REGISTERS AND COUNTERS

Applications of Flip-flops :-

(i) It can be used as a memory element.
(ii) It can be used to eliminate key debounce.
(iii) It is used as a basic building block in sequential circuits such as counters and Registers.
(iv) It can be used as a delay element.

REGISTERS :-

(i) Register is a group of flip-flops.
(ii) n-bit Register consists of 'n' number of flip-flops and it stores 'n' bit binary information.

Types of shift Registers :-

1. Serial In serial OUT shift registor (SISO) :-

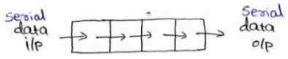


Fig:- Data flow in SISO shift right registor

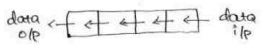


Fig:- Data flow in SISO shift left registor.

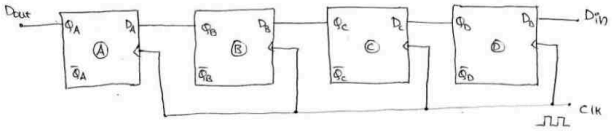


Fig:- 4-bit SISO shift left registor

i/p to reg 1011.

CLK	reg. o/p (Q_A)	Q_B	Q_C	Q_D	
0	0	0	0	0	
↑	0	0	0	1	
↑	0	0	1	0	
↑	0	1	0	1	
↑	1	0	1	1	→ i/p data stored in register.
↑	0	1	1	0	
↑	1	1	0	0	
↑	1	0	0	0	→ all 4 data i/p bits are transfered to o/p side
↑	0	0	0	0	→ reg. cleared.

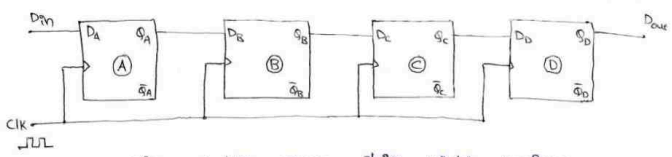


Fig:- 4-bit SISO shift right registor

2. Serial IN Parallel OUT shift Registor (SIPO) :-

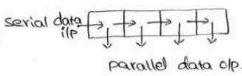


Fig:- data flow in SIPO shift right registor.

In this case, data bits enters into register in serial, but o/p is taken in parallel.

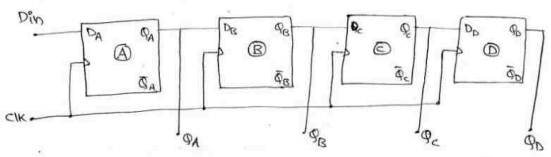


Fig:- 4-bit SIPO shift right register.

D_{in} → Serial data i/p

$Q_A Q_B Q_C Q_D$ → 4-bit parallel data o/p.

Register with parallel load :-

* the transfer of new information into a register is refered to as loading Register.
* If all the bits of Register are loaded simultaneously with a common clock pulse, we say that the loading is done in parallel.

3. Parallel IN parallel OUT shift register (PIPO):-

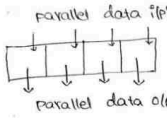


Fig:- data flow in PIPO shift register.

Data bits are entered into reg in parallel and o/p is taken in parallel.

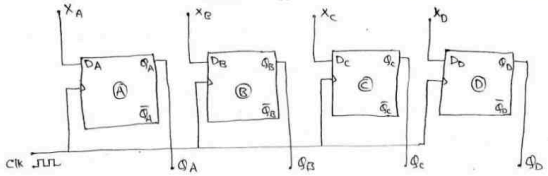


Fig:- 4-bit PIPO shift register.

$X_A X_B X_C X_D$ → parallel data i/p's

$Q_A Q_B Q_C Q_D$ → 4-bit parallel data o/p.

07/10/2014

4. Parallel In serial out shift Register (PISO) :-

Fig:- Dataflow in PISO shift right register.

I/p data entered into register parallely but o/p is taken serially.

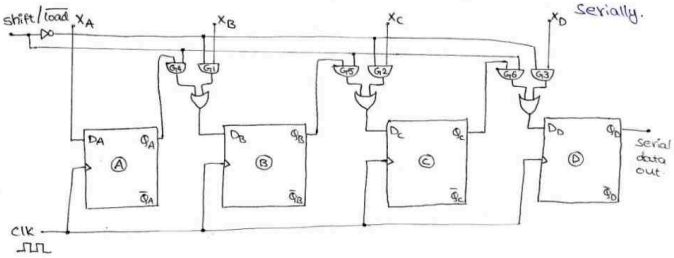


Fig:- 4-bit PISO shift right register.

$X_A X_B X_C X_D$ → parallel data i/p's

If shift/$\overline{load}$ = 0 then parallel data loads into reg.

when shift/$\overline{load}$ = 1, reg shifts the stored data to right

* Asynchronous (or) direct i/p's of flip-flops :-

(i) Set (or) pre-set :-

If set i/p is high then flip-flop o/p is '1' for any data i/p) irrespective of data i/p's & clk i/p

If set is high i.e 1 then ⇒ flip-flop will set

If $\overline{set}$ is low i.e 0 then ⇒ flip-flop will set.

(ii) clear (or) reset :-

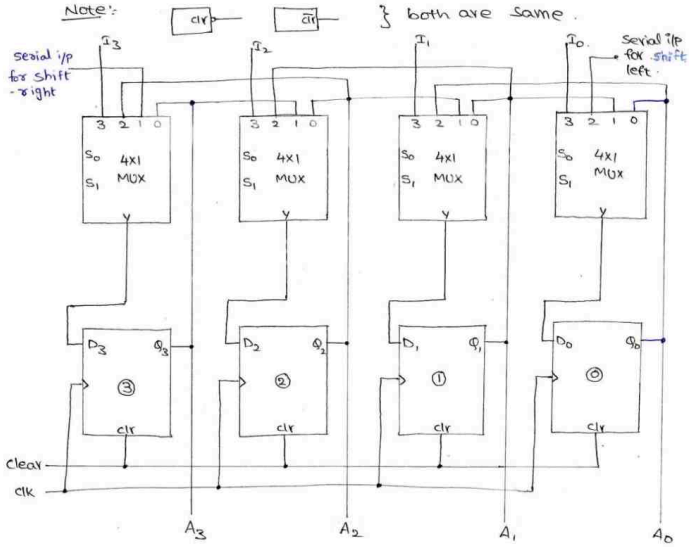
If reset i/p is high then flip-flop o/p is '0' (for any data i/p) irrespective of clk & data i/p's

* Universal shift register :-

If the register has both shift (left & Right) and Parallel load capabilities, it is refered to as Universal shift register.

→ acts like PISO, PIPO, SISO, SIPO
→ acts like shift (right & left)
→ (i.e) performs all operations.

Note :- clr, clr } both are same.



S_0, S_1 → are common selection i/p's, but not o/p's of previous mux. For simplicity we have drawn in the above way.

Fig:- 4-bit Universal shift register.

Function table :-

Mode control S_1	S_0	Reg. operation
0	0	NO change
0	1	shift right
1	0	shift left
1	1	parallel load.

$I_3 I_2 I_1 I_0$ → parallel i/p's
$A_3 A_2 A_1 A_0$ → parallel o/p's.

for '00' we observe all '0' connections in MUX, gives o/p to D_1, D_2, D_3, D_0.

Similarly, for '11' we observe all '3' connections in MUX, gives o/p to D_0, D_1, D_2, D_3.

* Serial Adder :-

It performs serial addition.

Block diagram :-

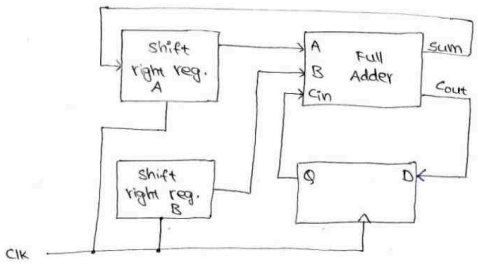


Fig: Serial Adder

let us consider two 4-bit numbers → 1011 and 1001 → Intermediate carry

1011
+ 1001
10100
end carry, Sum

considering A & B are 4 bit registers (SISO shift right reg.)

A reg. content is 1011.
B reg. content is 1001.

serial adder performs operation.

$A \leftarrow A+B$ Arrow indicates dataflow direction.

when clk is applied Reg.'s performs shift right operation.

During shift right, LSB comes out & goes to full Adder and FF adds A, B, C_{in} bits and transfers sum to A reg. & C_{out} to D-FF.

After performing serial addition contents of

reg A → 0100 (sum)

B → 0000

FF D → 1 (end carry)

Serial Transfer :-

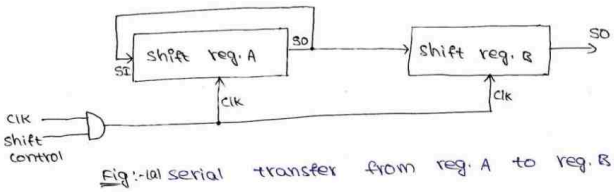


Fig :- (a) serial transfer from reg. A to reg. B.

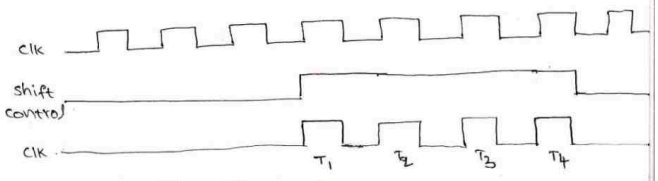


Fig :- (b) Timing diagram

considering A & B are 4-bit SISO shift right reg.'s.

SO → serial out ; SI → serial in.

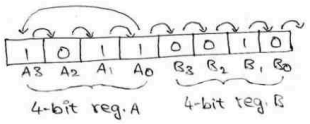
The above ckt performs operation

$B \leftarrow A$ (or) $A \rightarrow B$ Arrow indicates direction of dataflow.

After performing register transfer operation A=B

Fig (a) performs shift operation only when shift control is high because reg.'s receives clk i/p only when shift control = 1.

Initial values of A, B ⇒



serial transfer

Timing pulse	Shift reg. A	Shift reg. B
Initial value of reg.	1011	0010
After T_1	1101	1001
T_2	1110	1100
T_3	0111	0110
T_4	1011	1011

After T_4, both A & B reg.'s content is same.

(i.e) A = B.

COUNTERS :-

A counter is a register capable of counting the number of clock pulses arriving at it clock input. Count represents the number of clock pulses arrived.

(or)

A counter is an essentially a register that goes through a predetermined sequence of binary states.

* counters are available in two categories.

1. Asynchronous counters.
2. Synchronous counters

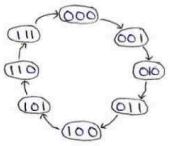
Asynchronous counters	synchronous counters
1. In this type of counter, flip-flops are connected in such a way that o/p of 1st FF drives the clk for the next FF.	1. In this type, there is no connection b/w o/p of 1st FF and clk i/p of the next FF.
2. All the FF are not clocked simultaneously.	2. All the FF are clocked simultaneously.
3. logic circuit is very simple even for more number of states	3. Design involves complex logic circuits as no. of states increases.
4. Main drawback of these counters is their low speed as the clk is propagated through no. of FF before it reaches last FF.	4. As the clk is simultaneously given to all FFs there is no problem of propagation delay. Hence they are high speed counters

Asynchronous (or) Ripple (or) serial counters (or) Non-synchronous counter:-

1. 3-bit Asynchronous up counter :-

An upcounter is a counter which counts in the upward direction. (i.e) 0, 1, 2, 3, ... n

State diagram :-



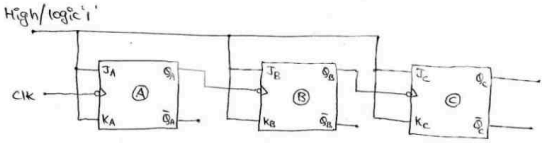
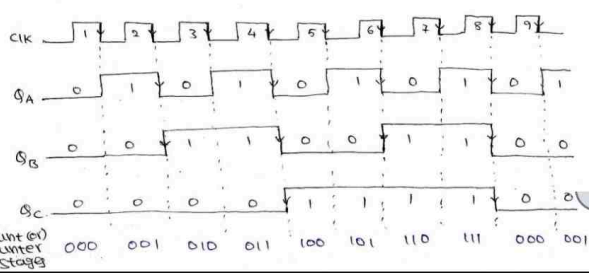


Fig:- 3-bit ripple up counter using -ve edge triggered JK FF's.

counter o/p → Q_C Q_B Q_A
MSB LSB

o/p waveforms



* the number of states through which the counter passes before returning to the starting state is called the Modulus of the counter.

(or)

the Mod number of a counter is the total no. of states it sequences through in each complete cycle

* Since a 3-bit counter has 8 states it is called a Mod-8 (or) Modulus-8 counter. (or) Modulo-8 counter.

3-bit ripple up counter using T-FF's:-

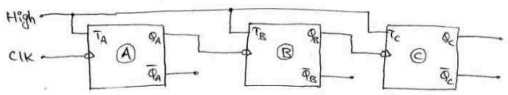


Fig:- 3-bit Asynchronous up counter using -ve edge triggered T-FF's.

counter o/p → Q_C Q_B Q_A
MSB LSB.

Note:- For designing ripple up counter using +ve edge triggered FF's connect $\bar{Q}$ o/p's to clk i/p's of next flip-flops.

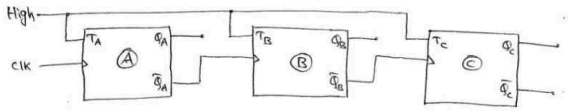


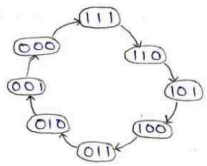
Fig:- 3-bit ripple up counter using +ve edge triggered T-FF's.

counter o/p → Q_C Q_B Q_A
MSB LSB.

2. 3-bit Asynchronous down counter :-

Down counter counts in downward direction. (i.e) n, n-1, n-2, ... 2, 1, 0.

State diagram:-



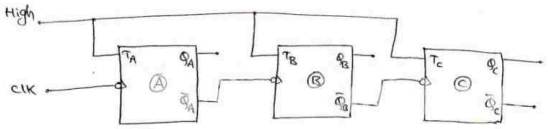
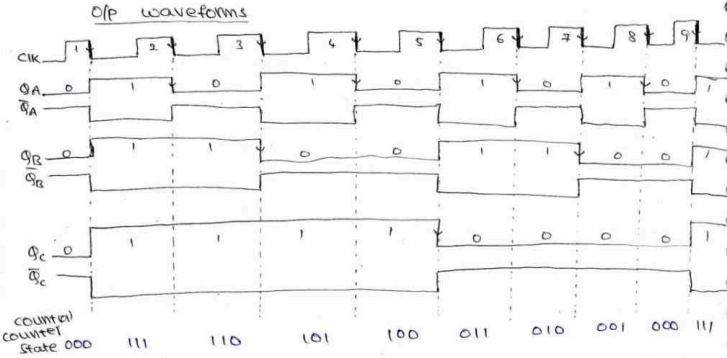


Fig:- 3-bit ripple down counter using -ve edge triggered T-FF's.

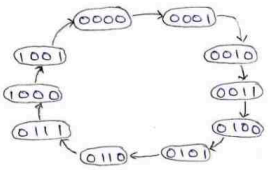
counter o/p → Q_C Q_B Q_A
MSB LSB

o/p waveforms



3. Asynchronous Mod-10 counter (or) Decade counter (or) BCD counter :-

State diagram:-



for NAND gate draw bubbles for clr i/p

for AND gate Dont draw bubbles

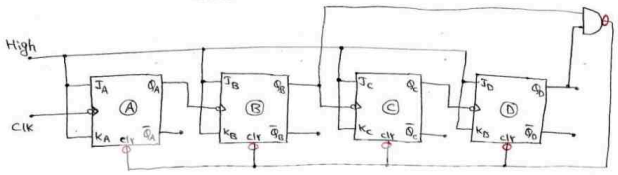


Fig:- MOD-10 ripple counter.

counter o/p → $Q_D\ Q_C\ Q_B\ Q_A$ (MSB … LSB)

* When counter o/p is $Q_D Q_C Q_B Q_A$ 1010, o/p of And gate becomes 1. Since o/p of And gate is connected to reset (or) clear i/p's of FF's, counter enters into 0000 state after 1010 state.

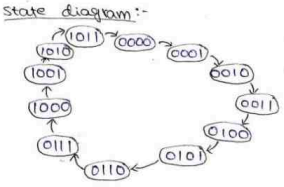
Truth table:-

CLK	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0

CLK	Q_D	Q_C	Q_B	Q_A
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	0	0	0	1
12	0	0	1	0
13	0	0	1	1
14	0	1	0	0

4. Asynchronous Mod-12 counter :-

State diagram:-



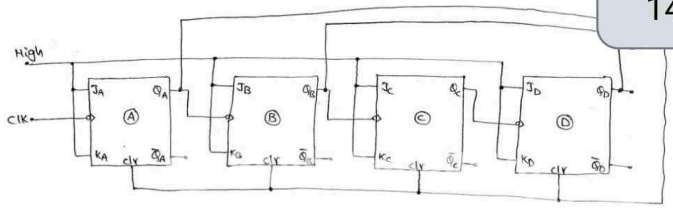


Fig:- MOD-12 ripple counter

counter o/p → $Q_D\ Q_C\ Q_B\ Q_A$ (MSB … LSB)

* When counter o/p is $Q_D Q_C Q_B Q_A$ 1011, o/p of And gate becomes 1. Since o/p of And gate is connected to reset (or) clear i/p's of FF's, counter enters into 0000 state after 1011 state.

Truth table:-

CLK	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

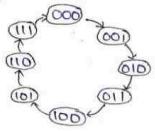
CLK	Q_D	Q_C	Q_B	Q_A
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	0	0	0	0
13	0	0	0	1
14	0	0	1	0
15	0	0	1	1
16	0	1	0	0
17	0	1	0	1

14/10/2014

Synchronous (or) Parallel counters :-

1. 3-bit synchronous Binary up counter (or) Mod-8 Sync. counter :-

State diagram :-



Logic diagram :-

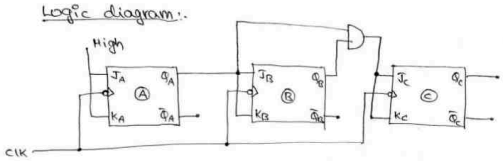


Fig :- MOD-8 Sync. counter

FF i/p's

$J_A = K_A = 1$

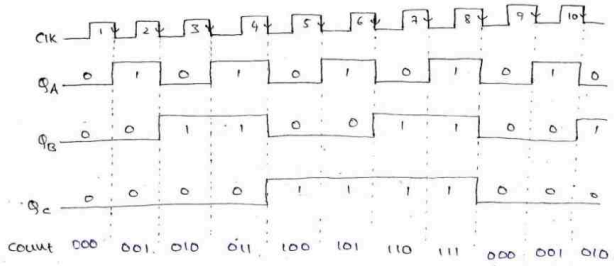
$J_B = K_B = Q_A$

$J_C = K_C = Q_A \cdot Q_B$

counter o/p $\longrightarrow$ Q_C Q_B Q_A

MSB LSB

output waveforms :-



2. 4-bit Synchronous Binary up counter (or) MOD-16 counter :-

Block diagram :-

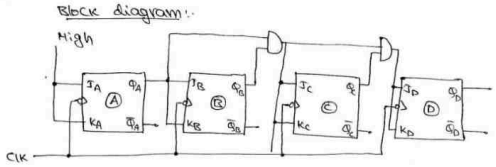


Fig :- MOD-16 Binary up counter

FF i/p's

$J_A = K_A = 1$

$J_B = K_B = Q_A$

$J_C = K_C = Q_A \cdot Q_B$

$J_D = K_D = Q_A \cdot Q_B \cdot Q_C$

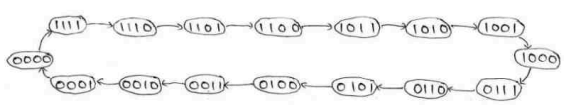
counter o/p $\longrightarrow$ Q_D Q_C Q_B Q_A

MSB LSB

16

3. 4-bit Synchronous Down counter (or) MOD-16 Down counter :-

State diagram :-



Logic diagram :-

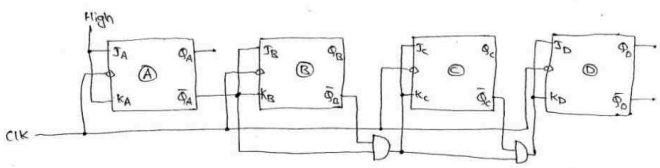


Fig : MOD-16 Down counter

FF i/p's

$J_A = K_A = 1$

$J_B = K_B = \bar{Q}_A$

$J_C = K_C = \bar{Q}_A \cdot \bar{Q}_B$

$J_D = K_D = \bar{Q}_A \cdot \bar{Q}_B \cdot \bar{Q}_C$

Counter o/p → $Q_D\ Q_C\ Q_B\ Q_A$ (MSB … LSB)

Truth table (or) Function table :-

CLK	counter o/p Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	1	1	1	1
2	1	1	1	0
3	1	1	0	1
4	1	1	0	0
5	1	0	1	1
6	1	0	1	0
7	1	0	0	1
8	1	0	0	0
9	0	1	1	1

CLK	counter o/p Q_D	Q_C	Q_B	Q_A
10	0	1	1	0
11	0	1	0	1
12	0	1	0	0
13	0	0	1	1
14	0	0	1	0
15	0	0	0	1
16	0	0	0	0
17	1	1	1	1
18	1	1	1	0
19	1	1	0	1

* Design of Synchrous counters :-

Q. Design synchrous MOD-5 counter using JK flip-flops.

Sol:- Note:- If counter type is not mentioned then design up counter.

Step 1:- Determine number of flip-flops.

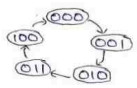
The no. of FF's required to design MOD-5 Synchronous up counter can be determine by the equation, $2^n \geq N$ where $n \rightarrow$ no. of FF's, $N \rightarrow$ MOD no.

The possible value of 'n' which satisifies the above equation is 3.

Thus MOD-5 counter uses 3 FF's.

Step 2:- Flip-flop type - JK FF.

Step 3:- State diagram



Step 4:- Excitation table

Present stage Q_C	Q_B	Q_A	Next stage Q_{C+1}	Q_{B+1}	Q_{A+1}	FF i/p's J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

∵ JK FF Excitation Rtable

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

18

Step 5:- k-Map simplification

for J_C

Q_C \ Q_BQ_A	00	01	11	10
0	X	X	1	X
1	X	X	X	X

∴ $J_C = Q_BQ_A$

for J_B

Q_C \ Q_BQ_A	00	01	11	10
0	1	X	X	X
1	X	X	X	X

∴ $J_B = Q_A$

for J_A

Q_C \ Q_BQ_A	00	01	11	10
0	1	X	X	1
1	X	X	X	X

∴ $J_A = \bar{Q}_C$

for K_C

Q_C \ Q_BQ_A	00	01	11	10
0	X	X	X	X
1	1	X	X	X

∴ $K_C = 1$

for K_B

Q_C \ Q_BQ_A	00	01	11	10
0	X	X	1	X
1	X	X	X	X

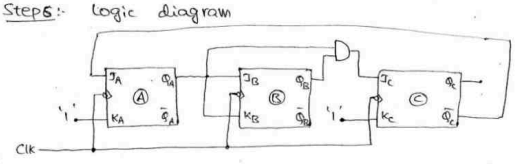
∴ $K_B = Q_A$

for K_A

Q_C \ Q_BQ_A	00	01	11	10
0	X	1	1	X
1	X	X	X	X

∴ $K_A = 1$

Step 6:- Logic diagram



15/10/2024 (10)

Q. Design MOD-10 synchronous up counter using T-Flip-flops.
(Decade counter / BCD counter / MOD-10 counter).

Sol: Step 1: Determine no. of FF's required.

$2^n \geq N$ where $N \rightarrow$ Mod No.
$n \rightarrow$ no. of FF's.

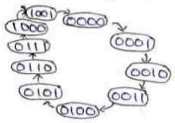
$2^n \geq 10$

$n = 4$

∴ No. of FF's required = 4

FF Type → T-FF

Step 2: State diagram



Step 3: Excitation table

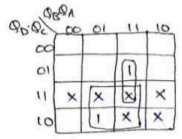
Present stage				Next stage				FF i/p's			
Q_D	Q_C	Q_B	Q_A	Q_{D+1}	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	x	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x

∵ T-FF excitation table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

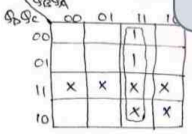
Step 4: K-map simplification

for T_D



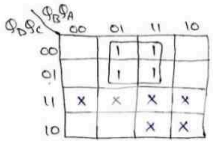
∴ $T_D = Q_D Q_A + Q_C Q_B Q_A$

for T_C



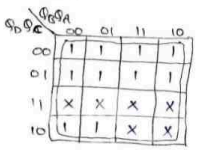
∴ $T_C = Q_B Q_A$

for T_B



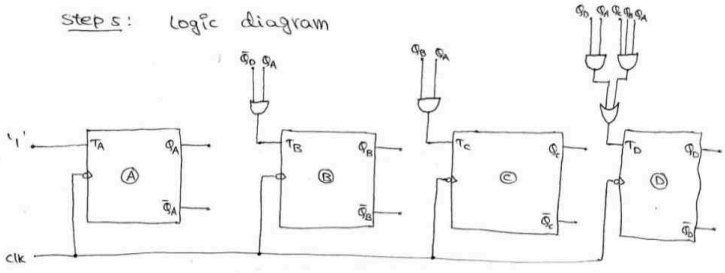
∴ $T_B = \bar{Q}_D Q_A$

for T_A



∴ $T_A = 1$

Step 5: Logic diagram



Counter o/p → Q_D Q_C Q_B Q_A
MSB LSB

20

20