classmate
Date
Page

Module V

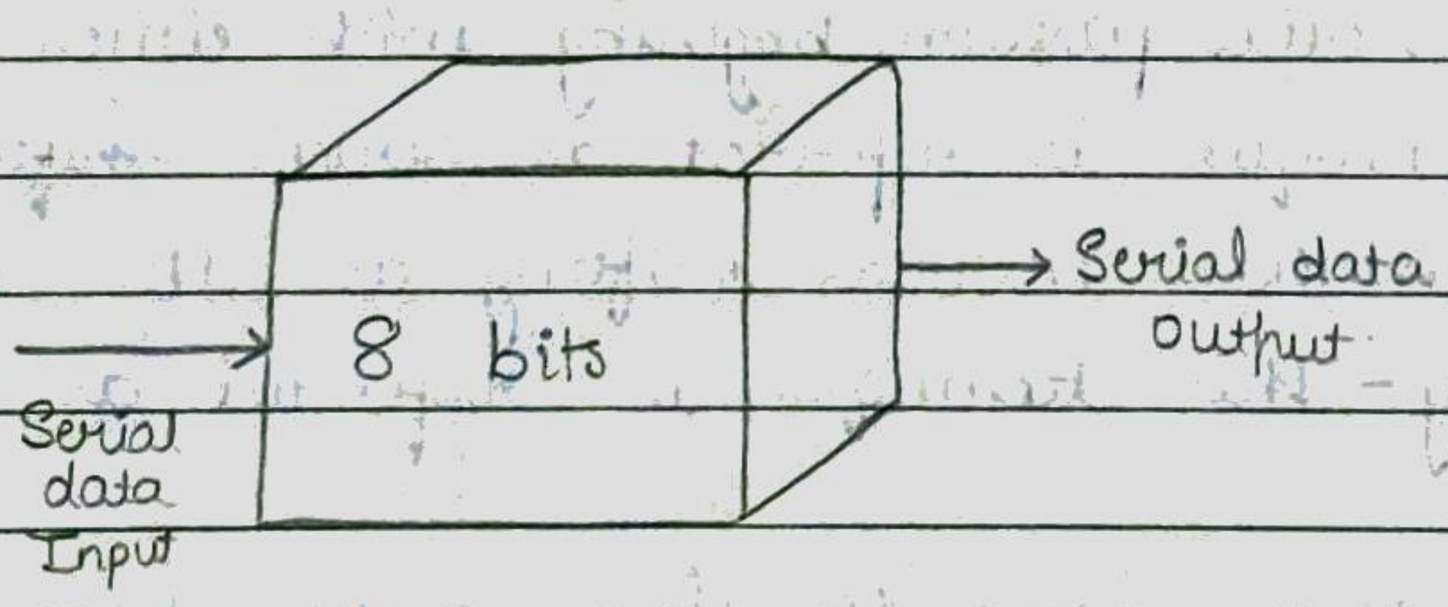
⇒ Registers and Counters

A register is a group of flip-flops that can be used to store a binary number.
There must be one flip-flop for each bit in the binary number. For example, a register used to store 8 bit binary number must have 8 flip-flops.
The flip-flops must be connected such that the binary number can be entered or 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.

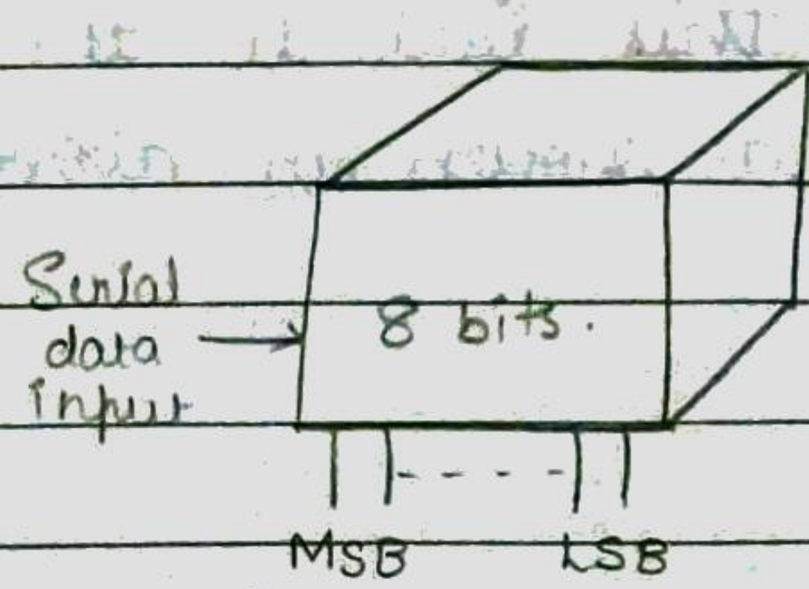
Types of Shift registers:

1. Serial In Serial Out
2. Serial In Parallel Out
3. Parallel In Serial Out
4. Parallel In Parallel Out

a SERIAL IN SERIAL OUT

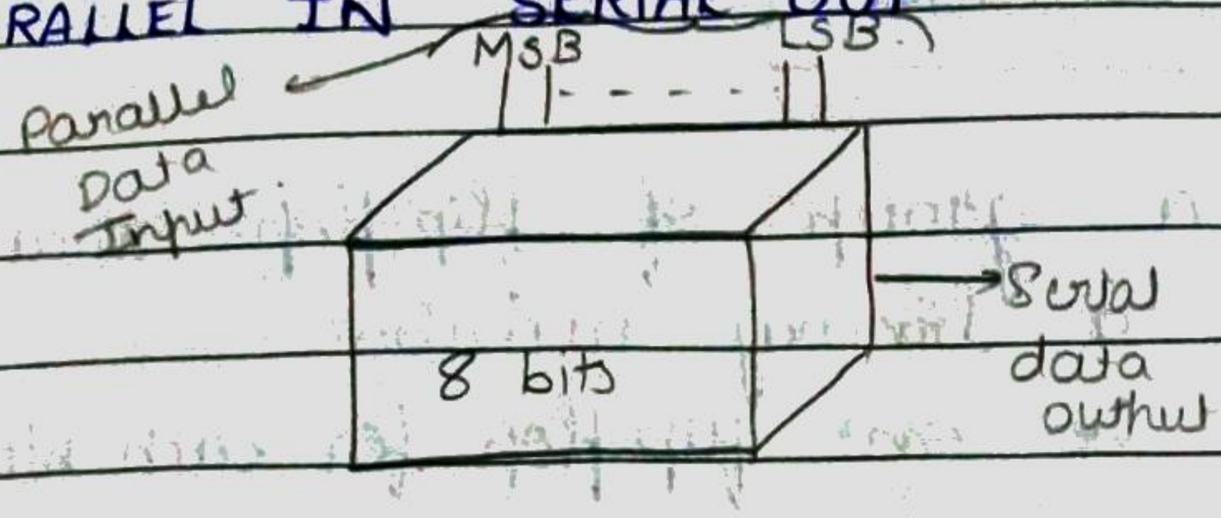


b SERIAL IN PARALLEL OUT

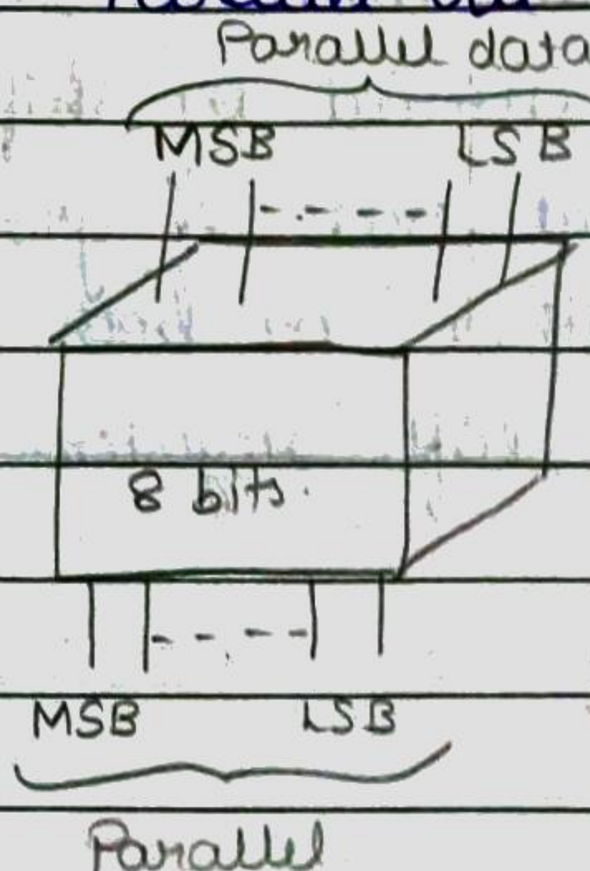


Parallel data output

c PARALLEL IN SERIAL OUT



d Parallel in Parallel out

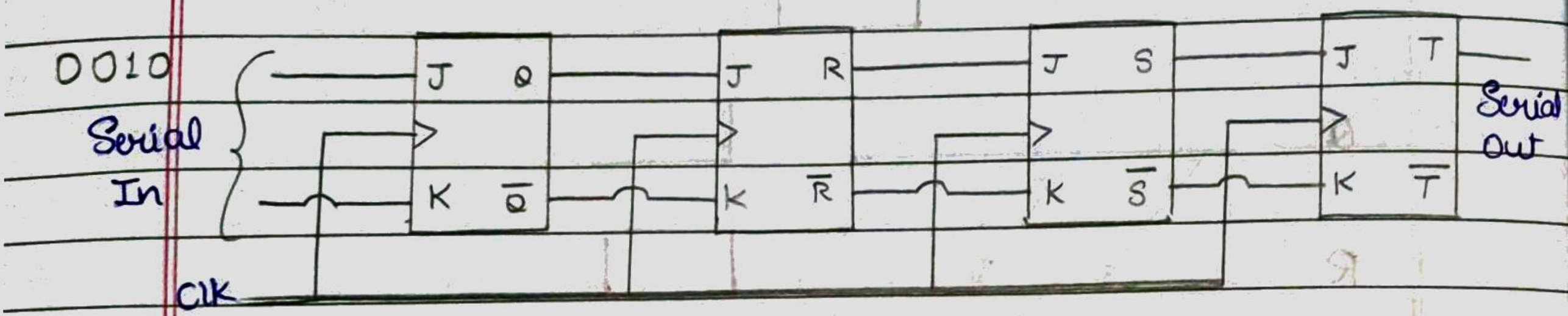


→ The bits in a binary number can be moved from one place to another in either of two ways.
→ The first method involves shifting the data one bit at a time in a serial fashion, beginning with either the MSB or LSB - This technique is referred as serial shifting.
→ The second method involves shifting all the data bits simultaneously - This technique is referred as parallel shifting.
→ There are two ways to shift data into a register, serial or parallel & similarly two ways to shift the data out of the register - This leads to the construction of 4 basic register type as shown in above figure.

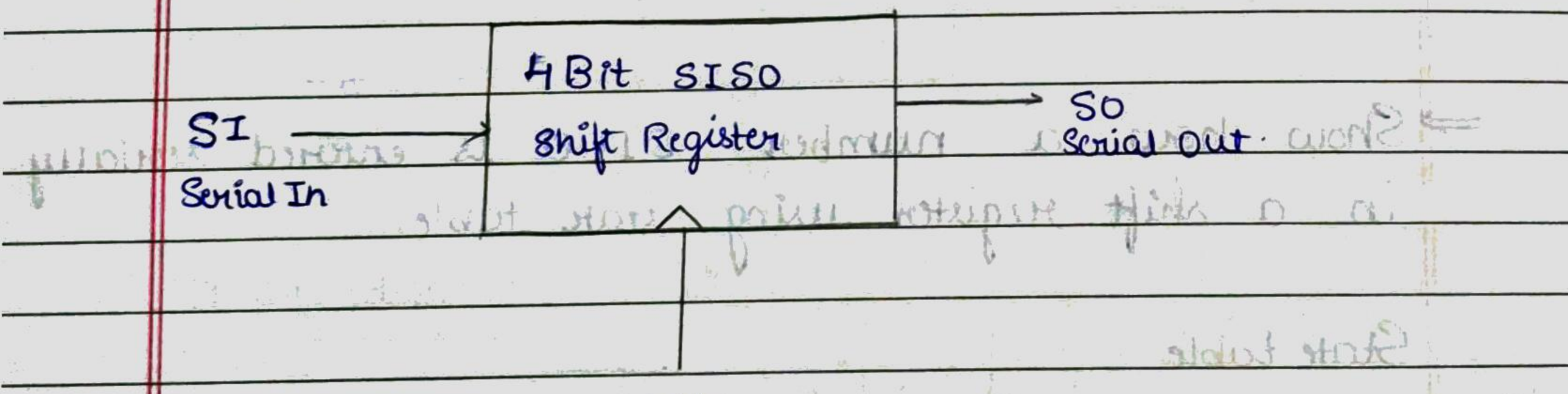
1 SERIAL IN SERIAL OUT (SISO)

→ Design 4 bit SISO Shift Register using JK flipflops

Logic diagram

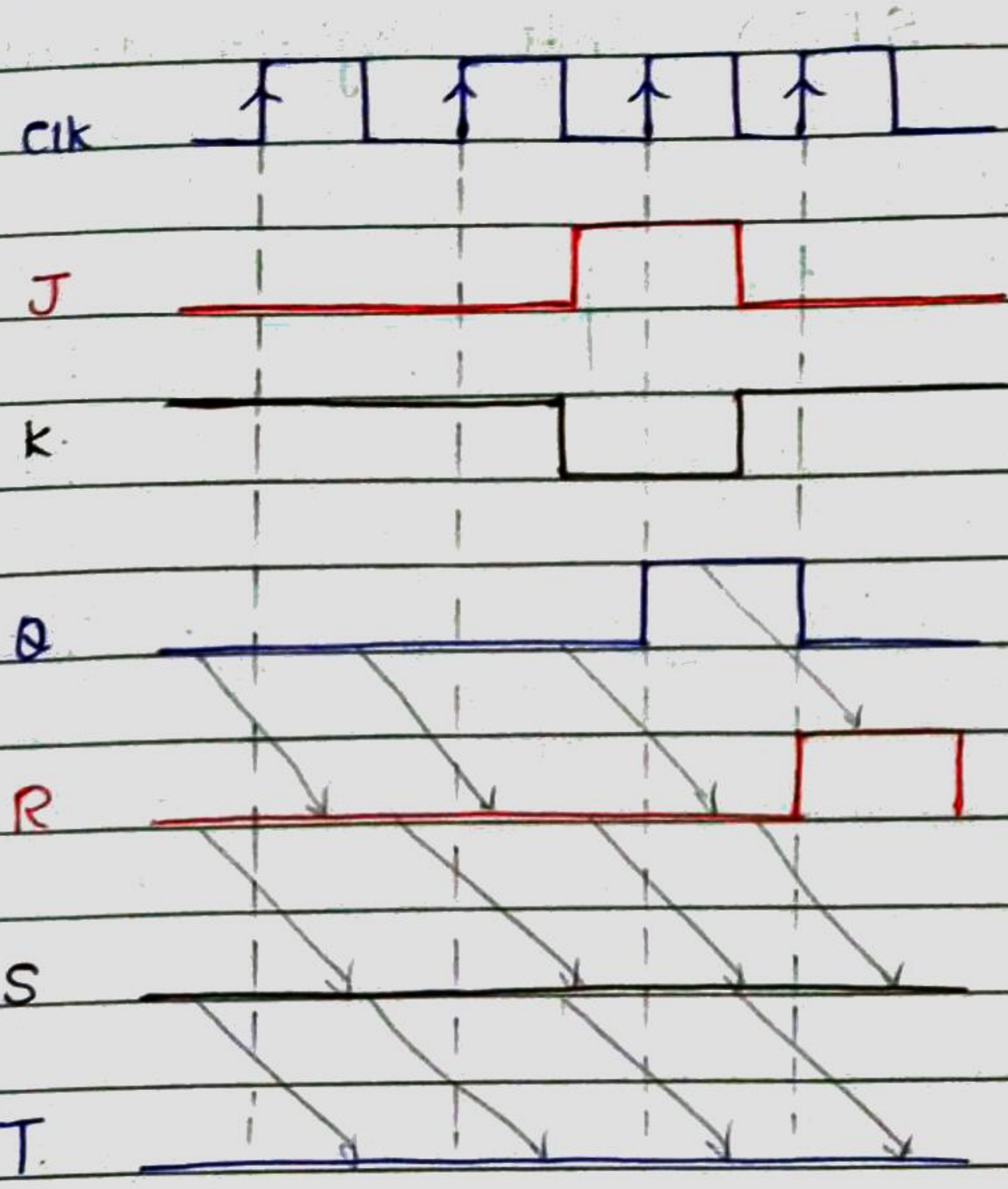


Block diagram



Date
Page

Timing diagram



⇒ Show how a number 0100 is entered serially in a shift register using state table.

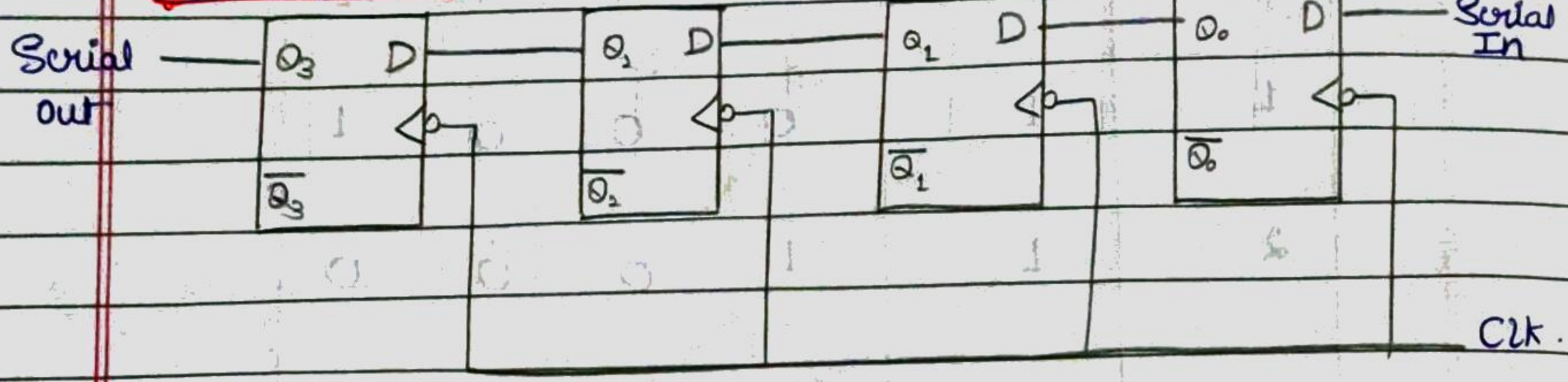
State table.

Clock	SI	Q	R	S	T
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4		0	0	1	0

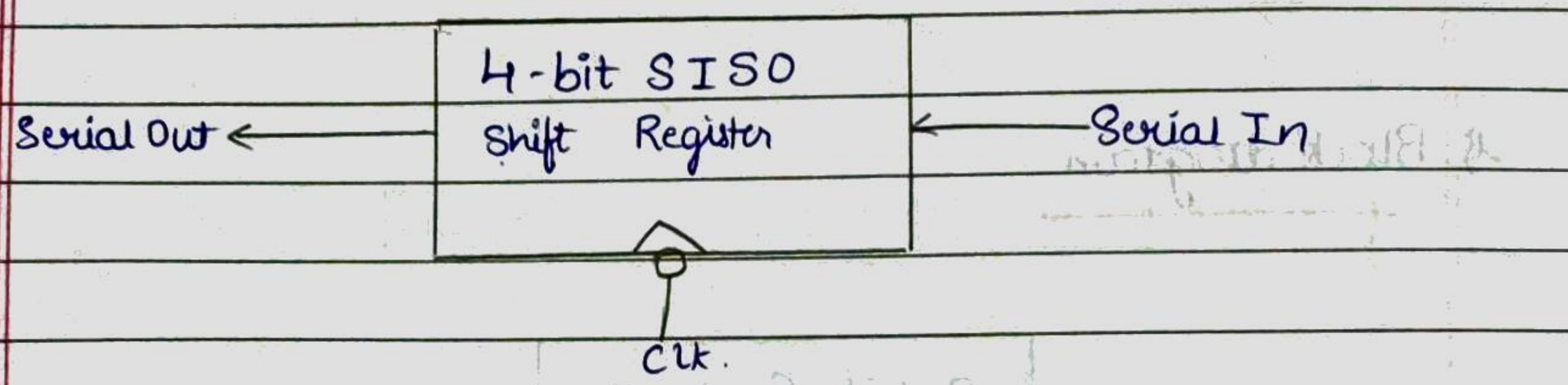
⇒ Design 4 bit SISO left shift register using D flip-flop (Negative edge triggered).

⇒ Initial contents of the register is 0011 ($Q_0 Q_1 Q_2 Q_3$)

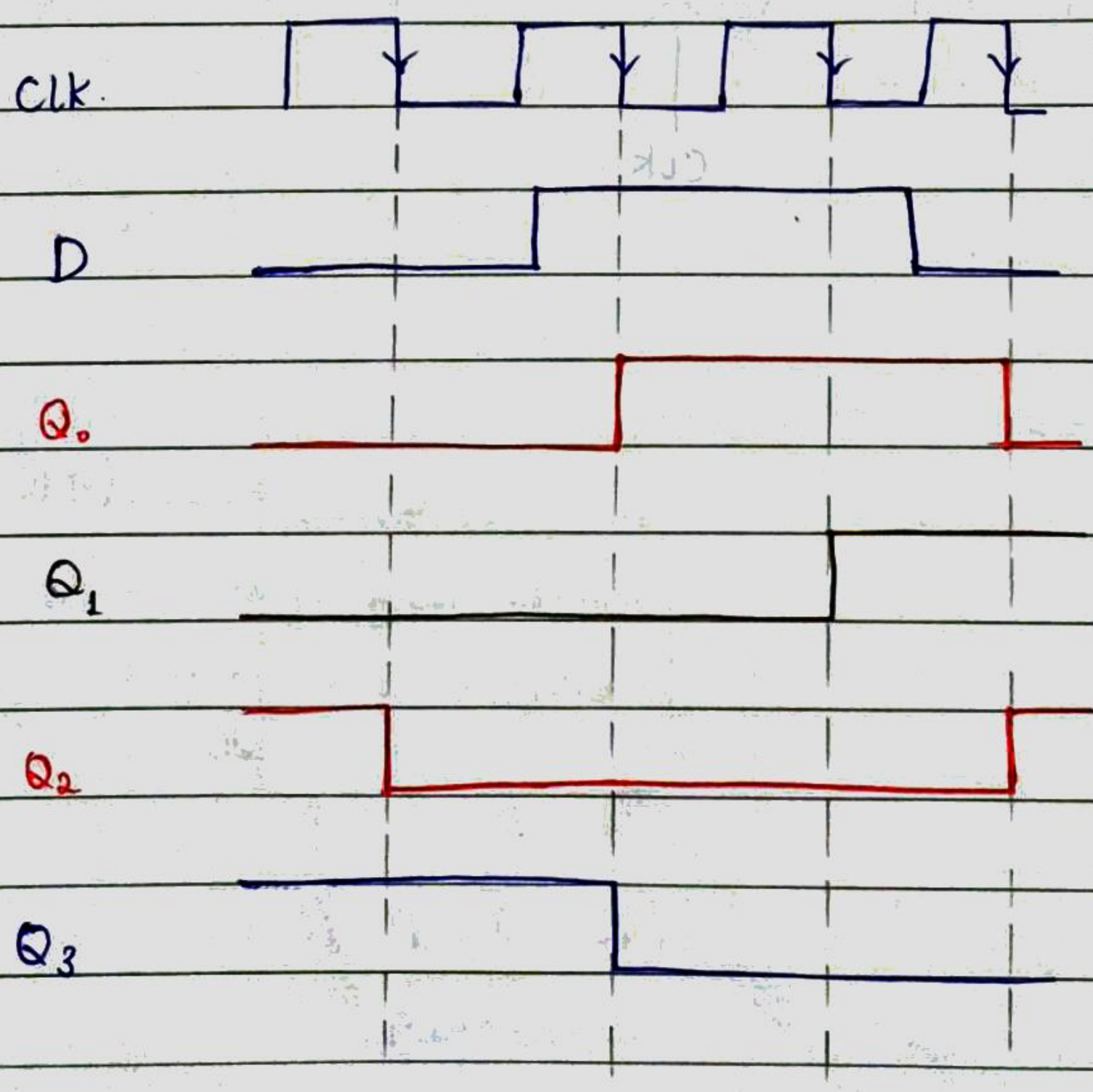
Logic Diagram



Block diagram.



Timing diagram



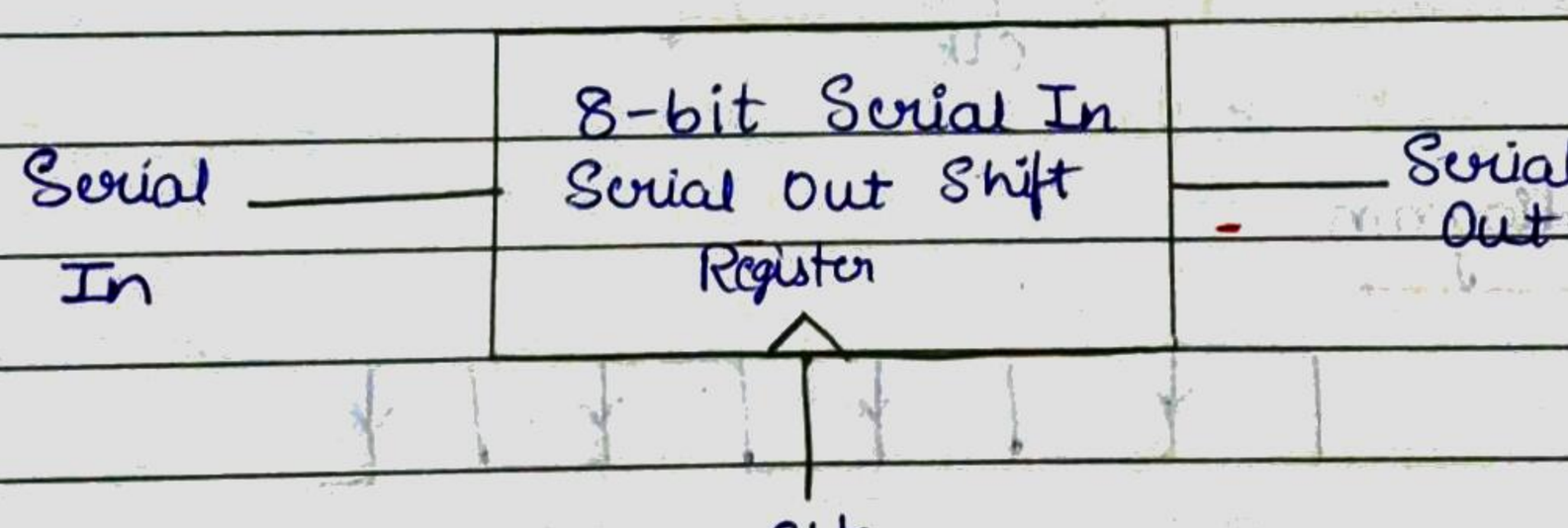
Date
Page

State table

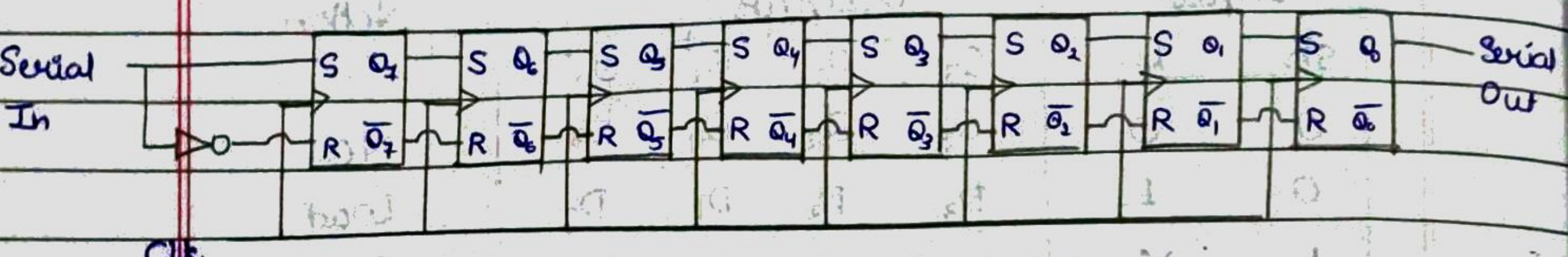
Clk	SI	Q_0	Q_1	Q_2	Q_3
0	0	0	0	1	1
1	1	0	0	0	1
2	1	1	0	0	0
3	0	1	1	0	0
4		0	1	1	0

$\Rightarrow$

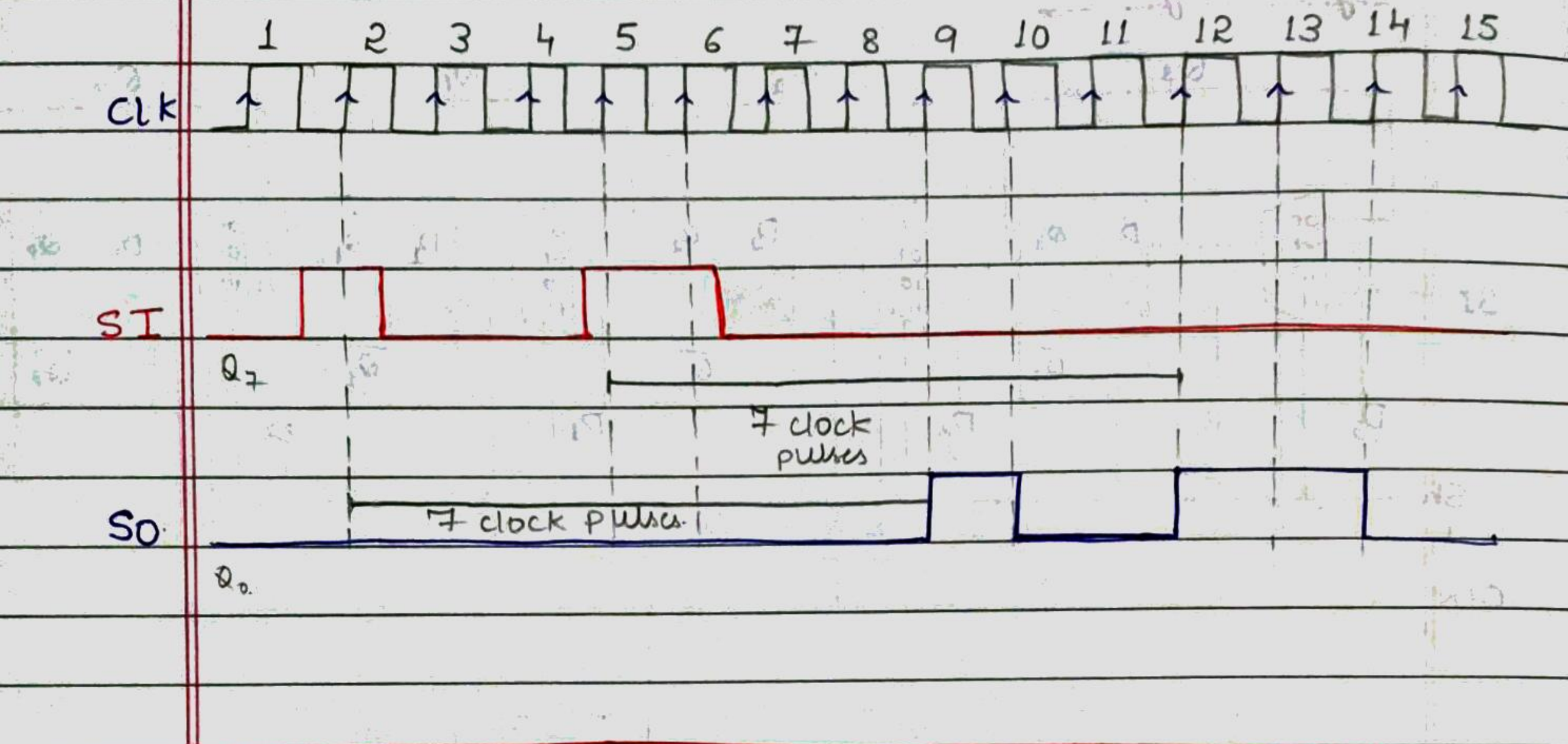
a) Block diagram



b) Logic diagram



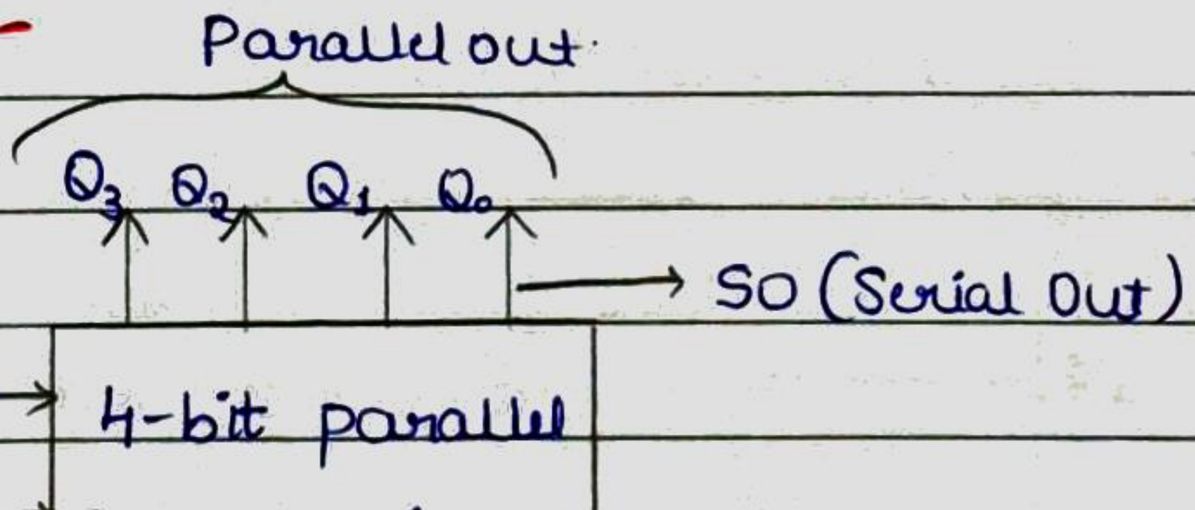
c) Timing diagram



10M ⟹ Parallel In Parallel out Right Shift Register.

⟹

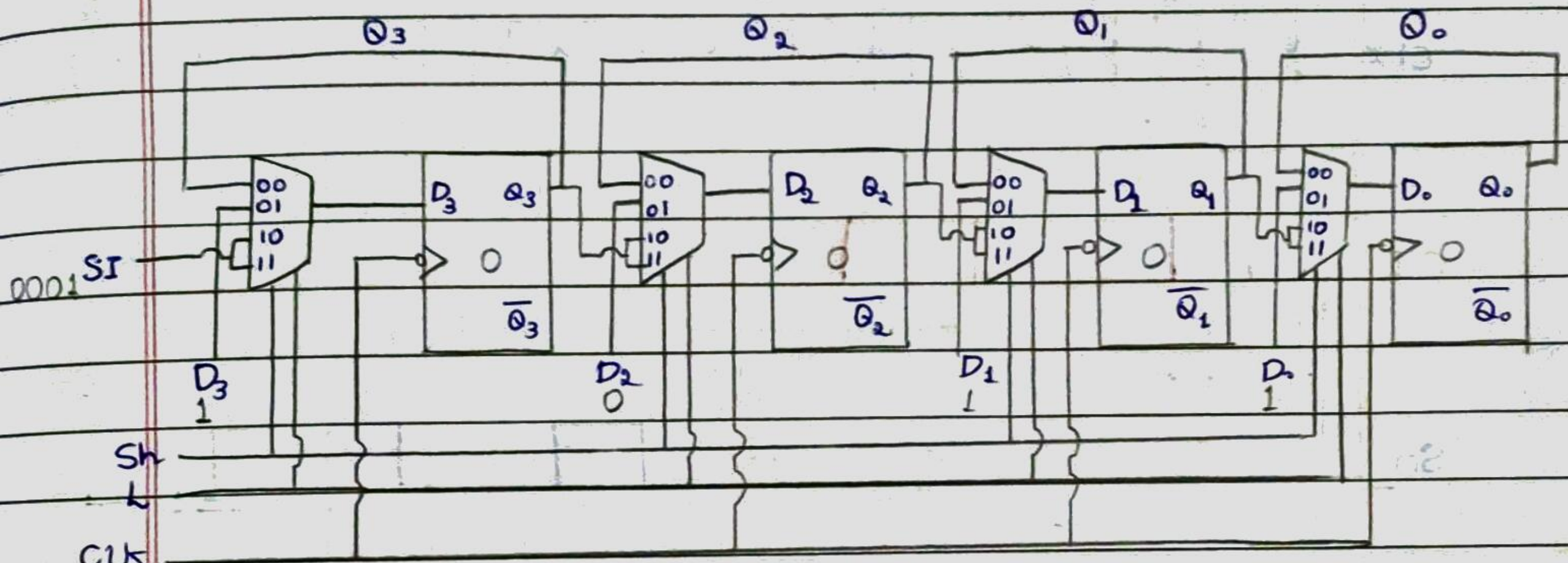
Block diagram.



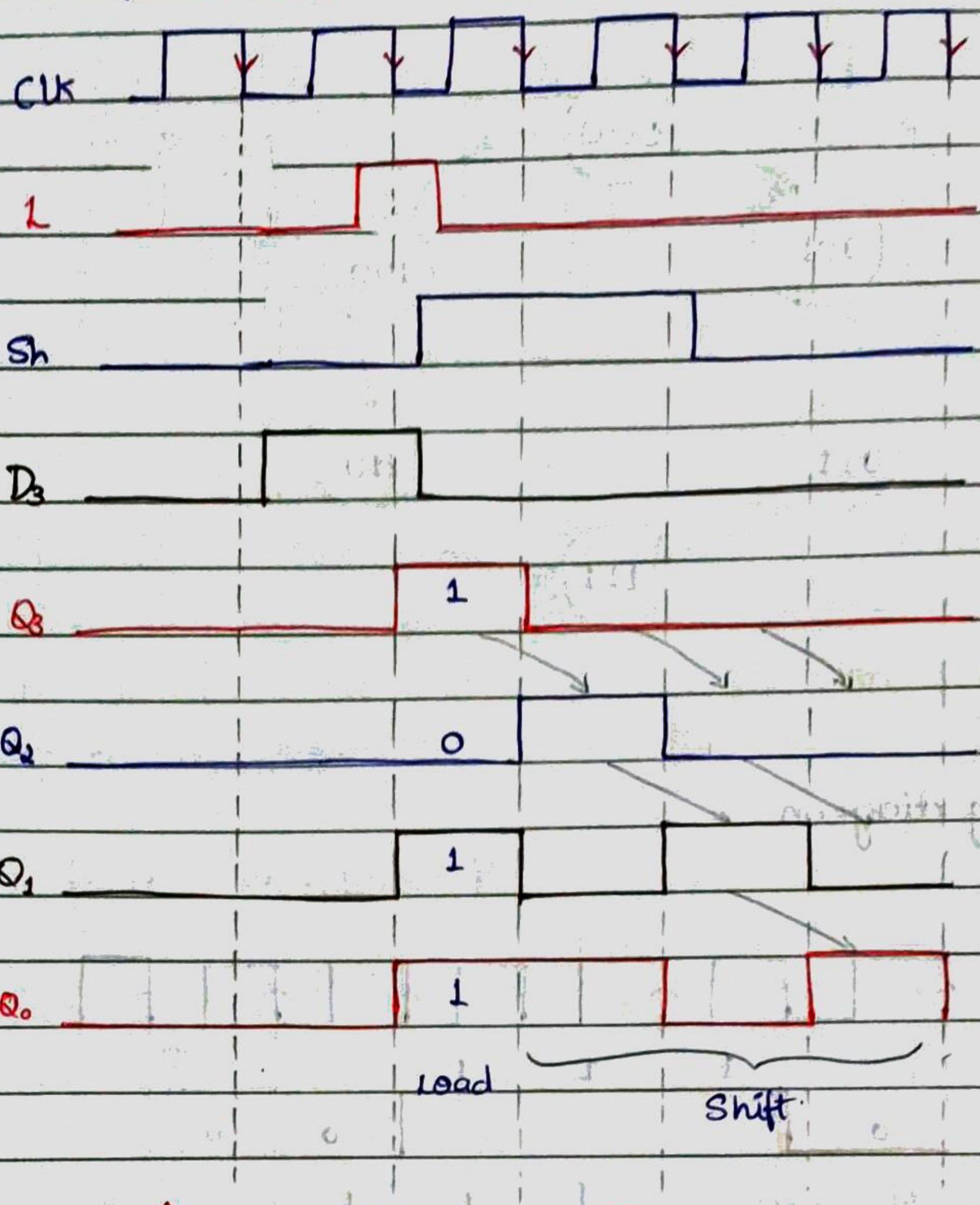
Shift Register Operation.

Inputs		Outputs				Action
Sh	L	Q_3^+	Q_2^+	Q_1^+	Q_0^+	
0	0	Q_3	Q_2	Q_1	Q_0	NC
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Shift

Logic diagram.



Timing diagram

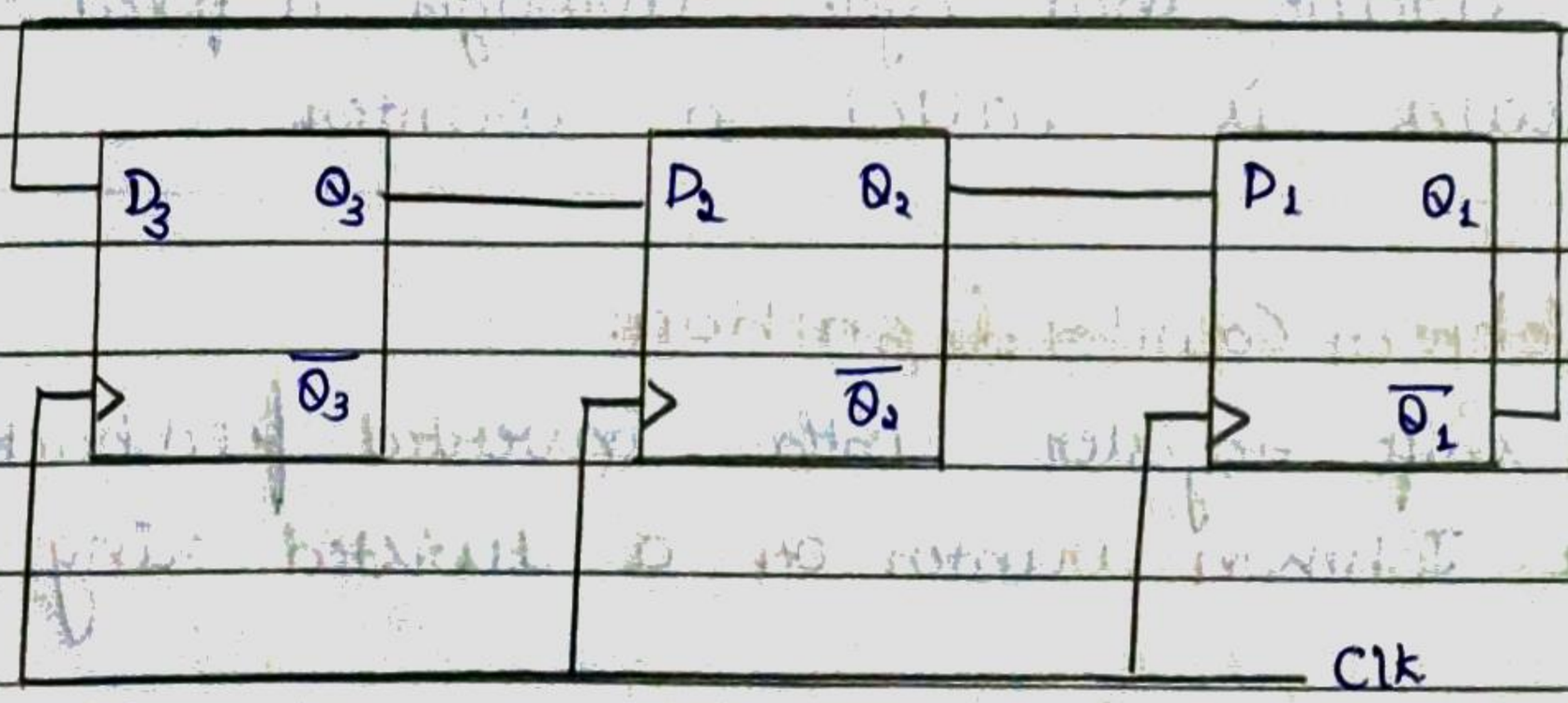


App of Register

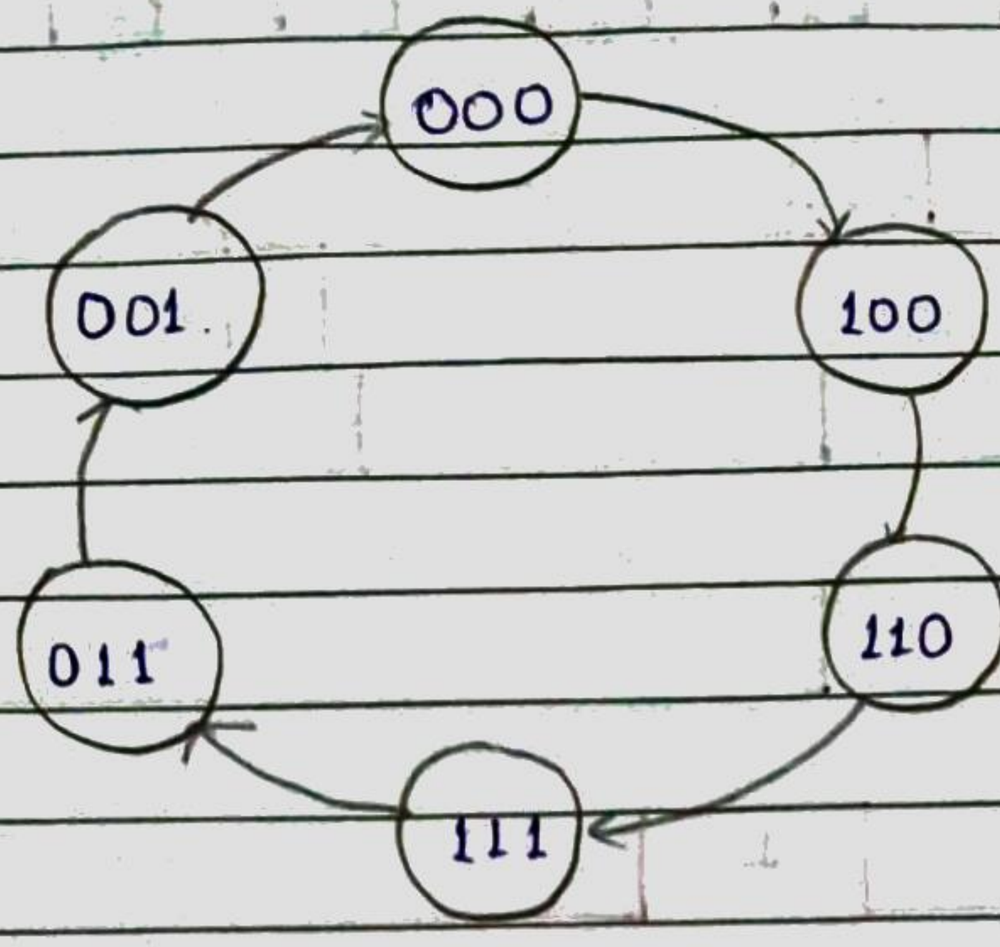
Applications of Registers

Johnson counter (or) twisted ring counter.

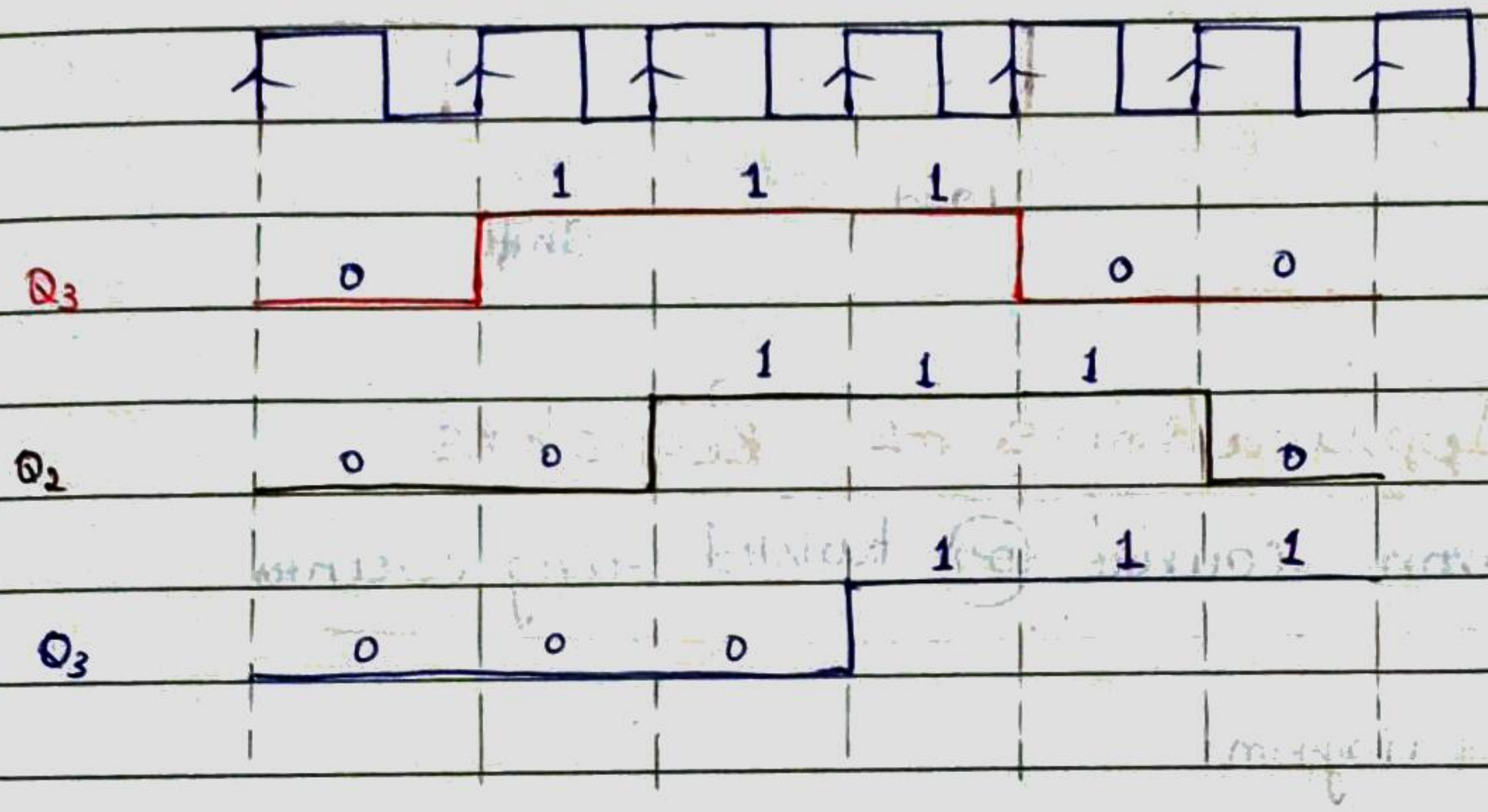
Logic diagram



Transition Graph or state diagram



Timing diagram



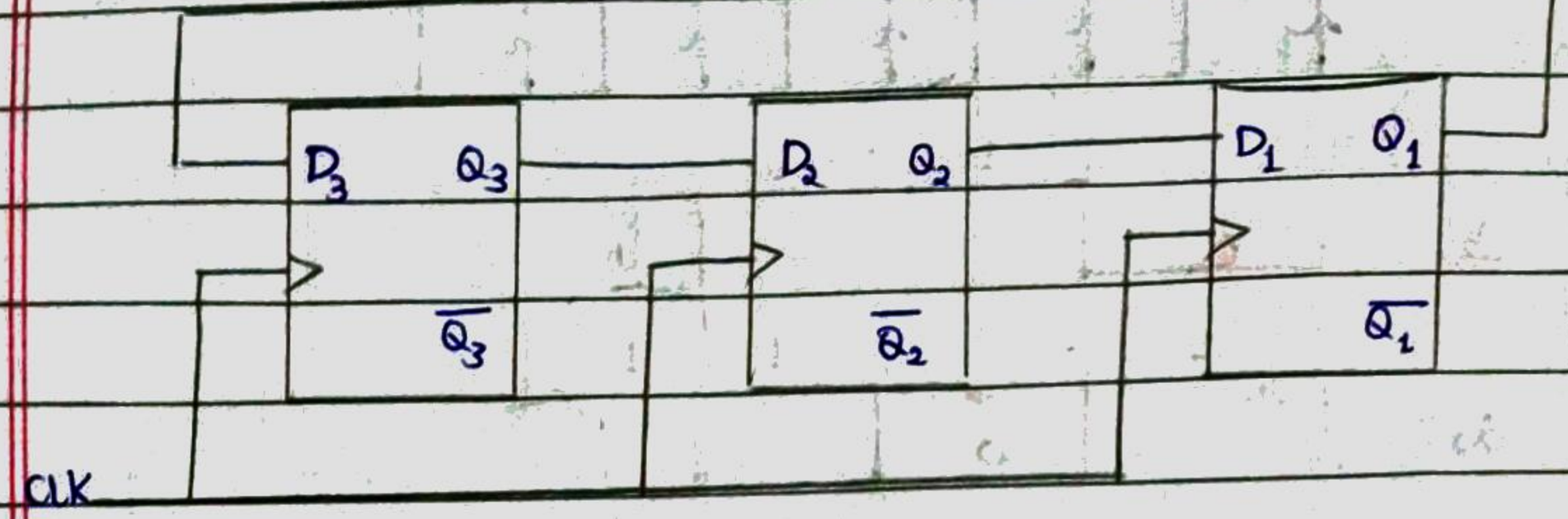
Counter definition:
A circuit that cycles through a fixed sequence of states is called a counter.

Johnson Counter definition:
A shift register with inverted feedback is called a Johnson counter or a twisted ring counter.

App of register

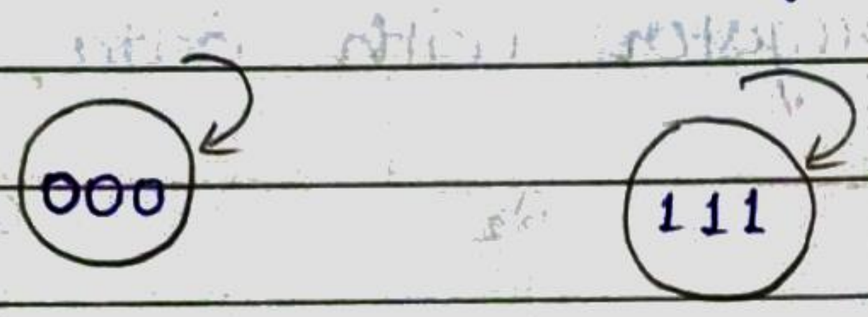
Applications of Registers

⇒ Ring counter

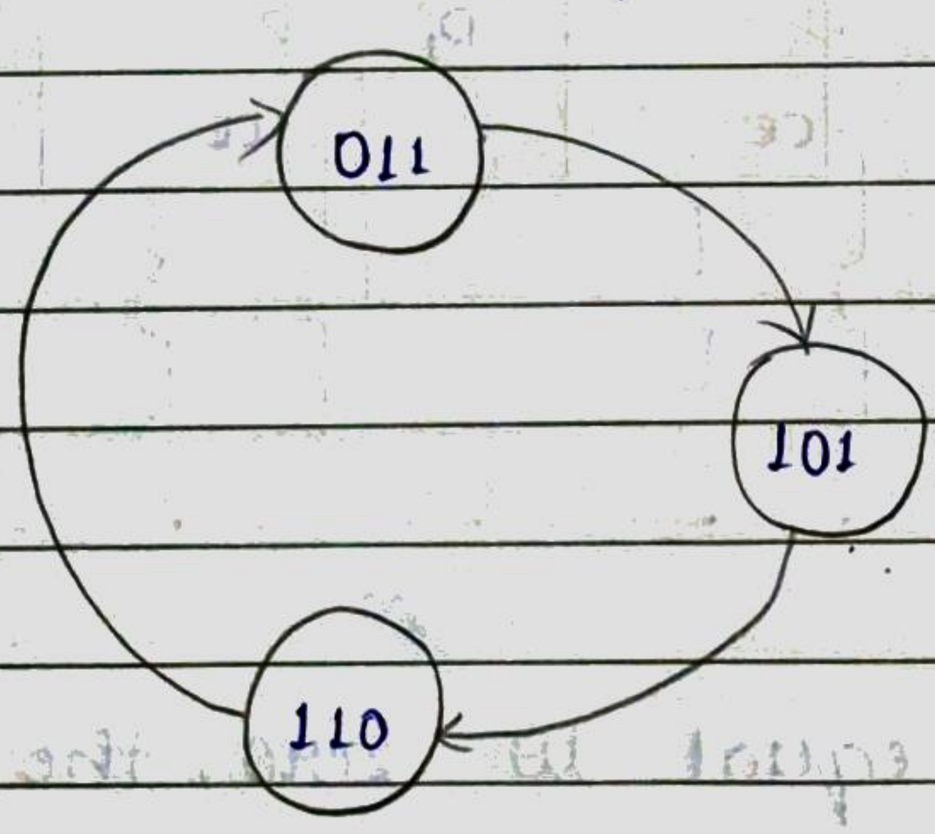


1 ⇒ If the data in flipflops is 000 (or) 111 then all the states will be similar i.e., 000 (or) 111
↳ we can use this for application which is repeatedly produce a same state.

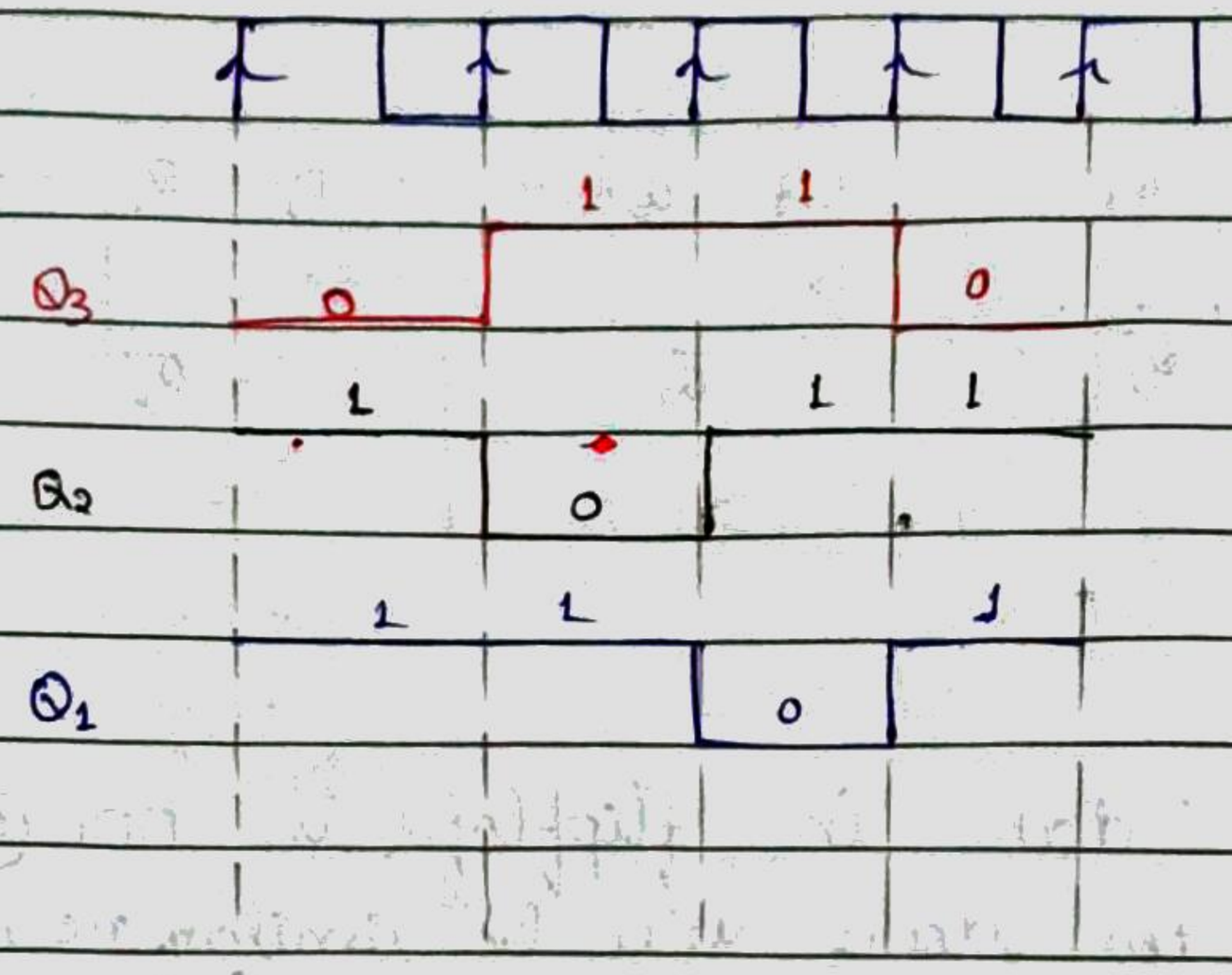
⇒ State or transition diagram



2. If the data in flipflops is 011, then

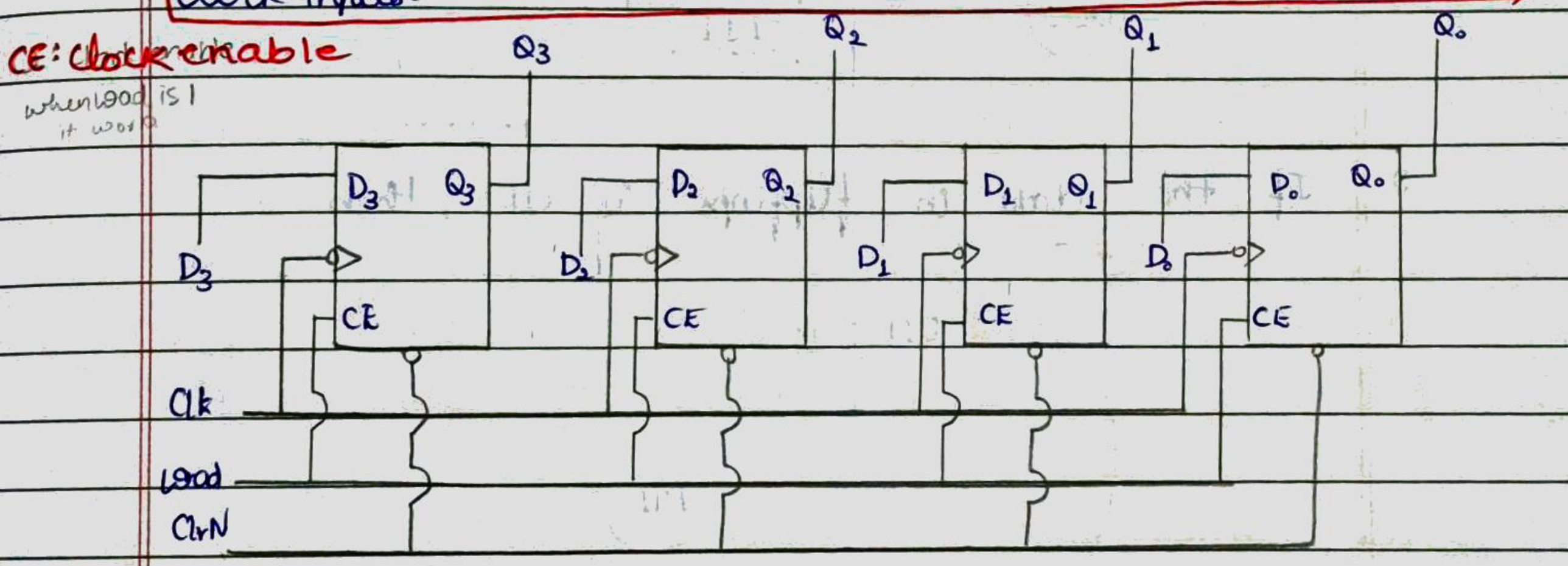


Timing assuming data in a register at 011.



A shift register with non inverted feedback is called a ring counter.

⇒ 4-bit D-flipflop register with data, load, clear & clock inputs.



→ When load is equal to zero, the register is not clocked & holds its present value.

→ When load is equal to one, the clock is transmitted to the flipflop & the data applied to D inputs will be loaded into the flipflops on the falling edge of the clock.

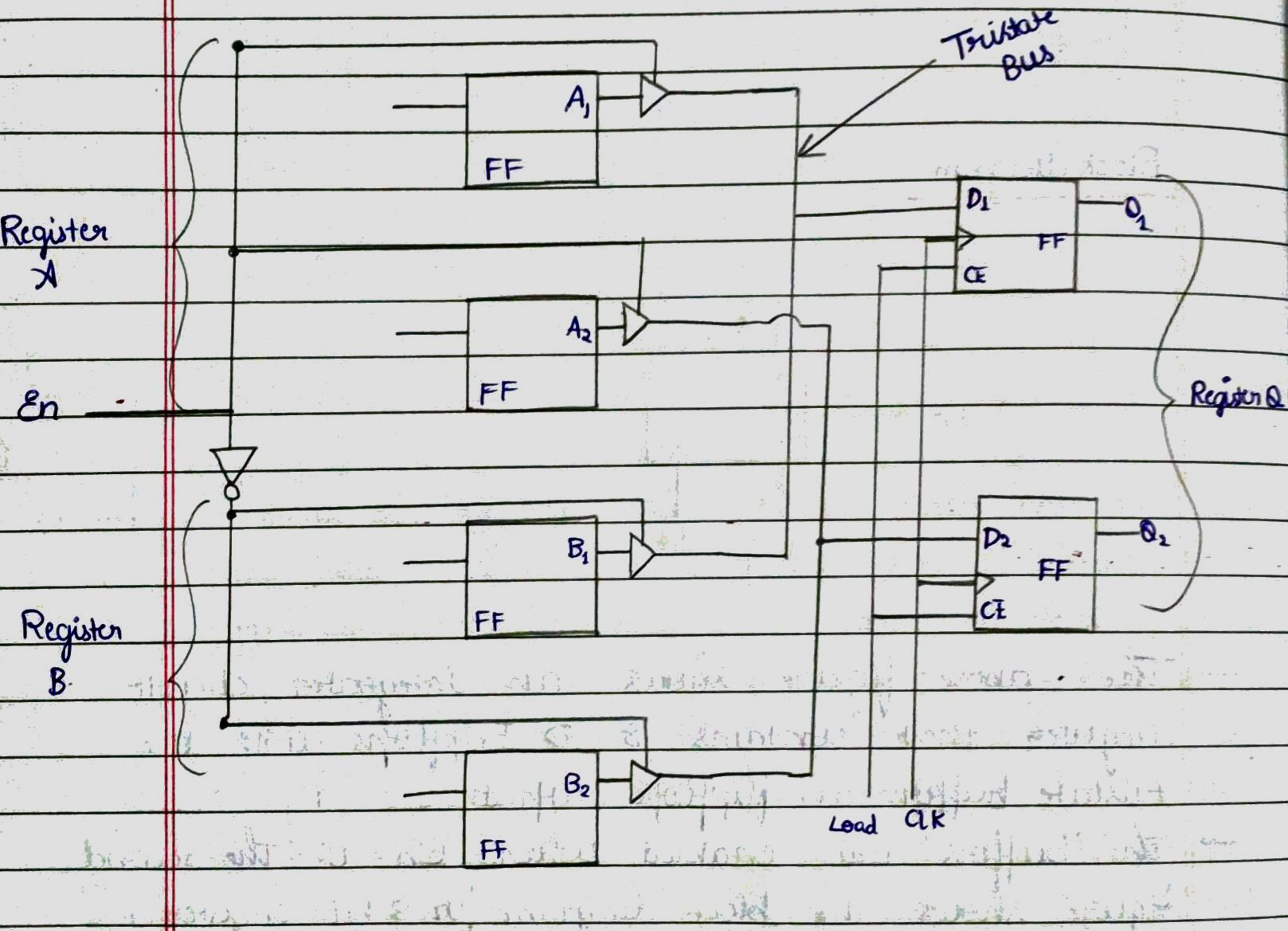
For example: if the Q outputs $Q_3Q_2Q_1Q_0$ is 0000 & the data inputs $D_3D_2D_1D_0$ is 1101

After the falling edge of the clock, Q will change from

0000 to 1101.

→ the clrN is a asynchronous input, the bubble at the clear input indicate that the logic '0' is required to clear the flipflop.

Data transfer between the register.



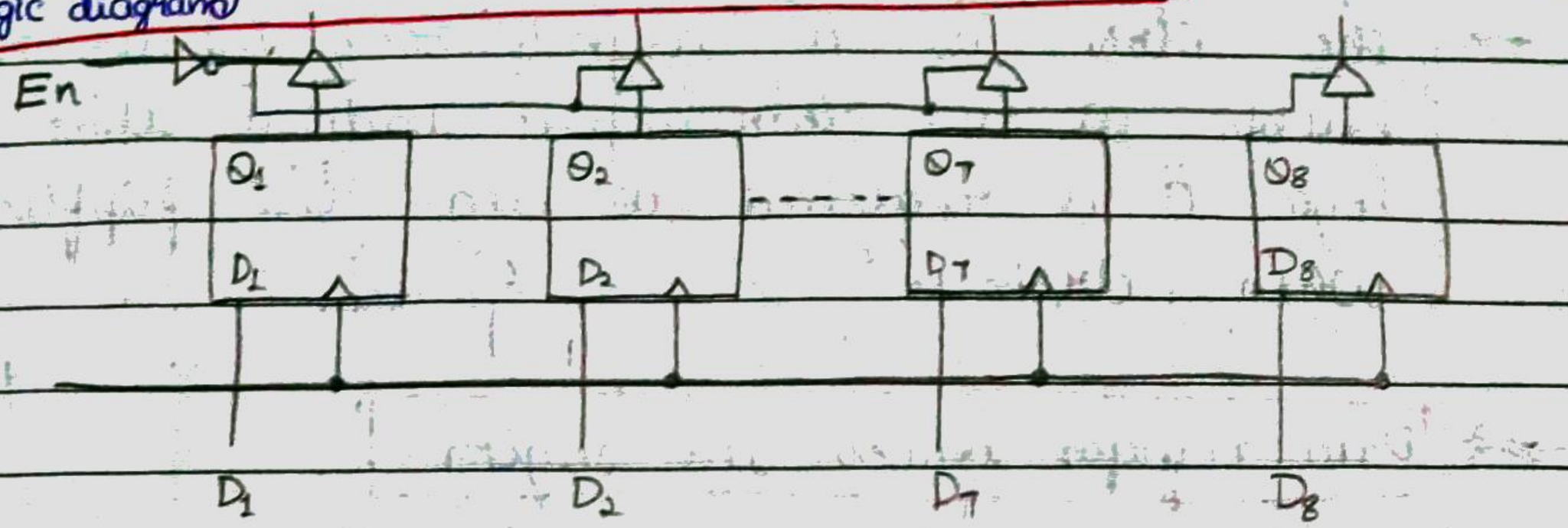
Register A = FF A_1 & A_2

Register B = FF B_1 & B_2

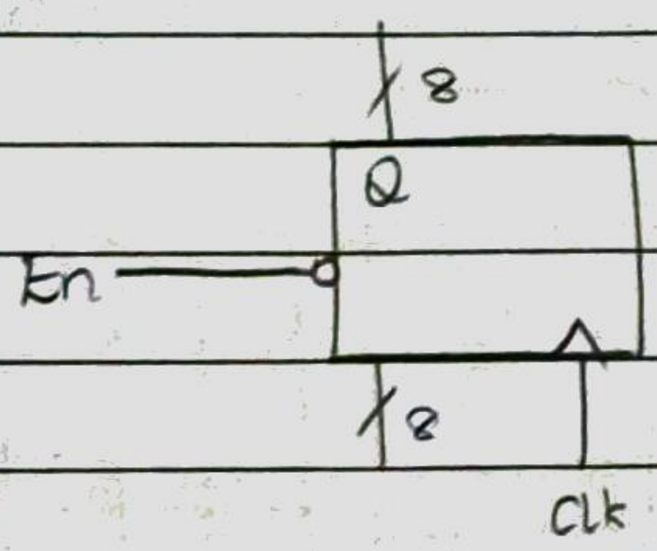
Register Q = FF Q_1 & Q_2.

⇒ 8-bit register with tristate output

Logic diagram



Block diagram

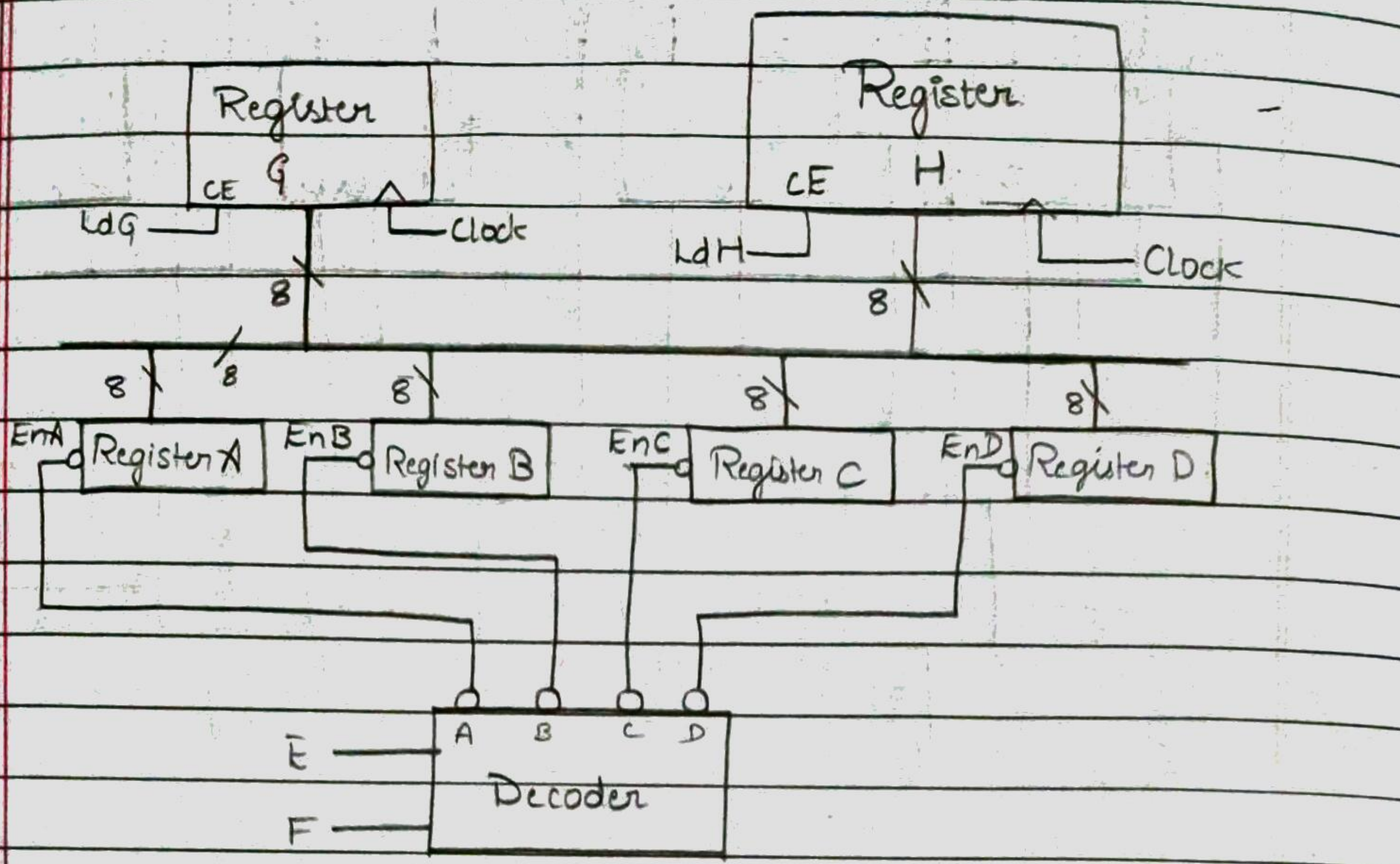


→ The above figure shows an Integrated circuit register that contains 8 D Flipflops with the tristate buffers at flipflop outputs.

→ The buffers are enabled when En = 0. The second figure shows the block diagram for 8 bit register.

Date
Page

Data Transfer using the tristate bus



→ The above figure shows how data can be transferred from one of the four 8-bit registers into one of two other registers.

→ The operation can be summarized as follows

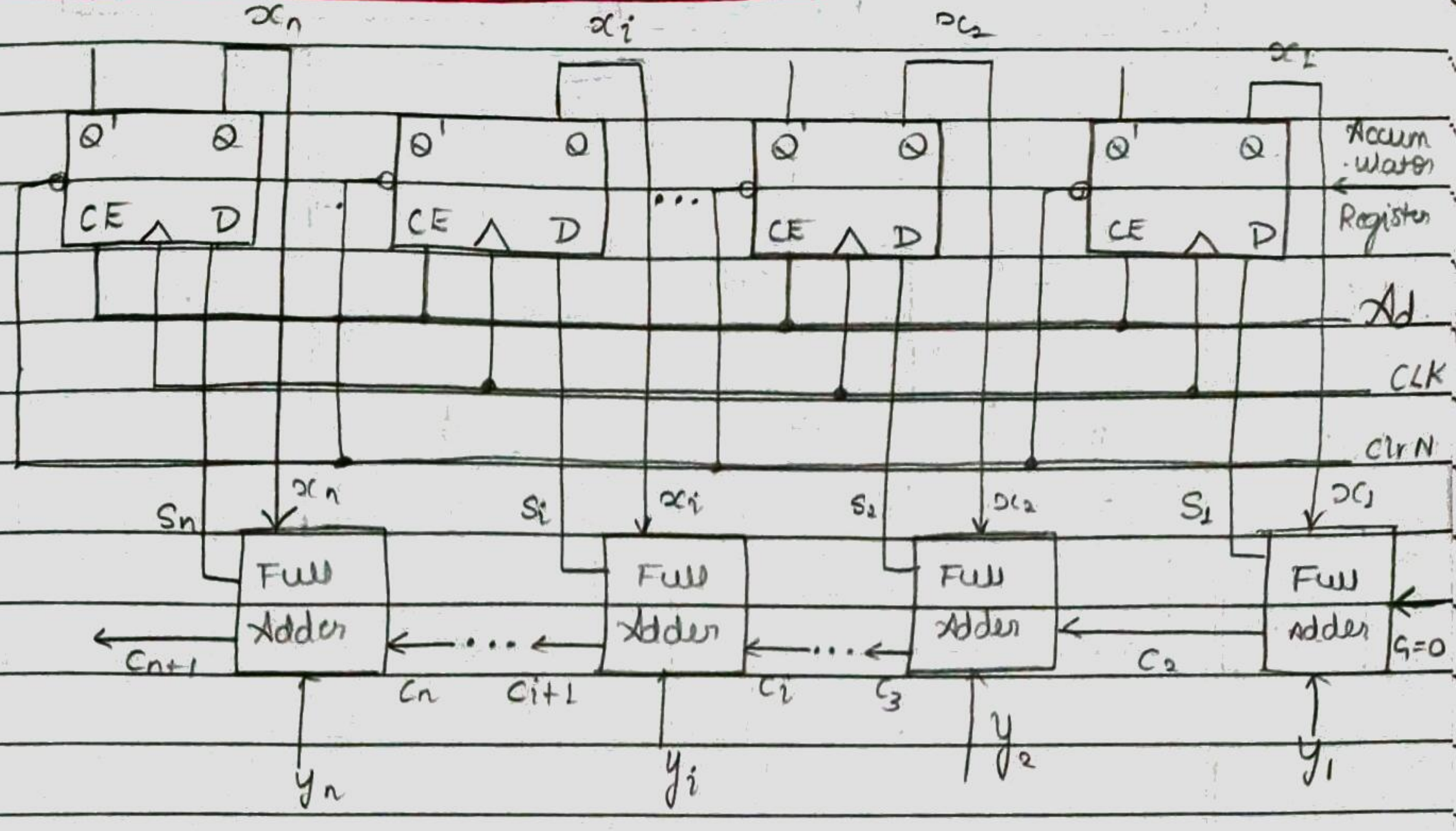
i If EF = 00, then 'A' is stored in 'G' or 'H'

ii If EF = 01, then 'B' is stored in 'G' or 'H'

iii If EF = 10, then 'C' is stored in 'G' or 'H'

iv If EF = 11, then 'D' is stored in 'G' or 'H'

Parallel Adder with Accumulator



→ In computer circuits, it is frequently desirable to store one number in a register of flipflops (called an accumulator) and add a second number to it, leaving the result stored in the accumulator.

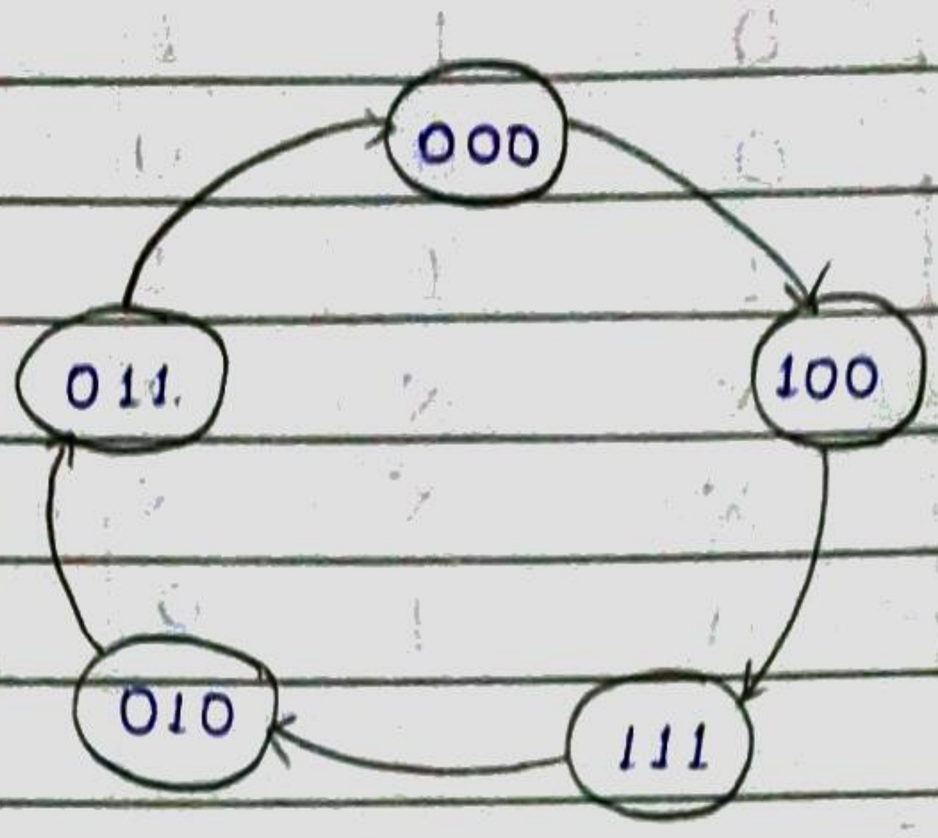
→ Suppose that the number $X = x_n \cdots x_2 x_1$ is stored in the accumulator. Then, the number $Y = y_n \cdots y_2 y_1$ is applied to the full adder inputs, and after the carry has propagated through the adders, the sum of X and Y appears at the adder outputs.

→ An add signal (Ad) is used to load the adder outputs into the accumulator flip-flops on the rising clock edge.

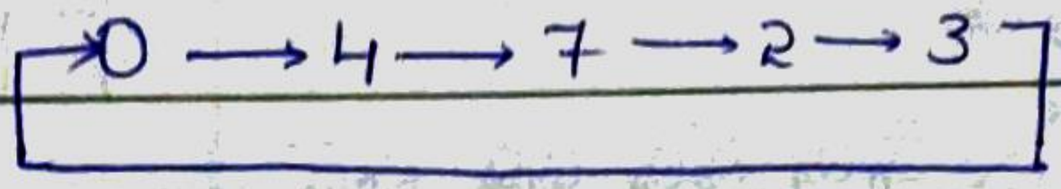
→ If $s_i = 1$, the next state of flip-flop x_i will be 1. If $s_i = 0$, the next state of flip-flop x_i will be 0. Thus, $x_i^+ = s_i$, and if Ad = 1, the number X in the accumulator is replaced with the sum of X and Y, following the rising edge of the clock.

⇒ Design of irregular counter (counter for other sequence)

⇒) 1 Design an upcounter for given state diagram using T flip-flop



(or)



i) T → 111 ⇒ 3 FF we need

ii) T-FF.

iii) Excitation table for T FlipFlop.

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

iv) Next state table.

Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	x	x	x	x	x	x
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	x	x	x	x	x	x
1	1	0	x	x	x	x	x	x
1	1	1	0	1	0	1	0	1

K-map

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	1	x	0	0
Q_C	0	x	1	x

$$T_C = Q_C'Q_B' + Q_CQ_B$$

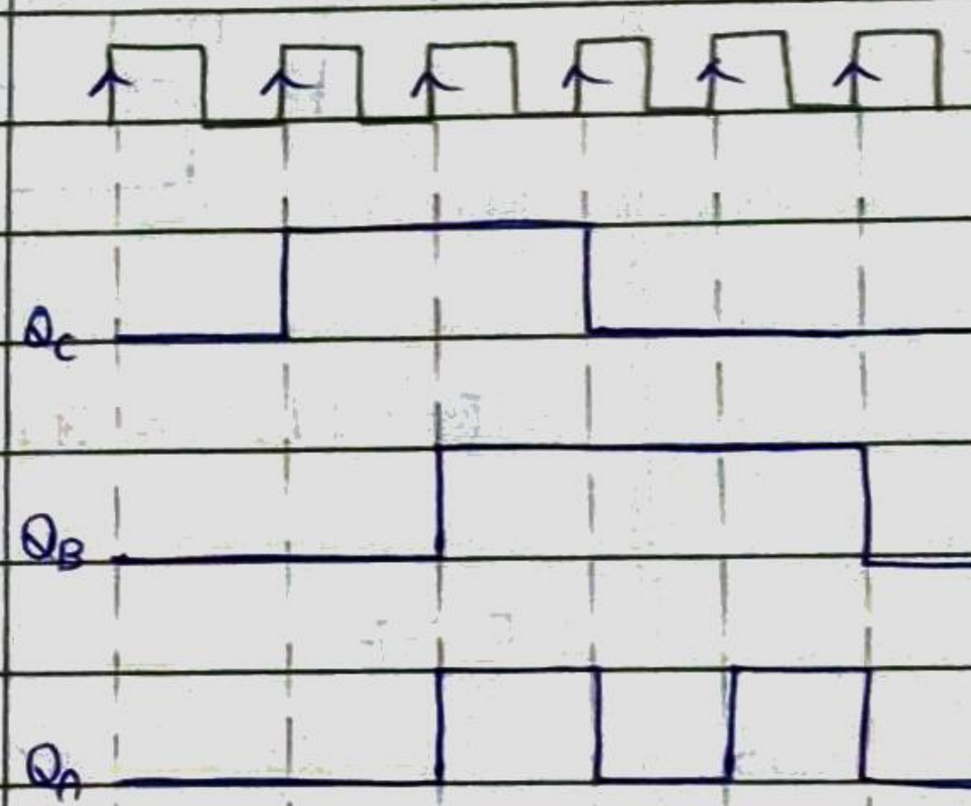
Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	0	x	1	0
Q_C	1	x	0	x

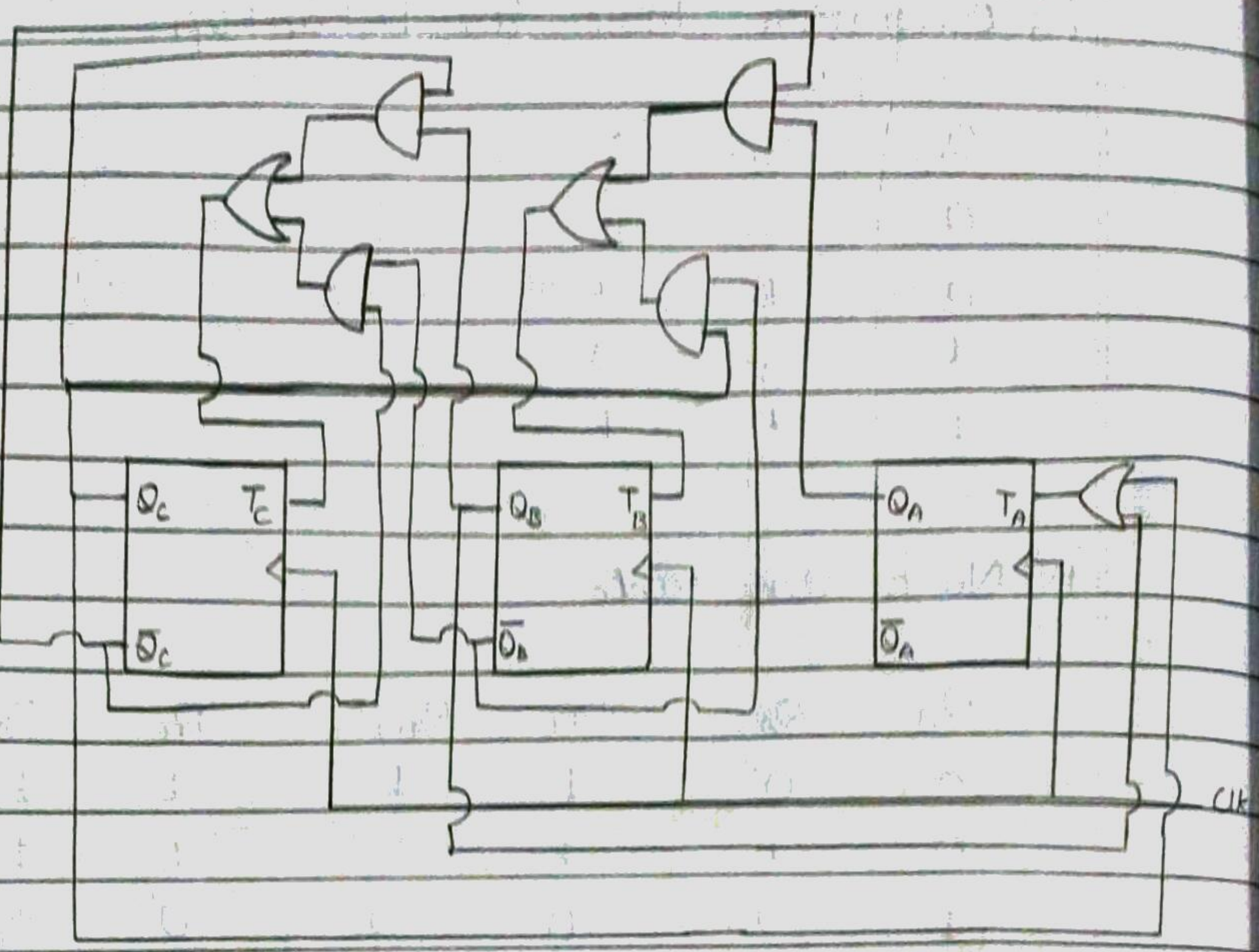
$$T_B = Q_C'Q_A + Q_CQ_B'$$

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	0	x	1	1
Q_C	1	x	1	x

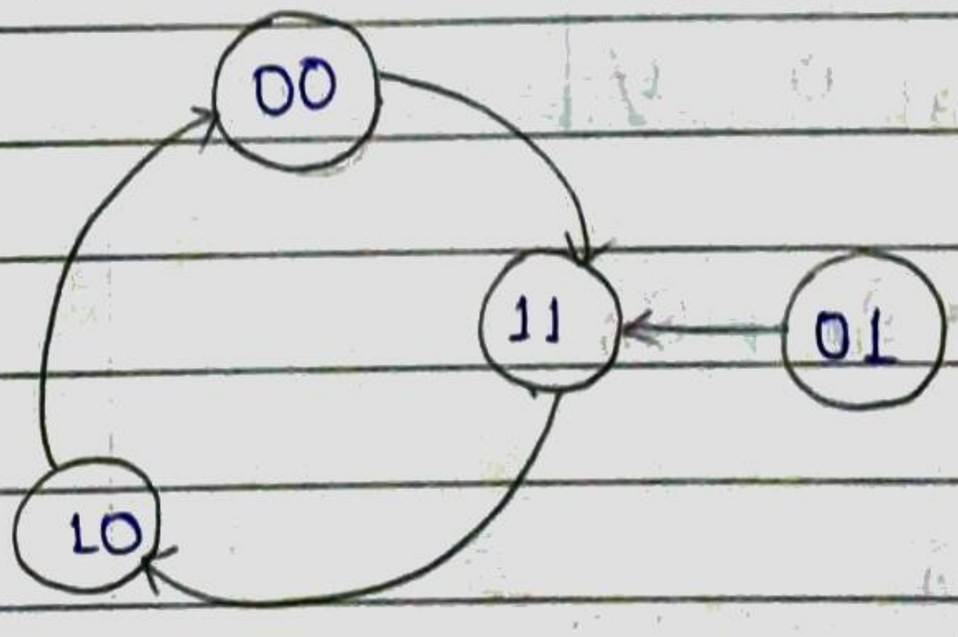
$$T_A = Q_C + Q_B$$

Timing diagram





2. Design an upcounter for given state diagram using D flip-flop.



i) D → 10 ⇒ 2FF

ii) D-FF

iii) Excitation table for D-Flip Flop.

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

iv) Next state table

Q_B	Q_A	Q_{B+1}	Q_{A+1}	D_B	D_A
0	0	1	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	1	1	0	1	0

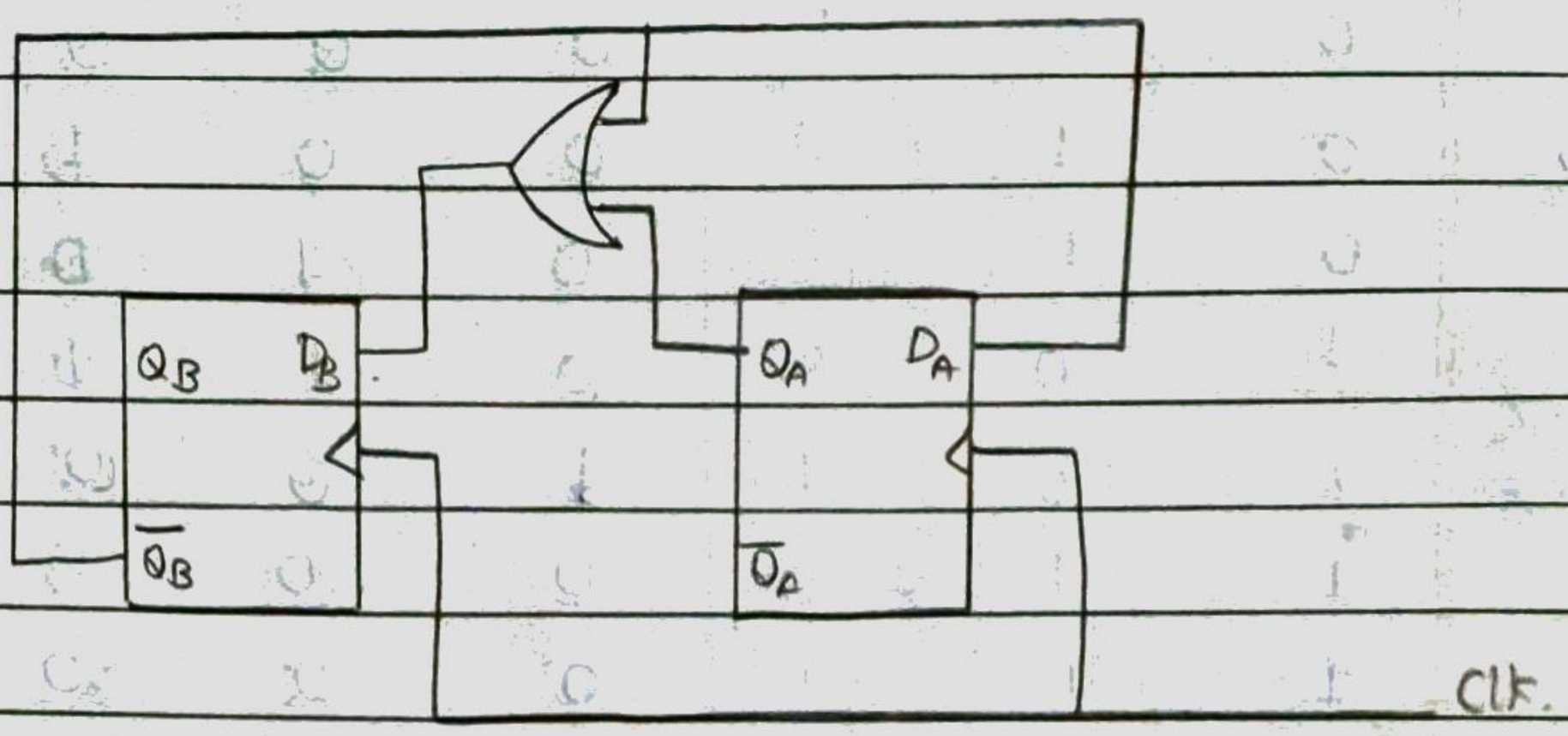
K-map-

Q_B \ Q_A	$\overline{Q_A}$	Q_A
$\overline{Q_B}$	1	1
Q_B	0	1

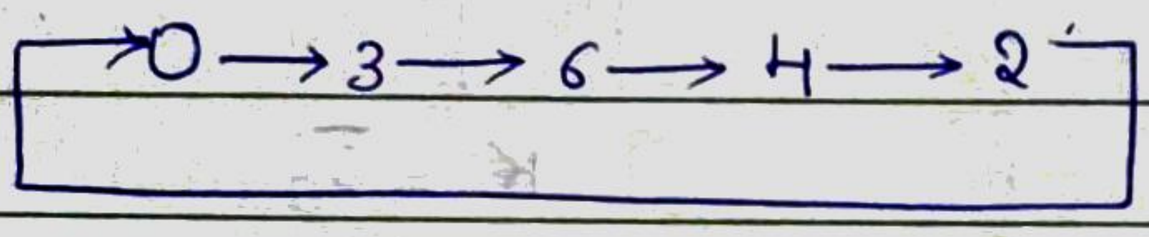
$$D_B = \overline{Q_B} + Q_A$$

Q_B \ Q_A	$\overline{Q_A}$	Q_A
$\overline{Q_B}$	1	1
Q_B	0	0

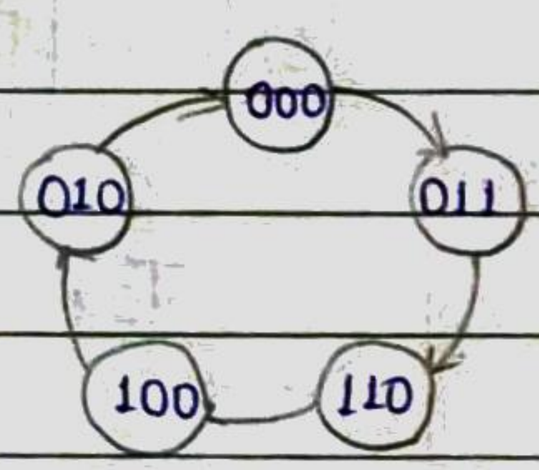
$$D_A = \overline{Q_B}$$



⇒ Design irregular counter using JK Flip Flop for the following sequence.



or



i) JK - 110 - 3FF

ii) JK Flip-Flop.

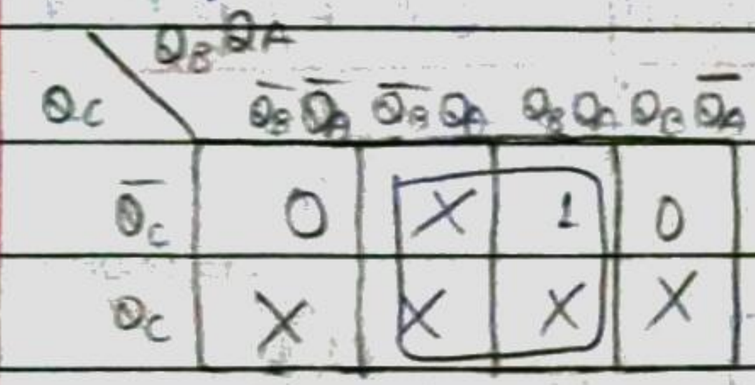
iii) Excitation table for JK Flip Flop.

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

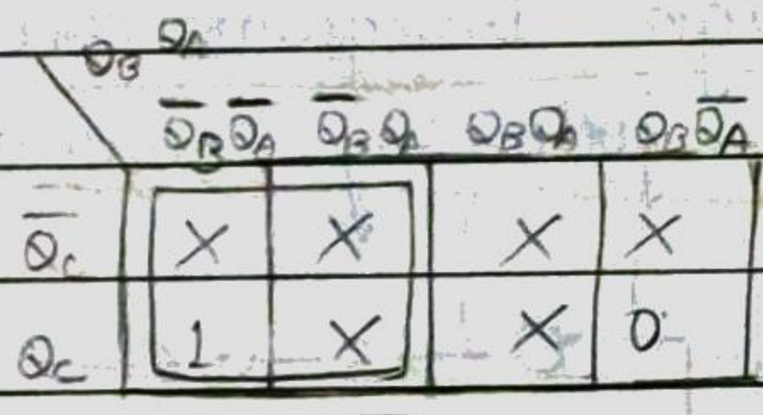
Next state table

Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	1	1	0	x	1	x	1	x
0	0	1	x	x	x	x	x	x	x	x	x
0	1	0	0	0	0	0	x	x	1	0	x
0	1	1	1	1	0	1	x	x	0	x	1
1	0	0	0	1	0	x	1	1	x	0	x
1	0	1	x	x	x	x	x	x	x	x	x
1	1	0	1	0	0	x	0	x	1	0	x
1	1	1	x	x	x	x	x	x	x	x	x

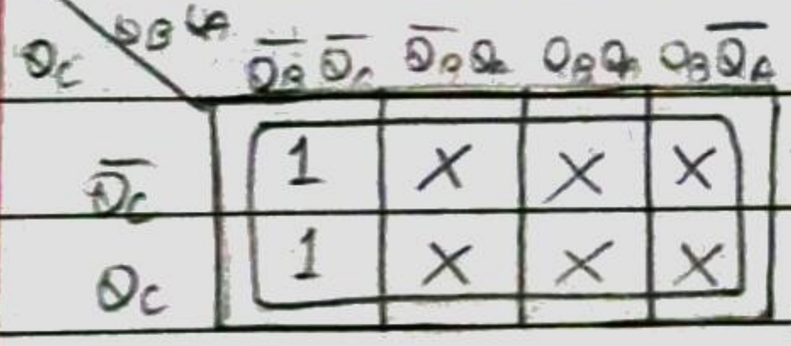
K-Map



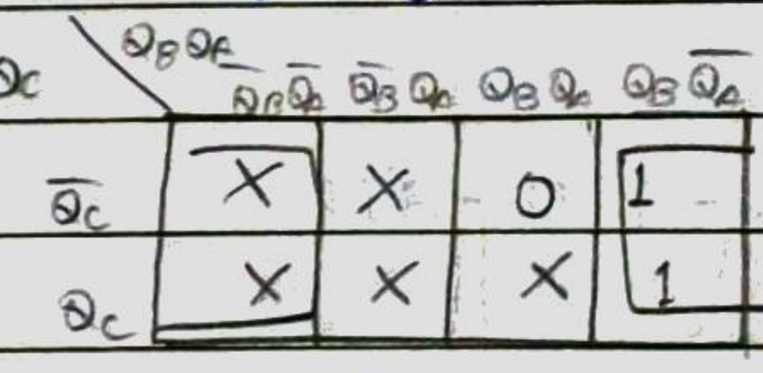
$J_C = Q_A$



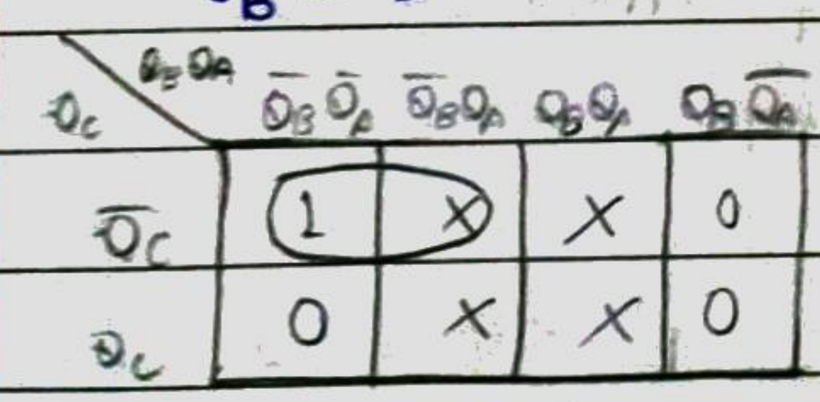
$K_C = \bar{Q}_B$



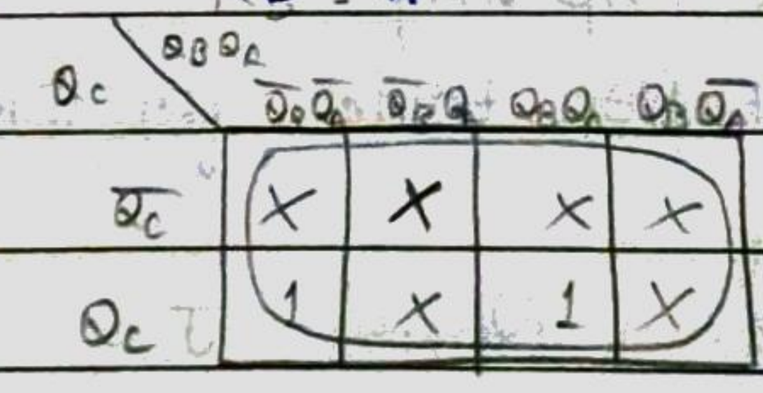
$J_B = 1$



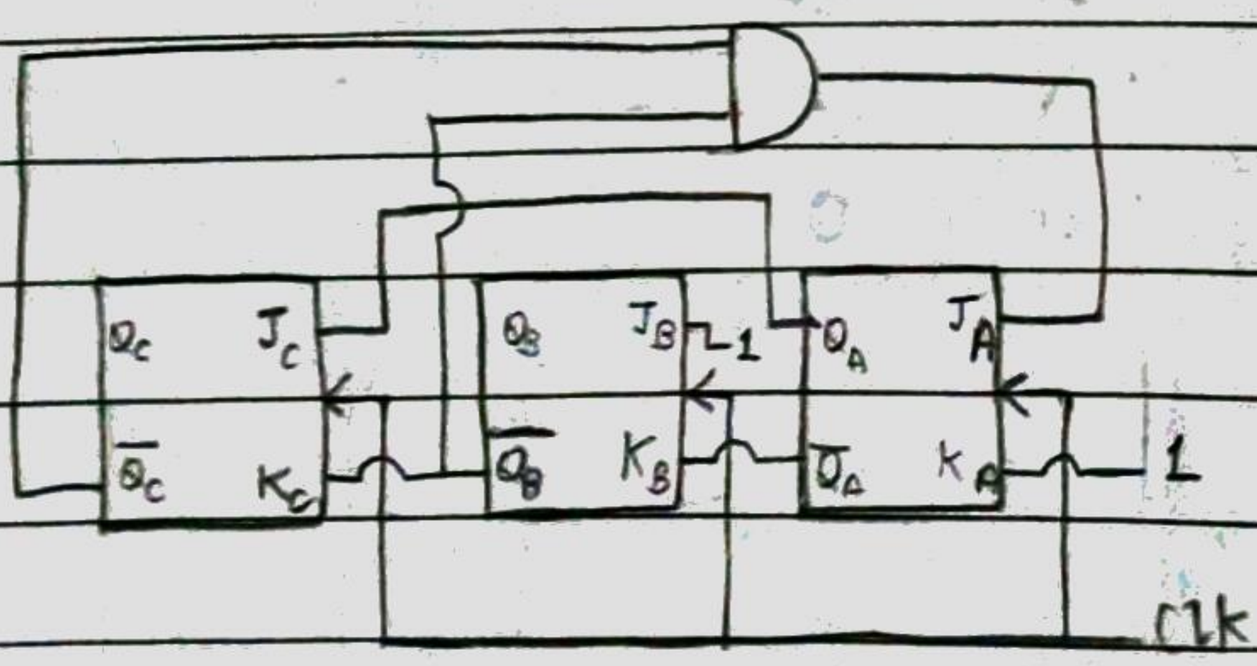
$K_B = \bar{Q}_A$



$J_A = \bar{Q}_C \bar{Q}_B$



$K_A = 1$



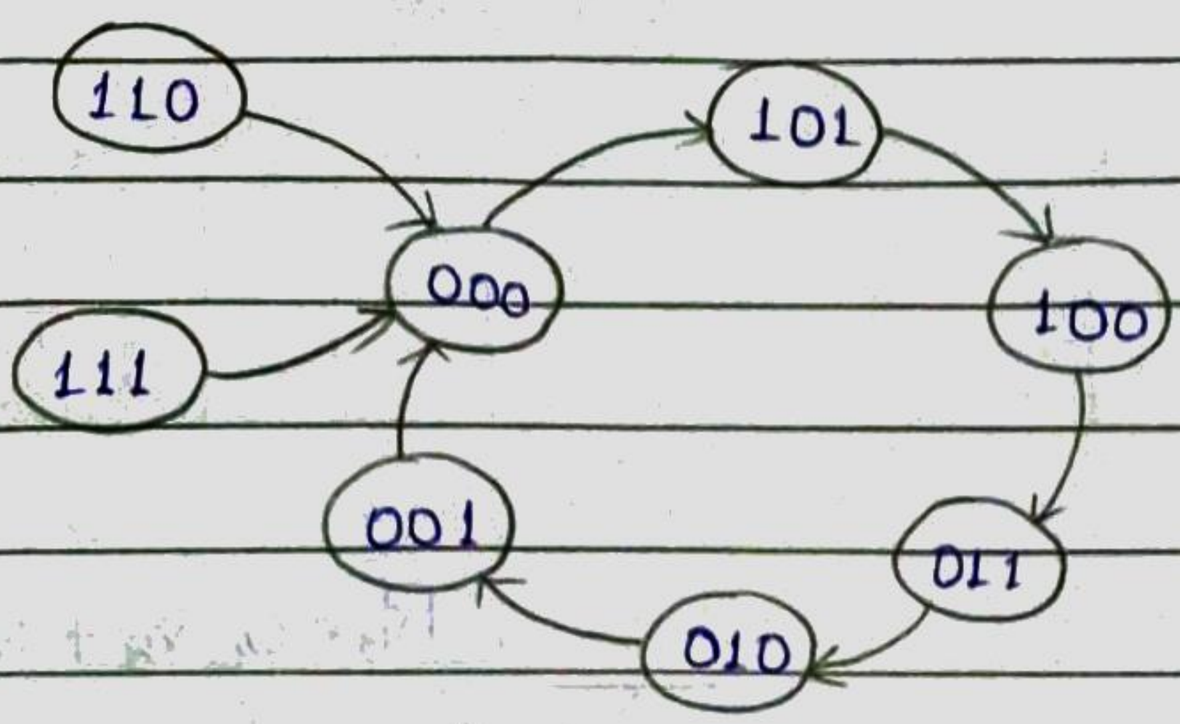
Date
Page

⇒ Design Mod 6 downcounter using SR flipflop where all unused state must lead to 000

5 → 0

101 → 000

3 SR FlipFlops.



Excitation table for SR Flip Flop

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Next state table.

Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	1	1	0	0	X	1	0
0	0	1	0	0	0	0	X	0	X	0	1
0	1	0	0	0	1	0	X	0	1	1	0
0	1	1	0	1	0	0	X	X	0	0	1
1	0	0	0	1	1	0	1	1	0	1	0
1	0	1	1	0	0	X	0	0	X	0	1
1	1	0	0	0	0	0	1	0	1	0	X
1	1	1	0	0	0	0	1	0	1	0	1

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	1	0	0	0
Q_C	0	X	0	0

$$S_C = \bar{Q}_A \bar{Q}_B \bar{Q}_C$$

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	0	X	X	X
Q_C	1	0	1	1

$$R_C = Q_B + Q_C \bar{Q}_A$$

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	0	0	X	0
Q_C	1	0	0	0

$$S_B = Q_C \bar{Q}_B \bar{Q}_A$$

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	X	X	0	1
Q_C	0	X	1	1

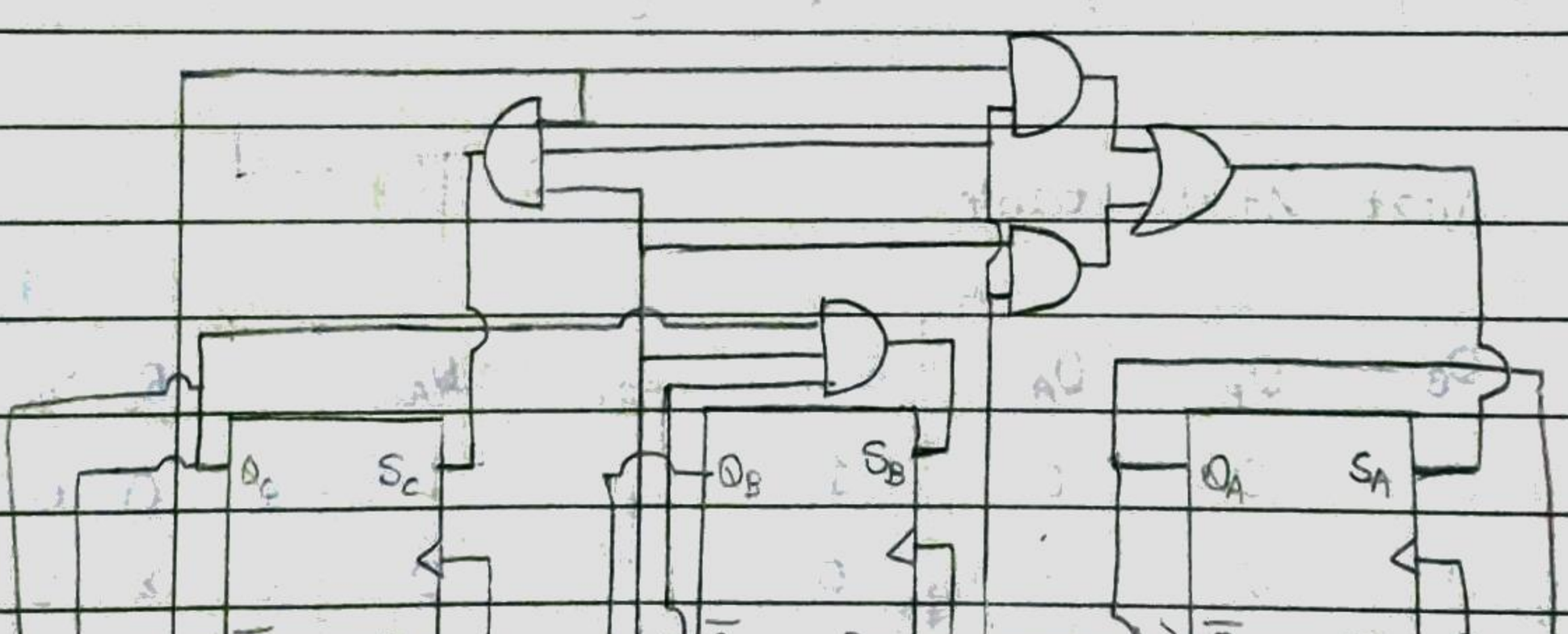
$$R_B = Q_C Q_B + Q_B \bar{Q}_A$$

Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	1	0	0	1
Q_C	1	0	0	0

$$S_A = \bar{Q}_B \bar{Q}_A + \bar{Q}_C \bar{Q}_A$$

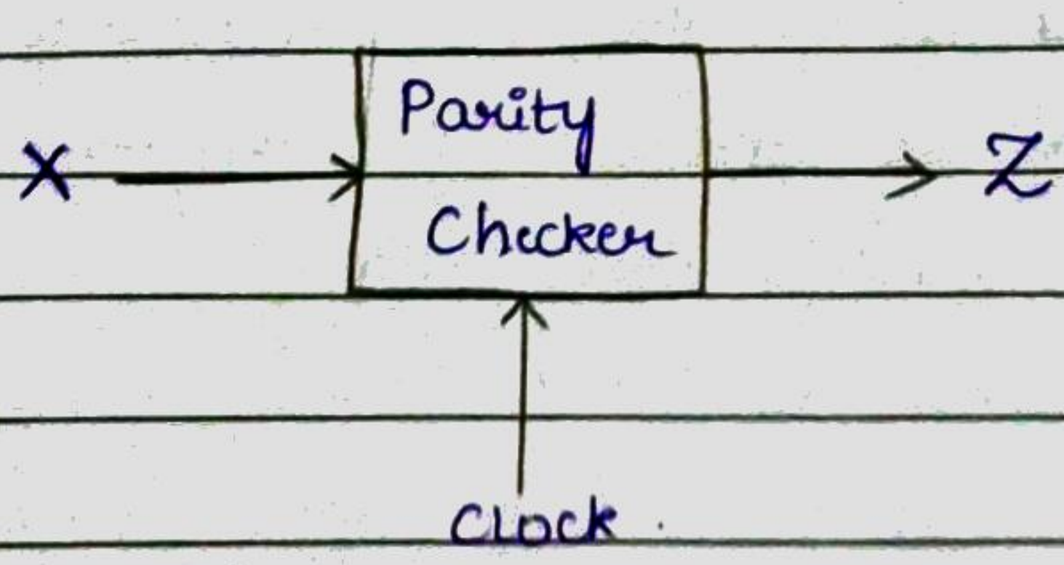
Q_C \ Q_BQ_A	$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_BQ_A$	Q_BQ_A	$Q_B\bar{Q}_A$
$\bar{Q}_C$	0	1	1	0
Q_C	0	1	1	X

$$R_A = Q_A.$$

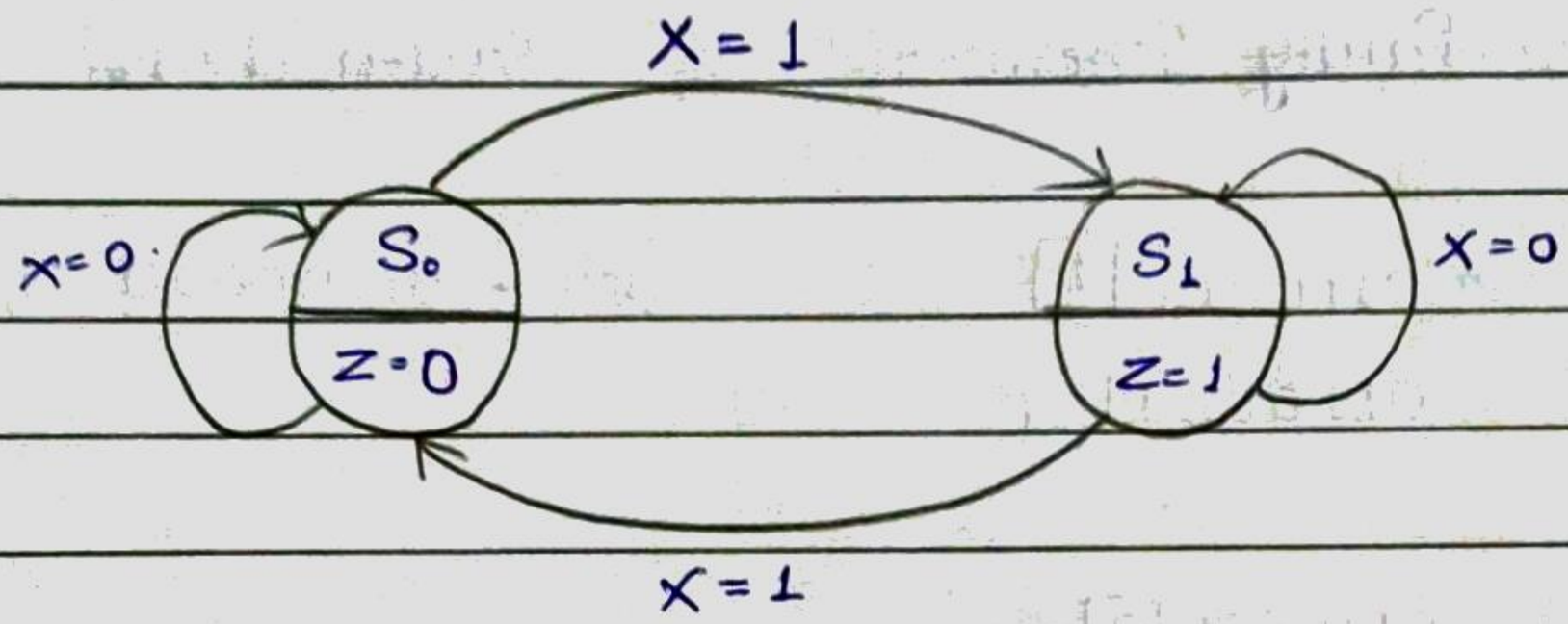


⇒ **A sequential parity checker**

i) Block diagram for parity checker.



ii) State Graph for parity checker.



iii) State & Transition Table.

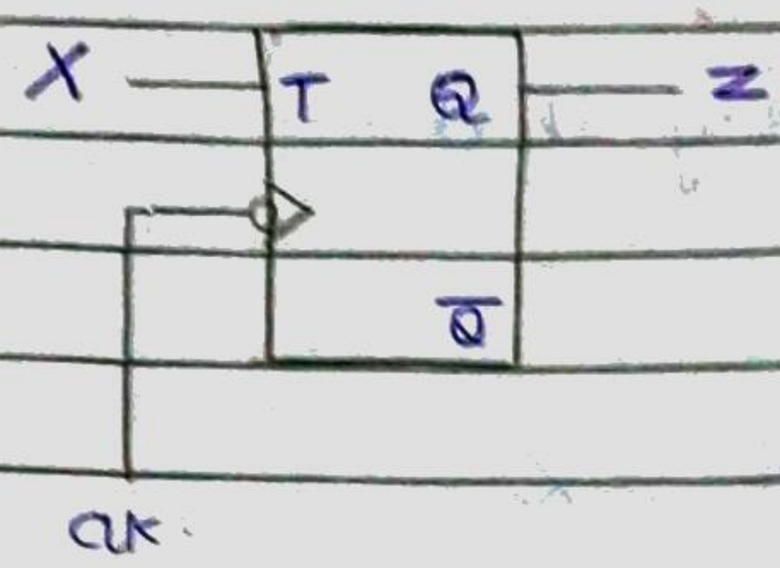
a) State table

Present state	Next state		Present
	X = 0	X = 1	O/P
S_0	S_0	S_1	0
S_1	S_1	S_0	1

b) Transition table using T flip flop

Q	Q^+		T	
	X = 0	X = 1	X = 0	X = 1
0	0	1	0	1
1	1	0	0	1

iv Sequential parity checker using T FlipFlop



Parity - No of 1's

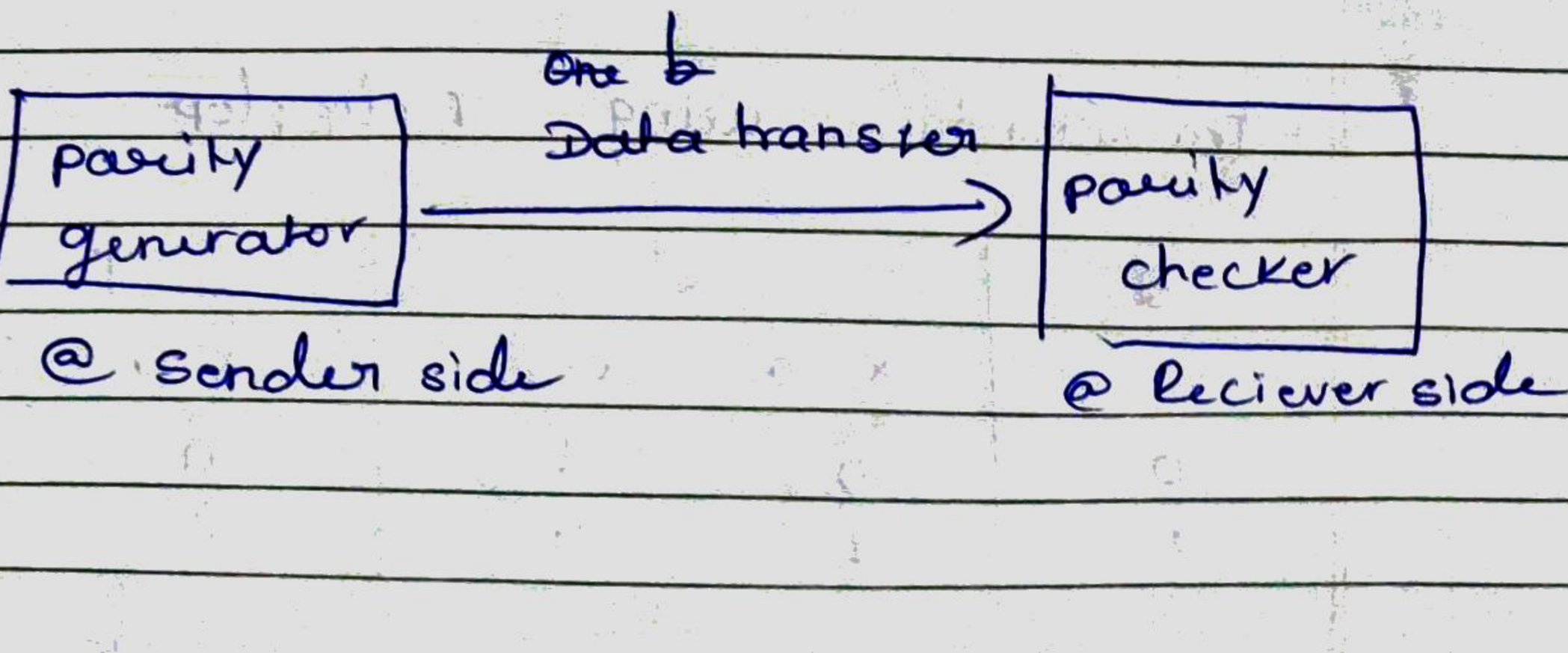
→ Odd parity → 0110001 → odd no^r of 1's

→ Even parity → 0111100 → Even no^r of 1's

	Parity Generator	Parity checker
ODD	0110000 [1]	OPP → 01100000
	0110001 [0]	
EVEN	0110000 [0]	
	0110001 [1]	

Application of parity checker and generator.

⇒ one bit error detection in data transmission



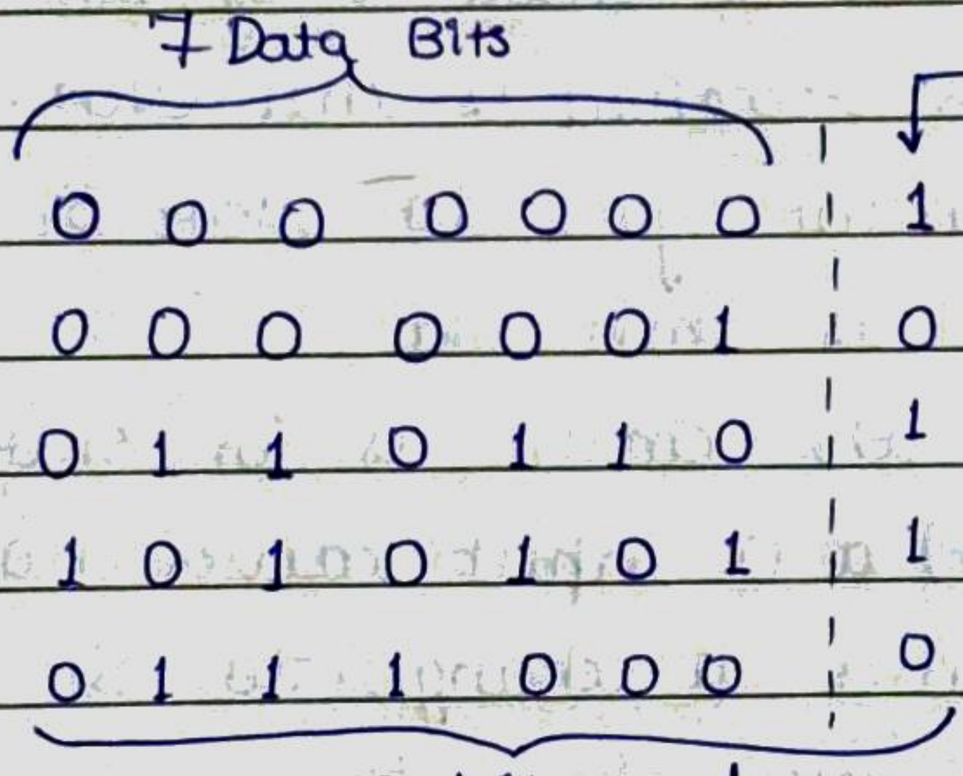
⇒ A Sequential Parity Checker

→ When binary data is transmitted or stored, an extra bit (called a parity bit) is frequently added for purposes of error detection.

→ For example, if data is being transmitted in groups of 7 bits, an eighth bit can be added to each group of 7 bits to make the total number of 1's in each block of 8 bits an odd number.

→ When the total number of 1 bits in the block (including the parity bit) is odd, we say that the parity is odd.

→ Alternately, the parity bit could be chosen such that the total number of 1's in the block is even, in which case we would have even parity.



→ If any single bit in the 8-bit word is changed from 0 to 1 or from 1 to 0, the parity is no longer odd.

→ Thus, if any single bit error occurs in transmission of a word with odd parity, the presence of this error can be detected because the number of 1 bits in the word has been changed from odd to even.

Date
Page

⇒ State Graph for Parity checker [Explaination].

→ We will start the design by constructing a state graph.

→ The sequential circuit must "remember" whether the total number of 1 inputs received is even or odd, therefore, only two states are required.

→ We will designate these states as S_0 and S_1, corresponding respectively to an even number of 1's received and an odd number of 1's received.

→ We will start the circuit in state S_0 because initially zero 1's have been received, and zero is an even number.

→ As indicated in the figure, if the circuit is in state S_0 (even number of 1's received) and $X=0$ is received, the circuit must stay in S_0 because the number of 1's received is still even. However, if $X=1$ is received, the circuit goes to state S_1 because the number of 1's received is then odd.

→ Similarly, if the circuit is in state S_1 (odd number of 1's received) a 0 input causes no state change, but a 1 causes a change to S_0 because the number of 1's received is then even.

→ The output Z should be 1 whenever the circuit is in state S_1 (odd number of 1's received). The output is listed below the state on the state graph.

⟹ State Tables and Graphs

→ **Rules**

The following method can be used to construct the transition table:

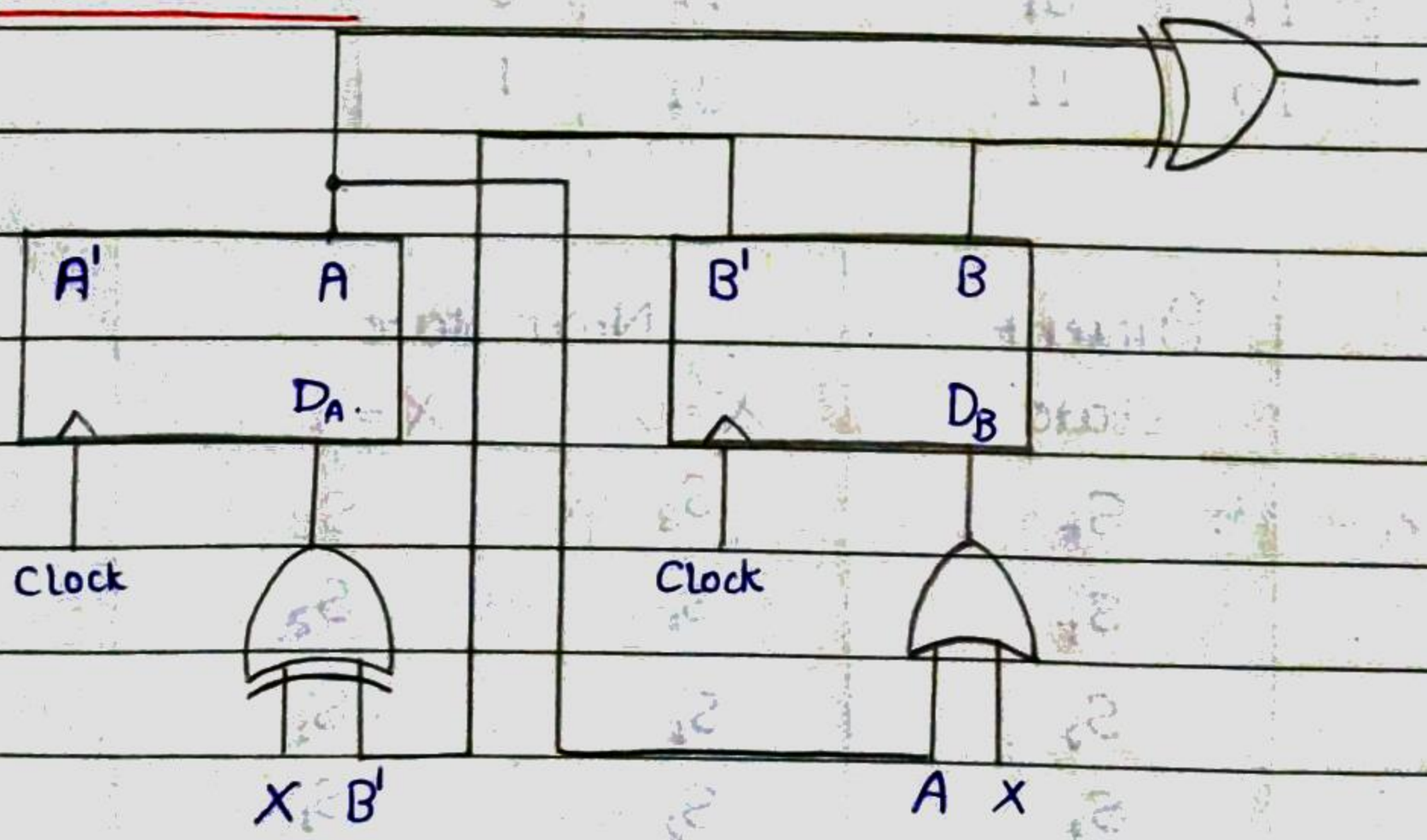
1. Determine the flip-flop input equations and the output equations from the circuit.
2. Derive the next-state equation for each flip-flop from its input equations, using one of the following relations:

D flip-flop	$Q^+ = D$
D-CE flip-flop	$Q^+ = D \cdot CE + Q \cdot CE'$
T flip-flop	$Q^+ = T \oplus Q$.
S-R flip-flop	$Q^+ = S + R'Q$.
J-K flip-flop	$Q^+ = JQ' + K'Q$.

3. Plot a next-state map for each flip-flop.
4. Combine these maps to form the transition table. Such a transition table, which gives the next state of the flip-flops as a function of their present state and the circuit inputs.

Example

1. **Moore State Machine**



1) The flip-flop input equations and output equation are

$DA = X \oplus B'$ $\quad DB = X + A$ $\quad Z = A \oplus B$.

2) The next-state equations for the flip-flops are

$A+ = X \oplus B'$ $\quad B+ = X + A$.

3) The corresponding maps are.

AB \ X	0	1
00	1	0
01	0	1
11	0	1
10	1	0

A^+

AB \ X	0	1
00	0	1
01	0	1
11	1	1
10	1	1

B^+.

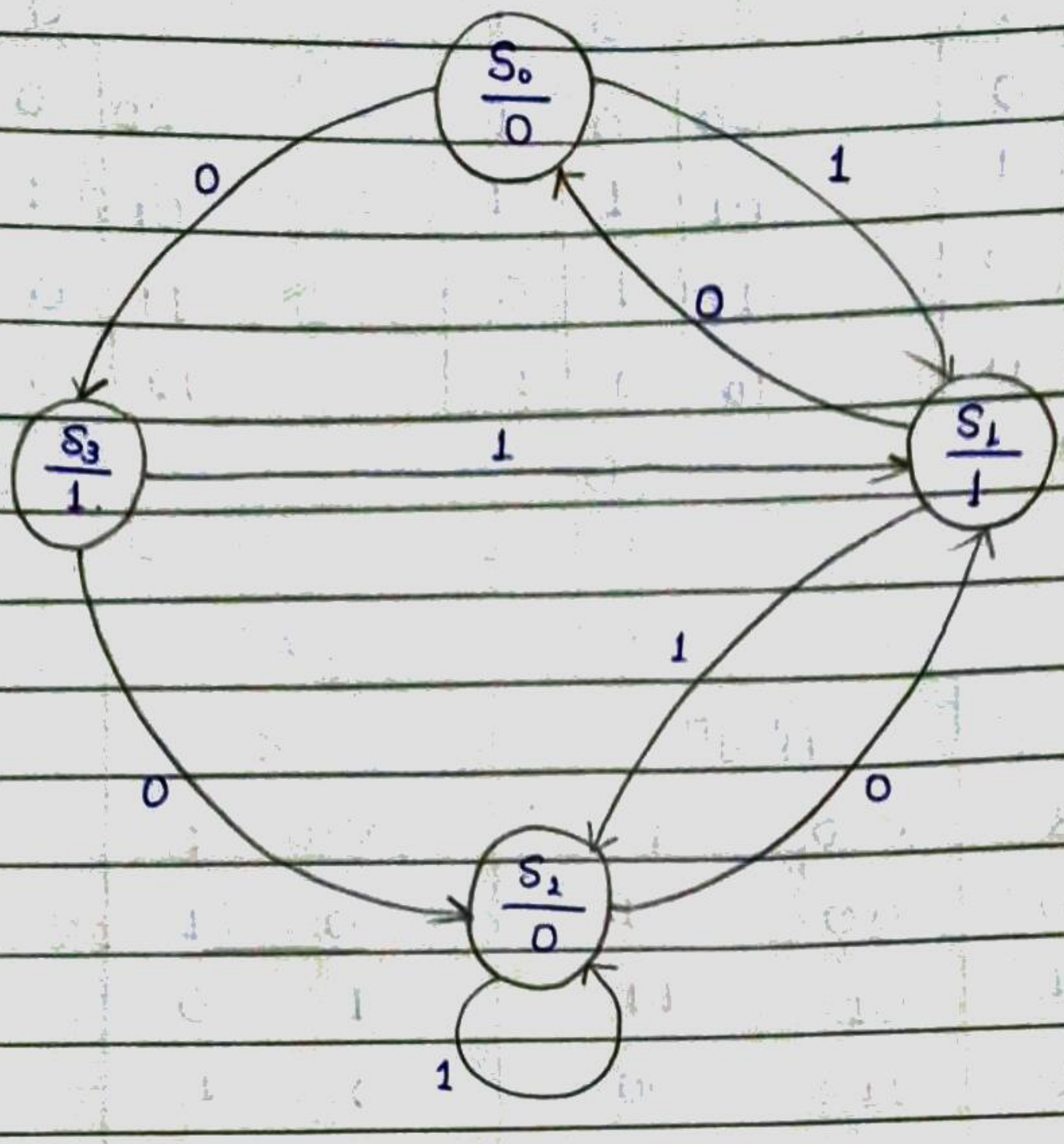
(a)

	A^+B^+		
AB	X = 0	X = 1	Z
00	10	01	0
01	00	11	1
11	01	11	0
10	11	01	1

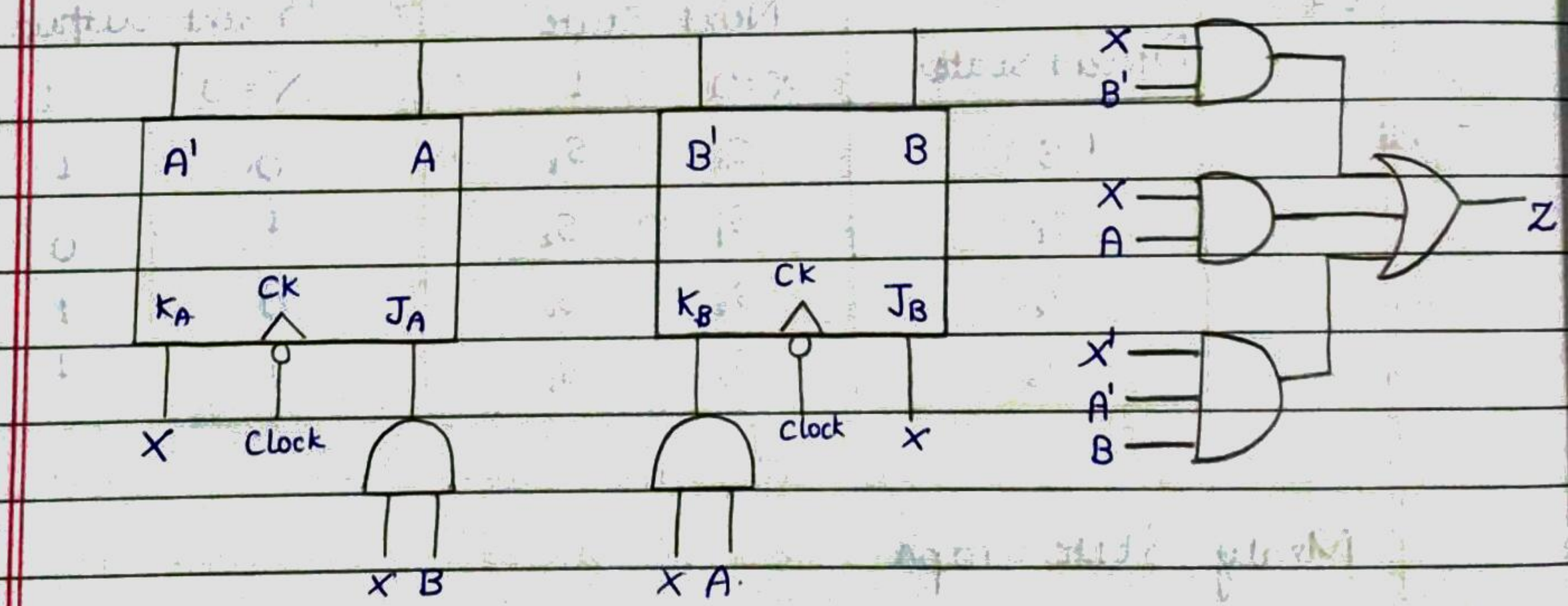
(b)

Present State	Next State X = 0	Next State X = 1	Present Output (z)
S_0	S_3	S_1	0
S_1	S_0	S_2	1
S_2	S_1	S_2	0
S_3	S_2	S_1	1

Moore State Graph.



2. Mealy State Machine.



$A^+ = J_A A' + K_A' A = XBA' + X'A.$

$B^+ = J_B B' + K_B' B = XB' + (AX)' B = XB' + X'B + A'B.$

$Z = X'A'B + XB' + XA.$

AB \ X	0	1
00	0	0
01	0	1
11	1	0
10	1	0

A^+

AB \ X	0	1
00	0	1
01	1	1
11	1	0
10	0	1

B^+

AB \ X	0	1
00	0	1
01	1	0
11	0	1
10	0	1

Z

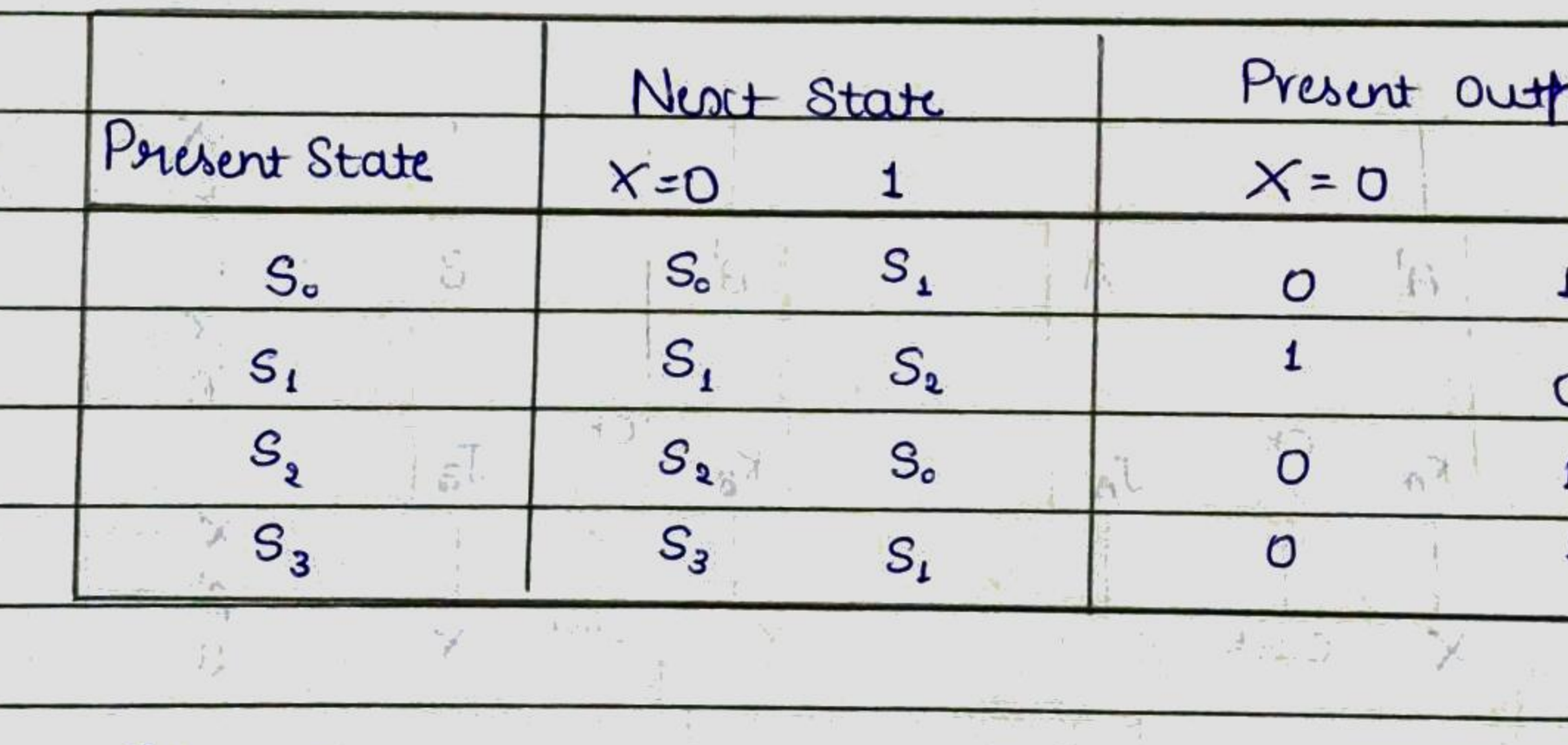
(a)

	A^+B^+		Z	
AB	X=0	1	X=0	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)

Present State	Next State		Present output	
	X=0	1	X=0	1
S_0	S_0	S_1	0	1
S_1	S_1	S_2	1	0
S_2	S_2	S_0	0	1
S_3	S_3	S_1	0	1

Mealy State Graph.



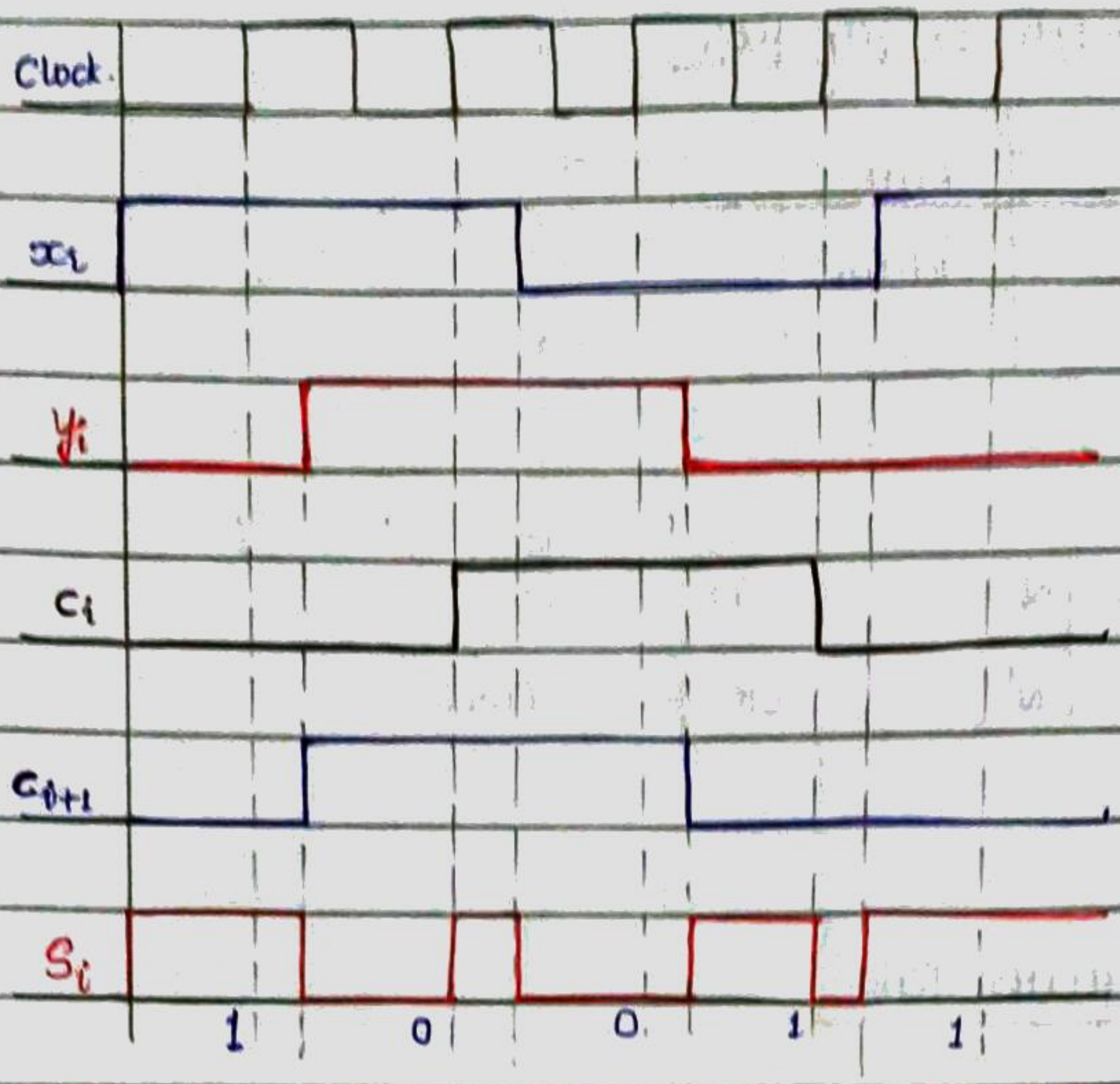
⇒ Serial Adder.

(a) With D flip-flop.

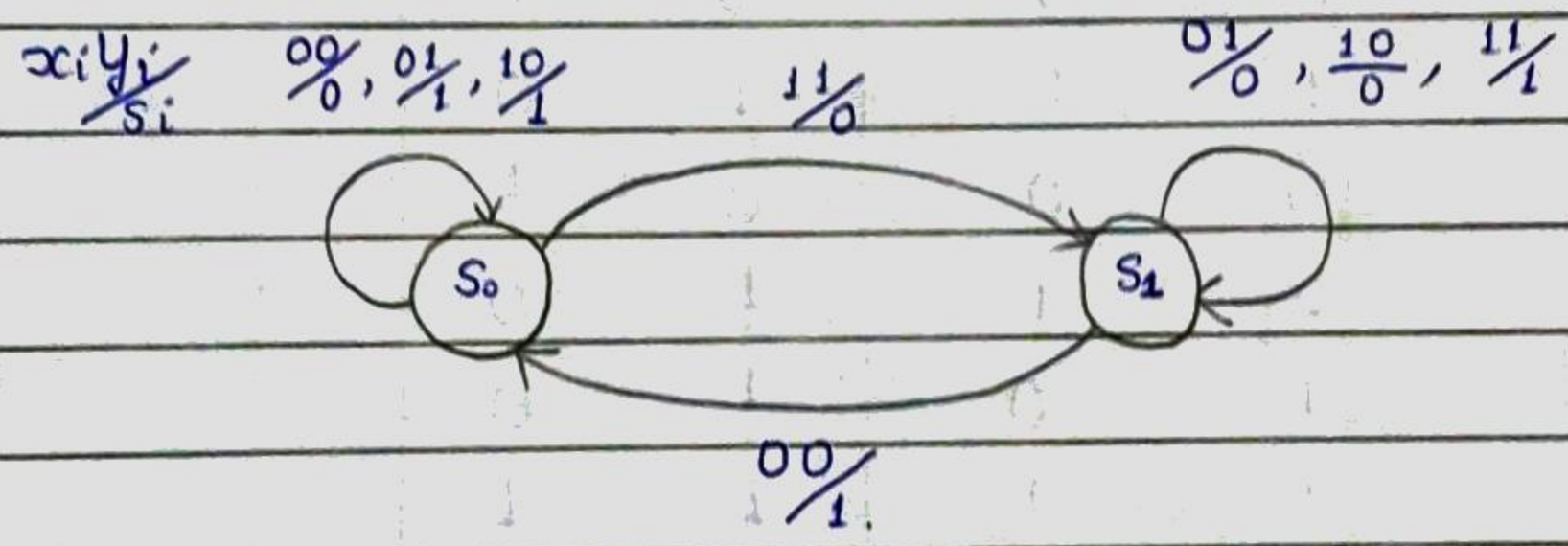
(b) Truth Table.

x_i	y_i	c_i	c_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Timing diagram for Serial Adder.



State Graph for Serial Adder



→ We can construct a state graph for the serial adder. The serial adder is a Mealy machine with inputs x_i and y_i and output S_i.

→ The two states represent a carry (c_i) of 0 and 1, respectively.

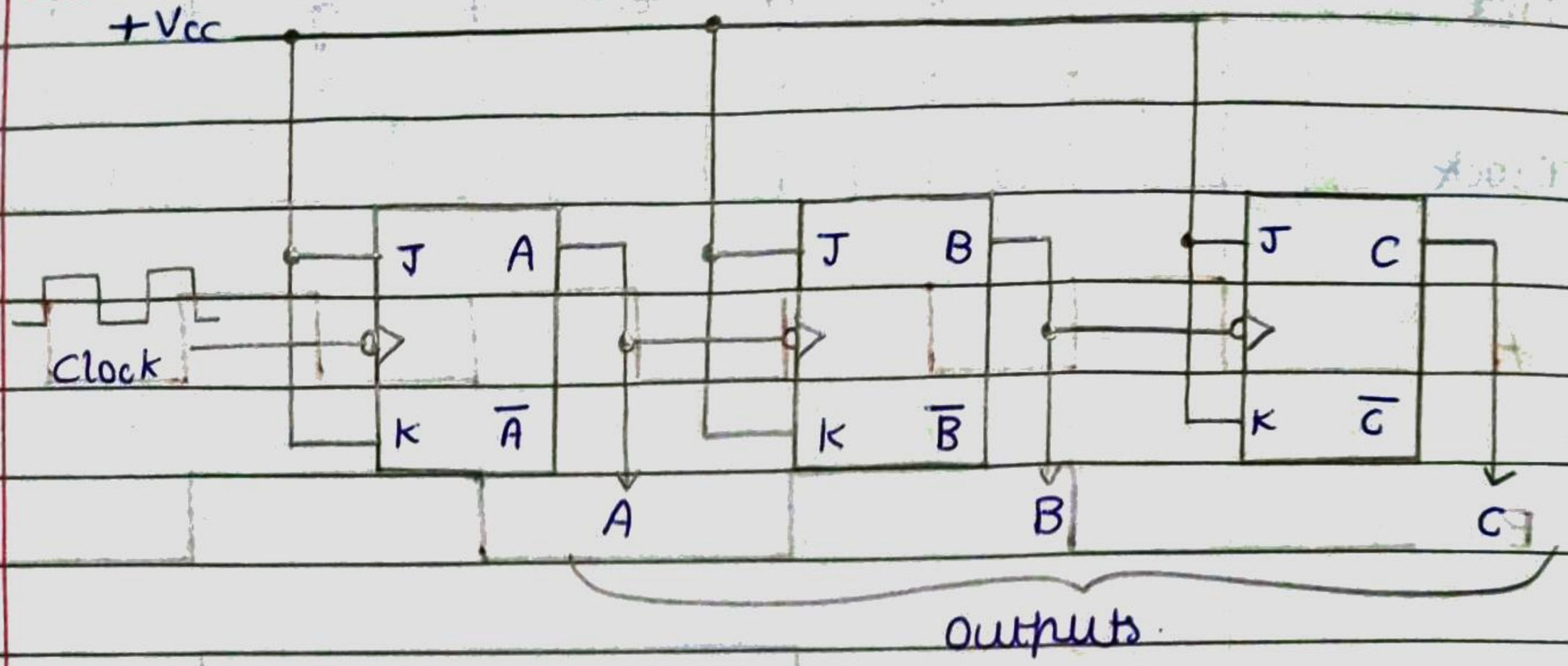
→ From the table, c_i is the present state of the sequential circuit, and c_{i+1} is the next state.

→ If we start in S_0 (no carry), and $x_i y_i = 11$, the output is $S_i = 0$ and the next state is S_1. This is indicated by the arrow going from state S_0 to S_1.

Date
Page

⇒ Asynchronous counters.

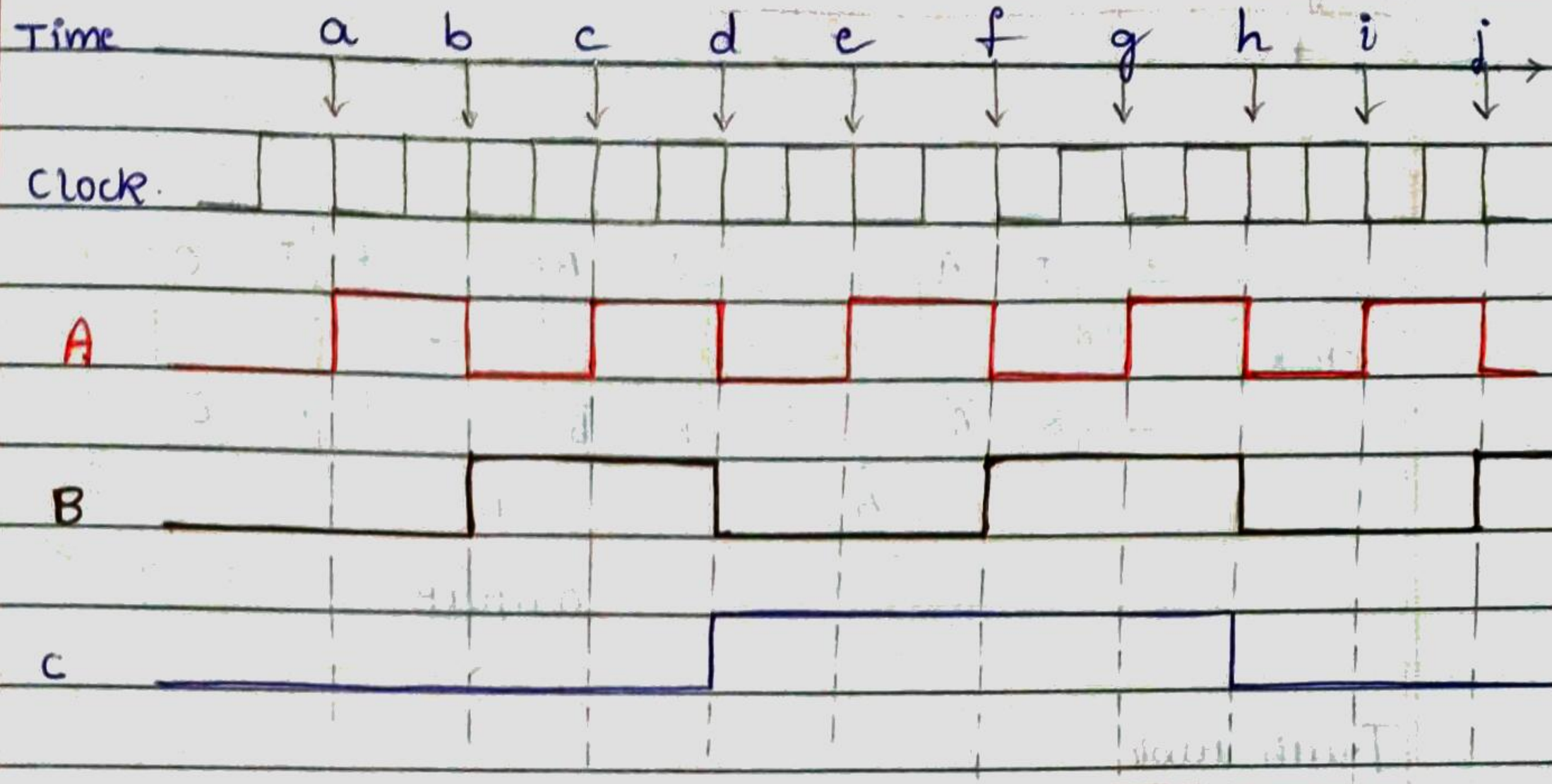
→ Asynchronous UP Counters.



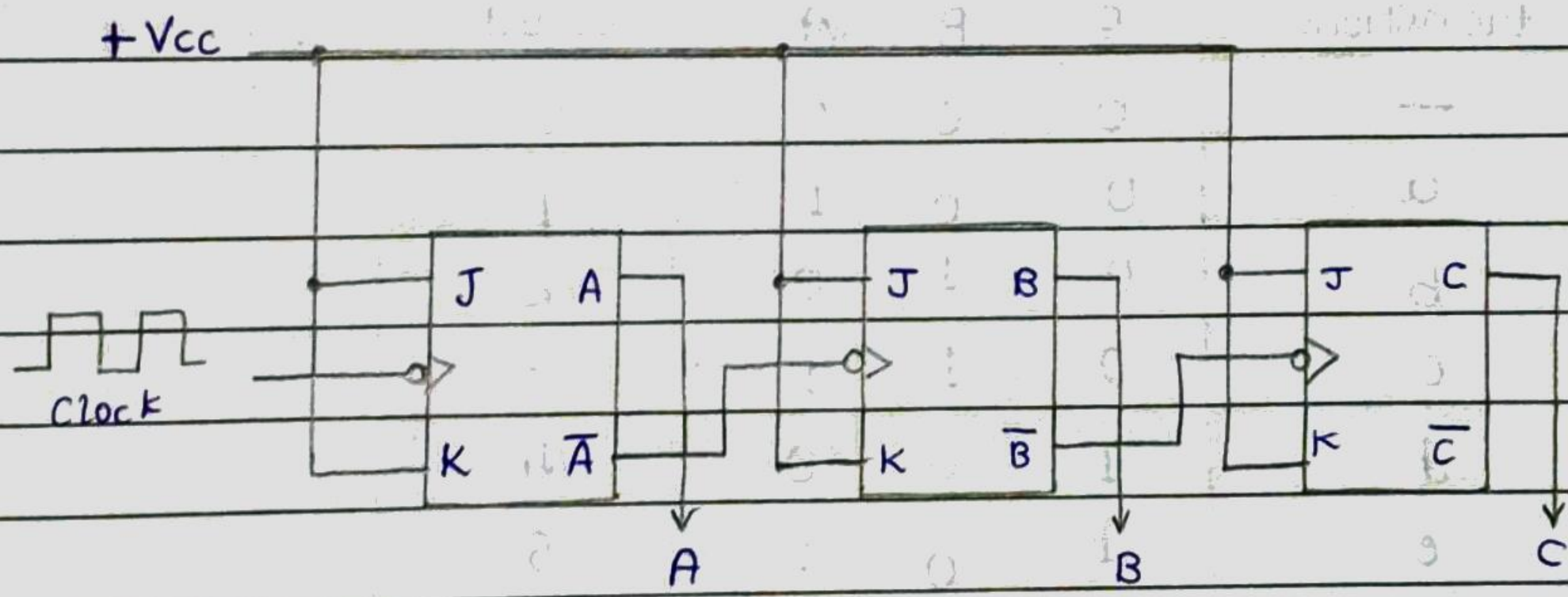
Truth table.

Negative clock transitions	C	B	A	State or count
—	0	0	0	0
a	0	0	1	1
b	0	1	0	2
c	0	1	1	3
d	1	0	0	4
e	1	0	1	5
f	1	1	0	6
g	1	1	1	7
h	0	0	0	0

Waveforms.



→ ## Asynchronous Down counters.



Truth Table

Count	C	B	A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
7	1	1	1

Timing diagram / Waveforms.