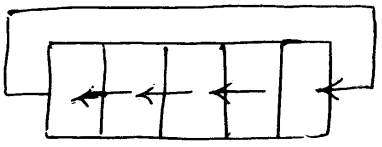
(c) Parallel shift in



(d) Parallel shift out.



(e) Rotate Right



(f) Rotate left.

FIG: Basic Data Movement in Registers.

* TYPES OF REGISTERS -

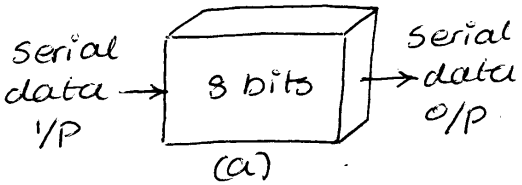
Register is simply a group of flip flops that can be used to store a binary number.

"There must be 1 flip-flop for each bit in the binary no. for eg; a register used to store an 8-bit binary no must have 8 flip-flops."

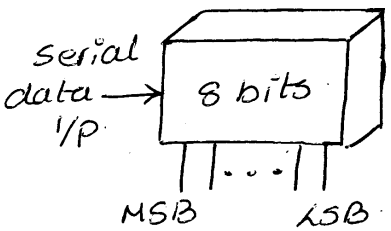
Naturally, the flip flops must be connected such that the binary no can be entered (shifted) into the register & possibly shifted out.
A group of flip flops connected to provide either or both of these functions is called a shift register.

The bits in a binary no can be moved from one place to another in either of 2 ways. The first method involves shifting the data 1 bit at a time in a serial fashion, beginning with either the MSB or the LSB. This technique is referred to as serial shifting. The second method involves shifting all the data bits simultaneously and is referred to as parallel shifting.

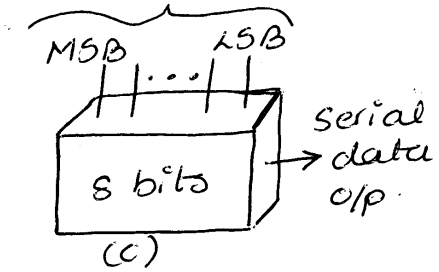
This leads to the construction of 4 basic register types.



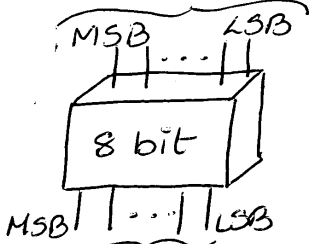
(a)
Serial In - Serial Out



Parallel data O/P's.
(b)
Serial - In Parallel Out



(c)
Parallel In Serial Out



Parallel data O/P.
(d)

1> Serial In - Serial Out (SISO) Register.

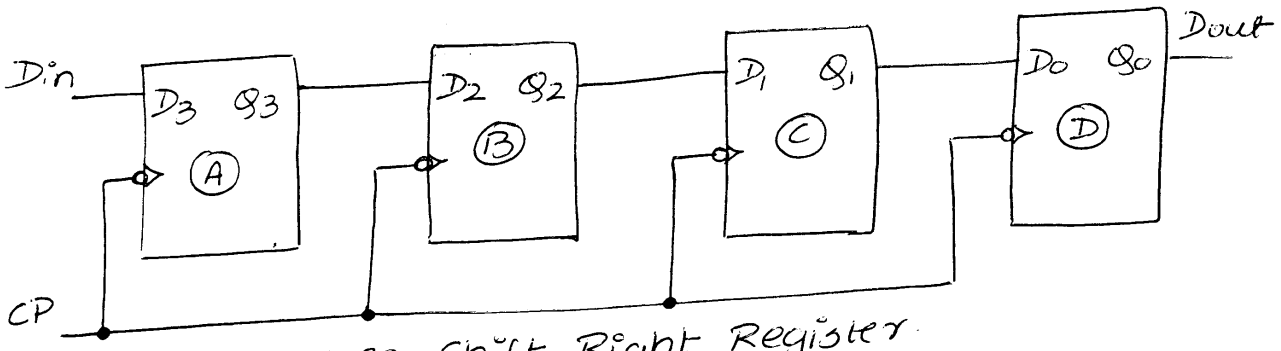


FIG: SISO Shift Right Register

Initially, register is cleared. So, $Q_3Q_2Q_1Q_0 = 0000$

a) When data 1111 is applied serially, ie; left most 1 is applied as Din,

$D_{in} = 1$, $Q_3Q_2Q_1Q_0 = 0000$

The arrival of the first falling clock edge sets the left most flip flop, and the stored word becomes,

$Q_3Q_2Q_1Q_0 = 1000$.

b) When the next falling clock edge hits, the Q_2 flip flop sets and the register contents become,

$Q_3Q_2Q_1Q_0 = 1100$.

c) The third falling clock edge results in,

$Q_3Q_2Q_1Q_0 = 1110$

d) The fourth falling clock edge gives,

$Q_3Q_2Q_1Q_0 = 1111$

In SISO, for loading 4 bit data, we require 4 clocks, and also at the 4th clock we will see the LSB.

At 5th clock, we will see 2nd bit of the I/p.

At 6th clock, we will see 3rd bit of the I/p.

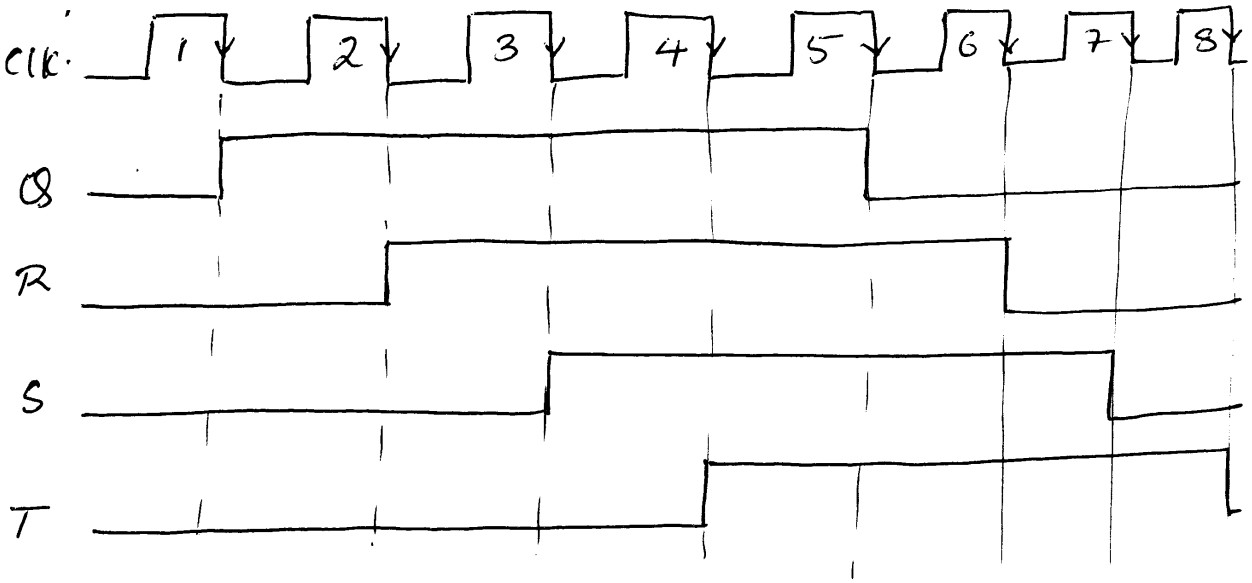
At 7th clock, we will see 4th bit of the I/p.

In brief, we require $n+(n-1)$ clocks to see the final bit of the I/p, where 'n' is the

At clock 0, Q = R = S = T = 0.

clock	D	Q	R	S	T	
0	0	0	0	0	0	
1	1 →	1	0	0	0	
2	1 →	1	1	0	0	
3	1 →	1	1	1	0	
4	1 →	1	1	1	1	→ 1 → LSB seen at clk 4
5	0 →	0	1	1	1	→ 1 → " " clk 5
6	0 →	0	0	1	1	→ 1 → " " clk 6
7	0 →	0	0	0	1	→ 1 → MSB " clk 7

Fig: Shift Right Operation.



Timing Diagram of SISO Register.

2> Serial In - Parallel Out (SIPO) Register.

A SIPO shift register is similar to the SISO shift register.

In SIPO, the data bits are entered serially as in SISO, but the data bits are taken out of each flip flop.

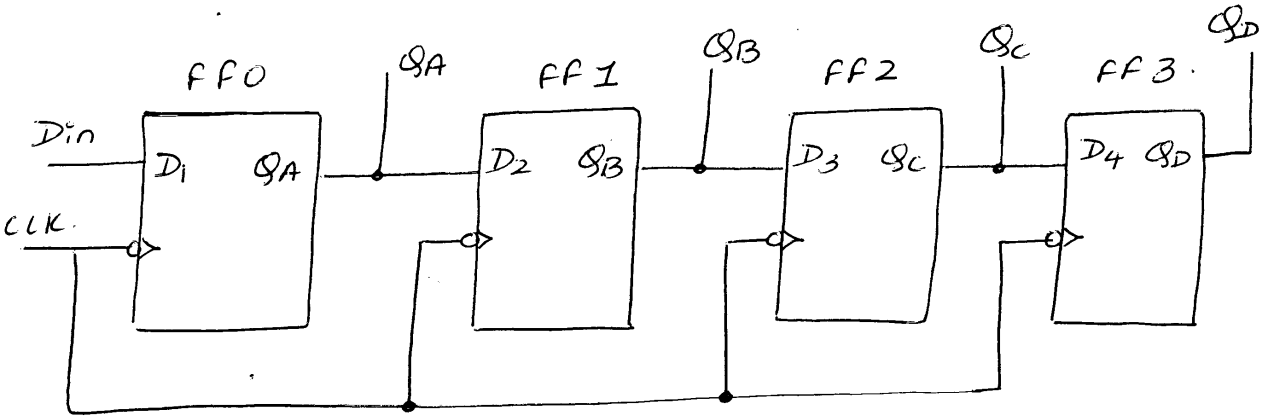


FIG: SIPO Shift Register.

Initially, register is cleared, so, $Q_A Q_B Q_C Q_D = 0000$.

a) When the data is applied serially, as 1111 ie; left most 1 is applied as Din,

$Din = 1$, $Q_A Q_B Q_C Q_D = 0000$.

The arrival of the first falling clock edge sets the left most flip flop, and the stored word becomes,

$Q_A Q_B Q_C Q_D = 1000$.

b) When the next falling clock edge hits, the Q_B flip flop sets and the register contents become,

$Q_A Q_B Q_C Q_D = 1100$.

c) The third falling clock edge results in,

$Q_A Q_B Q_C Q_D = 1110$.

d) The fourth falling clock edge gives,

$Q_A Q_B Q_C Q_D = 1111$

Here at clock 4, we are seeing the applied I/p at D.

In SIPO, we need n clocks to see the n-bit data at the o/p.
ie; we need 4 clocks to see 4-bit data at the o/p.

clock	Din	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	1	1	1	1	1

FIG: Shift operation.

3> Parallel In - Parallel Out (PIPO) Register.

For PIPO shift Registers, all data bits appear on the parallel o/p's immediately following the simultaneous entry of the data bits.

The following circuit is a 4-bit PIPO shift register constructed by D flipflop's.

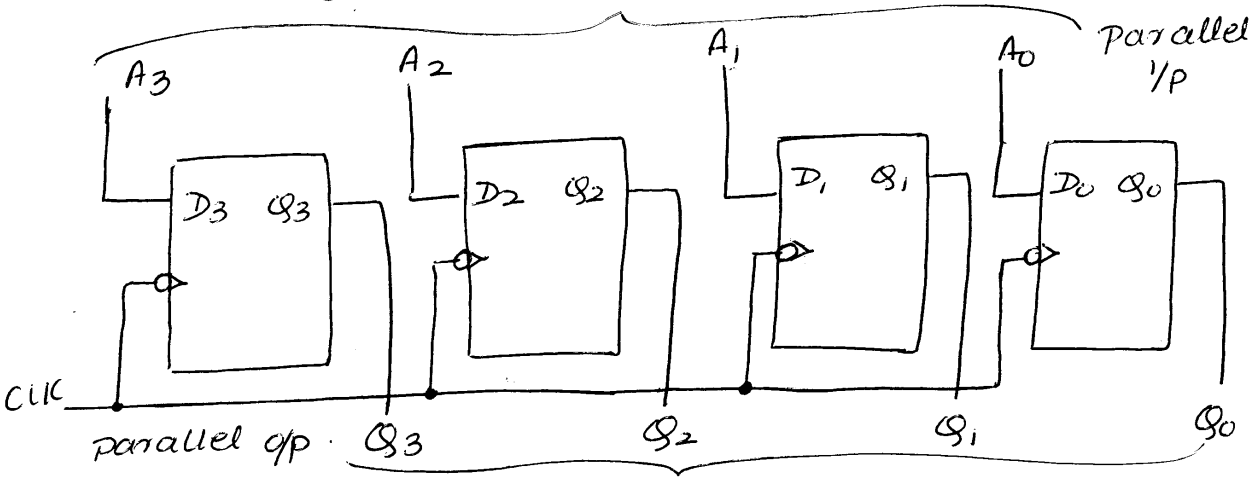


FIG: PIPO Shift-Register.

At first, all the o/p's are reset to '0'.

At clock '0',

$$Q_A = Q_B = Q_C = Q_D = \text{"0000"}.$$

Assign values $D_1 = 1$, $D_2 = 1$, $D_3 = 1$, $D_4 = 1$

At clock 1,

$Q_A = 1$ $Q_B = 1$ $Q_C = 1$ $Q_D = 1$.

Clock	D_1	D_2	D_3	D_4	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1

In PIPO, we need one clock to see the register o/p at a time.

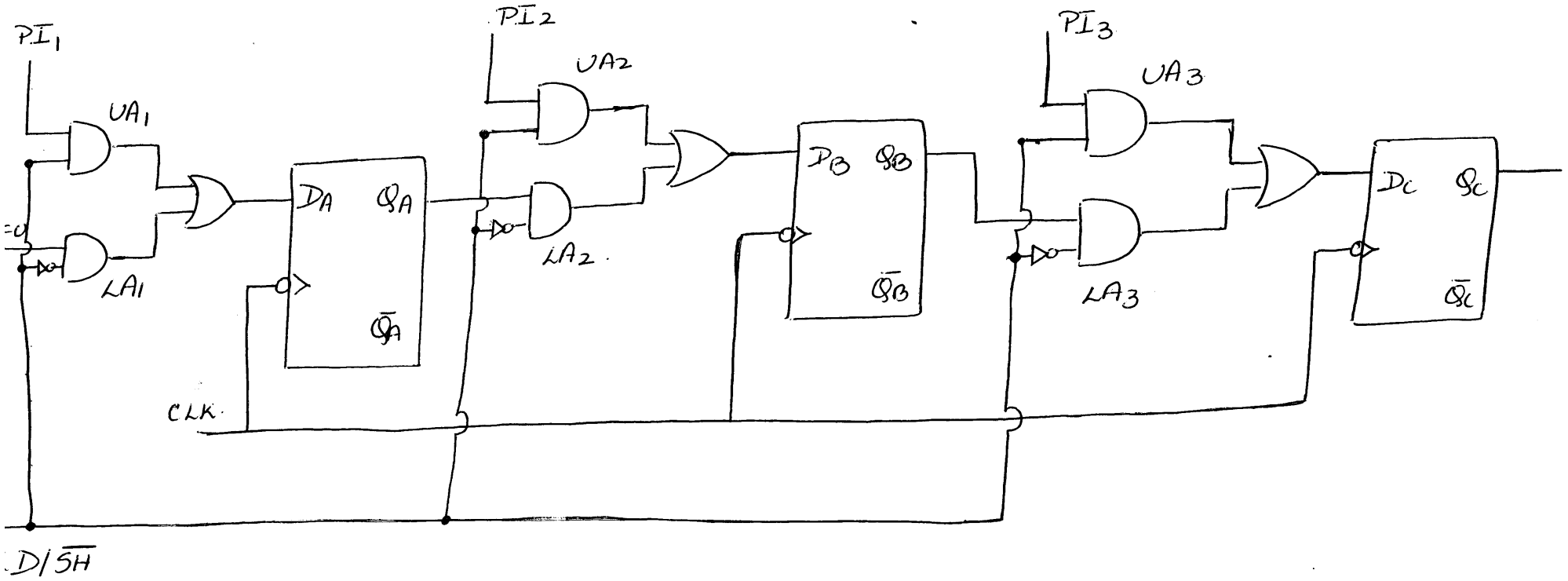
4) Parallel In - Serial Out (PISO) Register.

In this type, the bits are entered in parallel ie; simultaneously into their respective stages on parallel lines.

The figure below illustrates a four-bit Parallel In - Serial out register.

There are 4-i/p lines, PI_1, PI_2, PI_3 for entering data in parallel into the register.

SHIFT/$\overline{\text{LOAD}}$ is the control i/p which allows shift or loading data operation of the register. When SHIFT/$\overline{\text{LOAD}}$ is low, gates UA_1, UA_2, UA_3 are enabled, allowing each i/p data bit to be applied to D i/p of its respective flip flop. When a clock pulse is applied, the flip-flops with D=1 will SET and those with D=0 will RESET. Thus all 4 bits are stored simultaneously.



NOTE :

PI = Parallel Input SI = Serial Input

LD/SH = Load / Shift UA = Upper And Gate LA = Lower And Gate.

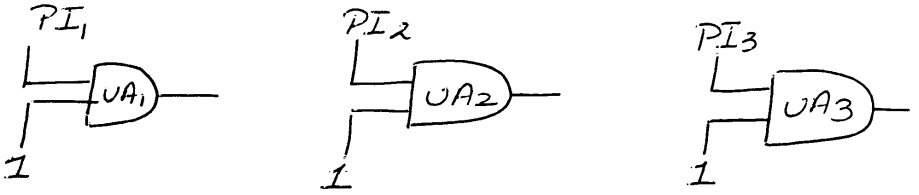
If LD/SH = 1, the circuit loads the data parallely through the parallel inputs PI1, PI2, PI3

If LD/SH = 0, the circuit shifts the data serially.

Parallel Loading -

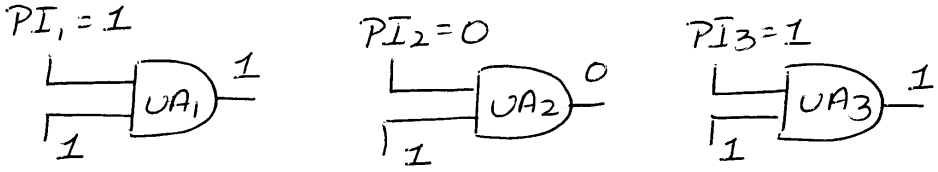
* Apply $LD/\overline{SH} = 1$, which makes one of the i/p's of upper AND gates UA1, UA2, & UA3 with 1.

ie;

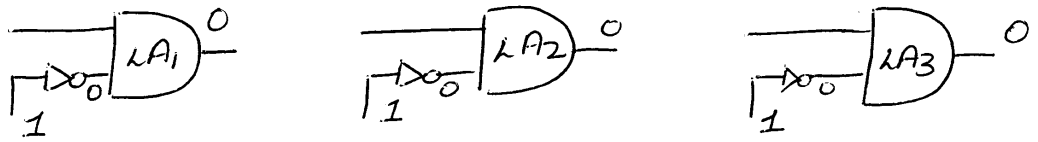


* Assign $PI_1 = 1$, $PI_2 = 0$, $PI_3 = 1$.

∴

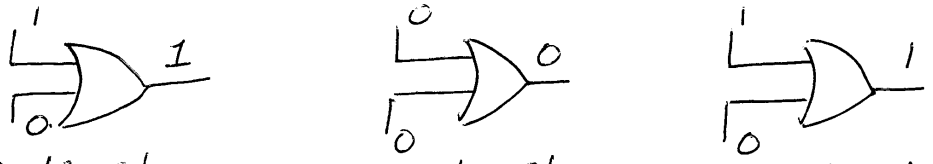


* With $LD/\overline{SH} = 1$, one of the lower and gate is treated as '0'.



And hence, the lower AND gates are result with '0'

wrt OR gates,



or gate of UA1 & LA1

or gate of UA2 & LA2

or gate of UA3 & LA3.

The OR gate o/p's are feeded to the i/p's of DA, DB & DC respectively.

ie; $D_A = 1$, $D_B = 0$, $D_C = 1$.

with the inputs of $D_A = 1$, $D_B = 0$ and $D_C = 1$ Apply the clk.

clock	D_A	D_B	D_C	Q_A	Q_B	Q_C.
↓	1	0	1	1	0	1

We can see that, we applied, $PI_1 = 1$, $PI_2 = 0$ & $PI_3 = 1$, All these values are replicated at $Q_A = 1$, $Q_B = 0$, $Q_C = 1$

By the above scenario, we come to know that the Data is loaded parallely.

Serial Shifting -

To shift the data serially, apply $LD/\overline{SH} = 0$. which makes $\overline{Shift} = 0$, which impacts $Shift = 1$.

with $LD/\overline{SH} = 0$, it makes one of the i/p's of $UA_1, UA_2, UA_3 = 0$ which makes the results of upper and gate o/p as '0'.

The lower and gates LA_1, LA_2 & LA_3 is having one of the i/p's with value '1'.

$LD/\overline{SH} = 1$

clk	D_A	D_B	D_C	Q_A	Q_B	Q_C
↓	1	0	1	1	0	1

In brief

clk	Q_A	Q_B	Q_C
clk 1 ↓	1	0	1

$LD/\overline{SH} = 0$

clk	SI	Q_A	Q_B	Q_C
clk 2 ↓	0 →	0	1	0
clk 3 ↓	0 →	0	0	1
clk 4 ↓	0 →	0	0	0

In Piso, we can see the o/p at the last flip flop.

At clk 1, $Q_C = 1$

At clk 2, $Q_C = 0$

At clk 3, $Q_C = 1$.

Hence, the o/p can be seen at n^{th} clock.

Circuit Operation –

If $LD/\overline{SH} = 1$, the circuit loads the data through the parallel inputs.

If $LD/\overline{SH} = 0$, the circuit shifts the data serially.

When $LD/\overline{SH} = 1$, the D_A, D_B and D_C are loaded with parallel inputs $\overline{PI_1}$, $\overline{PI_2}$, $\overline{PI_3}$.

When $LD/\overline{SH} = 0$, the $\overline{SI} = 0$ is feeded to D_A,

Q_A is treated as input of D_B.

Q_B is treated as input of D_C.

Hence when $LD/\overline{SH} = 1$

$\overline{PI_1} = D_A = 1$, $\overline{PI_2} = D_B = 0$, $\overline{PI_3} = D_C = 1$

Here we are using 3 FF's, to see all the possible o/p's serially. we will require 3 clocks.

In general, n FF's = n clocks to see the o/p.

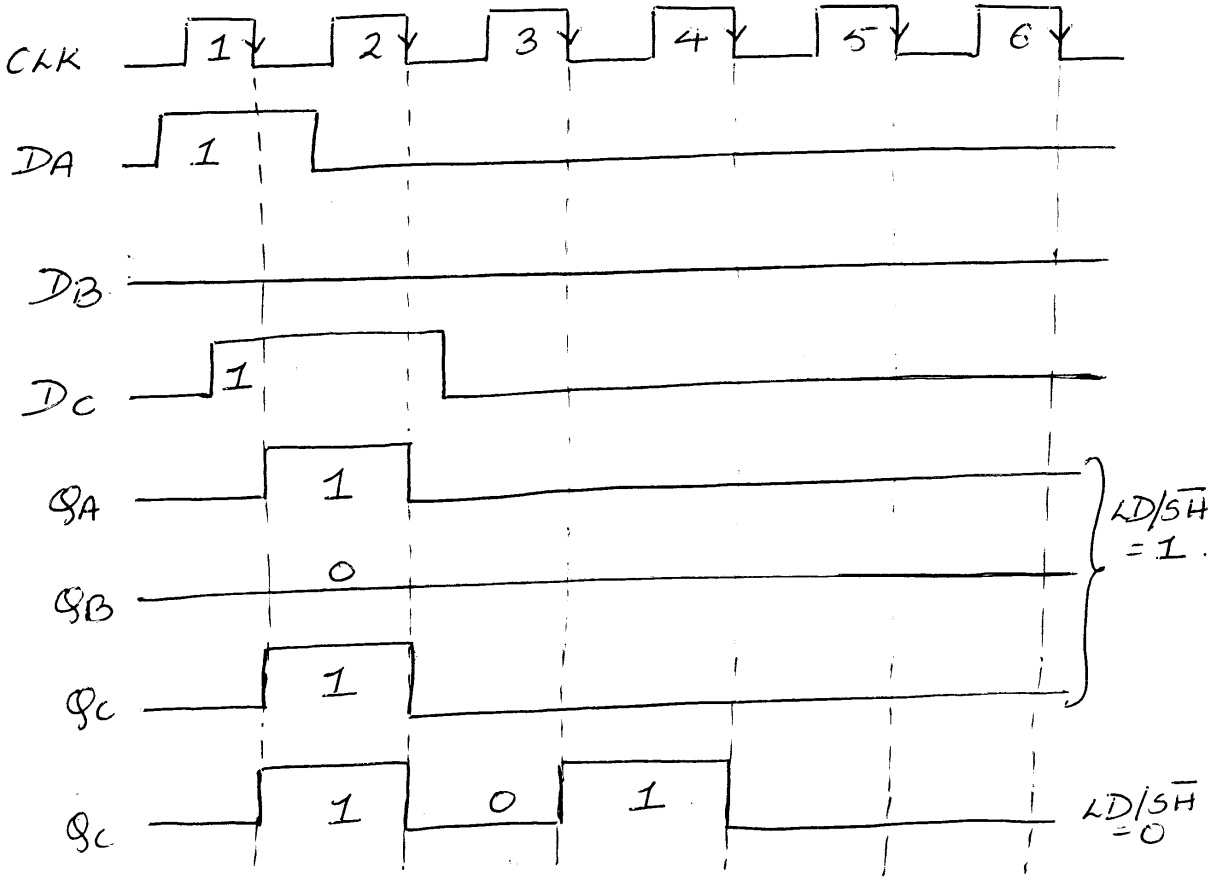


Fig: Timing Diagram of PISO.

UNIVERSAL SHIFT REGISTER.

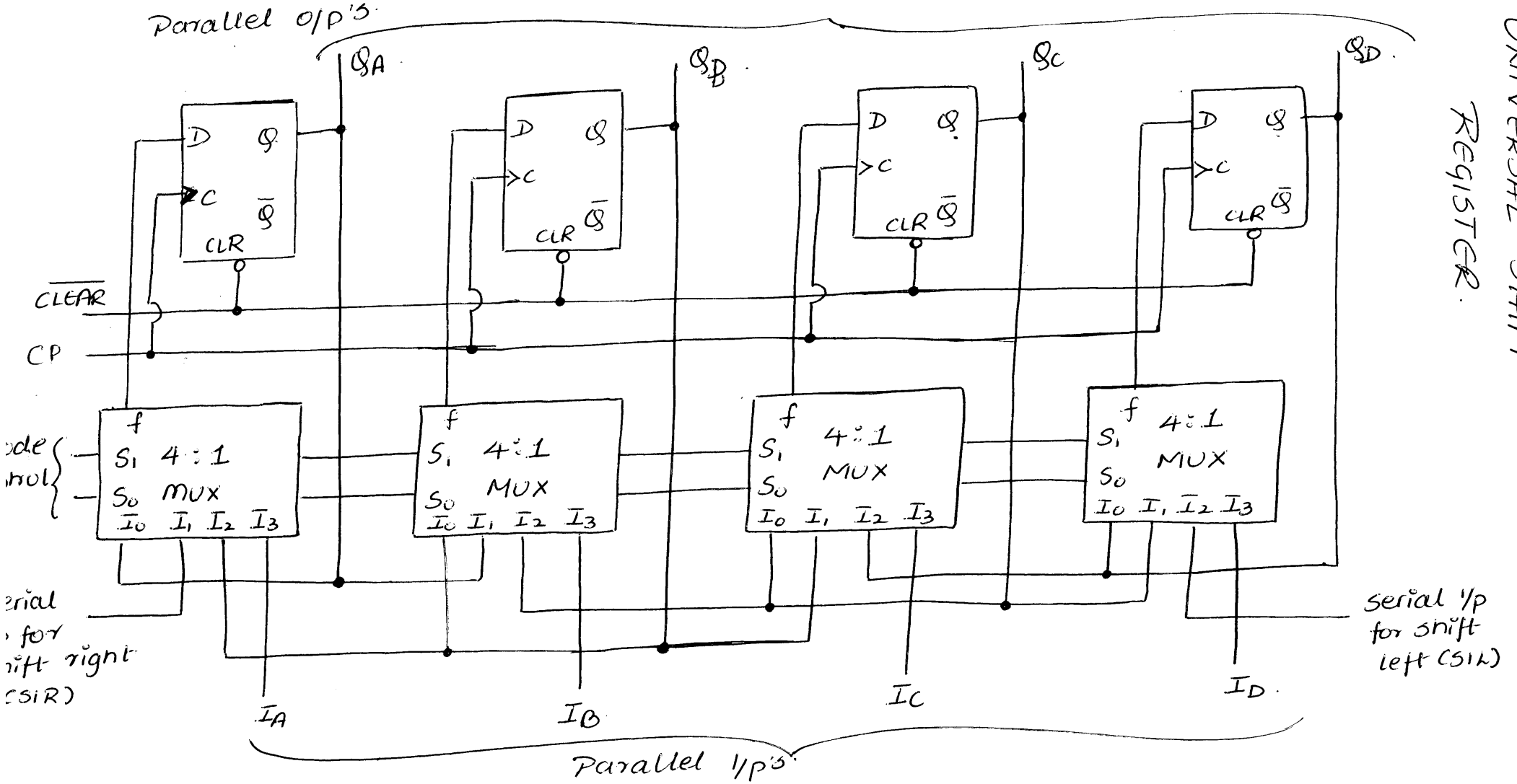


FIG: LOGIC DIAGRAM of UNIVERSAL SHIFT REGISTER.
(4-bit universal Shift Register)

UNIVERSAL SHIFT REGISTER-

A register capable of shifting in one direction only is a unidirectional shift register.
A register capable of shifting in both directions is a bidirectional shift register.
If a register has both shifts (right shift & left shift) & parallel load capabilities, it is referred to as "Universal shift Register".

The universal shift register performs the operations of all the types of shift registers. ie; SISO, SIPO, PISO and PIPO., Left shift register and Right shift Register.

Based on the following conditions it performs as different shift Register.

Mode control		Function	Next State			
S_1	S_0		Q_{An+1}	Q_{Bn+1}	Q_{Cn+1}	Q_{Dn+1}
0	0	Hold	Q_{An}	Q_{Bn}	Q_{Cn}	Q_{Dn}
0	1	Shift Right	Din	Q_{An}	Q_{Bn}	Q_{Cn}
1	0	Shift Left	Q_{Bn}	Q_{Cn}	Q_{Dn}	Din
1	1	Parallel Load	A	B	C	D

Fig: Mode control and Register operation

If $S_1 S_0 = 0\,0$, The shift register holds the previous value.

If $S_1 S_0 = 0\,1$, The data will be shifted Right

If $S_1 S_0 = 1\,0$, The data will be shifted Left.

If $S_1 S_0 = 1\,1$, The new data will be loaded parallely.

APPLICATIONS OF SHIFT REGISTERS -

i> RING COUNTER -

The figure below shows the logic diagram for a 4-bit ring counter.

As shown in the figure, the 'Q' output of each stage is connected to the D input of the next stage and the output of the last stage is fed back to the input of the first stage.

The $\overline{CLR}$ followed by $\overline{PRE}$ makes the output of the first stage to '1' and the remaining outputs are '0'.

i.e; Q_A is 1 and Q_B, Q_C, Q_D are '0'.

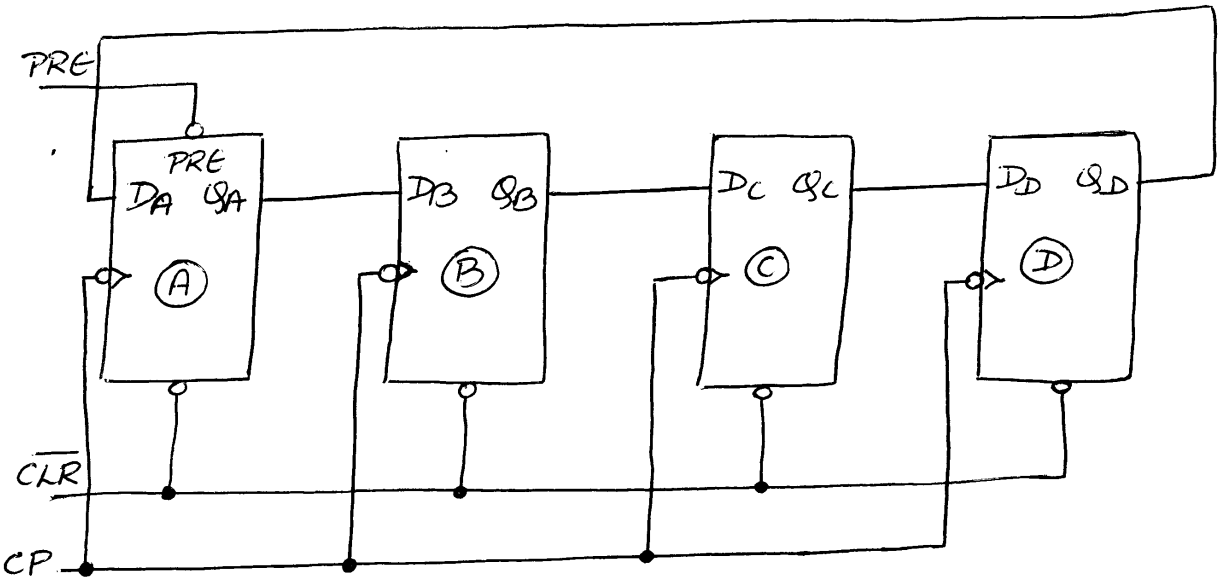


FIG: A 4-bit Ring Counter.

The first clock pulse produces $Q_A = 1$ and the remaining outputs are '0' According to the clock pulses applied at the clock input CP, a sequence of 4 states are produced. These states are listed in the Table given below.

In the above circuit, Assume the values for

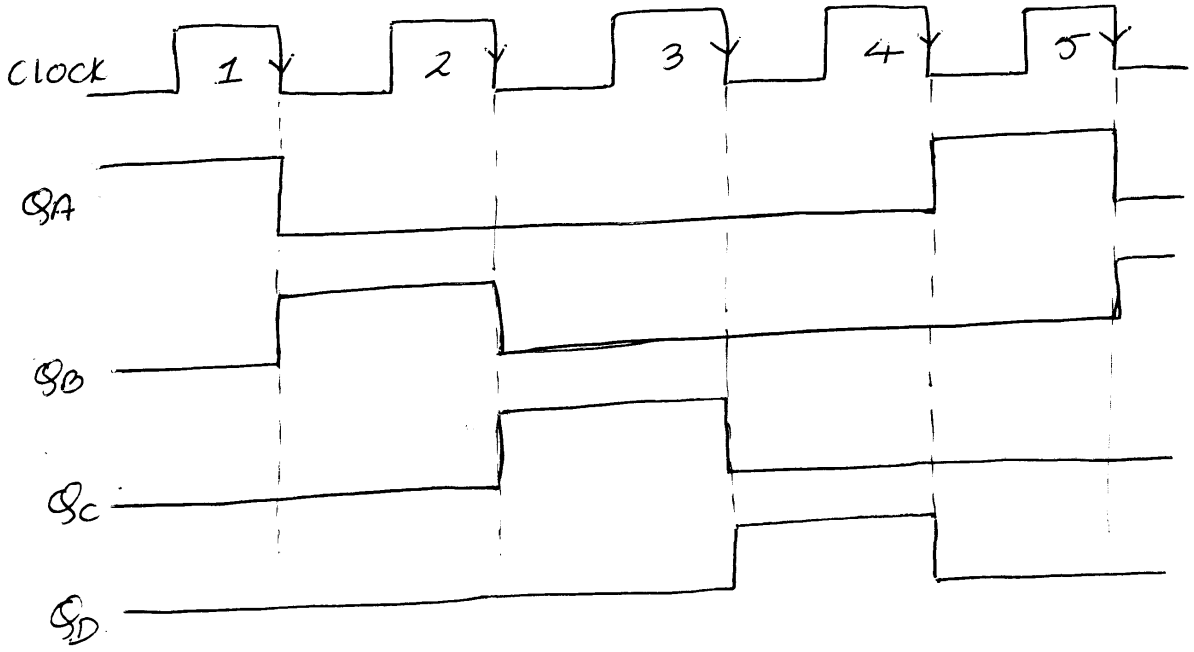
$Q_A = 1$, $Q_B = 0$, $Q_C = 0$, $Q_D = 0$

Clock Pulses	Q_A	Q_B	Q_C	Q_D
1 ↓	1	0	0	0
2 ↓	0	1	0	0
3 ↓	0	0	1	0
4 ↓	0	0	0	1
5 ↓	1	0	0	0

[cycle Repeats]

As shown in the table above, 1 is always retained in the counter and simply shifted around the ring, advancing one stage for each clock pulse. In this case, 4 stages of flip flops are used. So a sequence of 4 states is produced and repeated.

The ring counter can be used for counting the number of pulses. The number of pulses counted is read by noting which flip flop is in state 1. Since there is one pulse at the o/p for each of the N clock pulses, this circuit is also referred to as a divide-by-N-counter or an N:1 scalar.



ii) Johnson Counter / Twisted Tail Counter / Switched Tail Counter.

In a Johnson counter, the Q output of each stage of flip flop is connected to the D input of the next stage.
The single exception is that the complement output of the last flip flop is connected back to the D input of the first flip flop.

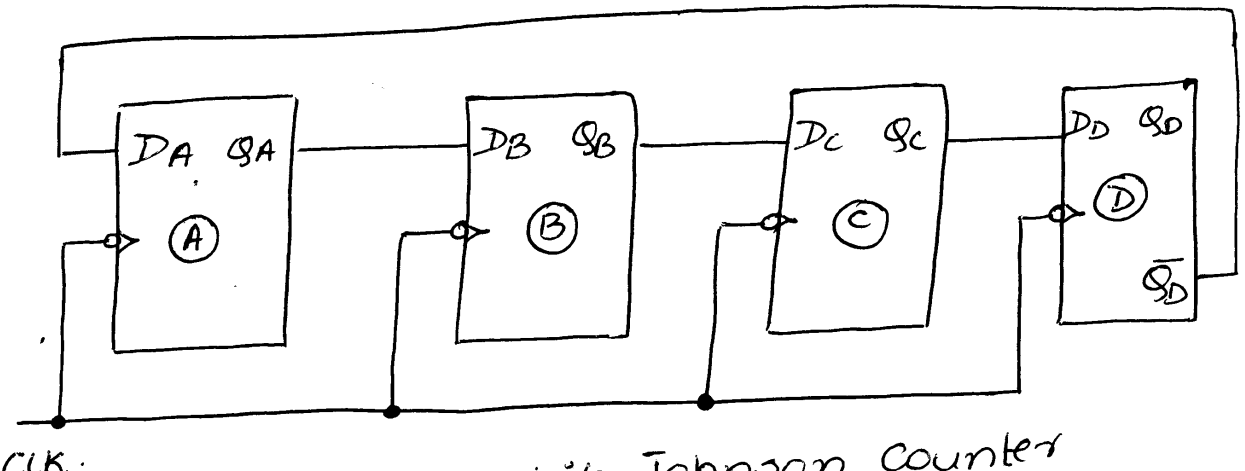


FIG: 4-bit Johnson Counter

As shown in the figure, there is a feedback from the rightmost flip flop's complemented output to the leftmost flip flop's input.

Initially, the register (all the flip flop's) are cleared.
So all the outputs, Q_A, Q_B, Q_C and Q_D are zero. The output of the last stage, Q_D is '0'. Therefore, the complemented output of the last stage, $\overline{Q}_D$ is '1'. This is connected back to the D input of the first stage. So, $D_A = 1$

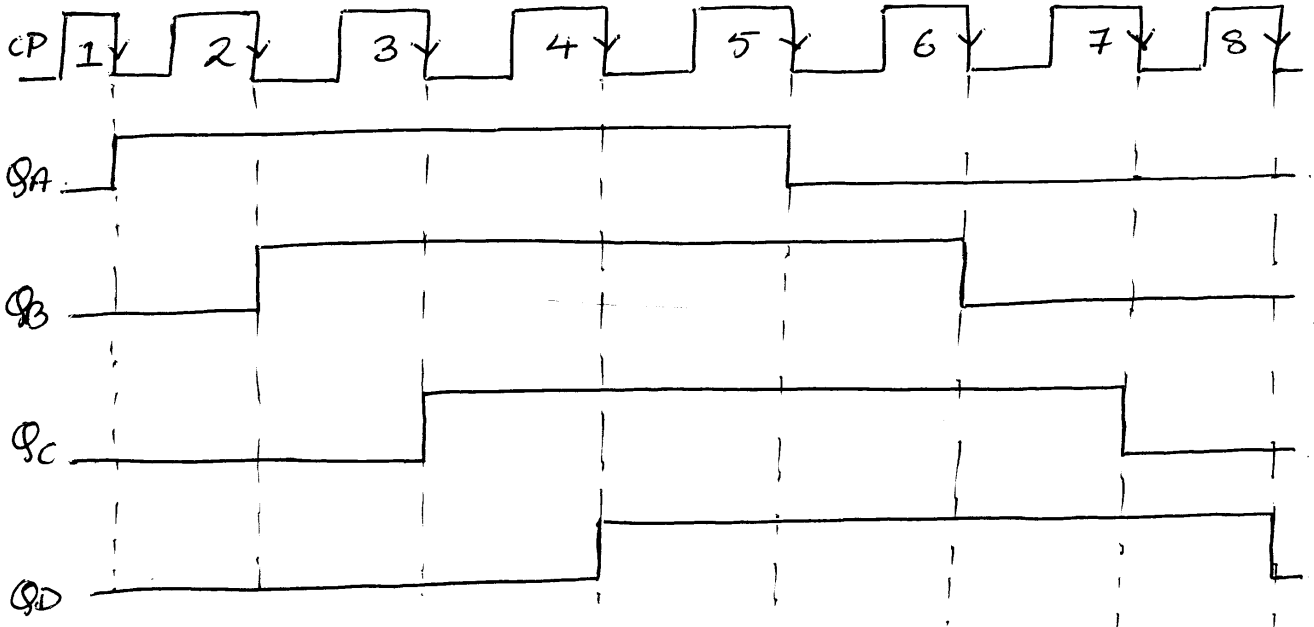
The first falling clock edge produces. $Q_A = 1$ and $Q_B = 0$, $Q_C = 0$ and $Q_D = 0$, since D_B, D_C and D_D are '0'.

The next clock pulse produces $Q_A = 1$, $Q_B = 1$, $Q_C = 0$ and $Q_D = 0$

The sequence of states is summarized in the table given below. After every 8 states, the same sequence is repeated.

Clock Pulse	Q_A	Q_B	Q_C	Q_D
0 ↓	0	0	0	0
1 ↓	1	0	0	0
2 ↓	1	1	0	0
3 ↓	1	1	1	0
4 ↓	1	1	1	1
5 ↓	0	1	1	1
6 ↓	0	0	1	1
7 ↓	0	0	0	1

Table: Four bit Johnson sequence.



Timing sequence of a 4-bit Johnson counter.

iii) Sequence Generator and Sequence Detector -

Sequence Generator.

The shift register can be used to generate a particular bit pattern repetitively.
The figure below shows the basic block diagram of a sequence generator.
Here, left most flip flop input accept the serial input and the right most flip flop gives the serial data output.
It is important to note that the serial data output signal is connected as a serial data in.
On every clock pulse, the data shift operation takes place. We get the loaded bit pattern at the serial output in a sequence.
Same bit pattern is again loaded in the register since serial o/p is connected serial in of the register.
Thus, the circuit generates a particular bit pattern repetitively.

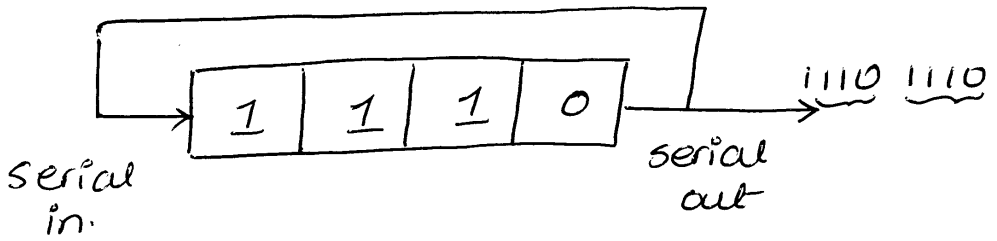


Fig: 4-bit Sequence Generator.

Sequence Detector -

The shift register can be used to detect the desired sequence.

The detection process requires 2 registers. One register stores the bit pattern to be detected, i.e; R_2 and the other register accepts the input data stream ie; R_1. input data stream enters a shift register as serial data in and leaves as serial out.

In every clock cycle, bit wise comparisions of these 2 registers are done using EX-NOR gates as shown in the figure. We know that the 2-I/p EX-NOR gate gives logic high O/p when both I/p's are either low or high. ie; when both are equal.

When O/p's of all the EX-NOR gates are logic high we can say that all bits are matched & hence the desired bit pattern is detected. The final output which indicates that the pattern is detected is taken from 4-I/p AND gate.

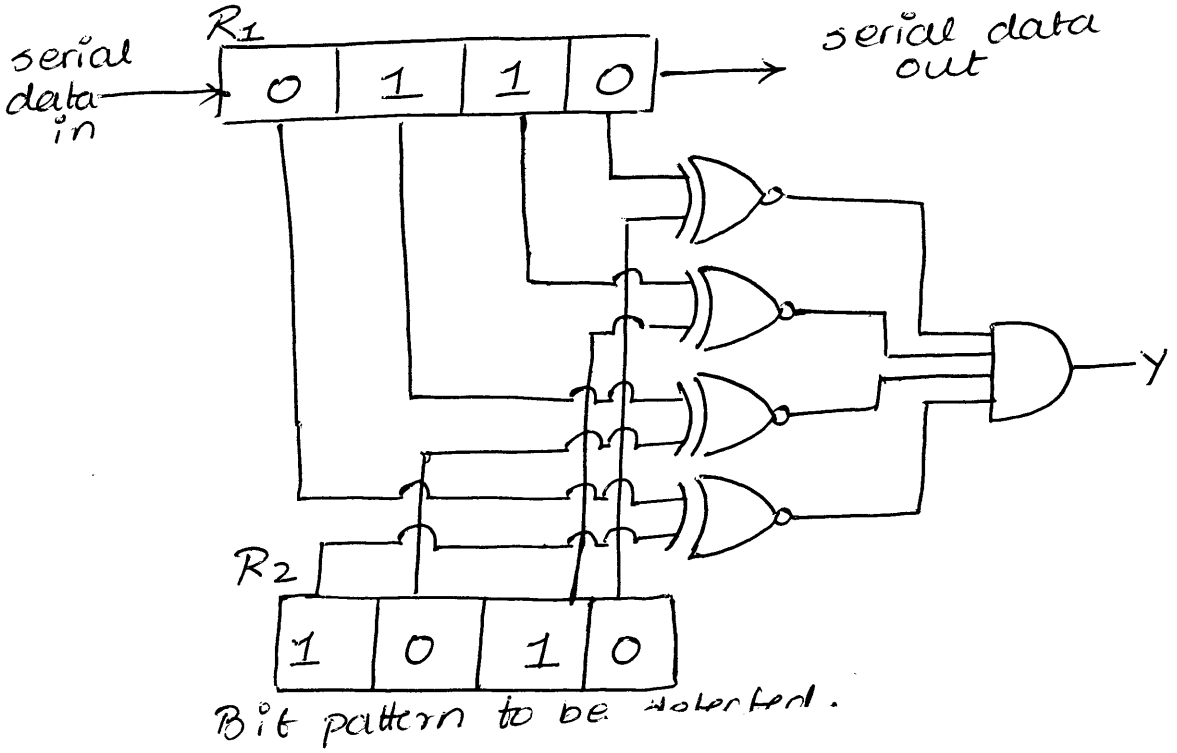


Fig: 4-bit sequen

The circuit that can detect a binary sequence is shown in the figure.

→ It has one register to store binary word we want to detect from binary stream.

→ And a SISO register is there where the data is serially entered and serially going out.

→ At every clock, the bitwise comparision of these 2 registers are done through EX-NOR gates as shown in figure.

→ The final output taken from 4-input AND gate ie, Y.

At the 1st step,

The first sequence 0101 is checked with the sequence 1010 resulting in 'zero', because all the bits are mismatched and the output Y is 'zero'.

In the next clk, the new sequence in upper register is 1010 and sequence to be detected in lower register is 1010.

The sequences are checked using XNOR and every bit is matched in upper & lower register, sending high signals to the AND gate resulting high o/p.
And Hence the sequence is detected.

iv) Serial Adder-

The addition of 3 bits (ie; Full Adder) can be done by using shift registers. The fig shows how serial addition takes place.

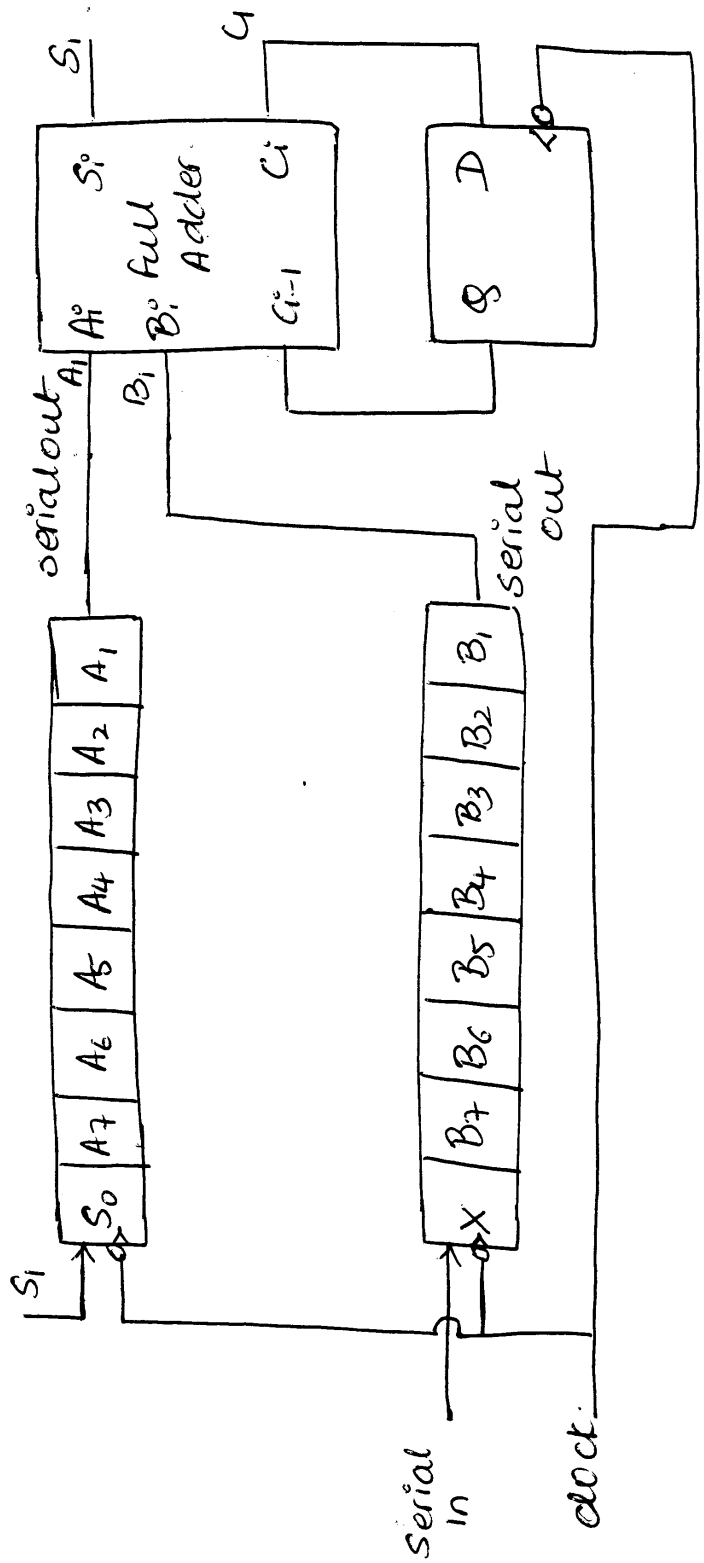


Fig: Serial Adder.

Two 8-bit nos, to be added. ($A_7 A_6 \ldots A_0$) and ($B_7 B_6 \ldots B_0$) are loaded in two 8-bit shift registers A & B.

The LSB is in right most position in 2 registers.
The MSB is in left most position in 2 registers.
Serial data out of A & B are fed to data i/p's of full adder. The carry-in is fed ~~to data~~ from its own carry o/p, which is initially cleared (ie; 0).
Both registers and D FF's are triggered by the same clock.

The sum(S) o/p of FA is fed to serial data in of shift register A.

- Serial Addition of two 8-bit numbers (Register values are at 2nd clk cycle).

Working:

→ LSB's of two no's A_0 & B_0 appearing out of respective registers are added by FA during the 1st clock and generate sum (S_0) & carry (C_0). So is available at serial i/p of register & C_0 at i/p of D FF.

→ At the next clk, S_0 becomes MSB of A & C_0 appears at D FF's o/p. ∴ in 2nd clk cycle Full Adder is fed by 2nd bit (A_1 & B_1) of A & B and previous carry C_0. S_1 & C_1 are generated & made available at serial data in of A register & i/p of D flipflop respectively.

→ This process goes on & its stopped by inhibiting the clk after 8 clk cycles.

* Register Implementation of HDL -

i) Give the HDL code for a shift register of 5-bits constructed using D flip flops.

```
module SR5 (D, CLK, T);
input CLK, D;
output T;
reg P, Q, R, S, T;
always @(negedge CLK)
begin
    P <= D;
    Q <= P;
    R <= Q;
    S <= R;
    T <= S;
end
endmodule
```

ii) HDL code for switched Tail counter / Johnson counter.

```
module STC (CP);
input CP;
output QA, QB, QC, QD;
reg QA, QB, QC, QD;
always @(negedge CP)
begin
    QA <= ~QD;
    QB <= QA;
    QC <= QB;
    QD <= QC;
end
endmodule
```