Sequential Circuits

If we add a memory unit with combinational circuit, through which output can be used as input, then it acts as sequential circuit.

Types:

(i) Synchronus (Adding Clock pulse (CP))

(ii) Asynchronus (without - clock pulse)

Flip-Flop: circuit used to store a single bit

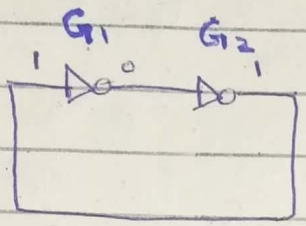
2 States $\begin{cases} 1 \\ 0 \end{cases}$ (either set, or reset)

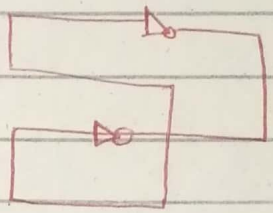
Types:

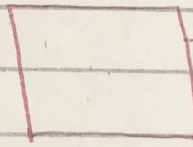
SR-flipflop, JK-flip flop, D-flip-flop, T-flipflop

Latch/Flip-Flop:

To store more bits, use more numbers of flip flops.

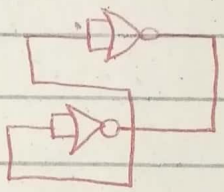


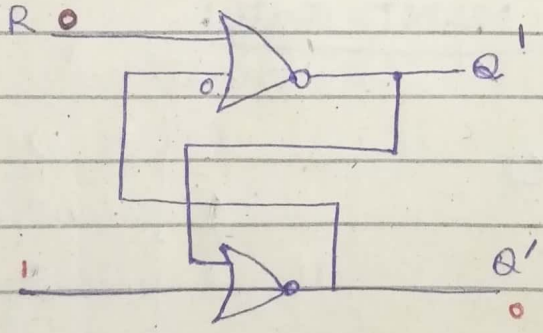




tells about the state of input

↳ Q, Q' always different





S	R	Q	Q'
→ 1	0	1	0
⊕ 0	0	1	0
→ 0	1	0	1
⊕ 0	0	0	1
Forbidden State → 1	1	0	0

↳ Active high for 1 means it propagates first

Undetermined state

⊗ un-desirable state where Q and Q' have same value

Propagation Delay:

To have already stored value of Q in flip-flop, send zero as input for both S, R.

S is set bit and R is reset bit.

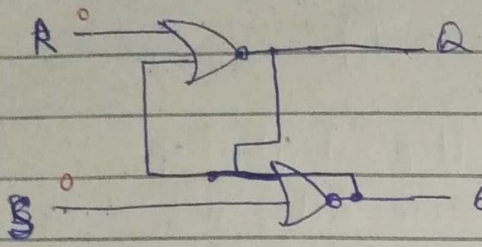
or set Q bit

* For S=1, R=0, flip-flop makes Q=1. For this S=0, R=0, flip-flop appears value of Q that is already stored.

* For S=0, R=1, flip-flop resets Q bit = 0.

Inially for S=0, R=0;

← Using NOR gate:



Depends on propagation time, if R works first then:

Q=1, S=0

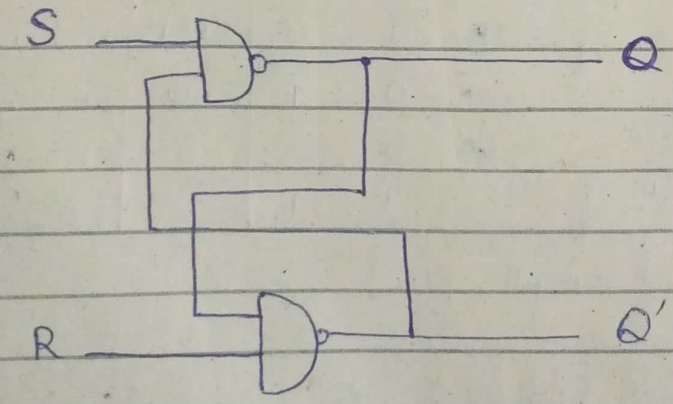
← (High Active state for NOR Gate)

works according to set-bit

S	R	Q_{t+1} ← new Q (new state)
0	0	Q_t ← previous Q (previous state)
0	1	0
1	0	1
1	1	Forbidden State

Implementation with NAND Gate:

↳ with NAND, S is reset, R is set bit



S	R	Q	Q'
0	1	1	0
1	1	1	0
1	0	0	1
1	1	0	1 ← unchanged
0	0	1	1

Forbidden State

Using NAND Gate: ← low active state

S	R	Q_{t+1}
0	0	Undesired
0	1	1
1	0	0
1	1	Q_t

Using NOR Gate:

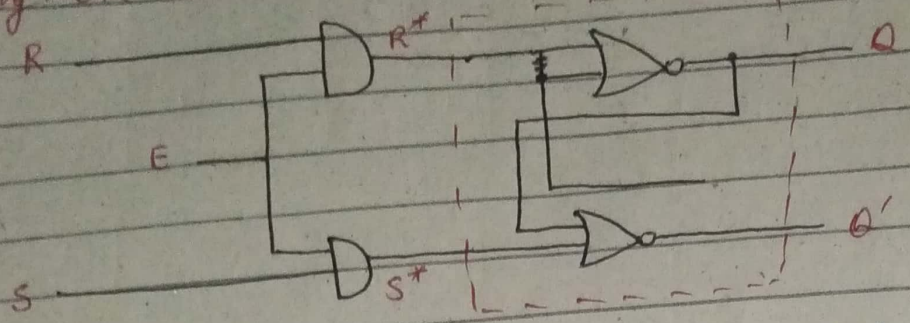
Q_t	S	R	Q_{t+1}
0	0	0	0 (Q_t)
0	0	1	0
0	1	0	1
0	1	1	X undesirable
1	0	0	1 (Q_t)
1	0	1	0
1	1	0	1
1	1	1	X

Using NAND Gate:

Q_t	S	R	Q_{t+1}
0	0	0	X
0	0	1	1
0	1	0	0
0	1	1	0 (Q_t)
1	0	0	X
1	0	1	1
1	1	0	0
1	1	1	1 (Q_t)

Gated SR-Latch: Latch with an Enable input / gated input

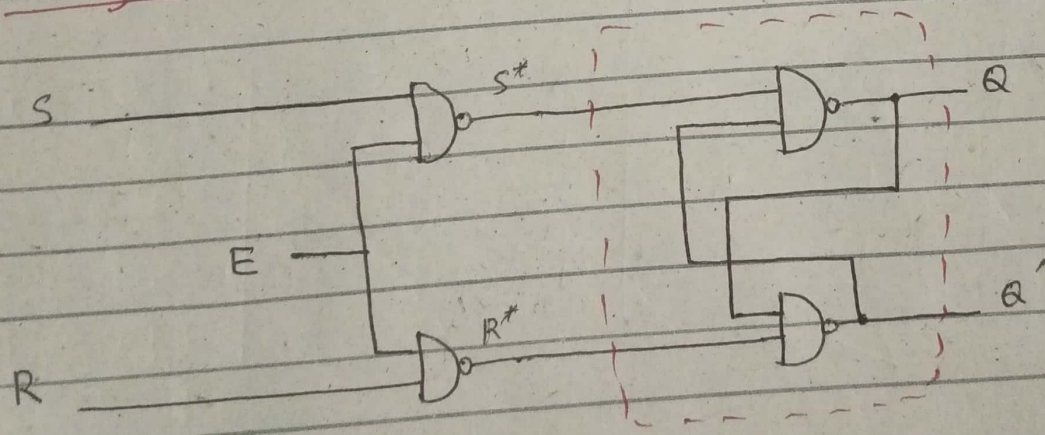
Using NOR Gate:



E	S	R	Q_{t+1}
→ 0	0	0	Q_t
→ 0	0	1	Q_t
→ 0	1	0	Q_t
→ 0	1	1	Q_t
1	0	0	Q_t
1	0	1	0
1	1	0	1
1	1	1	undesirable state

* If E is zero, no matter what input is in S, R → it retains Q_t in its new state (Q_{t+1})

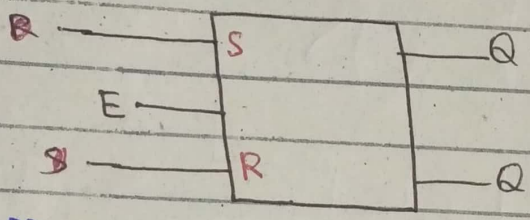
Using NAND Gate:-



E	S	R	Q_{t+1}	
0	0	0	Q_t	
0	0	1	Q_t	
0	1	0	Q_t	
0	1	1	Q_t	
1	0	0	Q_t	
1	0	1	0	
1	1	0	1	
1	1	1	X	undesirable

Gated SR-Latch works the same on both active high (NOR gate) and active low (NAND gate) state.

Block Diagram:



Truth Table assuming E=1:

	Q_t	S	R	Q_{t+1}
0	0	0	0	Q_t (0)
1	0	0	1	0
2	0	1	0	1
3	0	1	1	X
4	1	0	0	Q_t (1)
5	1	0	1	0
6	1	1	0	1
7	1	1	1	X

Characteristic Table:

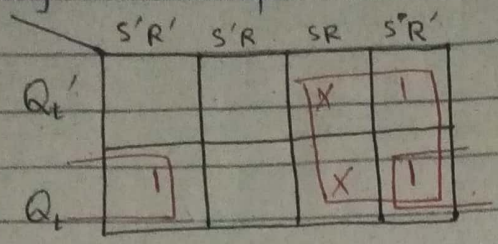
S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	X

Boolean Expression for Q_{t+1}:

$Q_{t+1} = Q_t'SR' + Q_tS'R' + Q_tS^{\bullet}R'$, $d = Q_t'SR + Q_tSR$

$Q_{t+1} = \Sigma(2,4,6)$ $d = \Sigma(3,7)$

By K-Map:

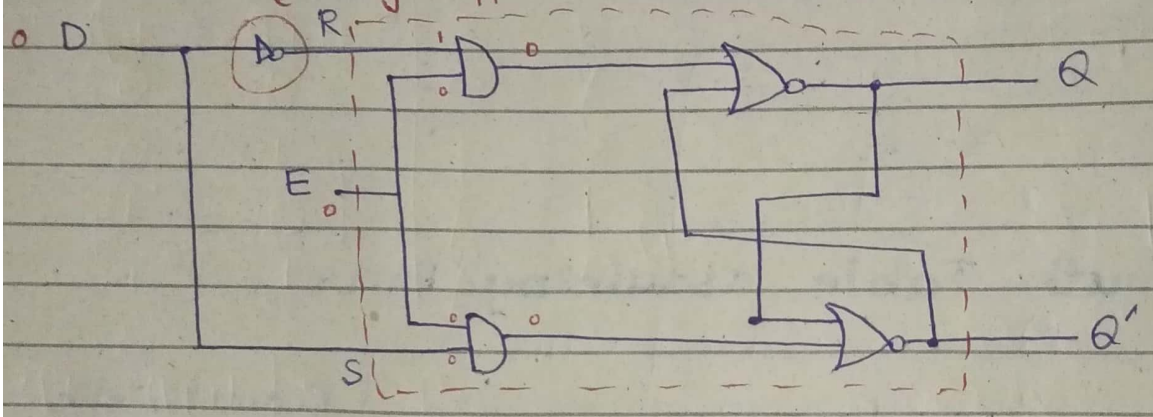


$$Q_{t+1} = S + R'Q_t$$

This is called characteristic equation ~~which~~ with condition: $S.R = 0$, so that S and R don't have 1 at the same which will lead towards undesirable state.

Gated D-Latch:

Only difference from Gated SR-Latch



E	D	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

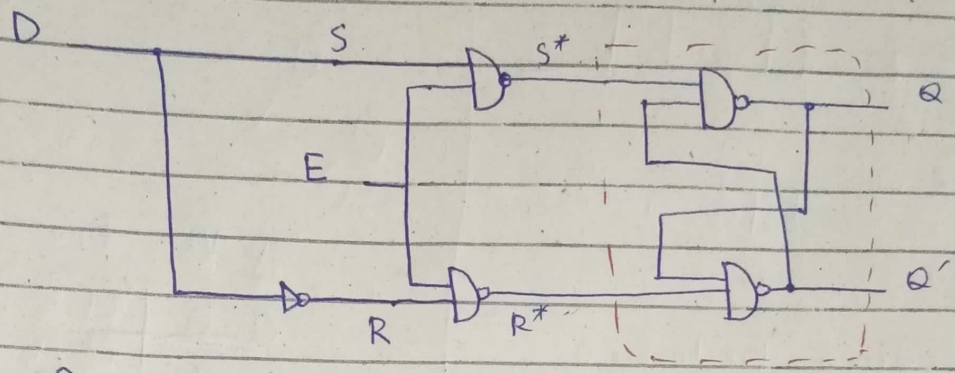
* For $E = 0$, $Q_{t+1} = Q_t$ dependent on Q_t for $E = 0$

* For $E = 1$, $Q_{t+1} = D$ independent of Q_t for $E = 1$

⇒ E = 1 :

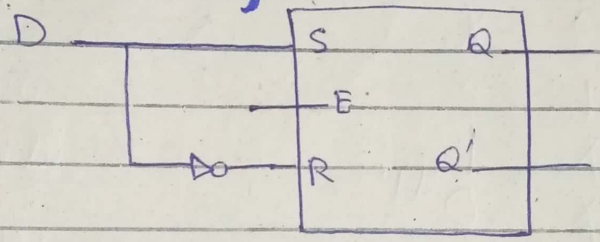
D	Q_{t+1}
0	0
1	1

D-Latch by using NAND Gate:



Same Truth Table as for D-Latch by NOR

Block Diagram:



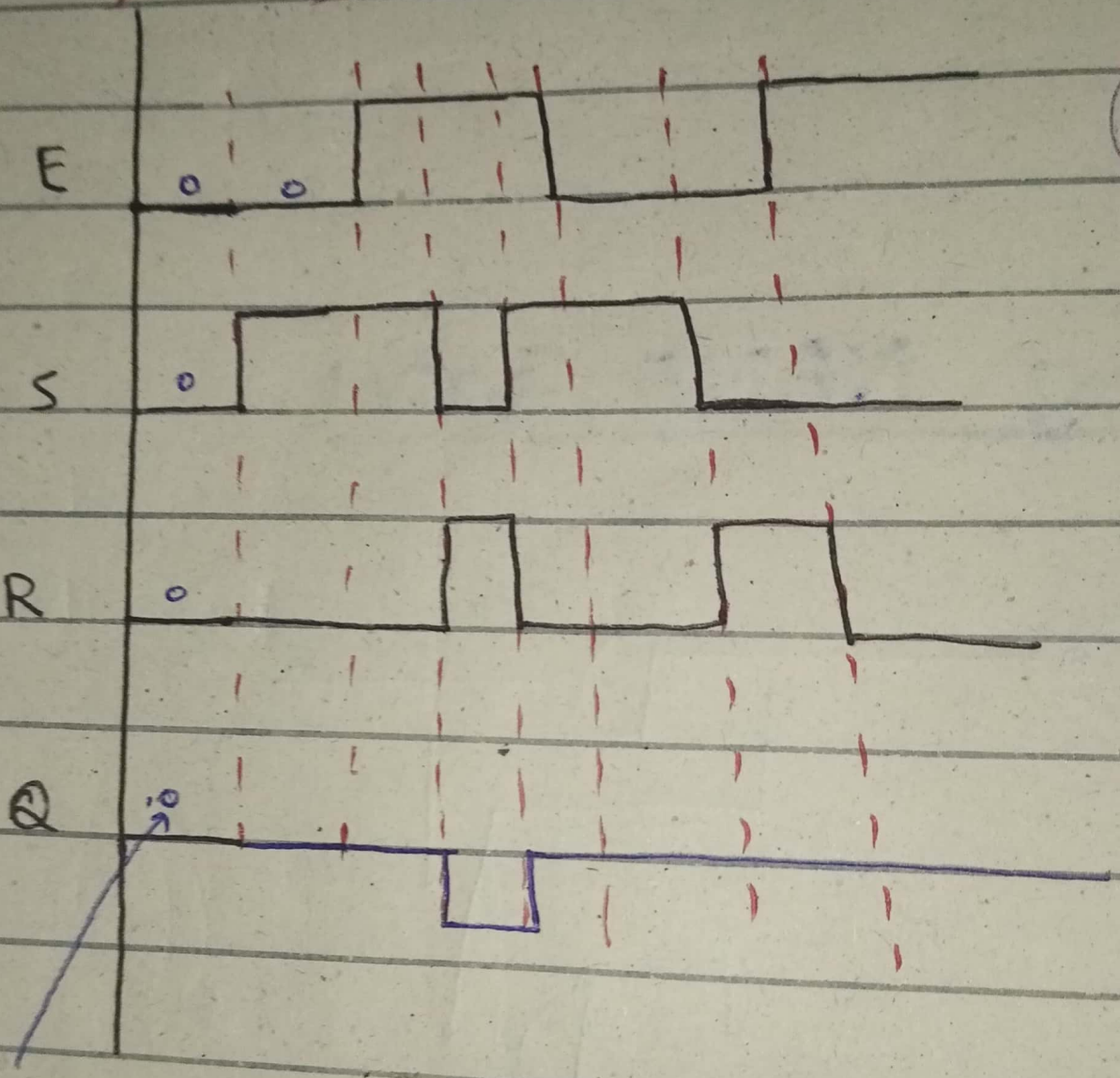
E	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1

$Q_{t+1} = D$

When Enable is zero, any state of D don't matter in Q_{t+1}

0	1	1
1	0	1
1	1	0

Timing Diagram of Gated SR-Lat



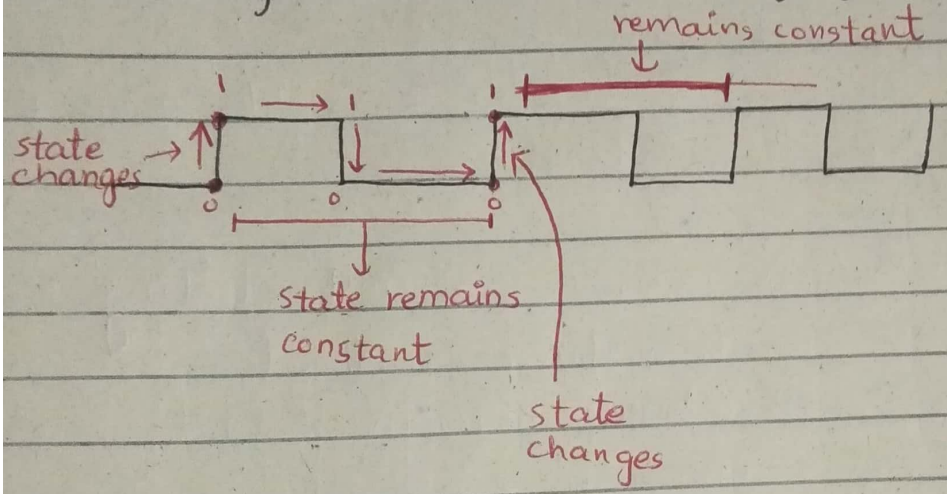
Initially, $Q_t = 0$ due to $E = 0$.

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X

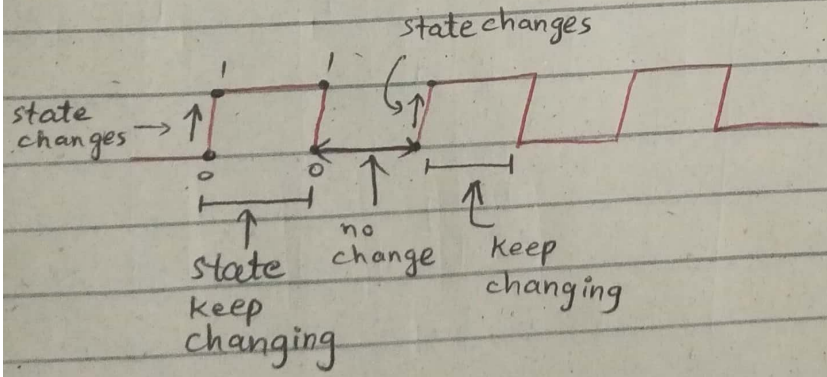
Level-triggered Flip Flops → Latch:
State is once changed when enable becomes high and then keep on changing until E becomes low.

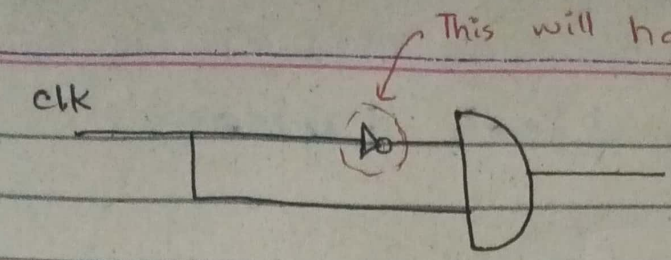
Edge-triggered Flip Flops → Clock Pulse:
↳ state changes once E is high, then no change in state, again state changes when enable E is high.

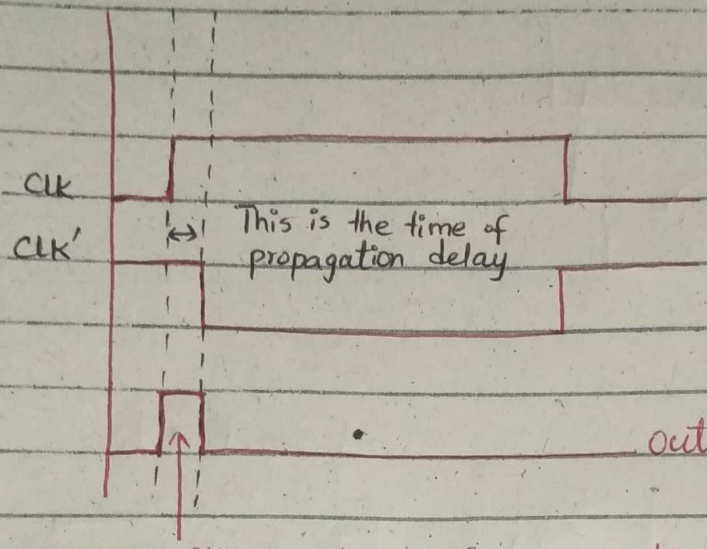
+ve Edge triggered (Clock Pulse):
Changes state on changing of E from 0 to 1.



-ve Edge triggered (Latch):

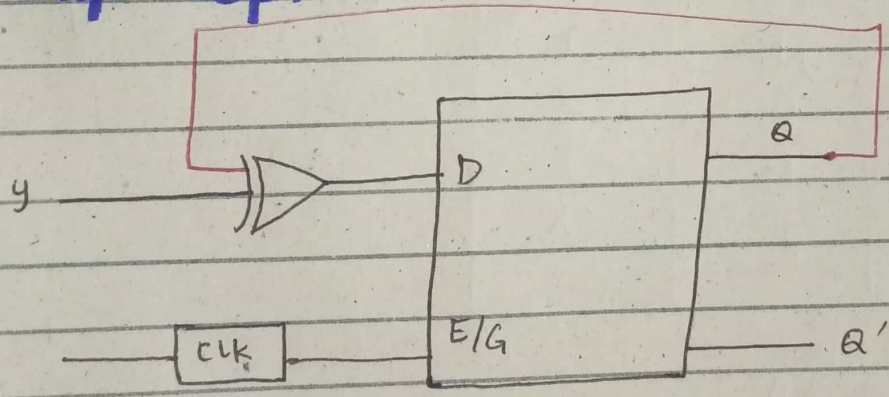






will be high for some time (when there is propagation delay due to NOT Gate)

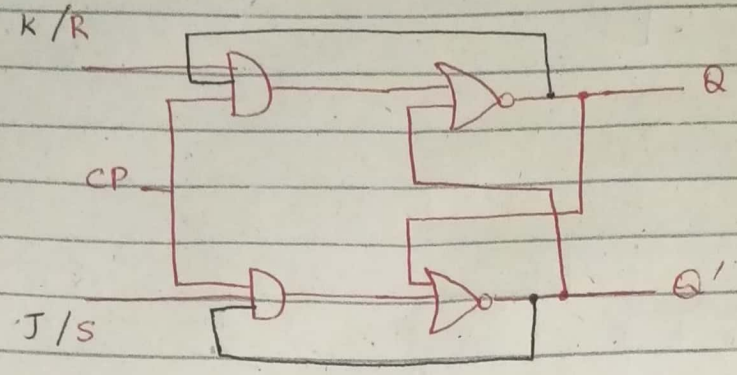
Constructing a +ve Edged triggered Flip Flop:



D will always change state only when E is 1 and will be constant aftewards untill E again becomes zero.

JK → Q' always with J
Q always with K
(whether NOR based or NAND based)

Gated SR using Clock Pulse:



JK - Flip Flop:

Q_t	J	K	Q_{t+1} (JK)	Q_{t+1} (SR)
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	X
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	0	X

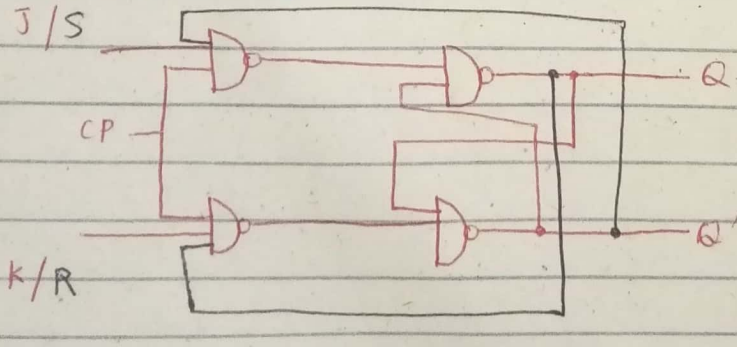
characteristic Table:

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q_t'

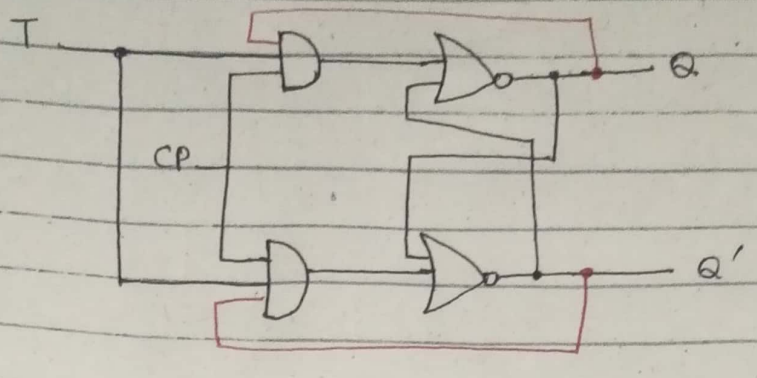
	$J'K'$	$J'K$	JK	JK'
Q_t'			1	1
Q_t	1			1

$$Q_{t+1} = Q_t'J + Q_tK'$$

NAND Based:



T-Flip Flop:



Q_t	T	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

← Characteristic Table of T:

$$Q_{t+1} = Q_t' T + Q_t T' = Q_t \oplus T$$

Characteristic Tables:

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	X

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q_t'

D	Q_{t+1}
0	0
1	1

T	Q_{t+1}
0	Q_t
1	Q_t'

Excitation Tables:

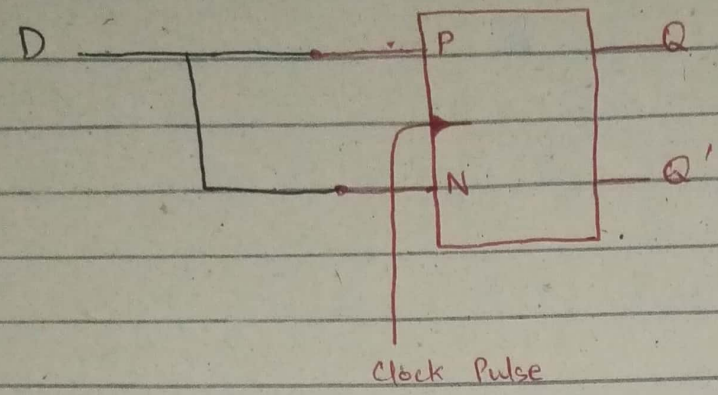
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q_t	Q_{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

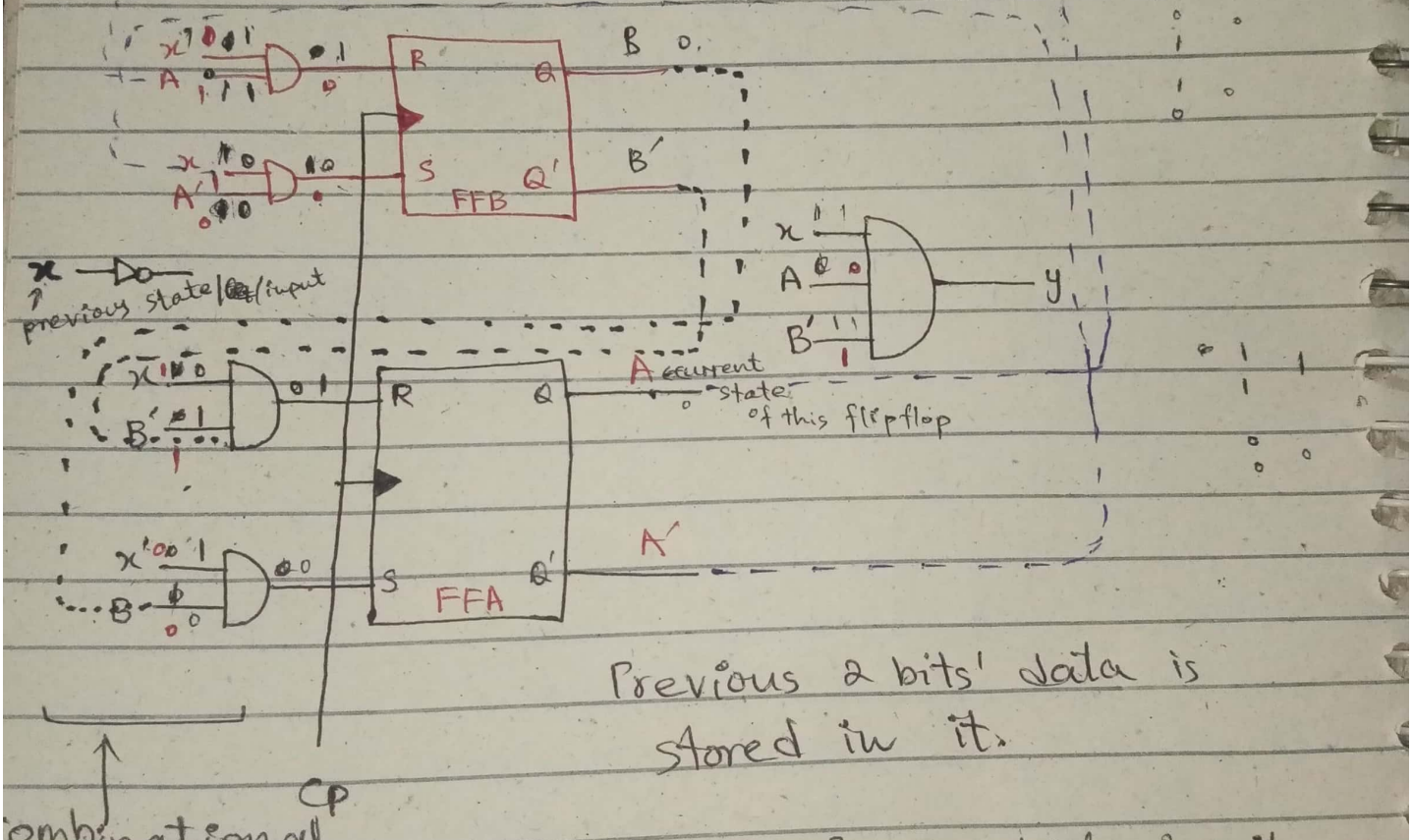
Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

D Latch using PN Flip Flop:



Analysis of Sequential Circuit:

R → K
S → J
→ JK



Previous 2 bits' data is stored in it.

Combinational circuits before memory units of sequential circuits

2 Flip Flops in memory units

A, B → 2 previous states

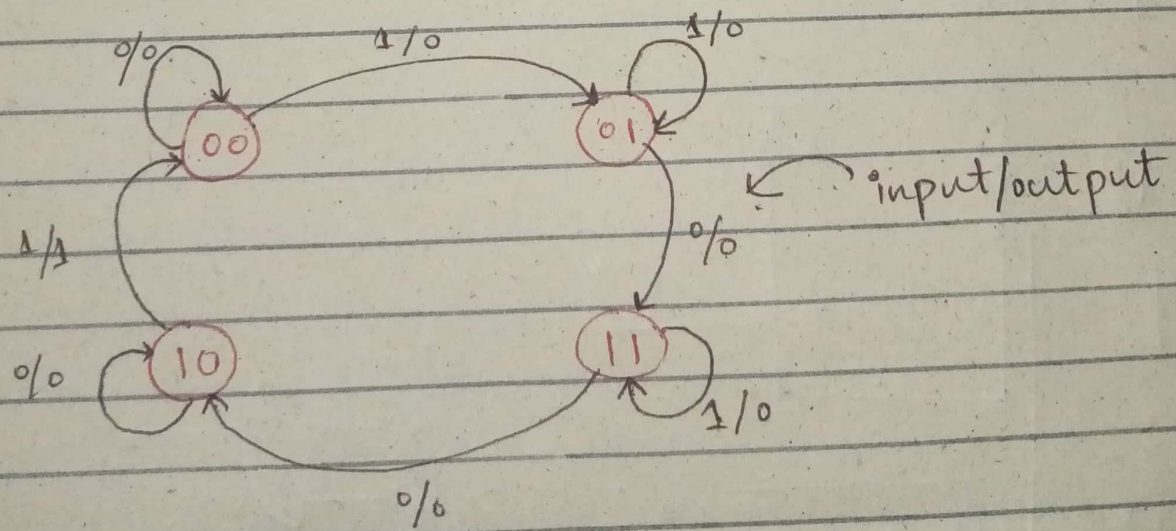
x → input

y → output

Previous State			Next State		
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	0	0
1	1	1	1	1	0

Previous State		Next State				Output	
		x=0		x=1		x=0	x=1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	1	0	0	0	0	1
1	1	1	0	1	1	0	0

State Diagram of Sequential Circuits:

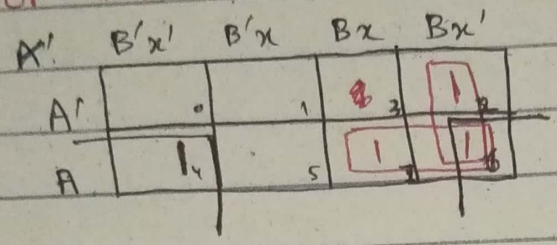


Analysis:

(i) Transition Table
(ii) Transition diagram
(iii) State Diagram
(iv) Equations / Boolean expression

Boolean Expression:

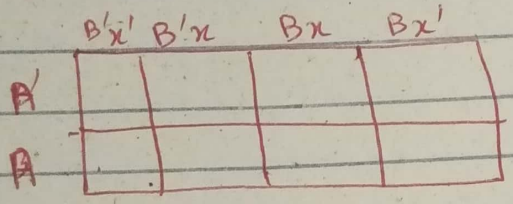
For A:



$$A_{(t+1)} = Ax' + A'B + Bx'$$

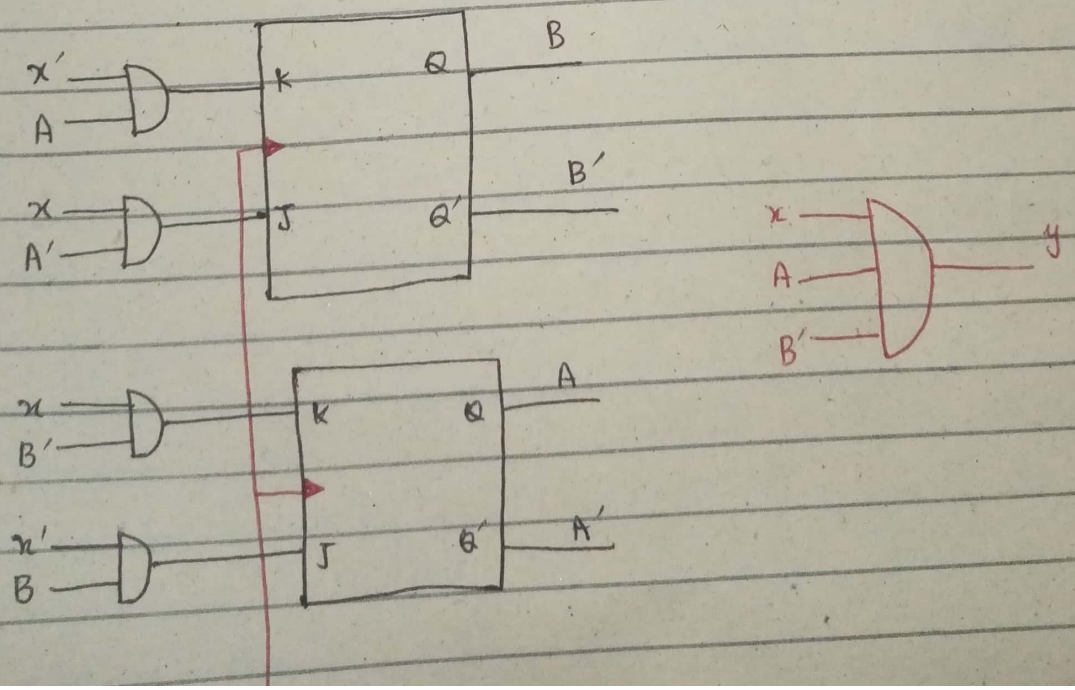
↑ next state

For B:



$B_{t+1} =$ ——

$y = xAB'$

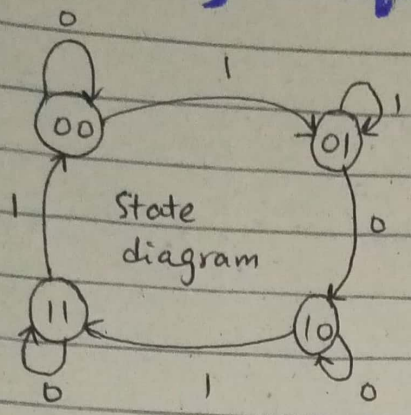


A	B	x
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

A	B	y
0	0	0
0	1	0
1	1	0
0	1	0
1	0	0
0	0	1
1	0	0
1	1	0

Designing Sequential Circuit:

Using D-flip flop:



No separate output

Steps in Designing Procedures

(i) Draw state diagram according to description.

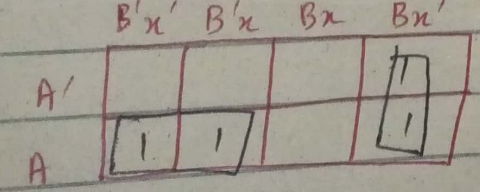
(ii) Draw transition table acc. to state diagram. also count no. of flip flops to be used (state diagram tells us about how many flip-flops to be used → no. of states = no. of flip-flops)

(iii) then, identify inputs of flip-flops using excitation table

Present State			Next State			
A	B	x	A	B	D_A	D_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	0	0	0	0

$D_A = \Sigma(2,4,5,6)$

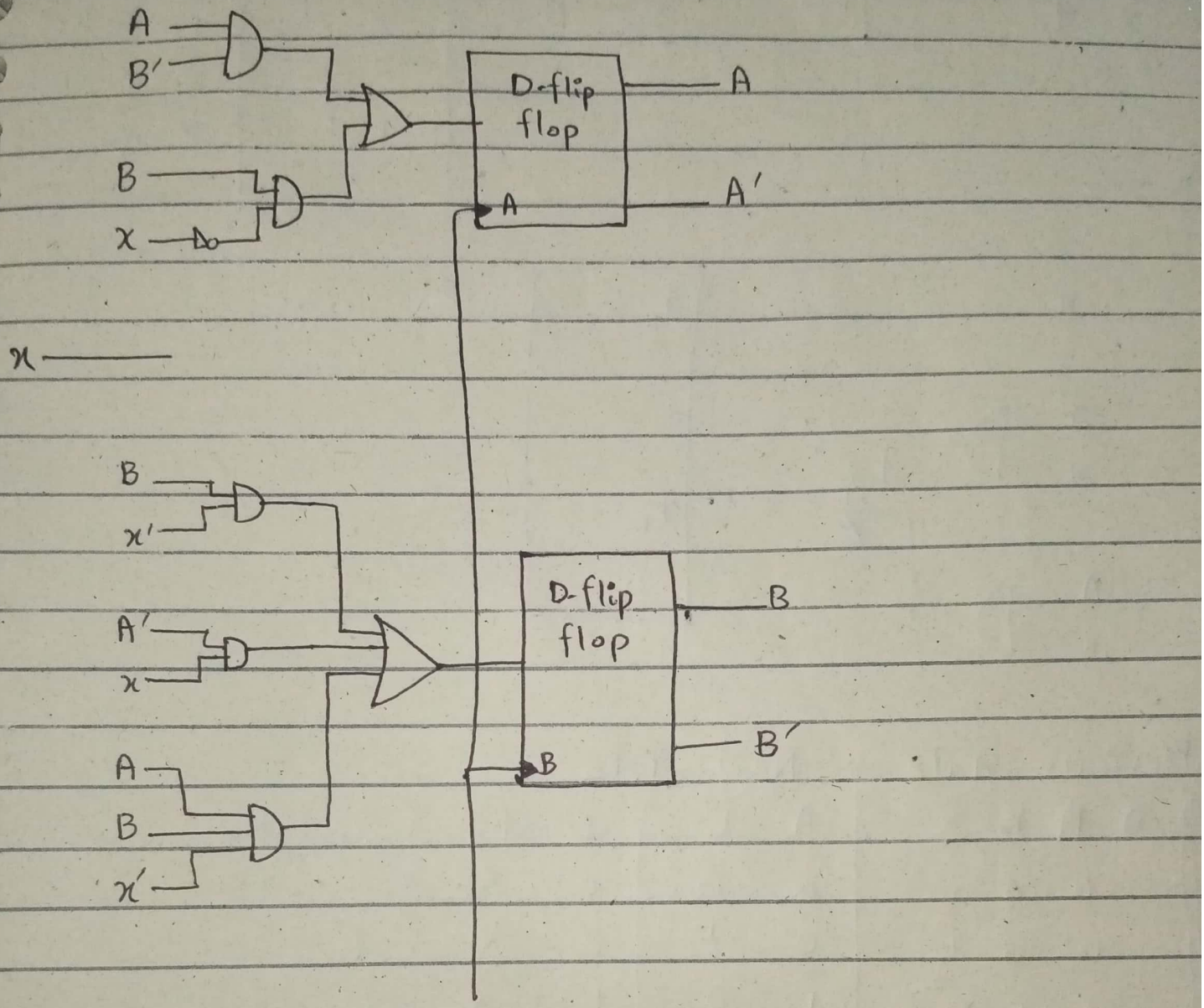
$D_B = \Sigma(1,3,5,6)$



$D_A = AB' + Bx'$

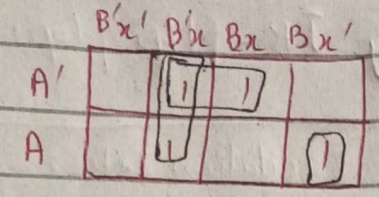
$D_B = B'x + A'x + ABx'$

Circuit Diagram:



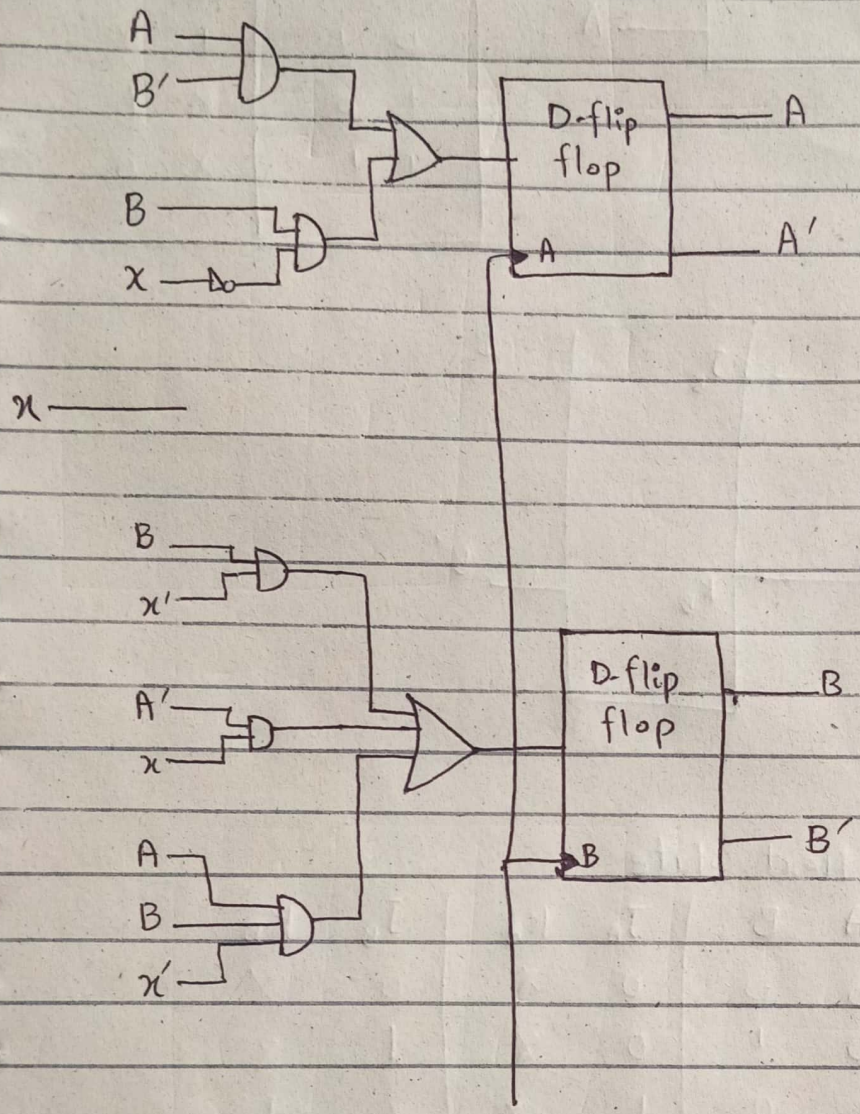
Summary of Circuit Designing:

1st Step: State Diagram
2nd Step: Transition Table / State Table
3rd Step: No. of flip flops
4th Step: Inputs of flip-flops using excitation tables
5th Step: Short form of inputs & outputs (if any) using K-map
6th Step: Draw circuit diagram



$D_B = B'x + A'x + ABx'$

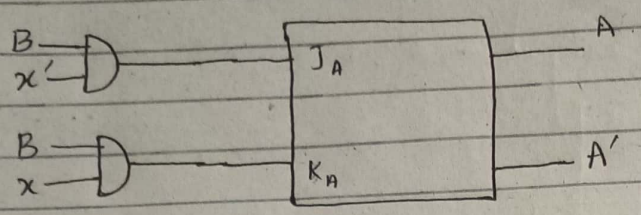
Circuit Diagram:



Summary of Circuit Designing:

1st Step: State Diagram
2nd Step: Transition Table / State Table
3rd Step: No. of flip flops
4th Step: Inputs of flip-flops using excitation tables
5th Step: Short form of inputs & outputs (if any) using K-map
6th Step: Draw circuit diagram

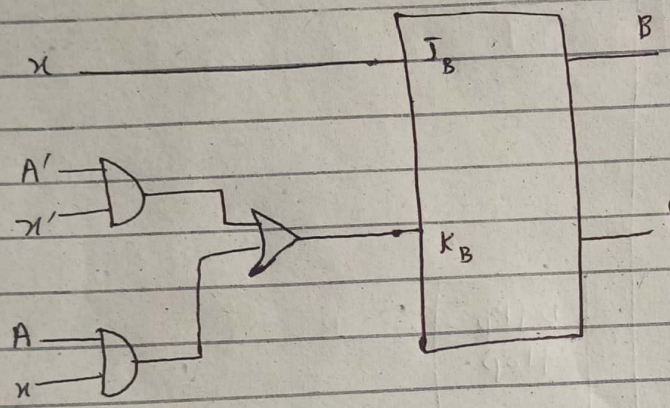
→ Design previous circuit using JK-flip flop:



Excitation Table:

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

x



Previous State			Next State					
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	1
1	1	1	0	0	X	1	X	0
0	0	0						
0	0	1						
0	1	0						
0	1	1						

$J_A = \Sigma(2)$ $\quad d = \Sigma(4,5,6,7)$

$K_A = \Sigma(7)$ $\quad d = \Sigma(0,1,2,3)$

$J_B = \Sigma(1,5)$ $\quad d = \Sigma(2,3,6,7)$

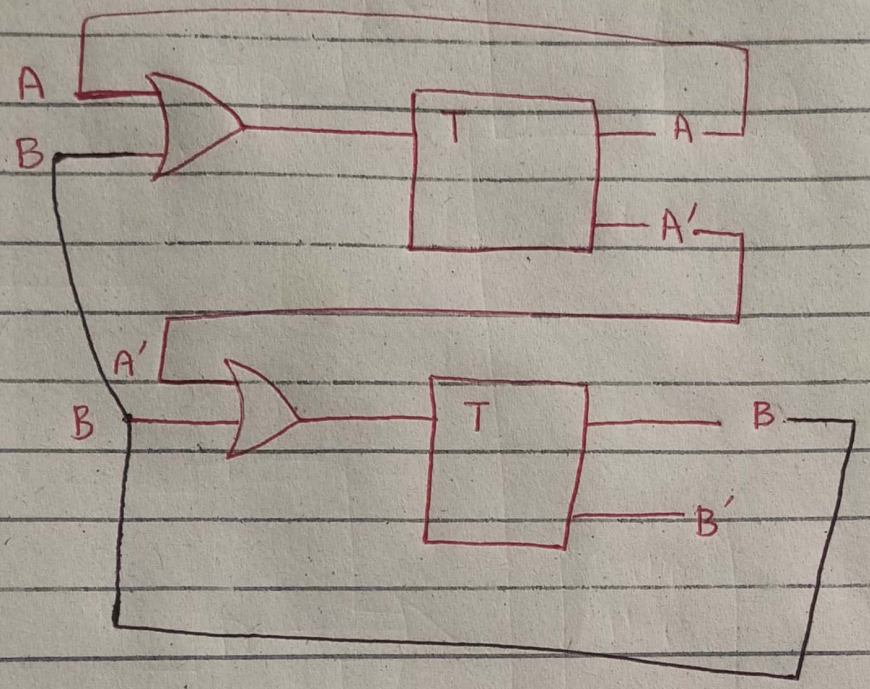
$K_B = \Sigma(2,7)$ $\quad d = \Sigma(0,1,4,5)$

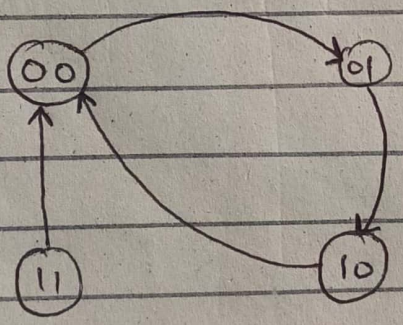
By K-Map, we get:

$J_A = Bx'$, $K_A = Bx$

$J_B = x$, $K_B = A'x' + Ax$

Q_t	Q_{t+1}
0	Q_t
1	Q_t'



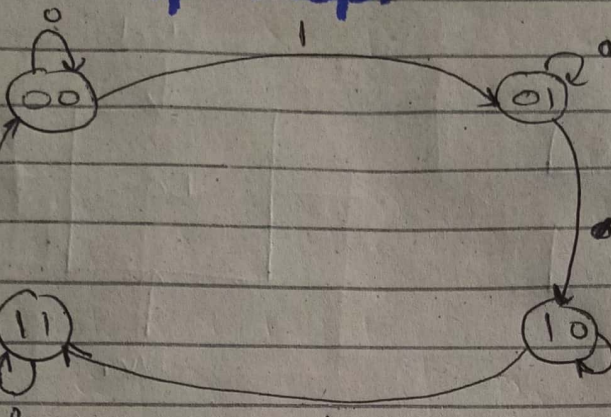


00	01
01	10
10	00
11	00

T-flip flop

14-12-23

SR - Flip Flop:



Q_t	Q_{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Previous State Next state → From excitation Table:

A	B	x	A	B	S_A	R_A	S_B	R_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	0
0	1	0	0	1	0	X	X	0
0	1	1	1	0	1	0	0	1
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	0
1	1	0	1	1	X	0	X	0
1	1	1	0	0	0	1	0	1

$S_A = A'Bx = \Sigma(3)$ $d = \Sigma(4,5,6)$

$R_A = \Sigma(7) = ABx$ $d = \Sigma(0,1,2)$

$S_B = \Sigma(1,5)$ $d = \Sigma(2,6)$

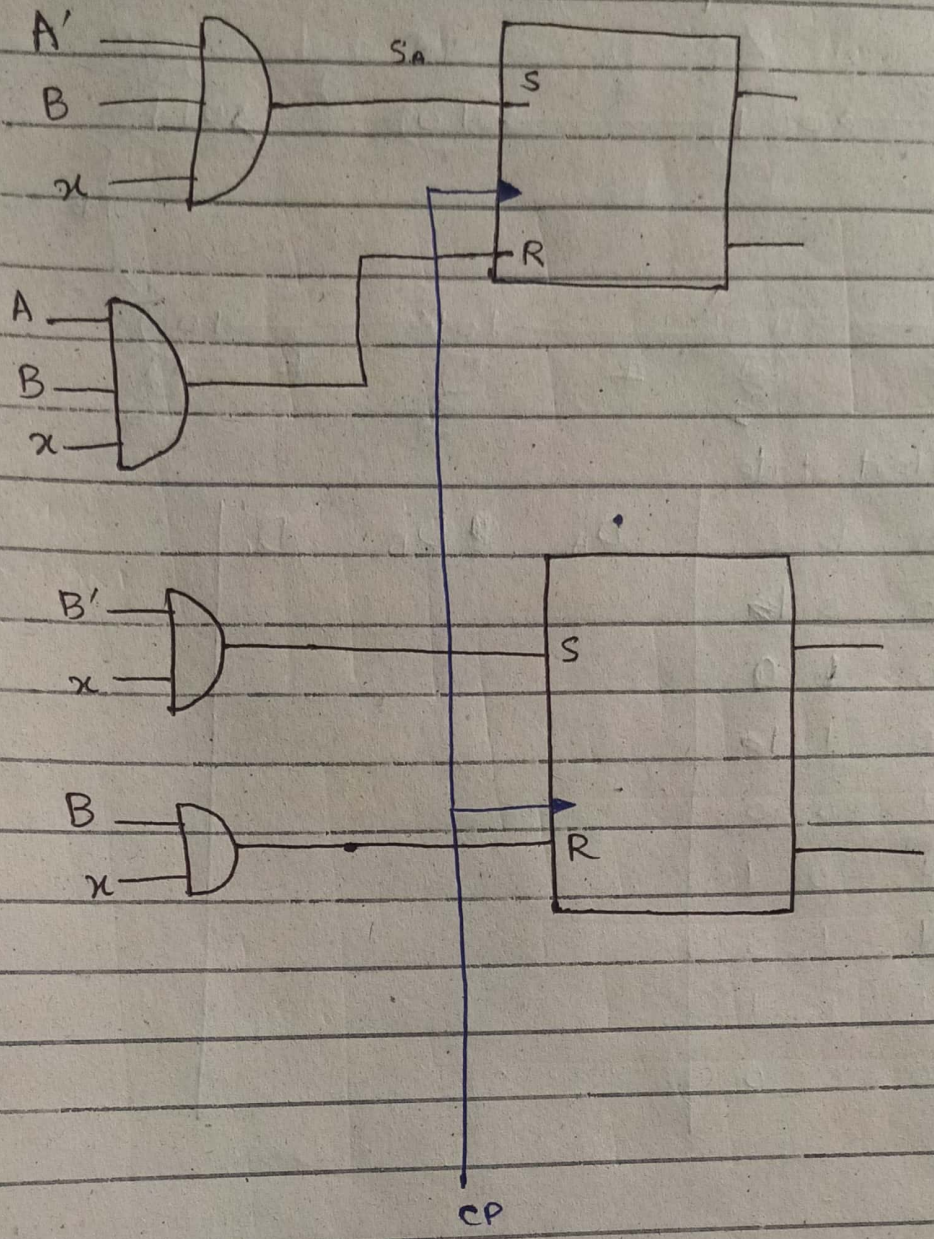
$R_B = \Sigma(3,7)$ $d = \Sigma(0,4)$

	$B'x'$	$B'x$	Bx	Bx'
A'			1 (3)	(2)
A	(4)	(5)	1 (7)	(6)

$R_B = Bx$

$S_B = B'x$

Circuit Diagram:

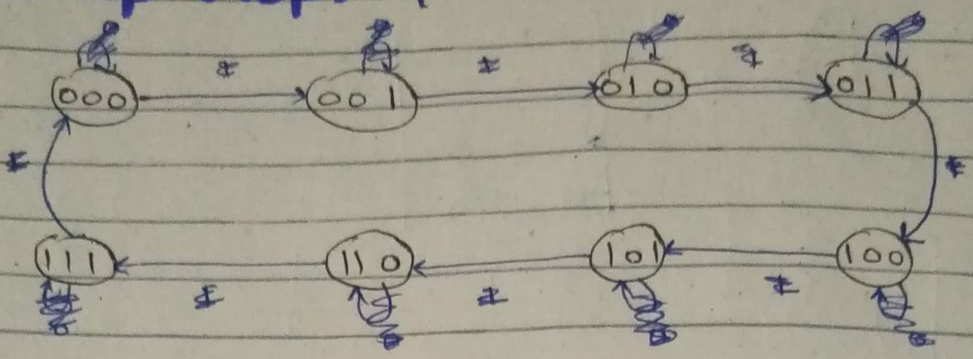


⇒ These circuits are called 3-bit counter circuits as they change their state in a (sequential) sequence (00 → 01 → 10 → 11) on x=1.

Design a 3-bit counter with SR-Flip Flop:

No input, transition on clock pulse

Design 3-Bit Counter by D-Flip Flop: (no input bit)



Previous State			Next State					
A	B	C	A	B	C	D_A	D_B	D_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

$D_A = \Sigma(3,4,5,6)$

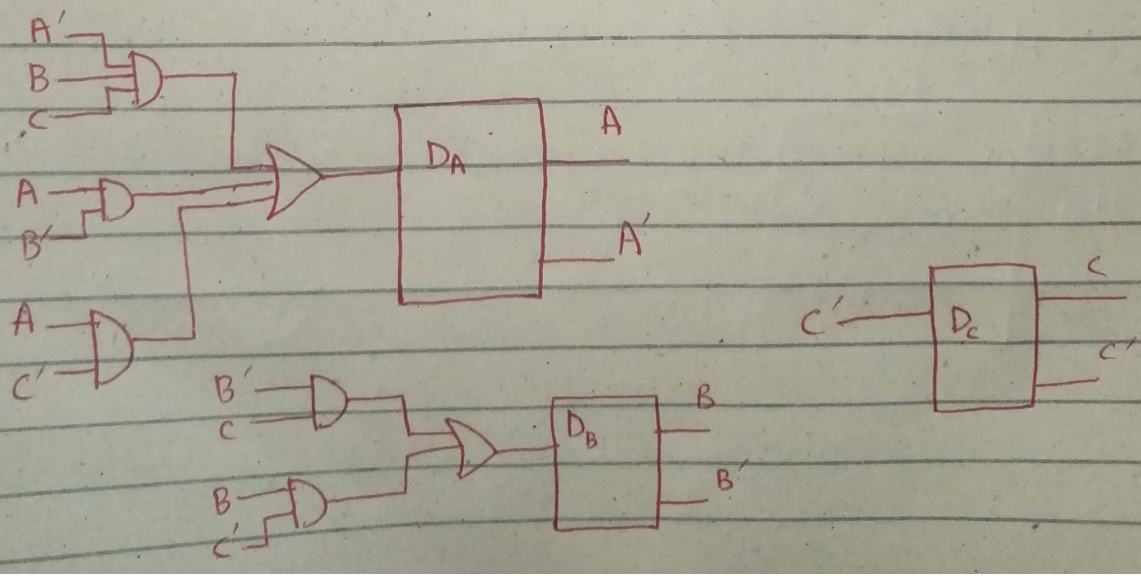
$D_B = \Sigma(1,2,5,6)$

$D_C = \Sigma(0,2,4,6)$

After K-Mapping:

$D_A = A'BC + AB' + AC'$, $D_B = B'C + BC'$

$D_C = C'$

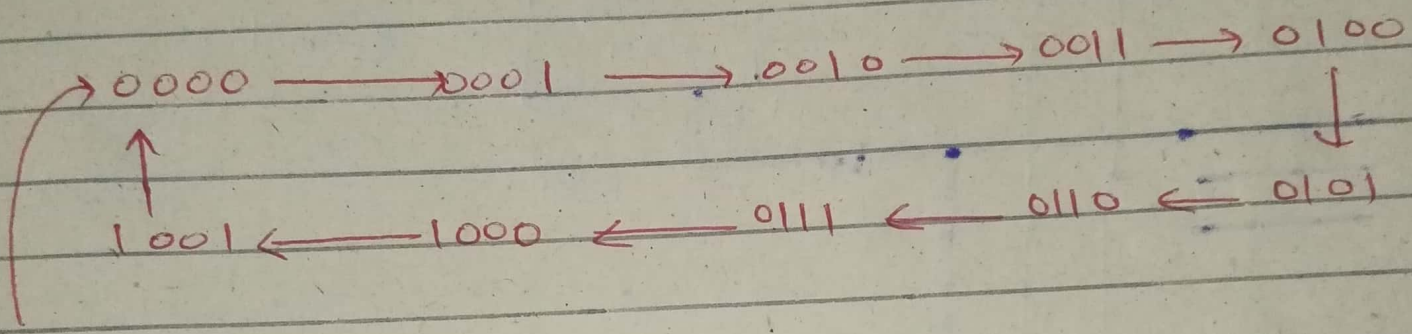


* Design a 4-bit counter

Design a BCD Counter:

0 → 9 ⇒ transits to next state

10 → 15 ⇒ resets to 0 (don't cares)

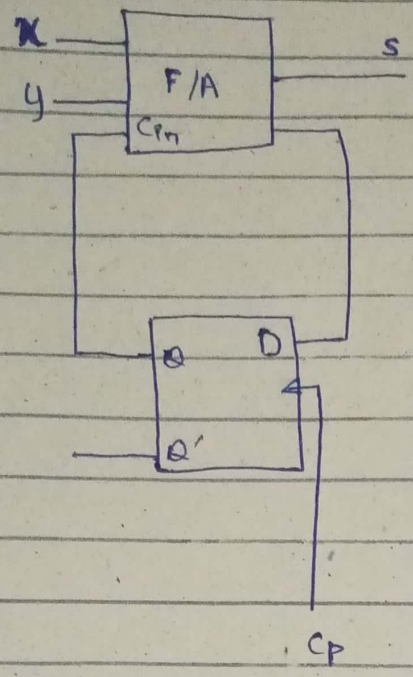


1010
1011
1100
1101
1110
1111

don't cares

Write all terms in transition table.

Full Adder Using D-Flip Flop:

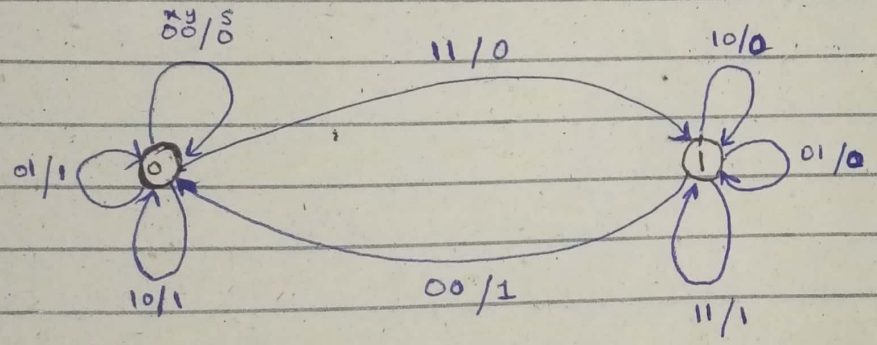


Cin from memory

Analysis:
(i) Circuit given
(ii) State table
(iii) State diagram

Design
(i) State diagram
(ii) State Table
(iii) Circuit Diagram

For one D-flip flop, there are 2 states: (i) 0 (ii) 1

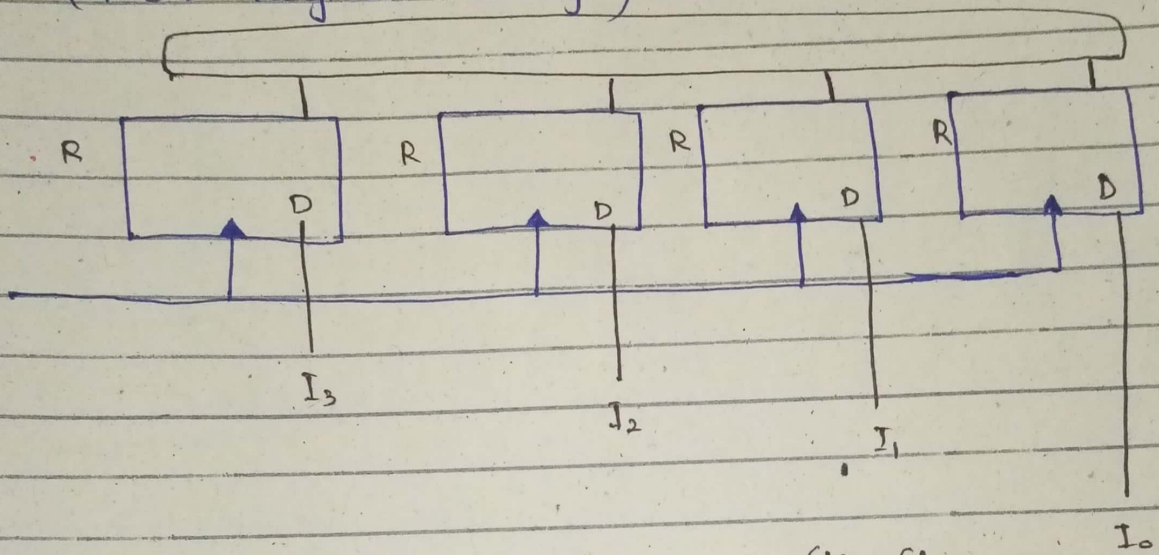


x	y	Cin/Q/D (D-flip flop)	S	Cout/D (Next State)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Memory:
(i) Primary ← RAM, Cache, Register (fastest memory)
↳ smallest unit of memory
(ii) Secondary
/
Store permanently

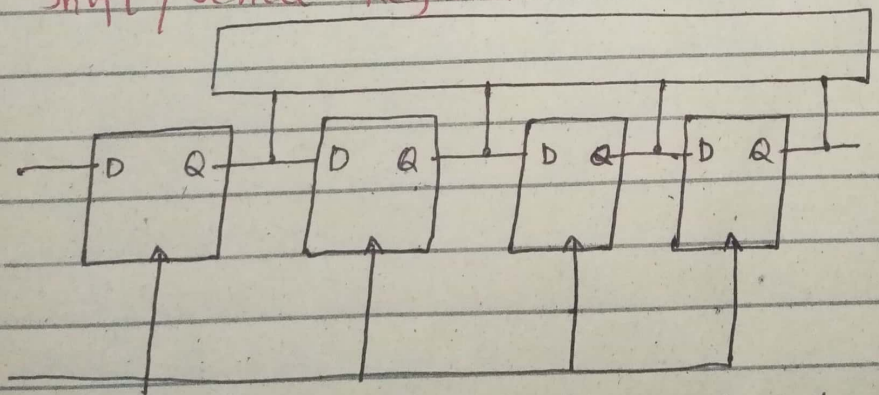
1 Bit Register is generally a Flip Flop

Design a D-Flip Flop: using register
(4-Bit Register Design) Parallel Registers

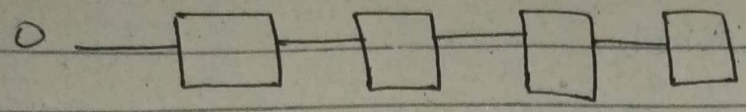


→ input is stored as it is in flip-flop.
→ We can also add one reset bit with every flip-flop. When this bit is 1, the information stored is reset.

Shift / Serial Register:

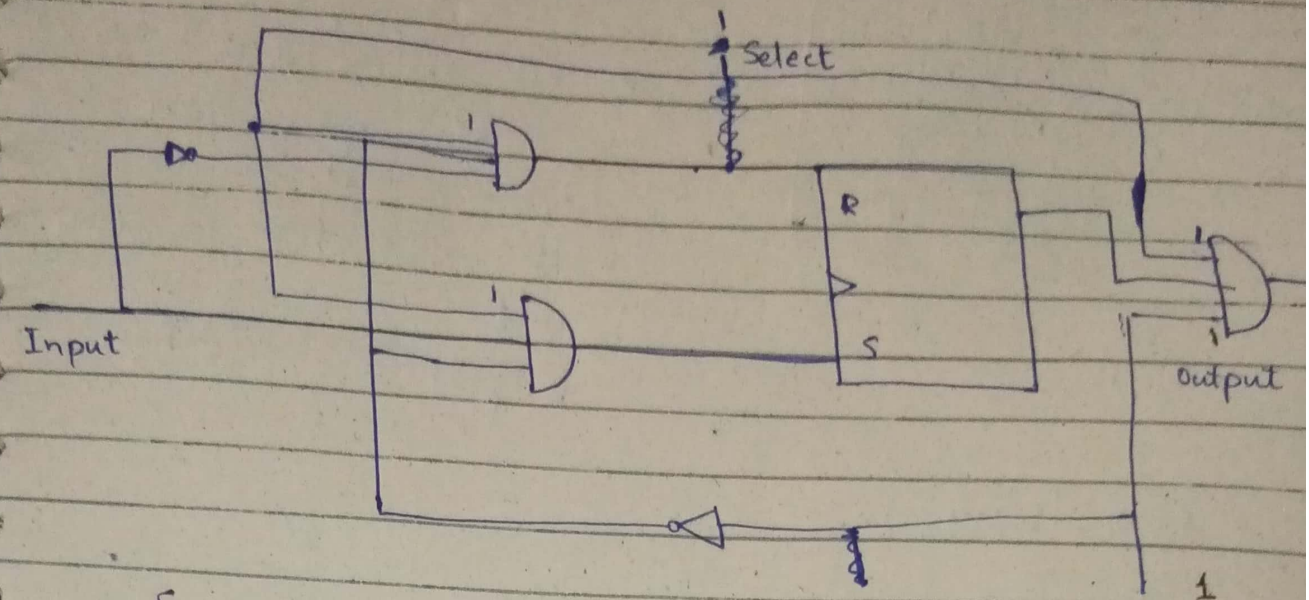


⇒ Such registers shift 1-bit data to left or right e.g. if register already had 1011, then on giving 0, the info stored is 0101.



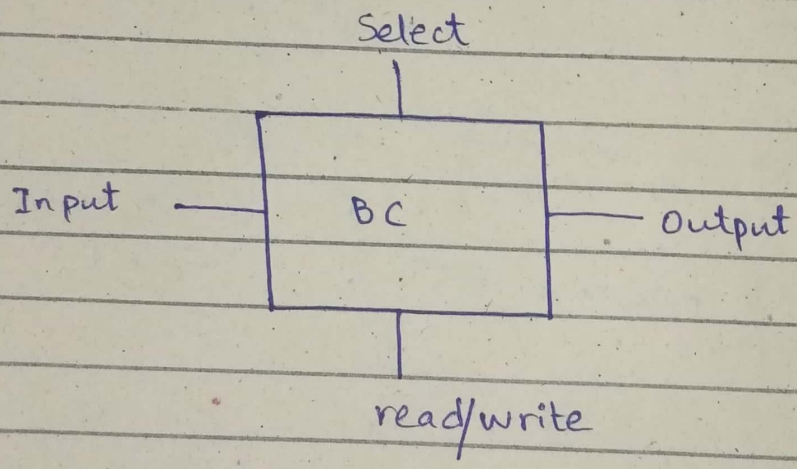
As clock pulse is same for every flip flop, so they work at same time i.e, 1st flip flop adopts the new input given by us. But 2nd flip flop is working at same time. So, it will not wait for 1st flip flop to complete its cycle., rather it will operate at same time and will adopt the already stored value of 1st flip-flop. That's why, the already stored info is shifted to right(1-bit) and input is stored at 1st place.

Binary Cell:



[0 - Write
1 - Reading] → on R/W

Black Box Circuit:



* To make it 1 – write and 0 → read, adding ~~remove~~ the NOT gate connecting R/W with last AND gate.

means hard disk have sectors and rotates around plates by reading and after finding start writing.

RAM → volatile memory

* Sequential Access
* Direct / Random Access

4 Byte RAM

1 Byte
1-Byte
1-Byte
1-Byte

* CPU tells address to RAM, hence RAM returns that due to its sequential memory units → that's why RAM is fast

4 x 2 RAM

↓

4 locations → have unique address (00, 01, 10, 11)

For 4 bit, address is 2-bit: $4 = 2^2$

2-bit
2-bit
2-bit
2-bit

2-bit Address

$16 = 2^4$

For 16 Bit, address ⇒ 4-bit

RAM size: 1MB Address=?

10^{20}

↓

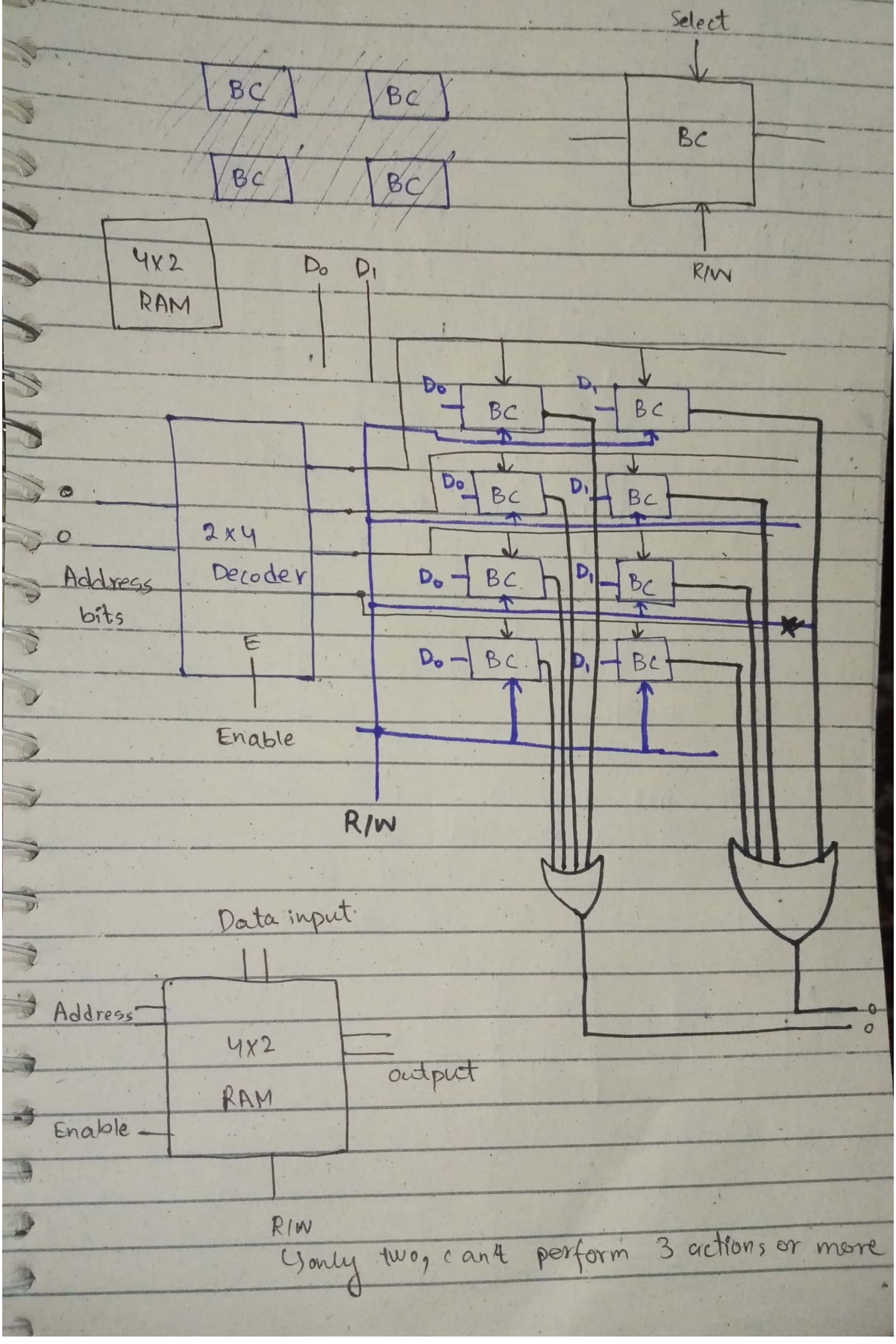
20-bit size address

8 kB = 8 x 2^{10} = $2^3 \times 2^{10}$ = 2^{13}

Address = 13

For every bit there is one

RAM has Direct Access to Data.



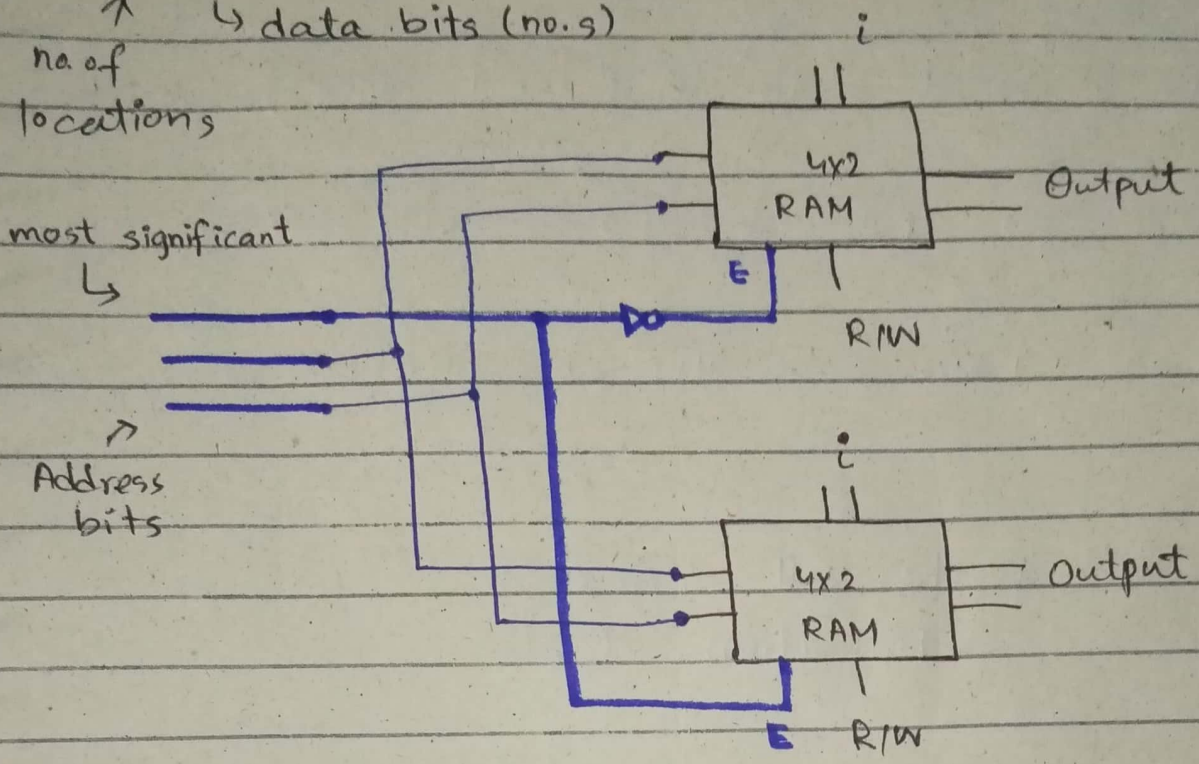
Only two, can't perform 3 actions or more

19-12-2023

8 x 2 RAM by using 4x2 RAM:

↑ no. of locations

↳ data bits (no.s)



Comparator, multiplier circuit, BCD converter, Decoder, Encoder, MUX, de-MUX

Sequential Circuits

Registers

RAM

⊛ Detailed topics' of Sir's Lecture