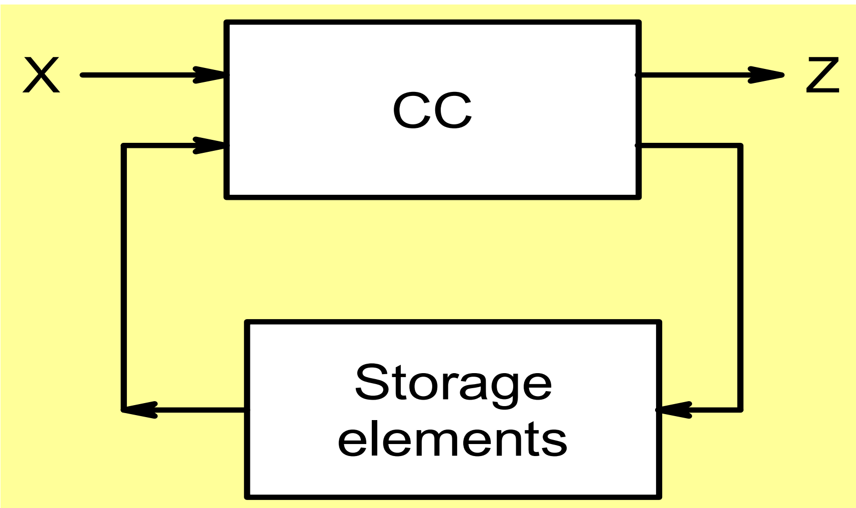
ESc201 : Introduction to Electronics

Sequential Circuit Design -2

Dr. Y. S. Chauhan
Dept. of Electrical Engineering
IIT Kanpur

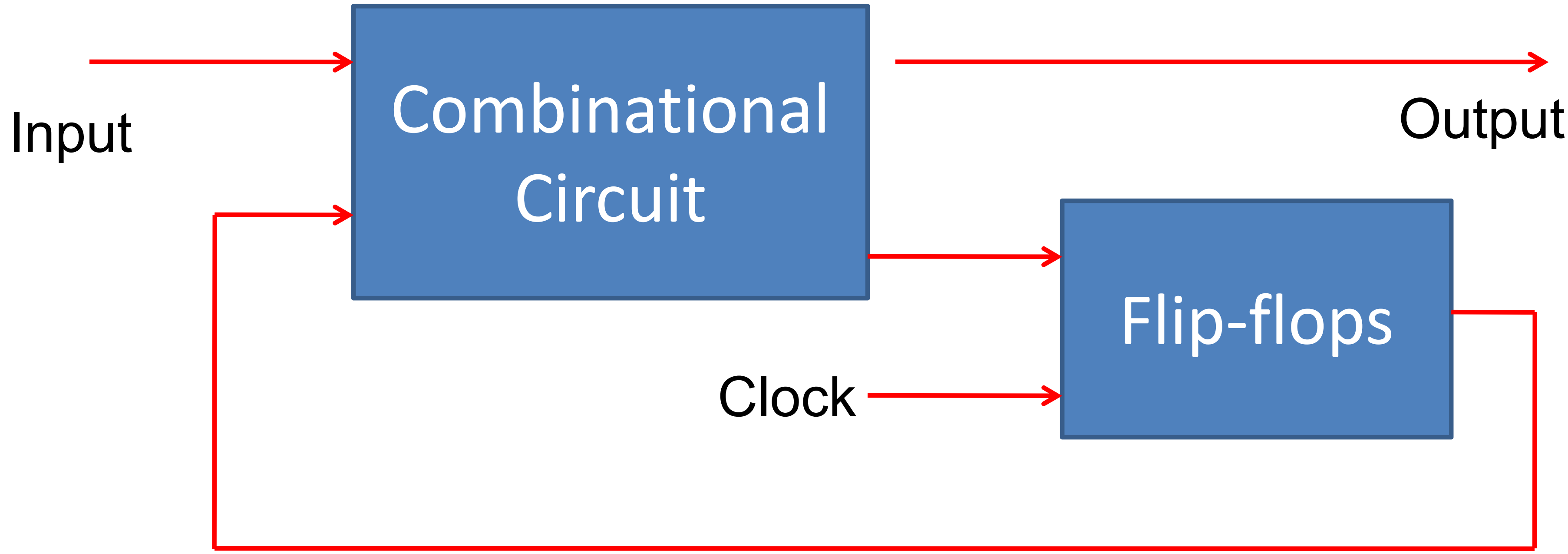
Sequential Circuits

The binary information stored in the storage elements at any given time defines the ***state*** of the sequential circuit at that time



Output is a function of input as well as the present state of the storage elements.

Synchronous Sequential Circuits

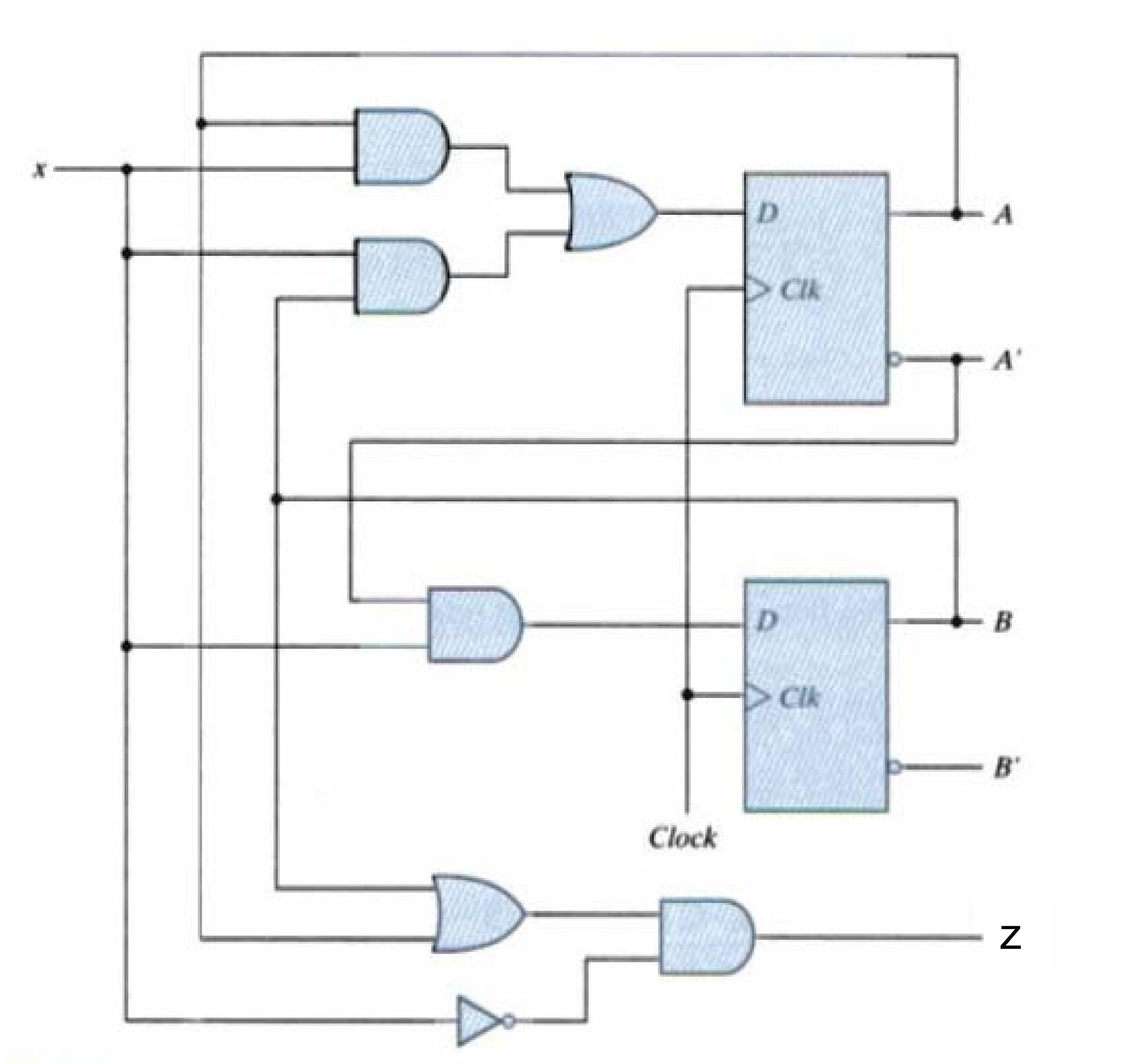


Employs signals that affect the storage elements only at discrete instants of time.

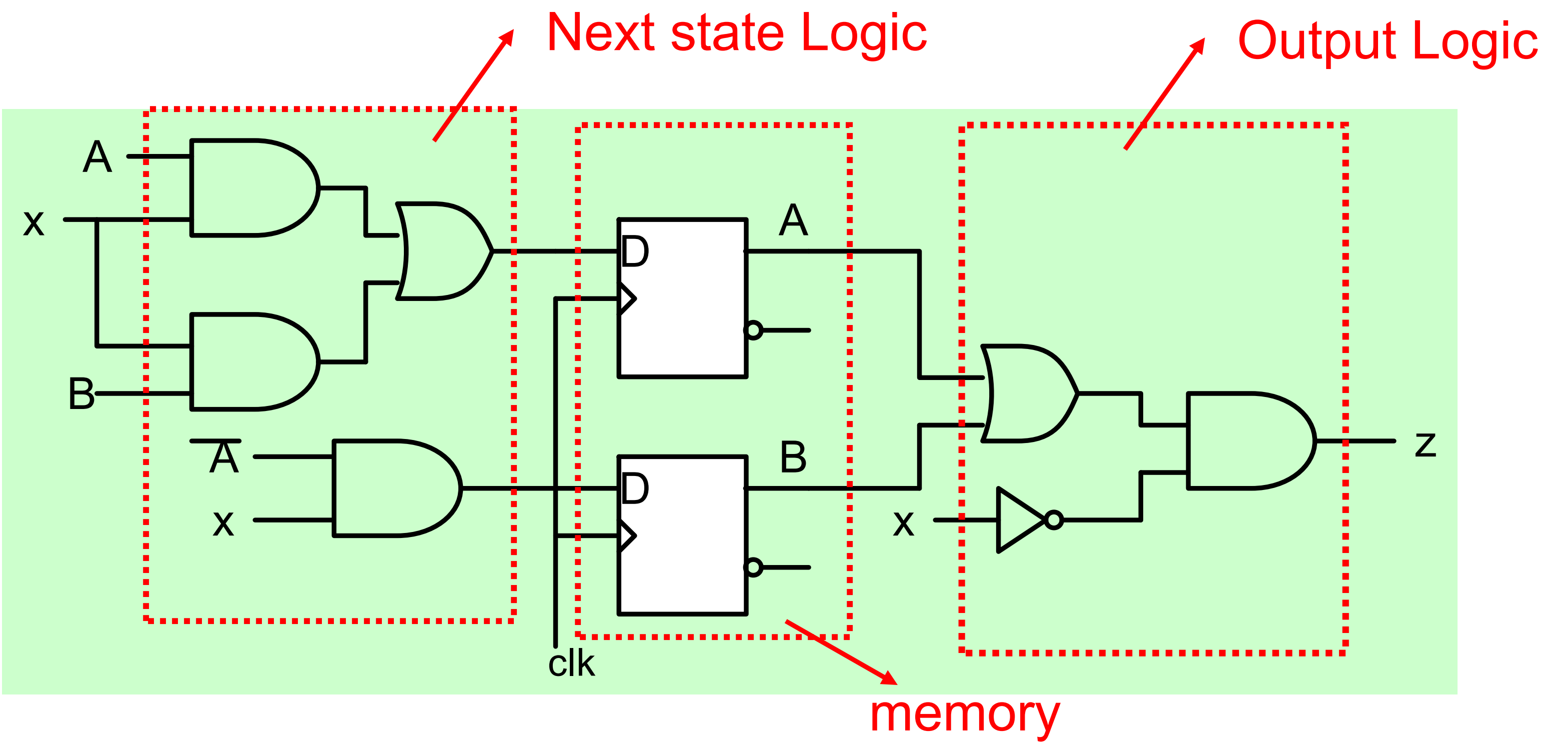
Synchronization is achieved via the ***clock pulses***.

Synchronous Clocked Sequential Circuits

Analysis



Analysis



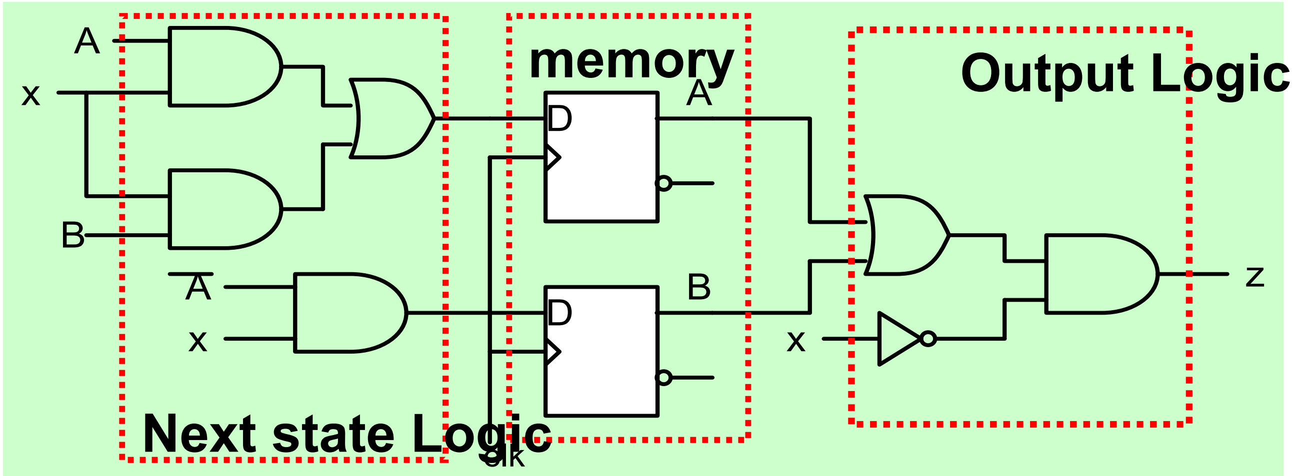
Output z depends on the input x and on the state of the memory (A,B)

The memory has 2 FFs and each FF can be in state 0 or 1. Thus there are four possible states: AB: 00,01,10,11.

To describe the behavior of a sequential circuit, we need to show

1. How the system goes from one memory state to the next as the input changes
2. How the output responds to input in each state

Analysis of Sequential Circuits



$$D_A = A.x + B.x \;\; ; \; D_B = \overline{A}.x \, ; \, z = (A+B).\overline{x}$$

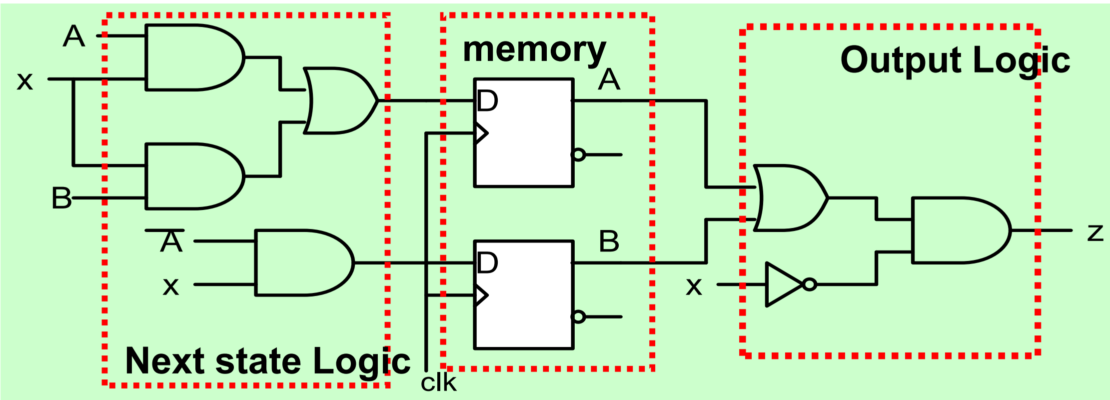
$$A(t+1) = A(t).x + B(t).x$$

$$B(t+1) = \overline{A(t)}.x$$

$$z(t) = (A(t)+B(t)).\overline{x}$$

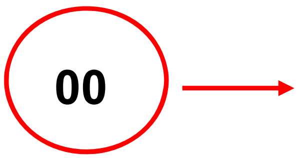
State Transition Table

Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	**0**	**0**	**0**
0	0	1	**0**	**1**	**0**
0	1	0	**0**	**0**	**1**
0	1	1	**1**	**1**	**0**
1	0	0	**0**	**0**	**1**
1	0	1	**1**	**0**	**0**
1	1	0	**0**	**0**	**1**
1	1	1	**1**	**0**	**0**

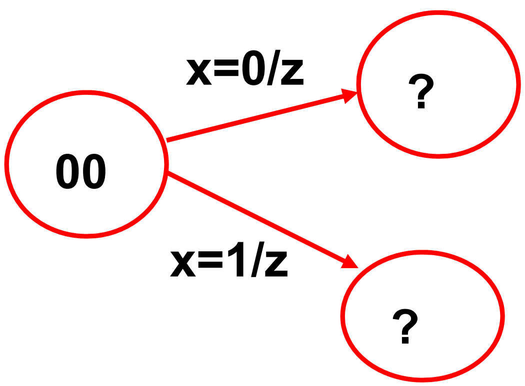


State Transition Table

Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

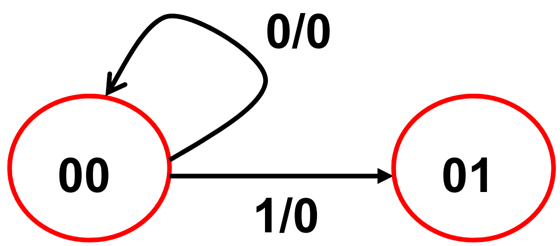


Memory state in which FF A& B have output values 00

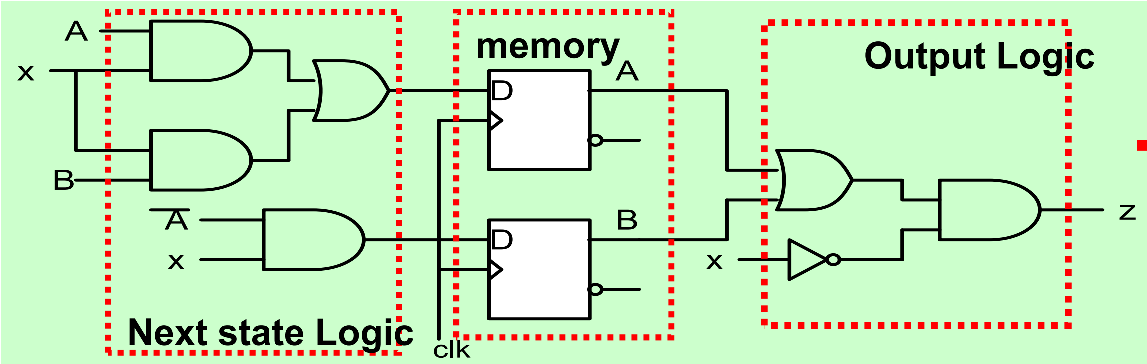


If x = 0 then z = 0, When the clock edge comes the system would stay in 00 state.

If x = 1 then z = 0. When the clock edge comes the system would go to 01 state.



Analysis of Sequential Circuits



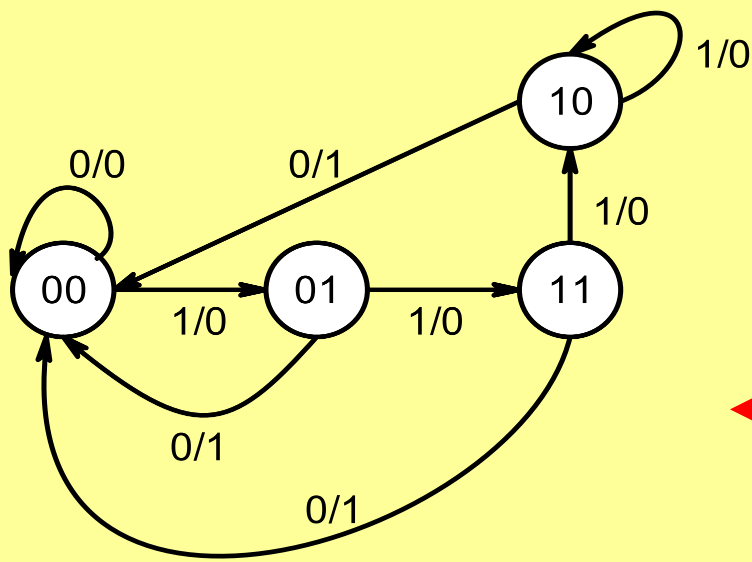
$$A(t+1) = A(t).x + B(t).x$$

$$B(t+1) = \overline{A(t)}.x$$

$$z(t) = (A(t) + B(t)).\overline{x}$$

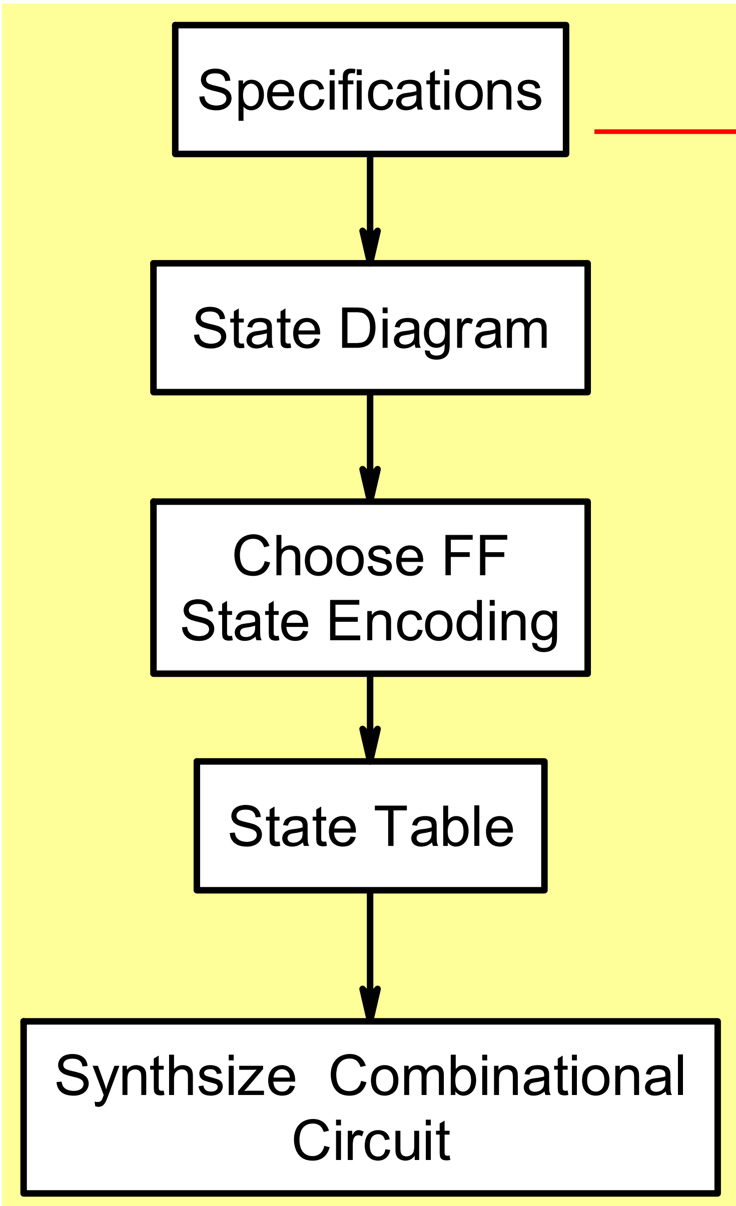
State Transition Table

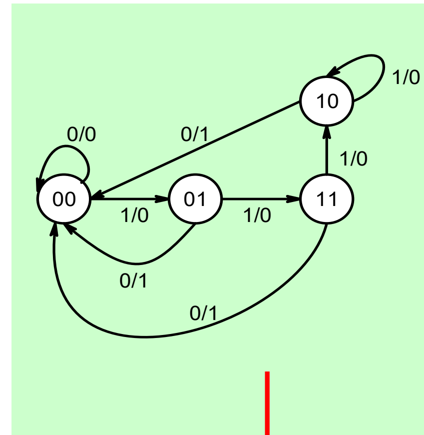
Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



State transition Graph

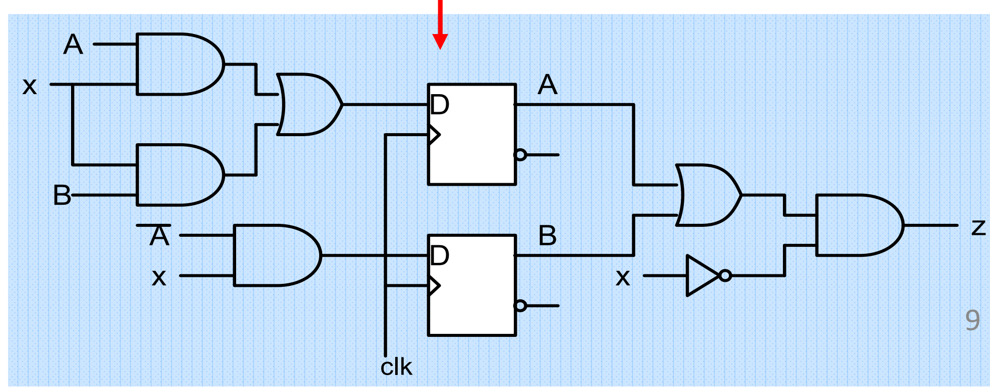
Design of Sequential Circuits





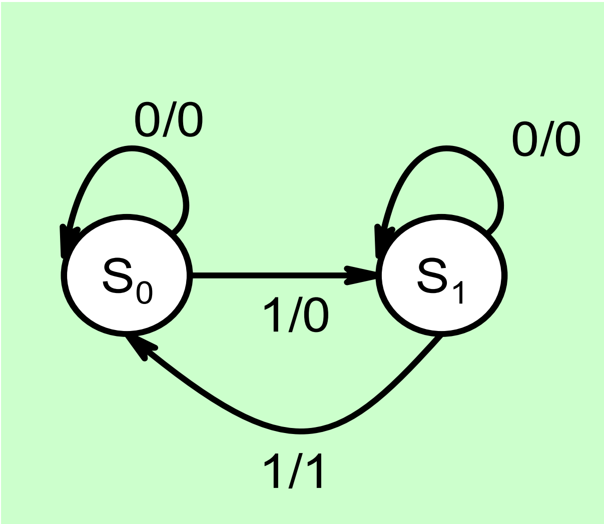
State Transition Table

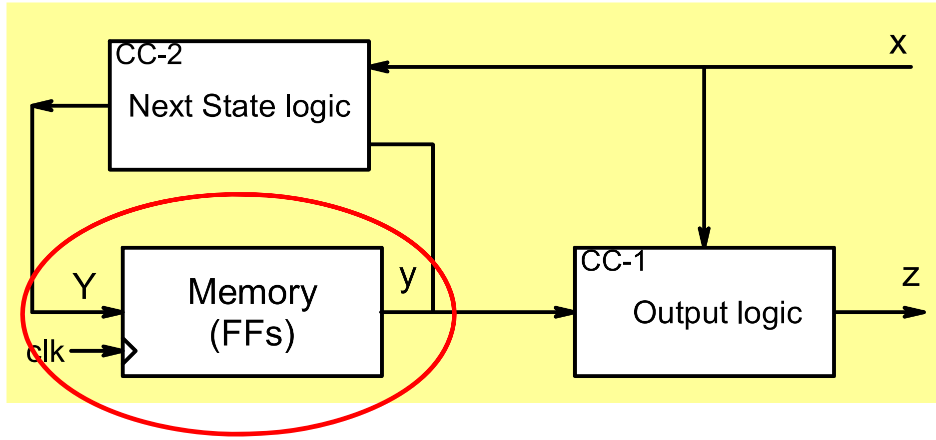
Present State A	Present State B	Input x	Next State A	Next State B	Output z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



Conversion of State transition graph to a circuit

Example-1



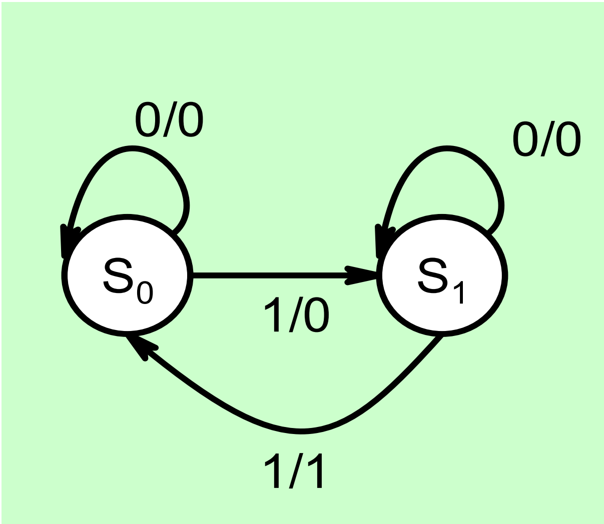


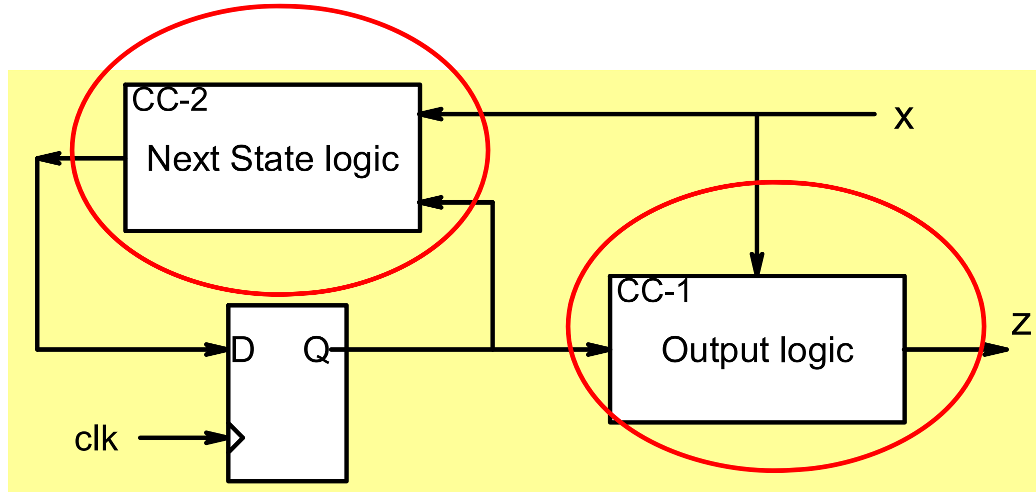
3 blocks need to be designed

1. How many FFs do we need? **N FFS can represent 2^N states so Minimum is 1**

2. Which FF do we choose? **Say D FF**

3. How are the states encoded? **Say FF output Q=0 represents S_0 and Q=1 represents S_1 state**





State Transition Table

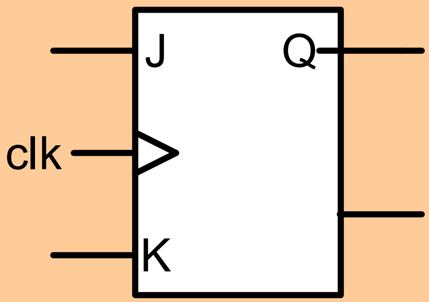
Present State Q(t)	Input x	Next State Q(t+1)	D	Output z
0	0	0		0
0	1	1		0
1	0	1		0
1	1	0		1

Excitation Table

What inputs are required to effect a particular state change

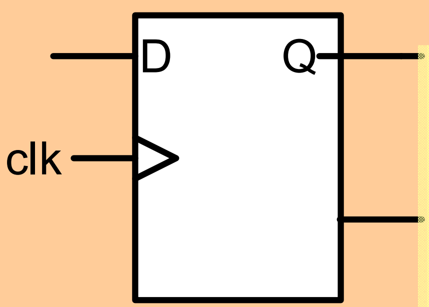
		Inputs
Q(t)	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table



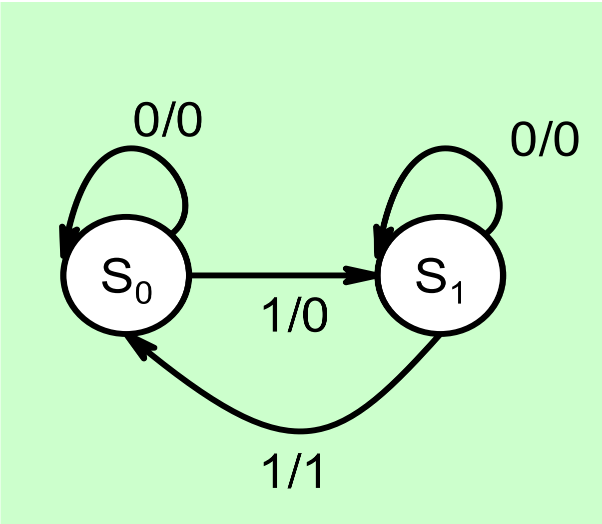
J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

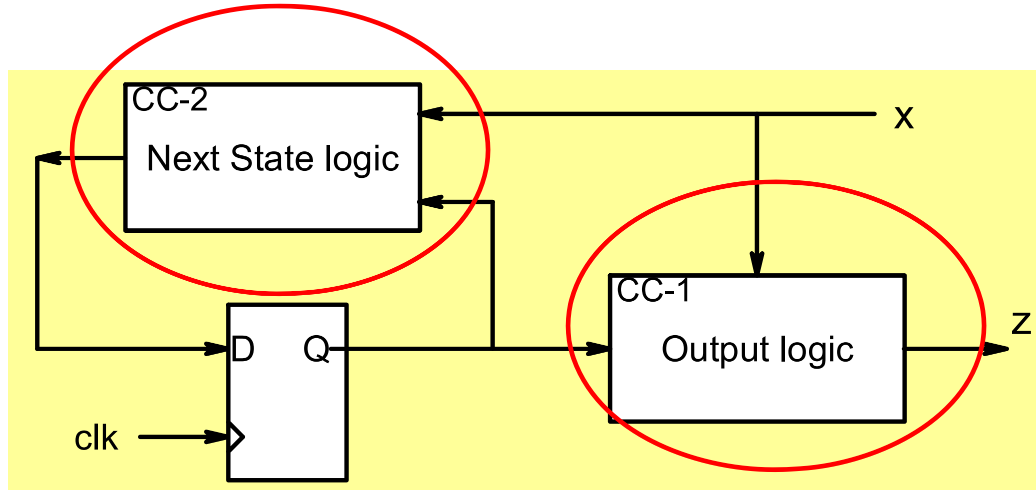
		Inputs
Q(t)	**Q(t+1)**	**J K**
0	0	**0 X**
0	1	**1 X**
1	0	**X 1**
1	1	**X 0**



D	Q(t+1)
0	0
1	1

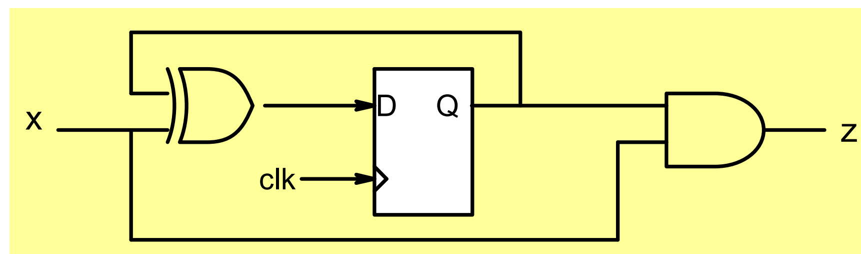
		Inputs
Q(t)	**Q(t+1)**	**D**
0	0	**0**
0	1	**1**
1	0	**0**
1	1	**1**





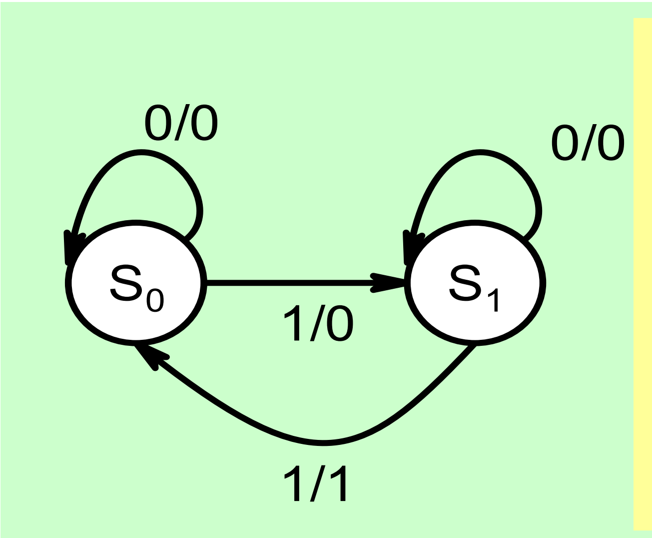
State Transition Table

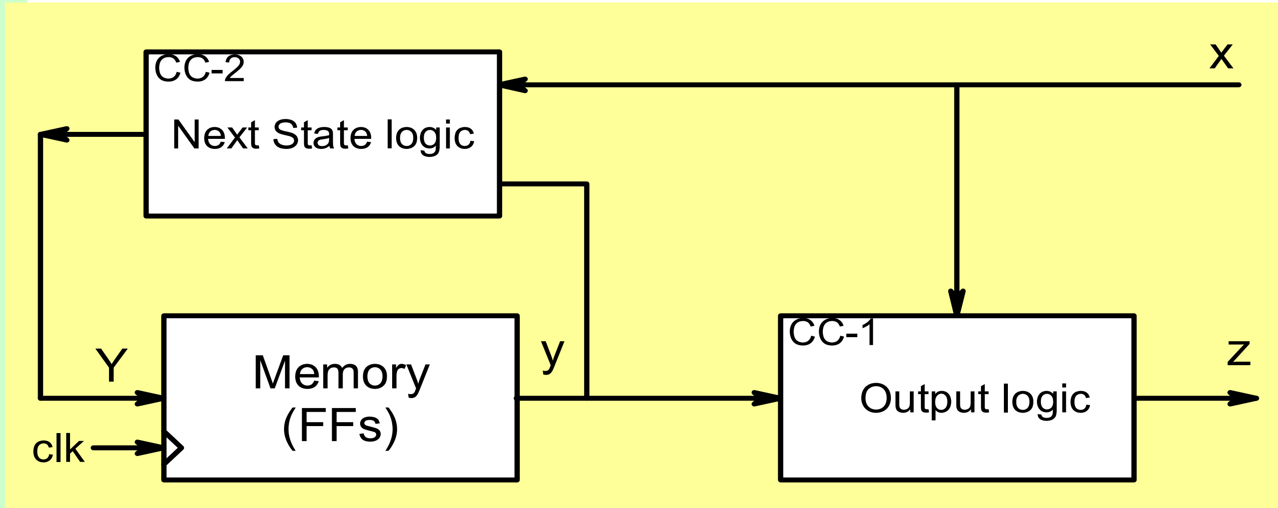
Present State Q(t)	Input x	Next State Q(t+1)	D	Output z
0	0	0	**0**	0
0	1	1	**1**	0
1	0	1	**1**	0
1	1	0	**0**	1



$$D = \overline{Q}.x + Q.\overline{x} \quad ; \quad z = Q.x$$

Example-2

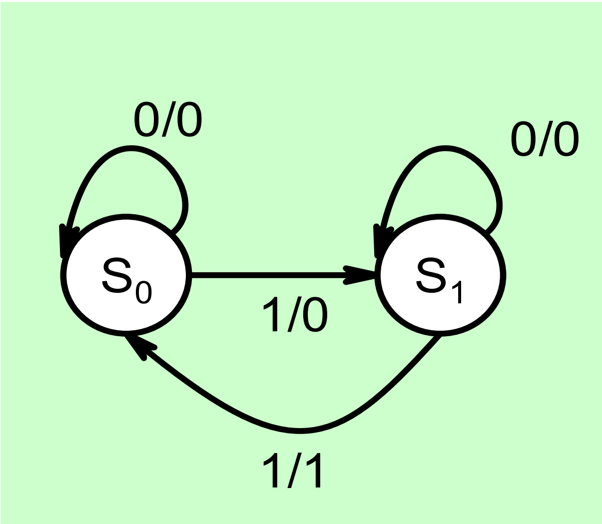


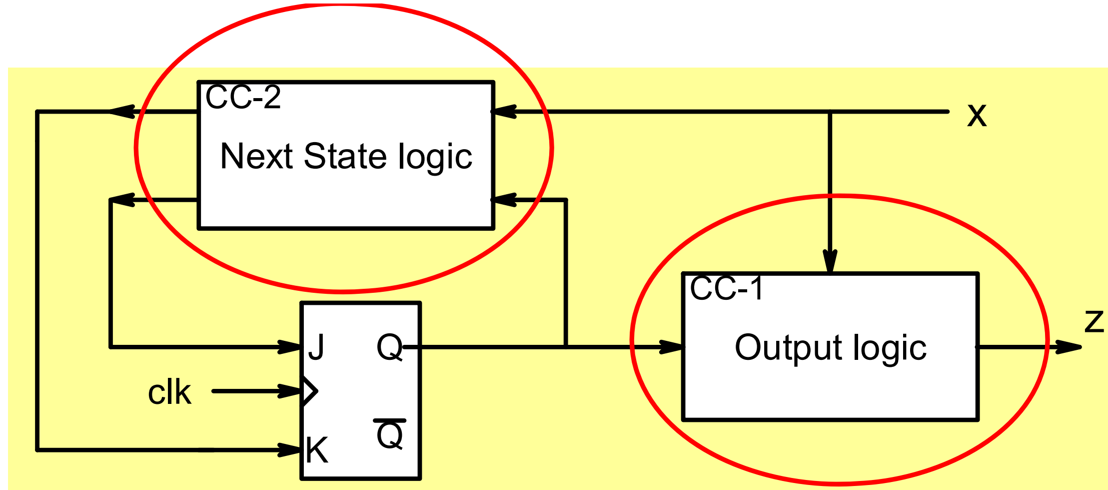


1. How many FFs do we need? **1**

2. Which FF do we choose? **Say JK FF**

3. How are the states encoded? **Say FF output Q=0 represents S_0 and Q=1 represents S_1 state**

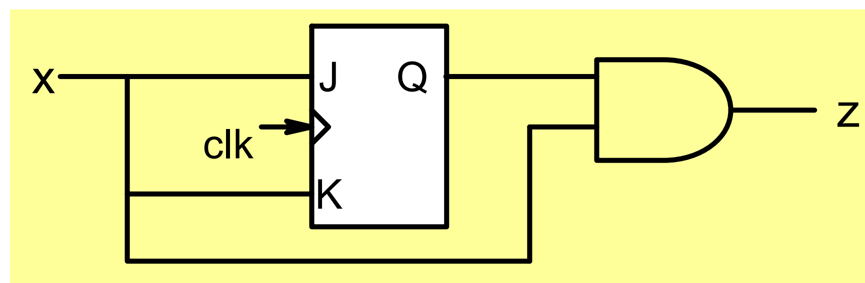




Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

State Transition Table

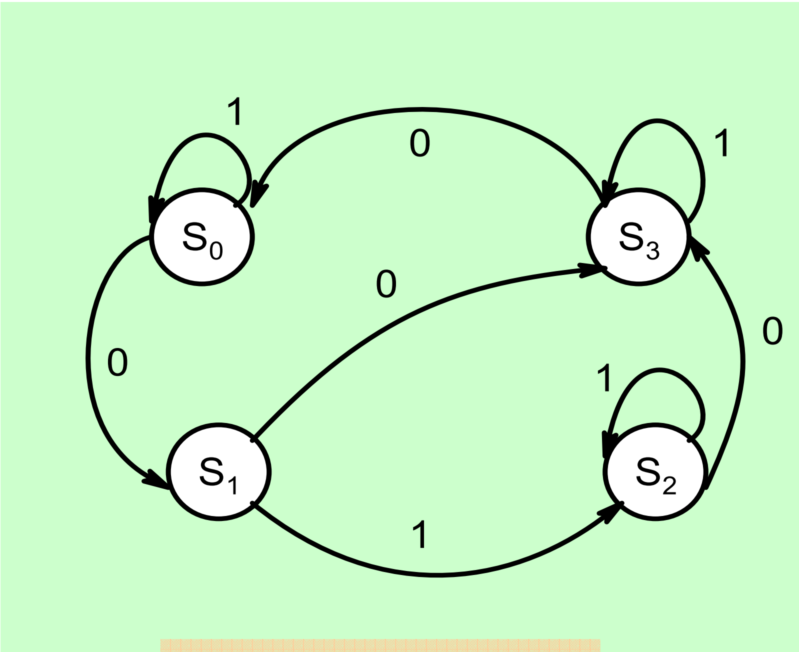
Present State Q(t)	Input x	Next State Q(t+1)	J	K	Output z
0	0	0	**0**	**X**	0
0	1	1	**1**	**X**	0
1	0	1	**X**	**0**	0
1	1	0	**X**	**1**	1

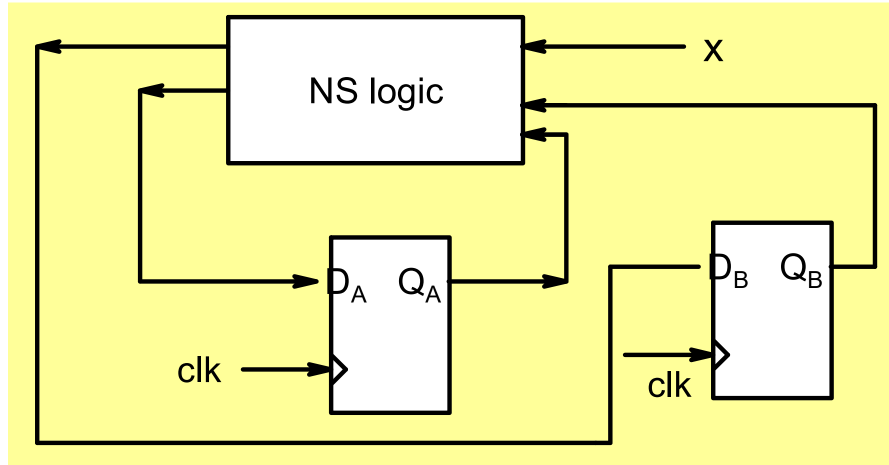


$$J = x \;\; ; K = x ; \;\; z = Q.x$$

Example-3

For 4 states a minimum of two FFs will be required. Let us choose 2 D FFs A &B





State	FF O/P A	B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

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

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

D_A

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

$$D_A = \overline{A}B + xB + A\overline{B}$$
$$= A \oplus B + x.B$$

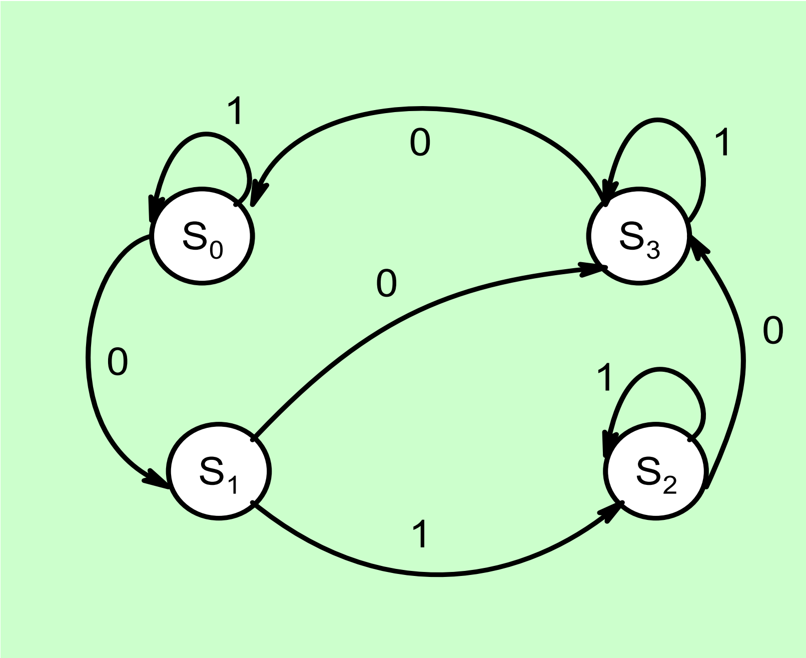
D_B

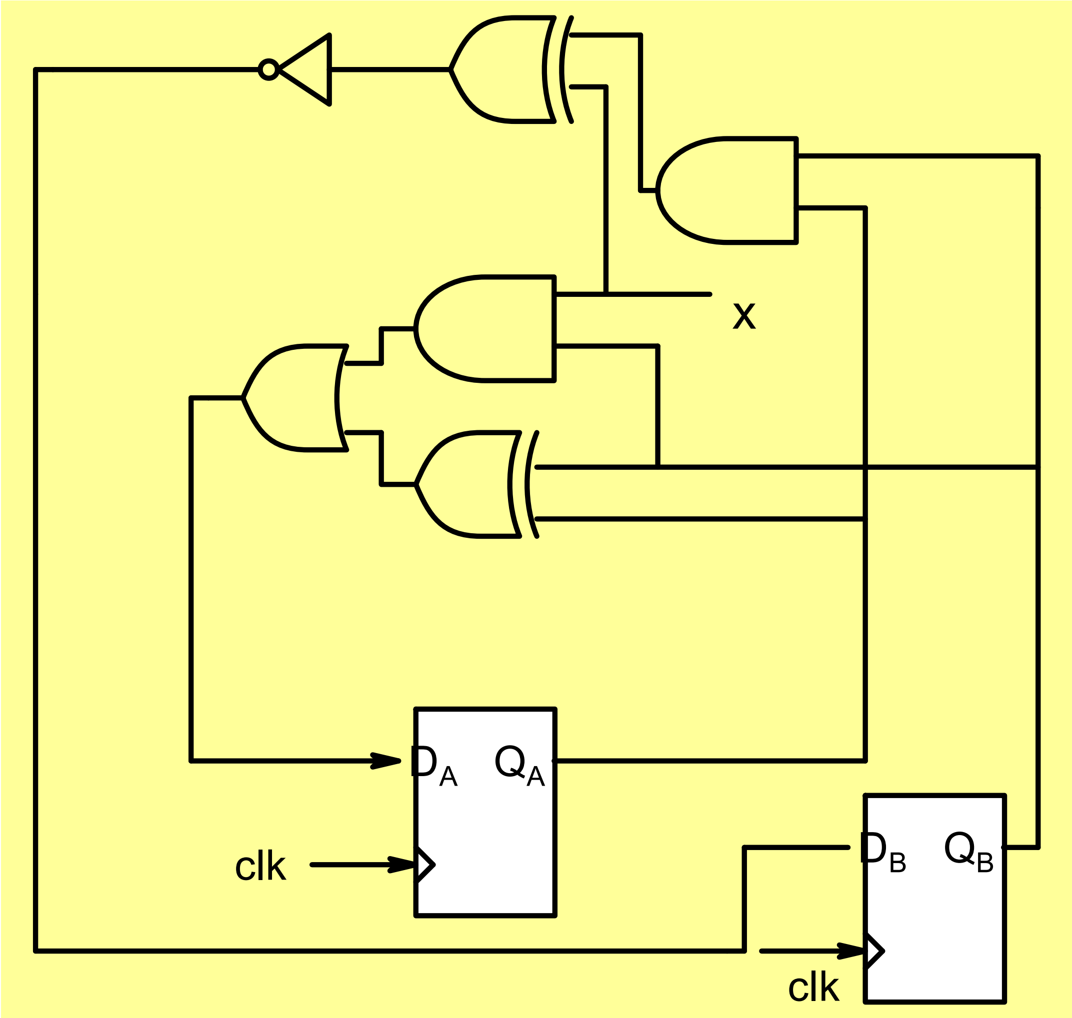
x \ AB	00	01	11	10
0	1	1	0	1
1	0	0	1	0

$$D_B = \overline{x}.\overline{A} + \overline{x}.\overline{B} + x.A.B$$
$$= \overline{x}.(\overline{A} + \overline{B}) + x.A.B$$
$$= \overline{x}.\overline{AB} + x.AB = \overline{x \oplus AB}$$

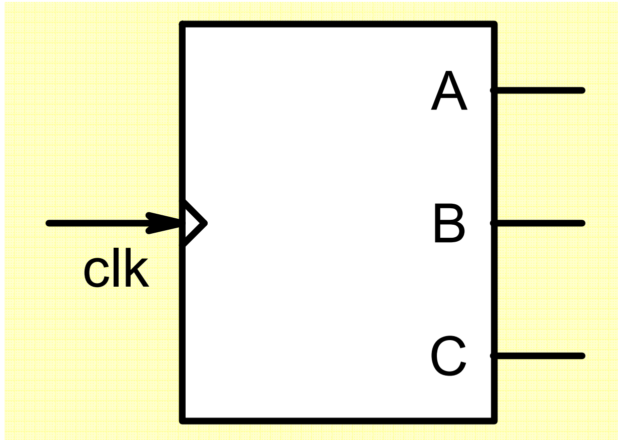
$$D_A = A \oplus B + x.B$$

$$D_B = \overline{x \oplus AB}$$

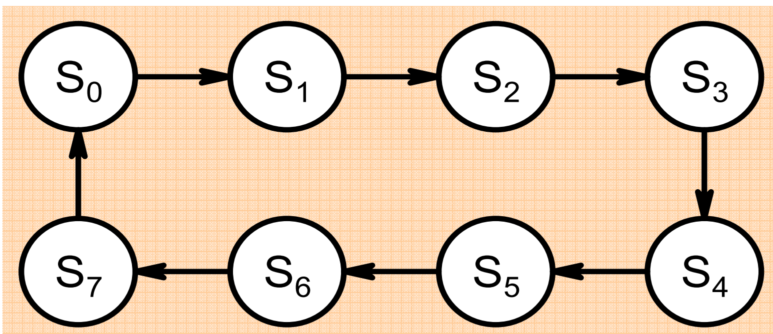




Counters

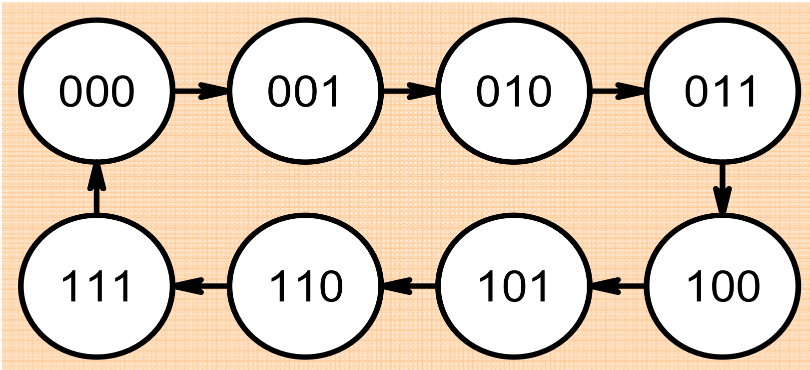


A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

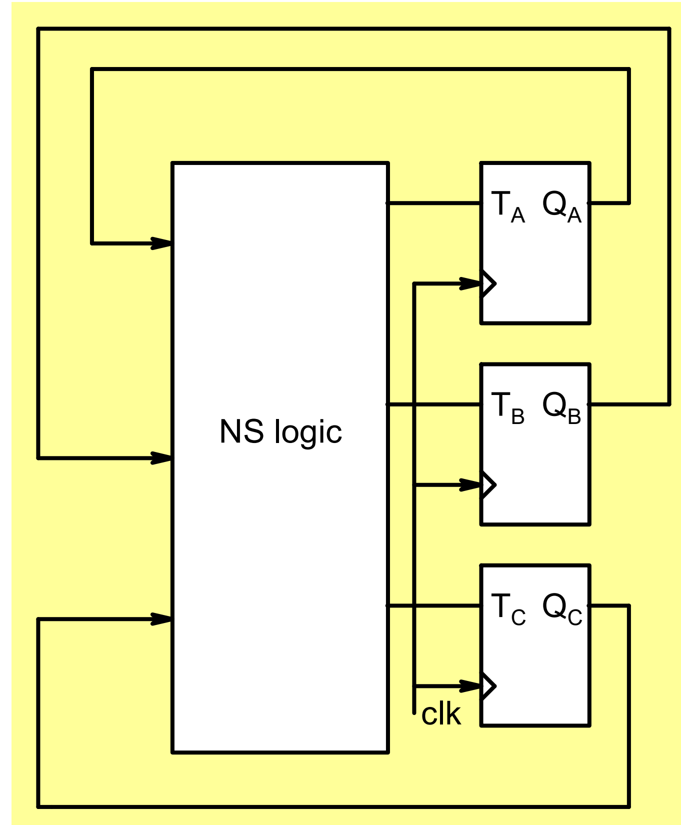


In state S_0, the output ABC is 000, in S_1 001 and so on

There are 8 states so 3 FFs are at least required. Let us choose T FF.



PS	NS	
A B C	A B C	T_A T_B T_C
0 0 0	0 0 1	0 0 1
0 0 1	0 1 0	0 1 1
0 1 0	0 1 1	0 0 1
0 1 1	1 0 0	1 1 1
1 0 0	1 0 1	0 0 1
1 0 1	1 1 0	0 1 1
1 1 0	1 1 1	0 0 1
1 1 1	0 0 0	1 1 1



$$T_A = B.C \; ; \; T_B = C \; ; \; T_C = 1$$

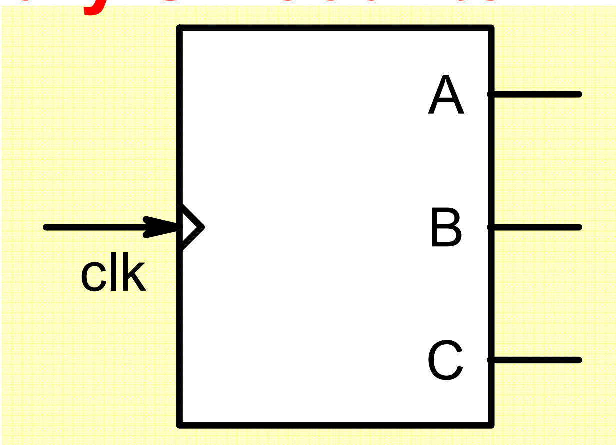
$$T_B = C$$

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

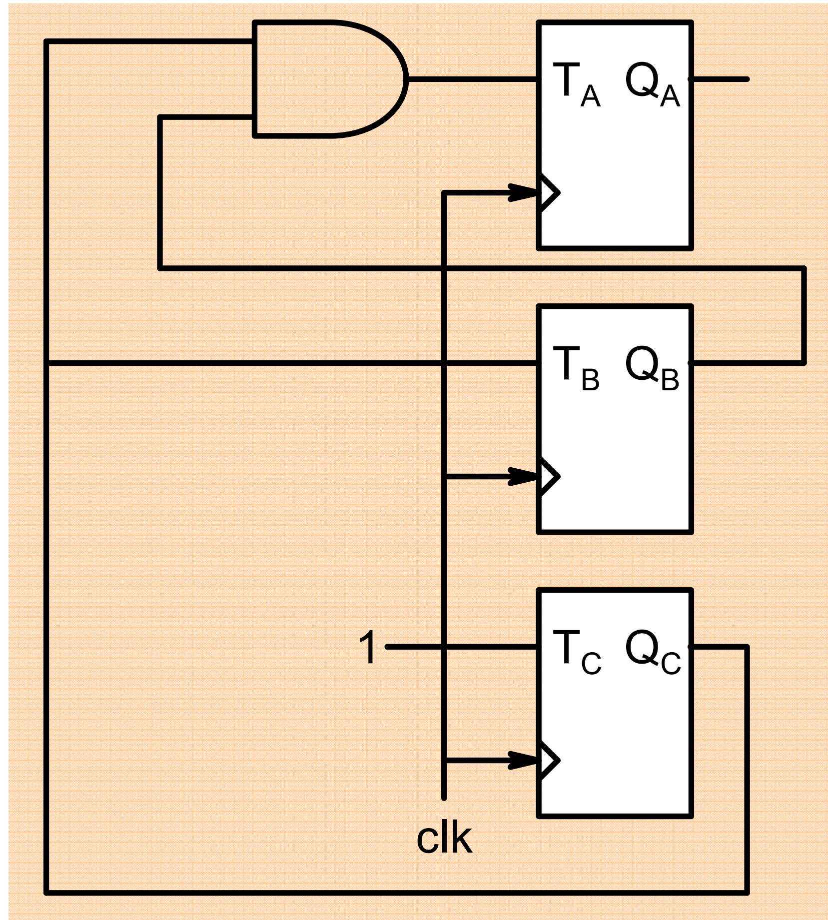
$$T_A = B.C$$

C \ AB	00	01	11	10
0				
1		1	1	

Binary UP counter



A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



$$T_A = B.C \; ; \; T_B = C \; ; \; T_C = 1$$

A B C D
0 0 0 0
0 0 0 1
0 0 1 0
0 0 1 1
0 1 0 0
0 1 0 1
0 1 10
0 1 1 1
1 0 0 0
1 0 0 1
1 0 1 0
1 0 1 1
1 1 0 0
1 1 0 1
1 1 1 0
1 1 1 1
0 0 0 0

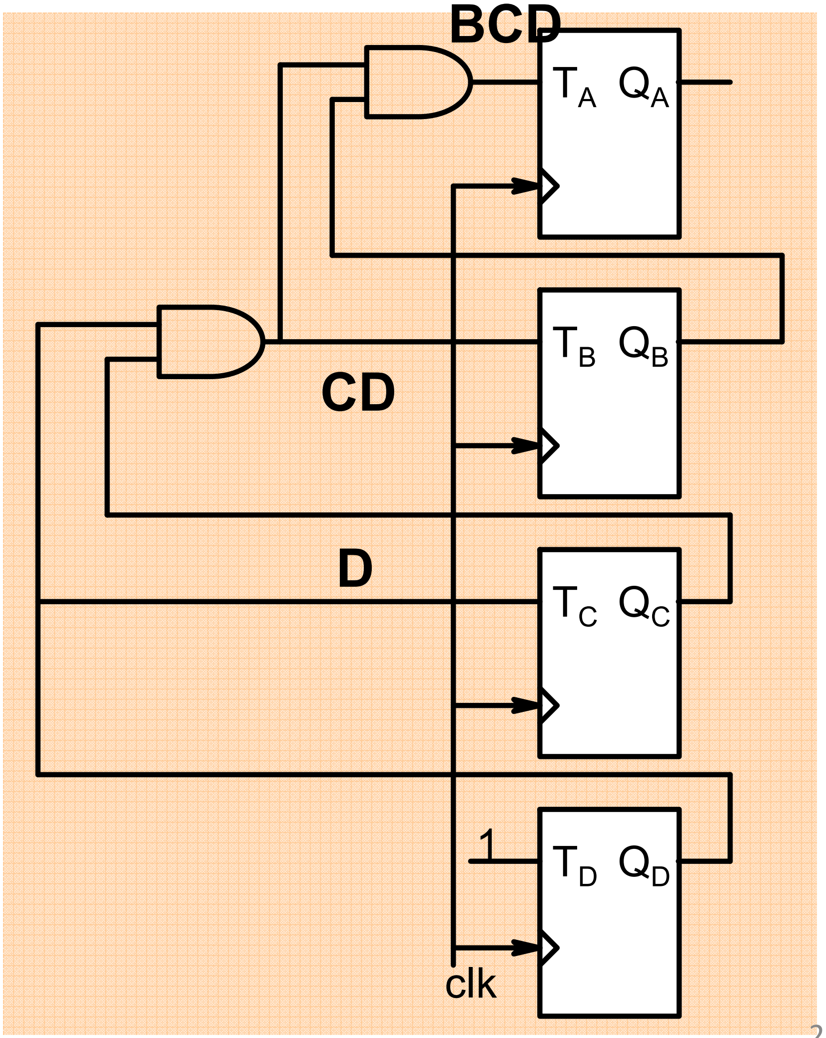
-D toggles every clock cycle

-C toggles only when D is 1

-B toggles only when both C and D are 1

-A toggles only when B C D are 1

T FF toggles when T=1



4-bit Down Counter

A B C D

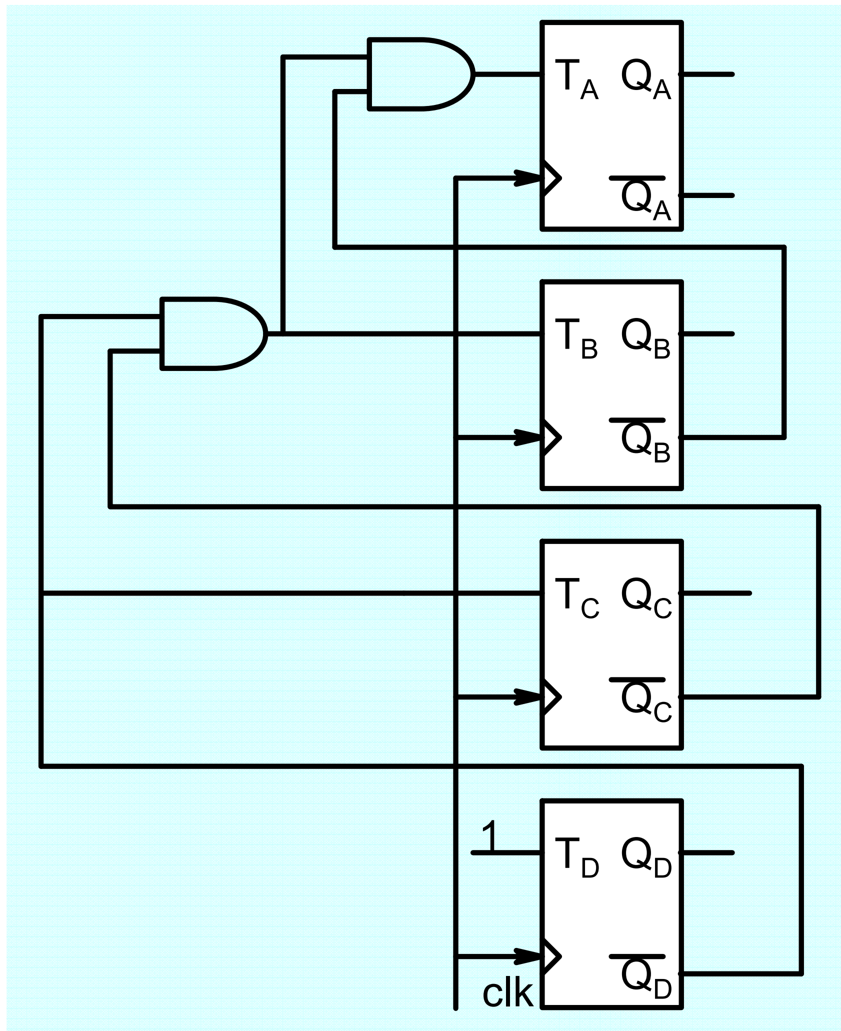
1 1 1 1
1 1 1 0
1 1 0 1
1 1 0 0
1 0 1 1
1 0 1 0
1 0 0 1
1 0 0 0
0 1 1 1
0 1 1 0
0 1 0 1
0 1 0 0
0 0 1 1
0 0 1 0
0 0 0 1
0 0 0 0
1 1 1 1

-D toggles every clock cycle

-C toggles only when D is 0

-B toggles only when both C and D are 0

-A toggles only when D C B are 0



Counters

A	B	C
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Binary down counter

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

Decade counter

Modulo-10 Counter

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

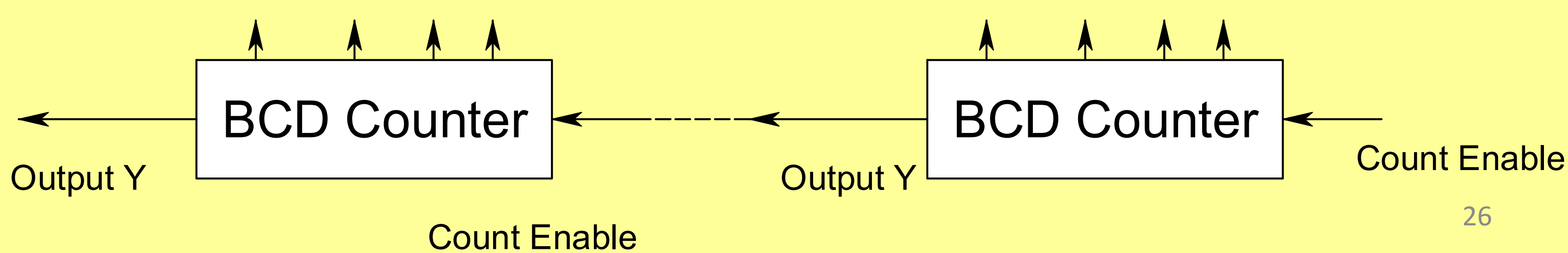
Modulo-5 Counter

BCD Counter

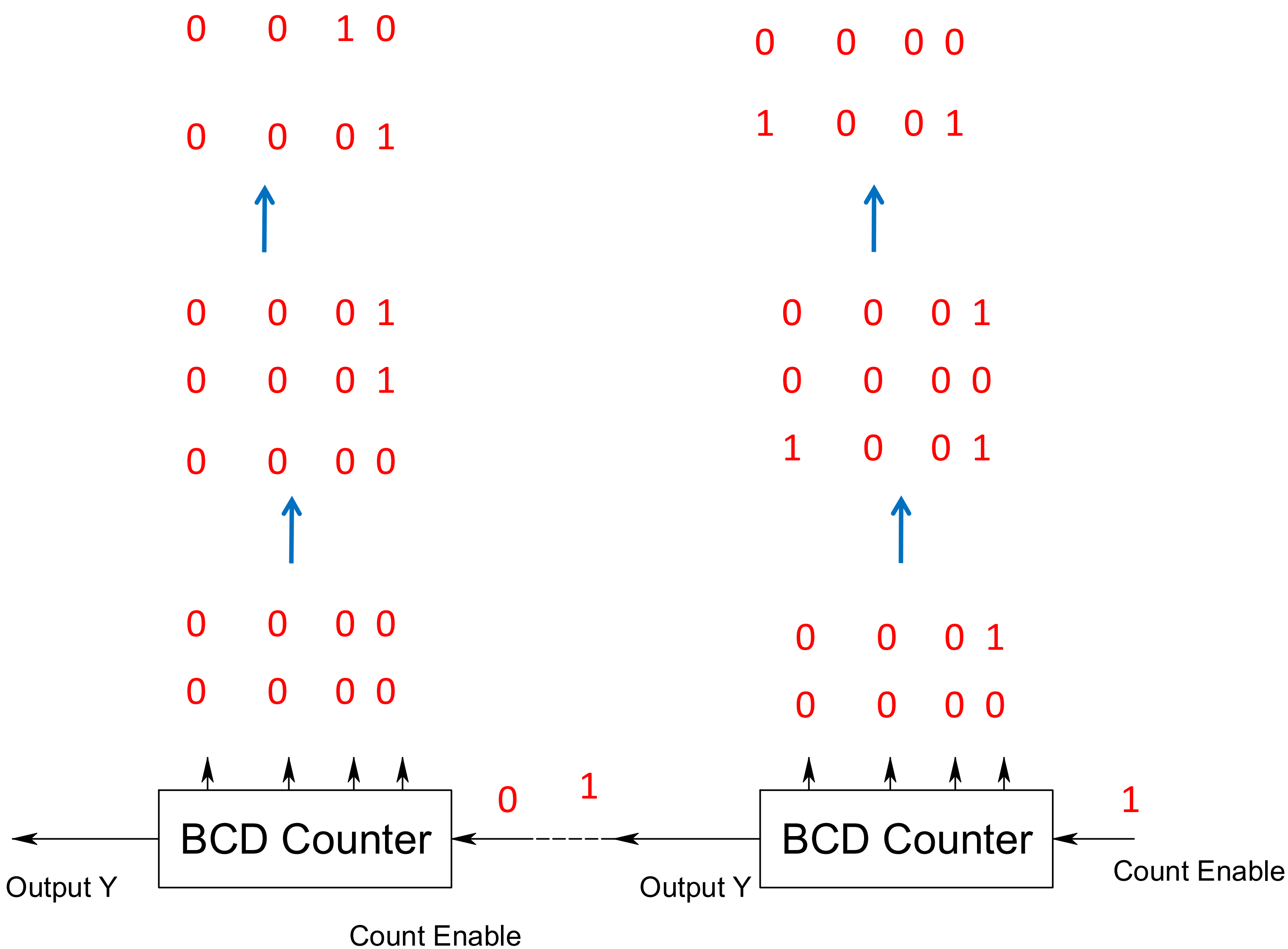
Binary Coded Decimal (BCD): each decimal digit is coded as a 4-bit binary number

25 = 0010 0101

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1



BCD counter from 0 to 99



Counter with Unused States

PS			NS								
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
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	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

There are two unused states 011 and 111.

one approach to handle this situation is that, while evaluating expressions for J K , we use don't care conditions corresponding to these unused states

Counter with Unused States

PS A	PS B	PS C	NS A	NS B	NS C	J_A	K_A	J_B	K_B	J_C	K_C
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	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

A \ BC	00	01	11	10
0	0	0	X	1
1	X	X	X	X

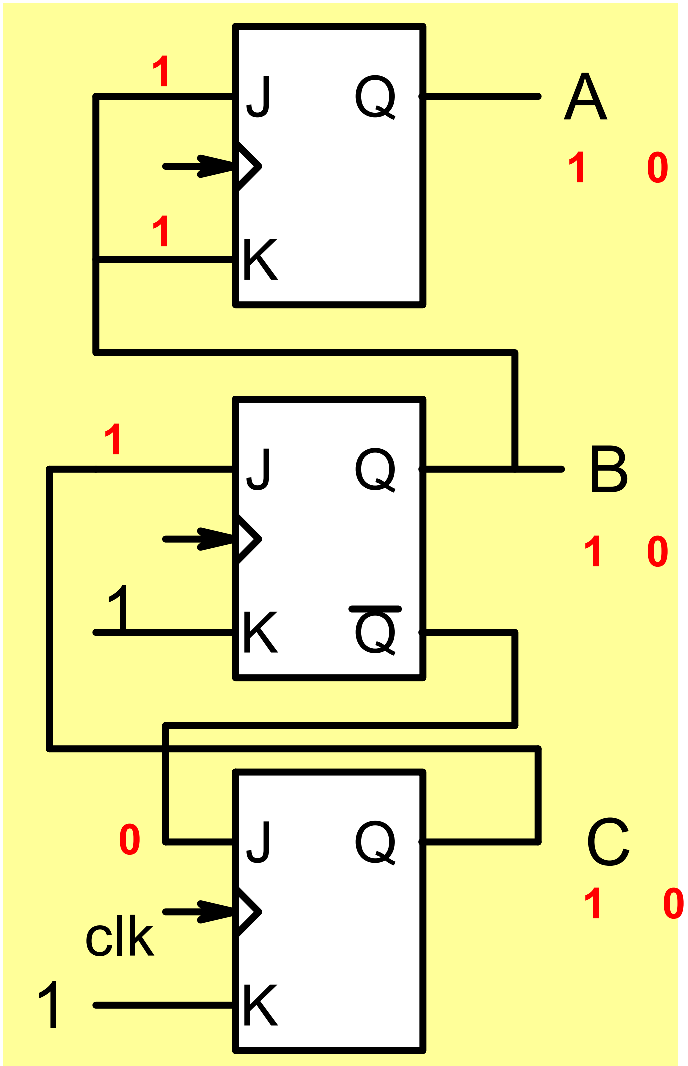
$$J_A = B$$

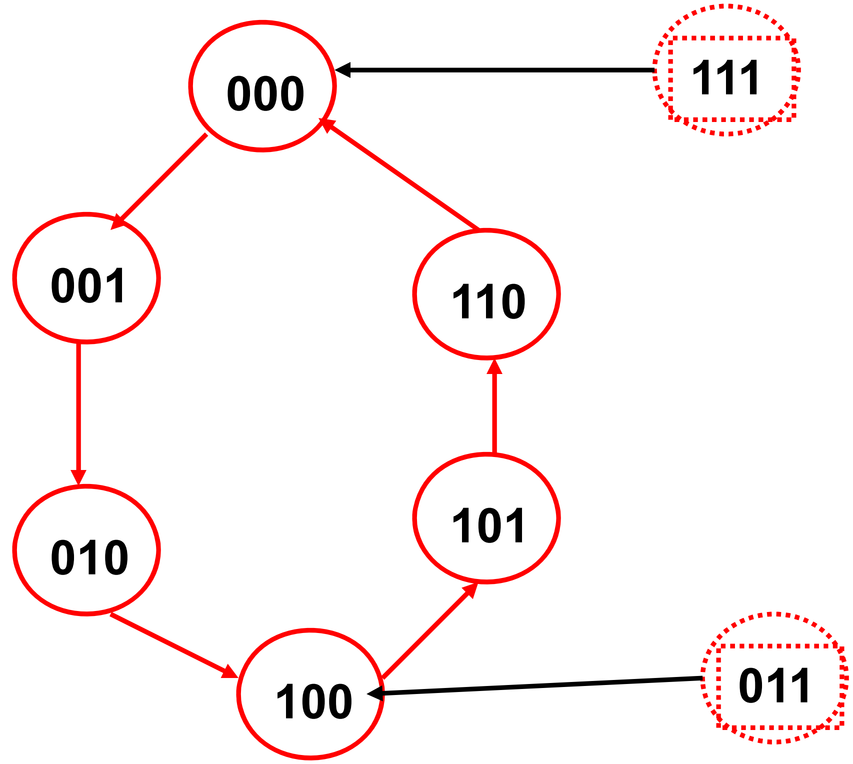
Counter with Unused States

PS A B C	NS A B C	J_A	K_A	J_B	K_B	J_C	K_C
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	1 0 0	1	X	X	1	0	X
1 0 0	1 0 1	X	0	0	X	1	X
1 0 1	1 1 0	X	0	1	X	X	1
1 1 0	0 0 0	X	1	X	1	0	X

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = \overline{B} \qquad K_C = 1$$

After synthesizing the circuit, one needs to check that if by chance the counter goes into one of the unused states, after one or more clock cycles, it enters a used state and then remains among the used states





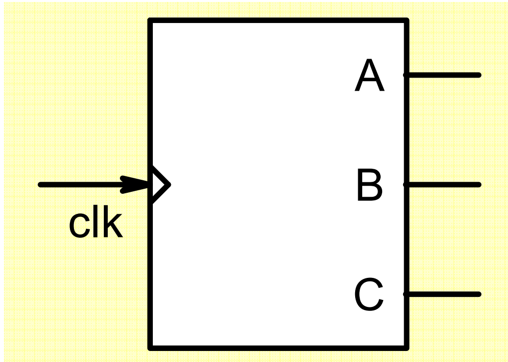
$$J_A = B \qquad K_A = B$$

$$J_B = C \qquad K_B = 1$$

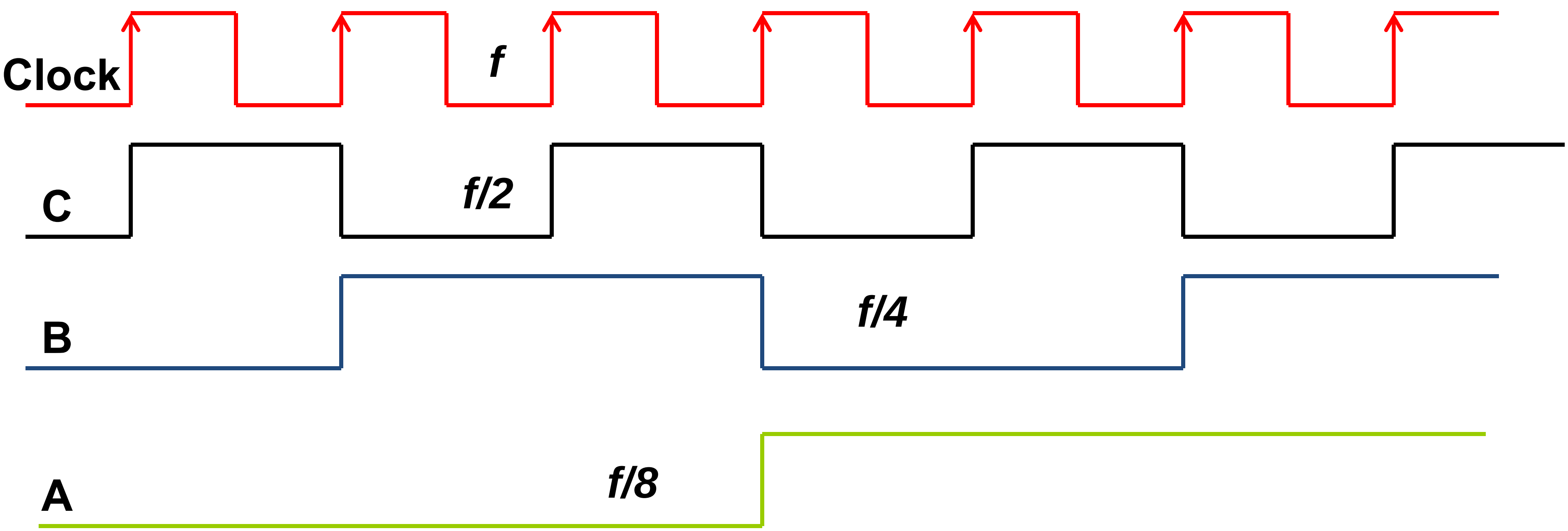
$$J_C = \overline{B} \qquad K_C = 1$$

We can see that if by chance the counter goes into unused states 111 or 011, then after a clock cycle it enters one of the used states.

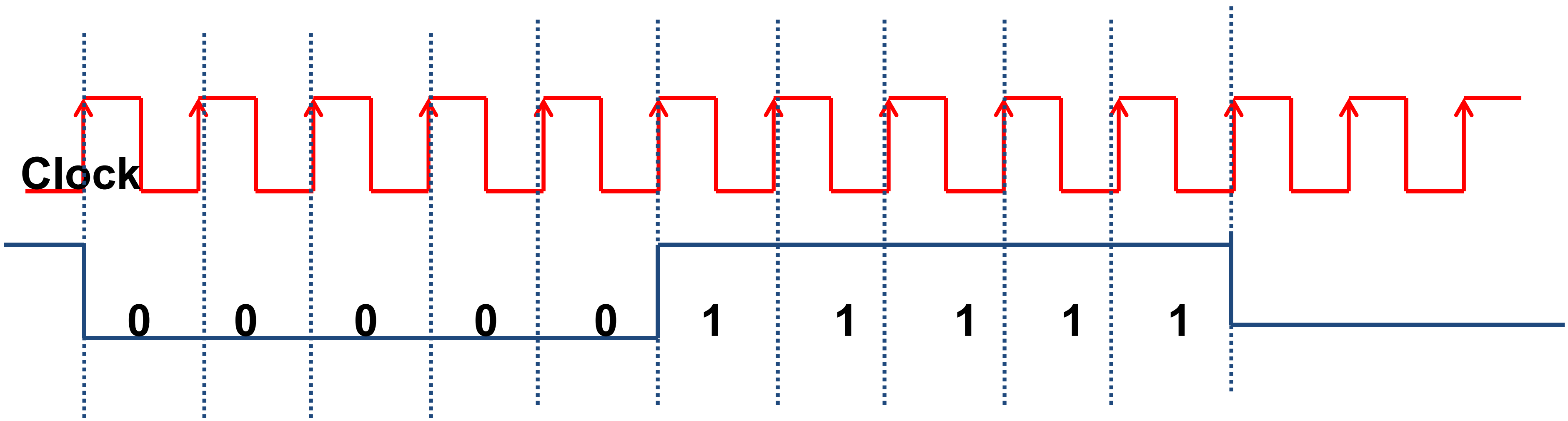
Counter as frequency divider



A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

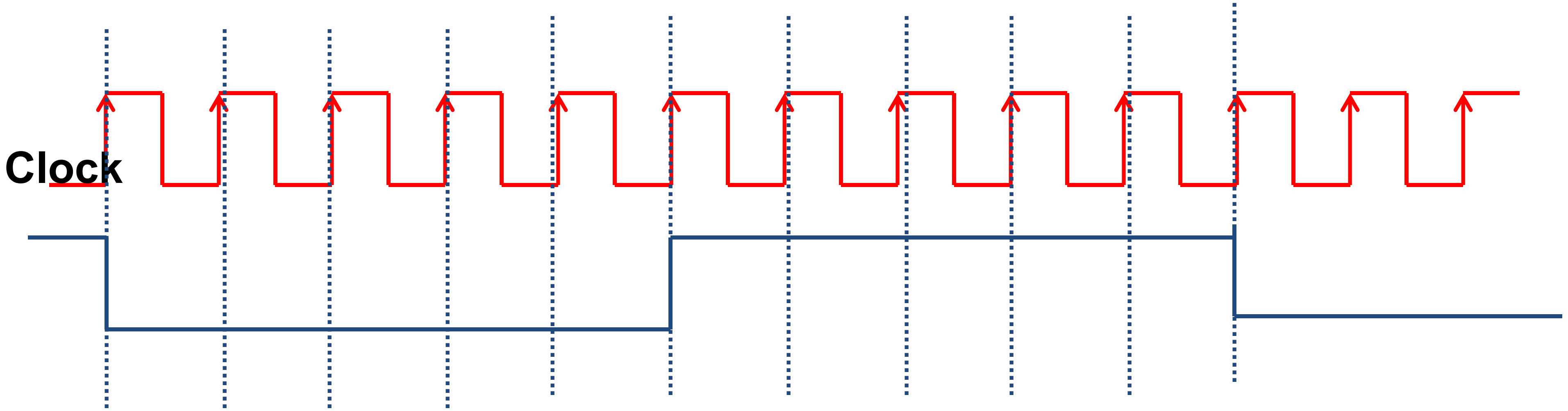


Example From a frequency of 10KHz, generate the following signal of frequency 1KHz



A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

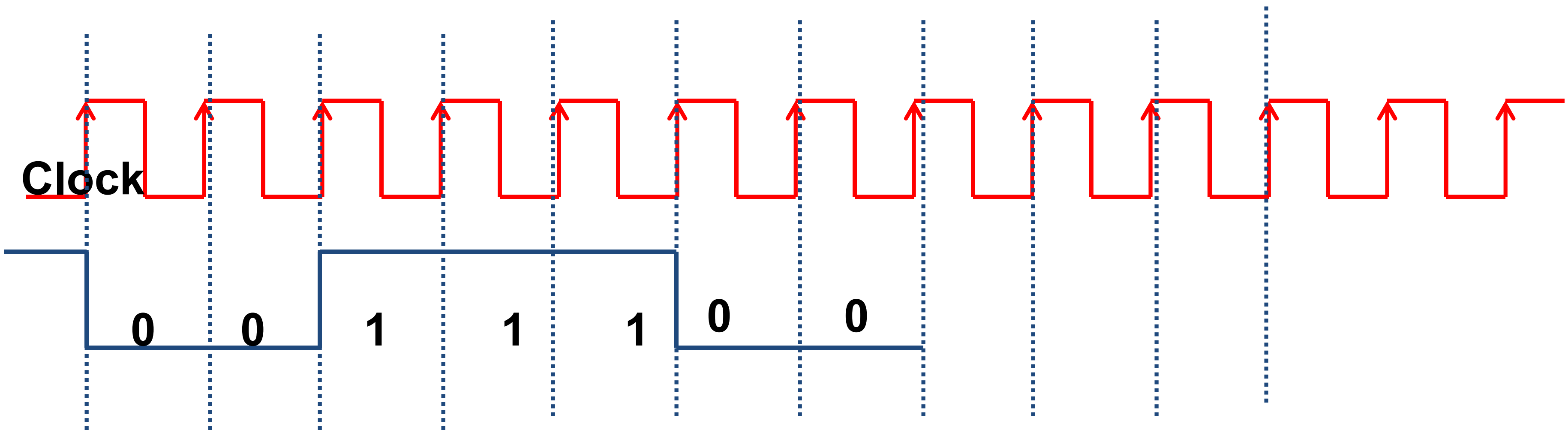
This will have a frequency of 1KHz but it will not have the same waveform



Design a divide by 10 counter with the following states

A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

Example From a frequency of 10KHz, generate the following signal of frequency 2KHz



A divide by 5 counter is required that has 5 states.

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0

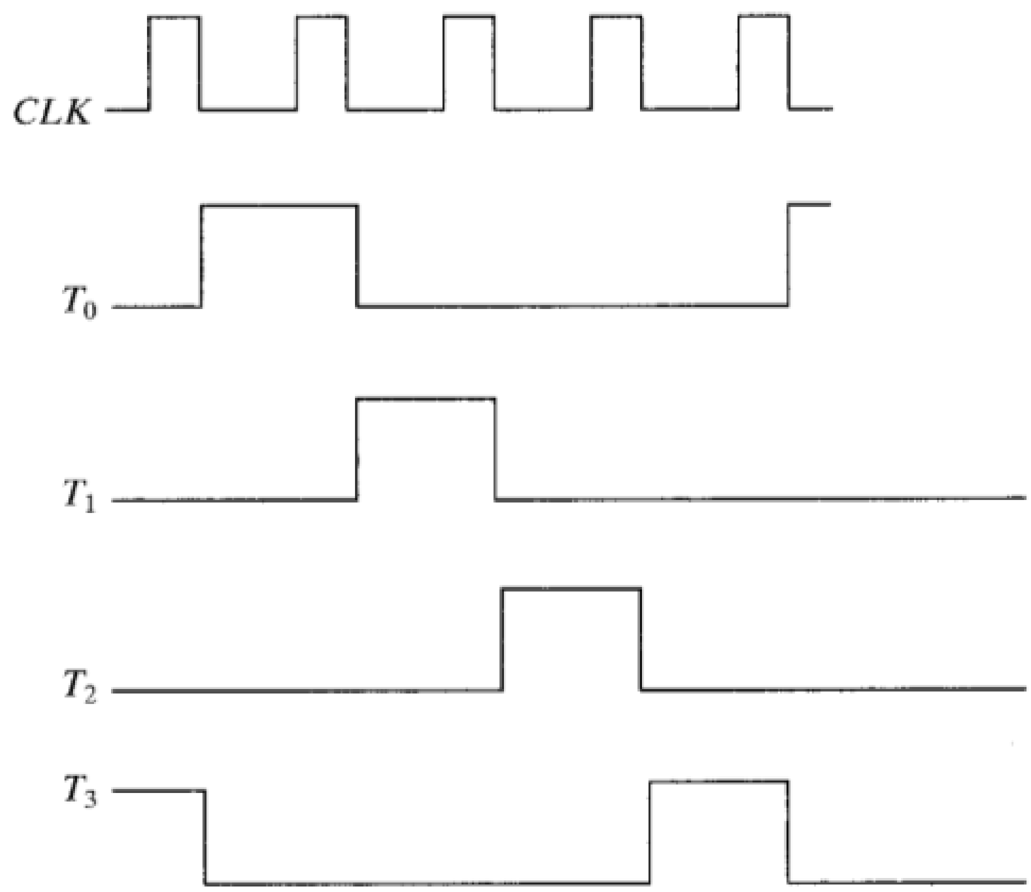
Modify →

A	B	C
0	0	0
0	0	1
1	1	0
1	1	1
1	0	0

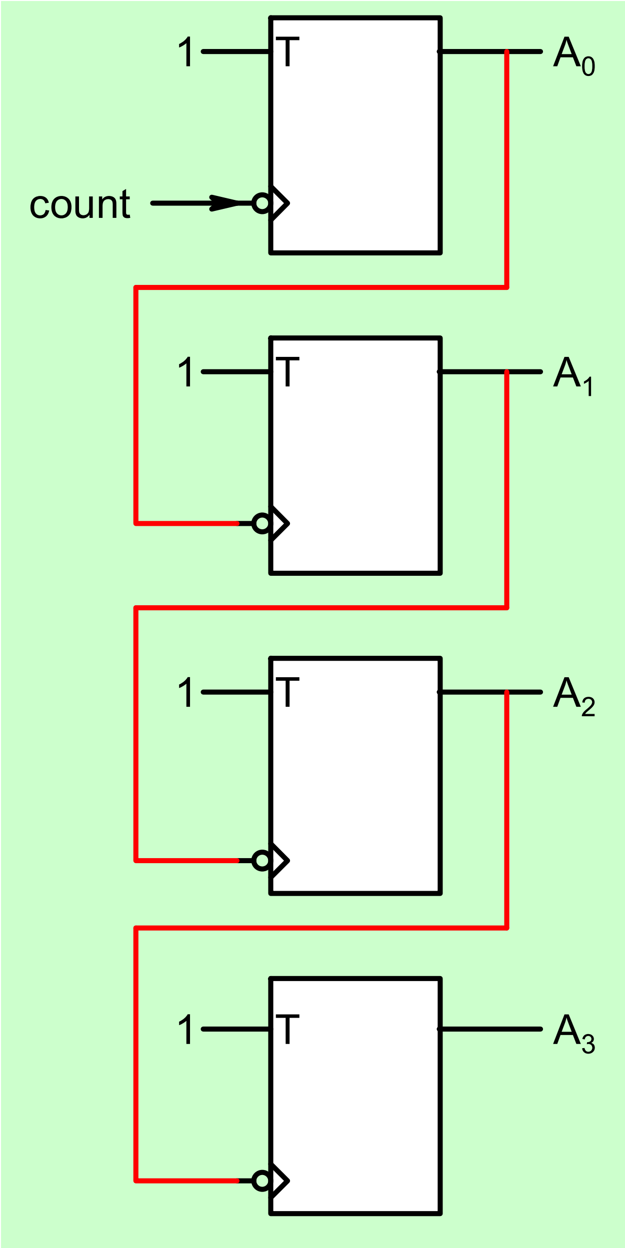
A will give the required waveform.

Ring Counter

T_3	T_2	T_1	T_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0



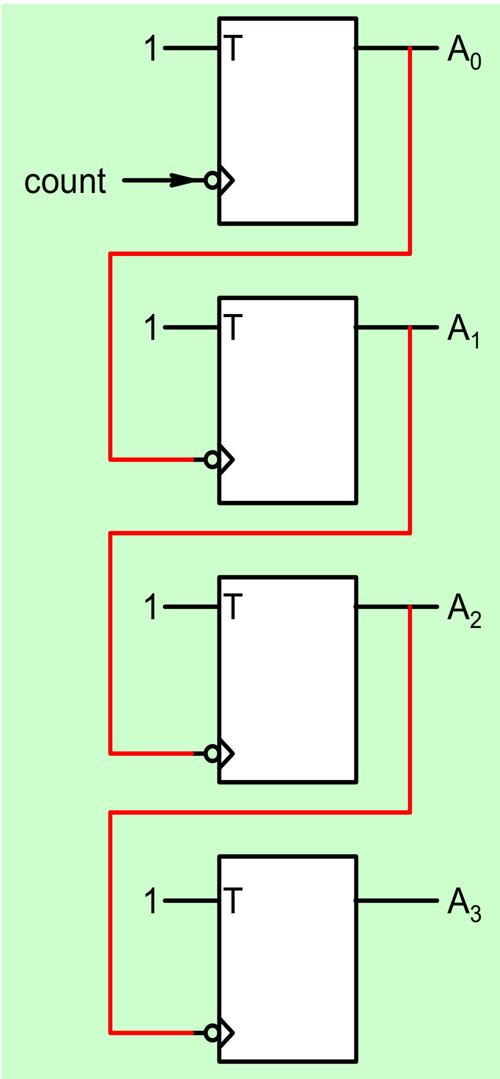
Ripple Counter



T FF toggles when T = 1; otherwise Hold state

F/F is negative edge Triggered

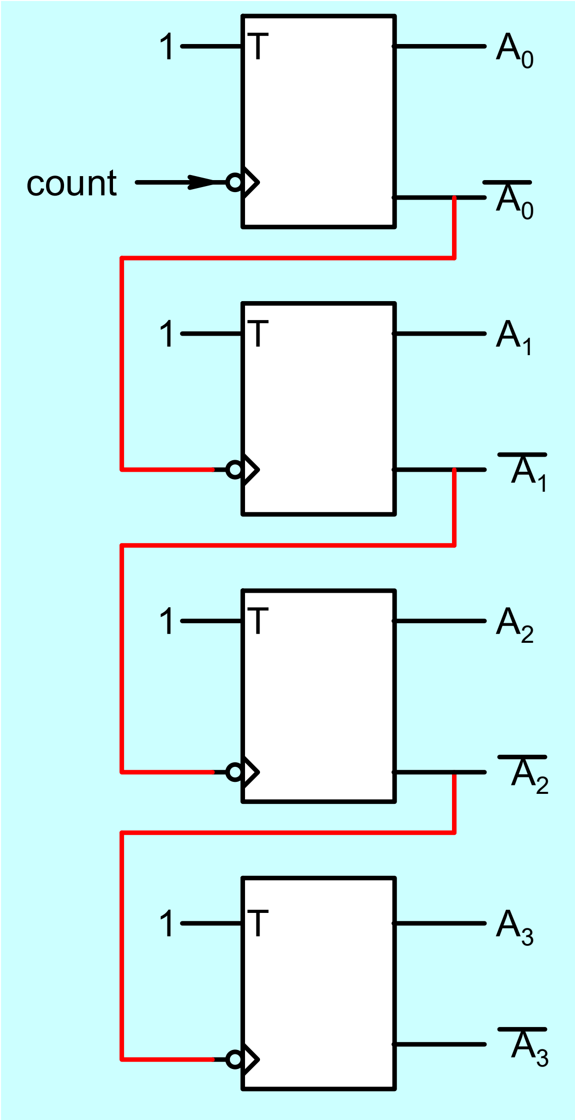
Ripple Counter



0	1	2	3	4	5	------15	
0	1	0	1	0	1	------1	0
0	0	1	1	0	0	------1	0
0	0	0	0	1	1	------1	0
0	0	0	0	0	0	------1	0

$$A_3 A_2 A_1 A_0$$

Ripple Down Counter



0 1 0

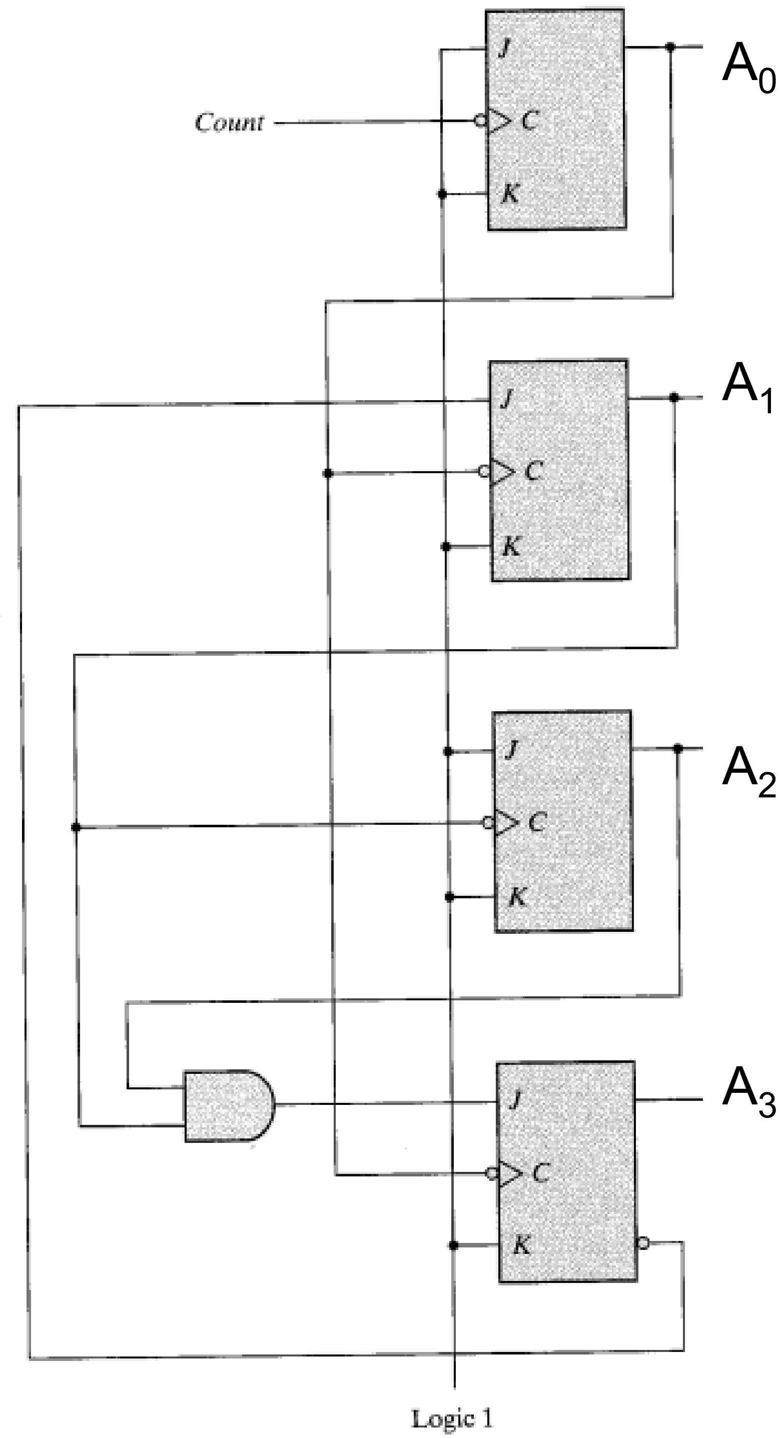
0 1 1

0 1 1

0 1 1

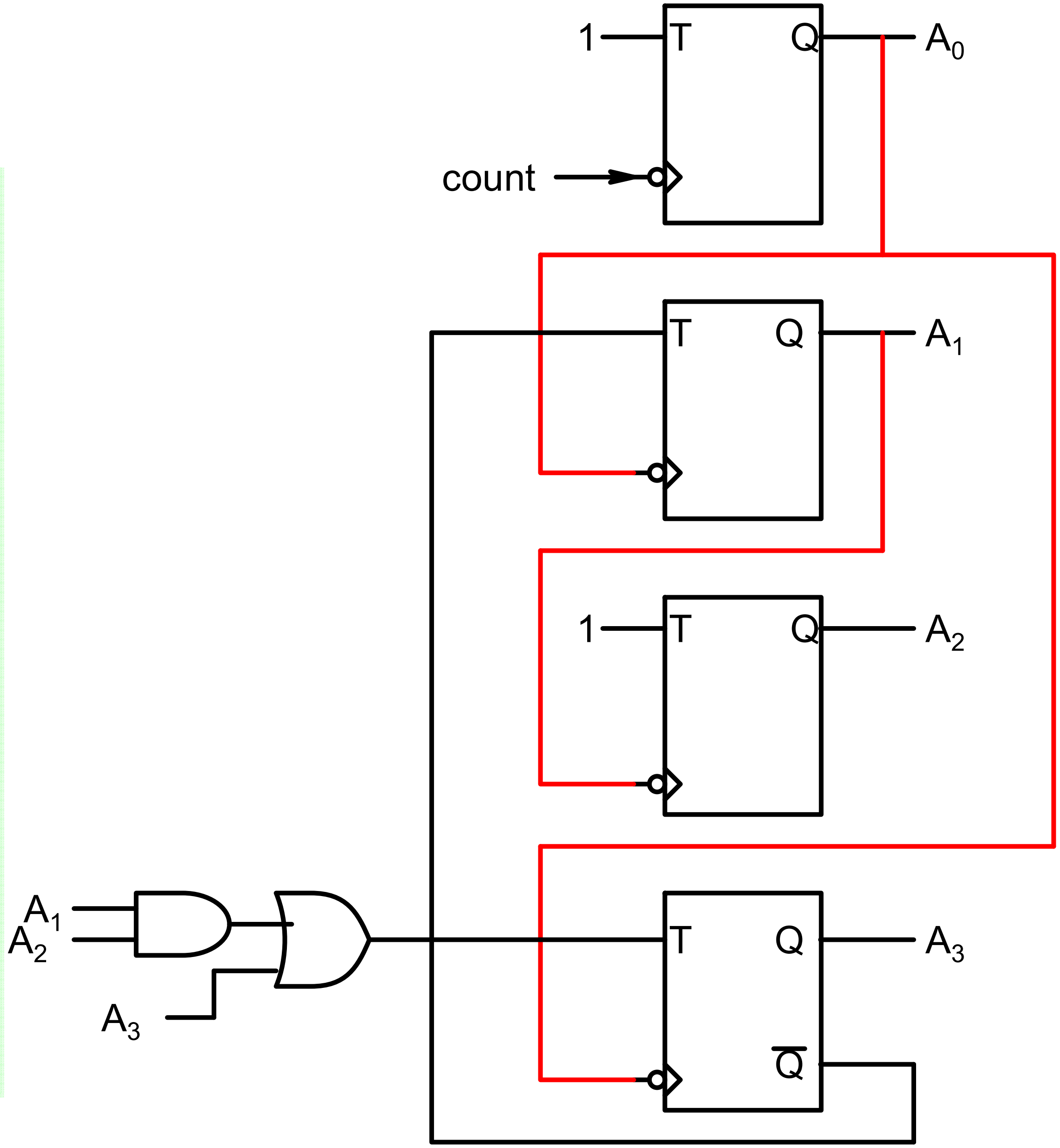
BCD Ripple Counter

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

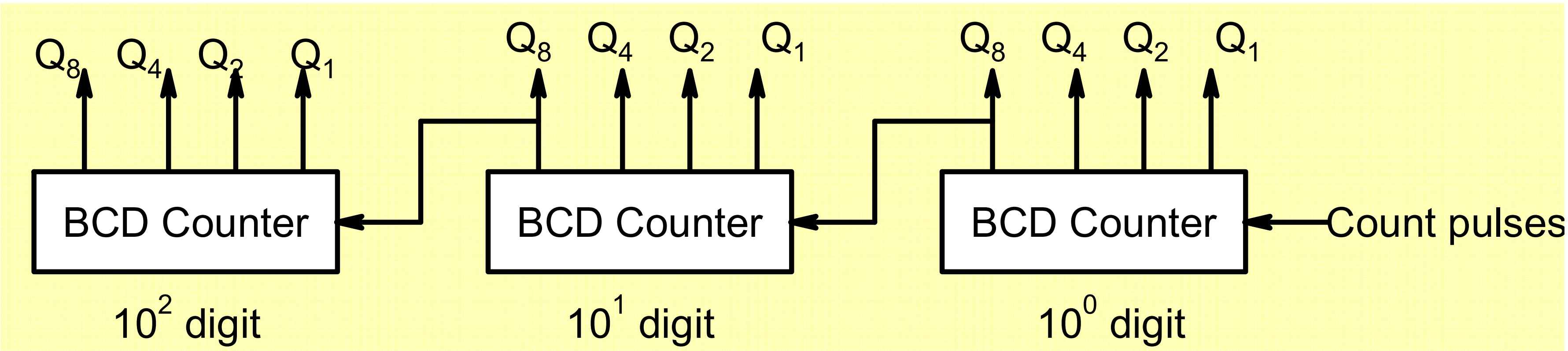


BCD Ripple Counter

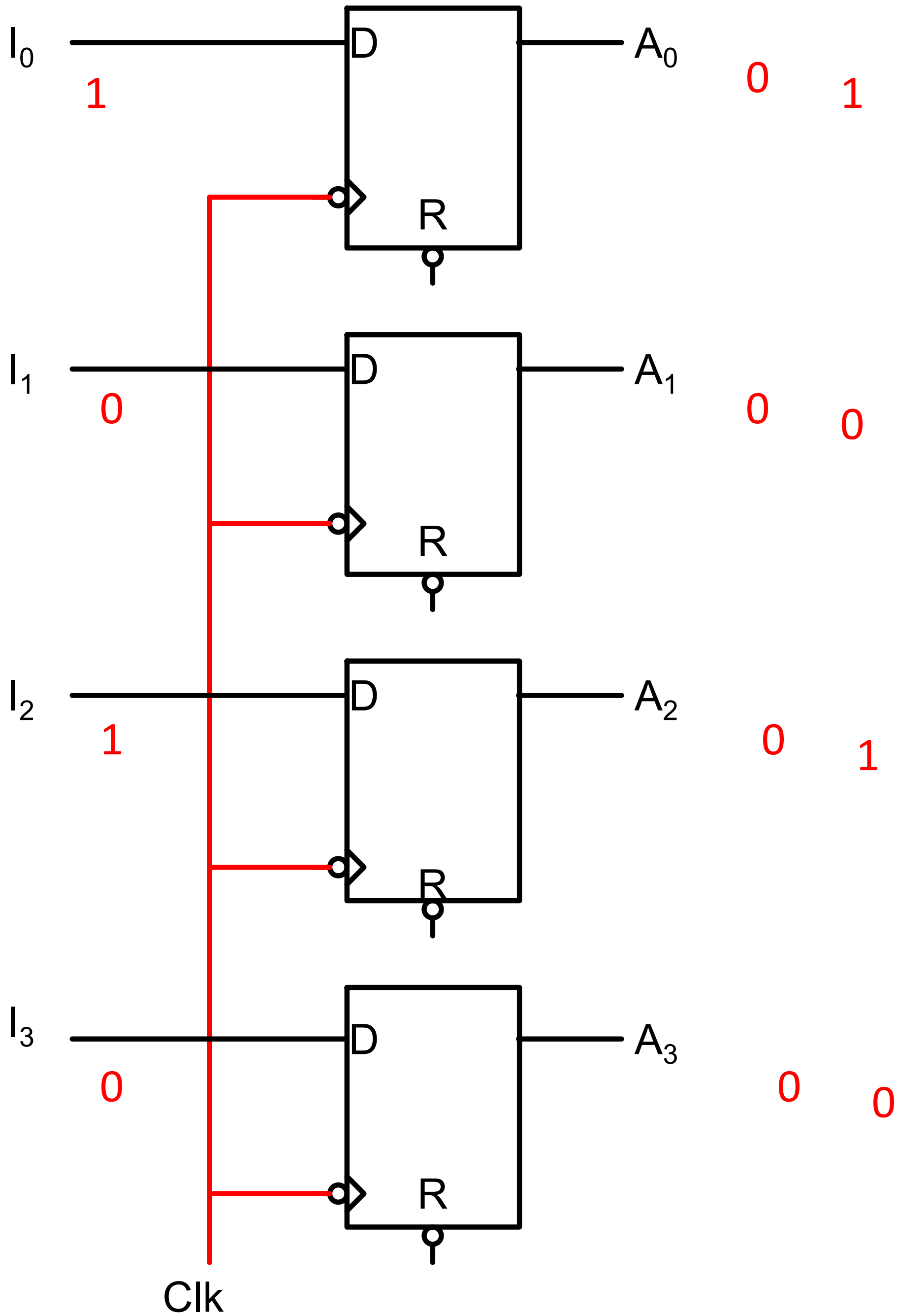
A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1



Cascading of BCD counters

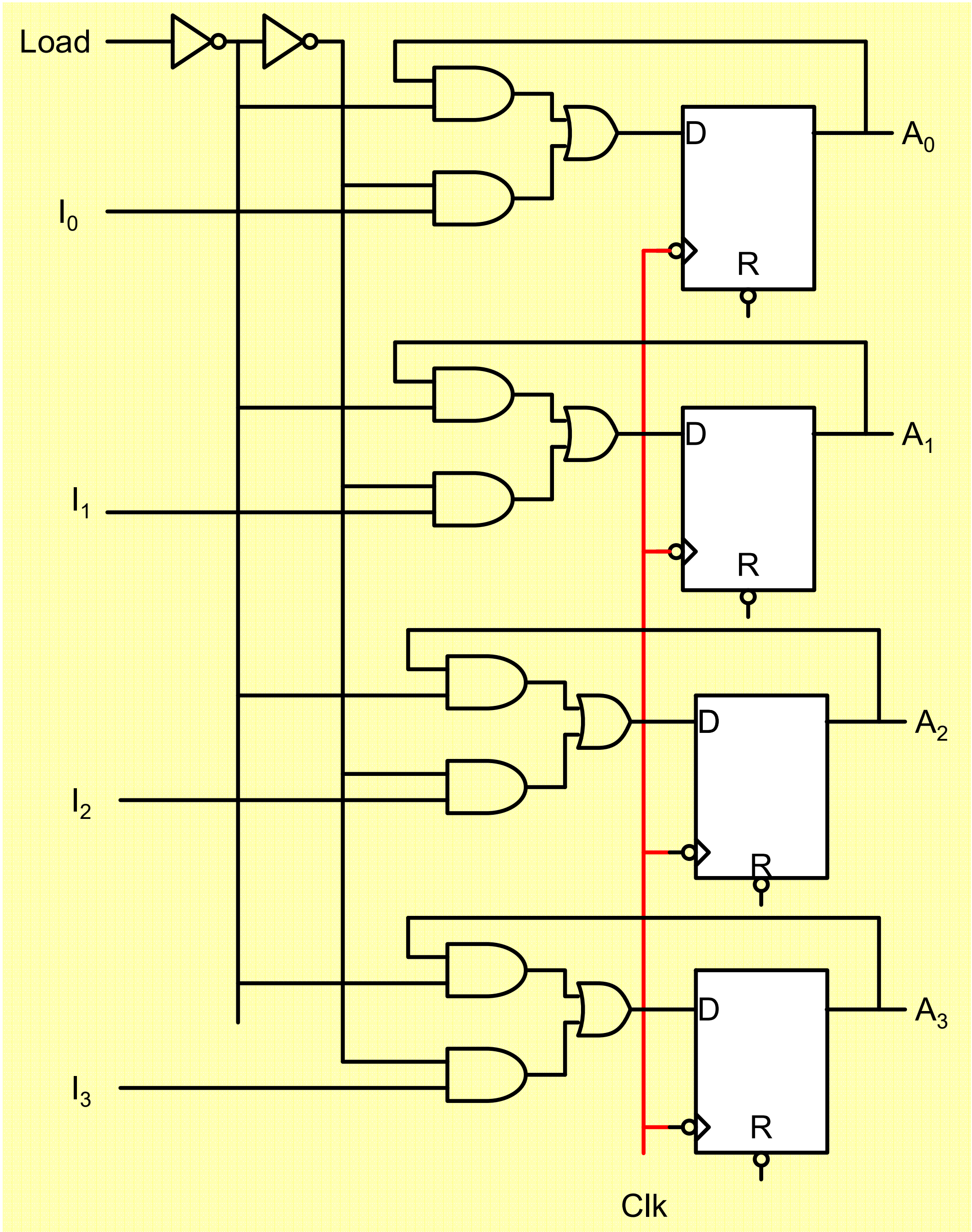


Register

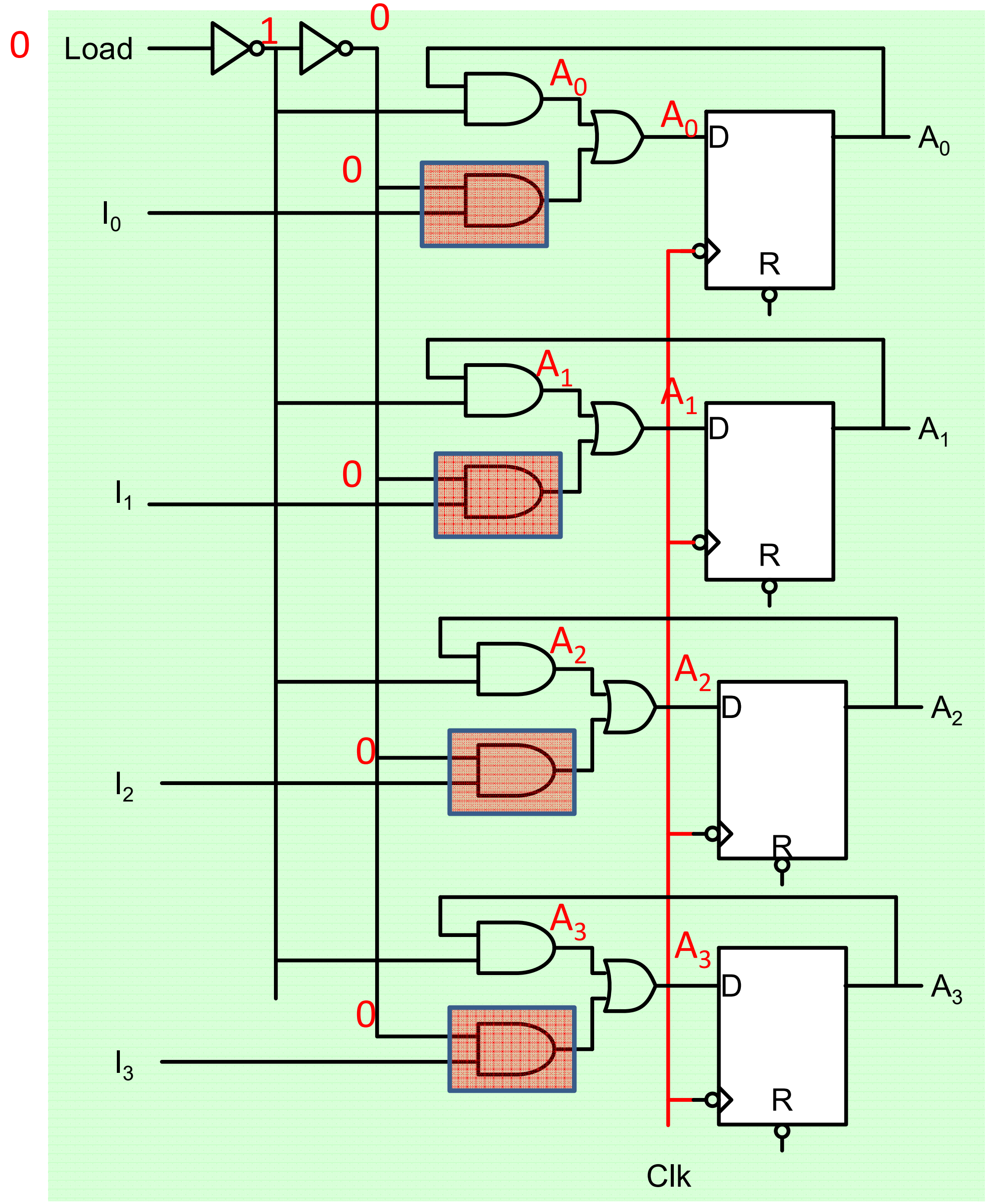


A register is a group of flip flop, each one of which is capable of storing one bit information.

4-bit Register with parallel load

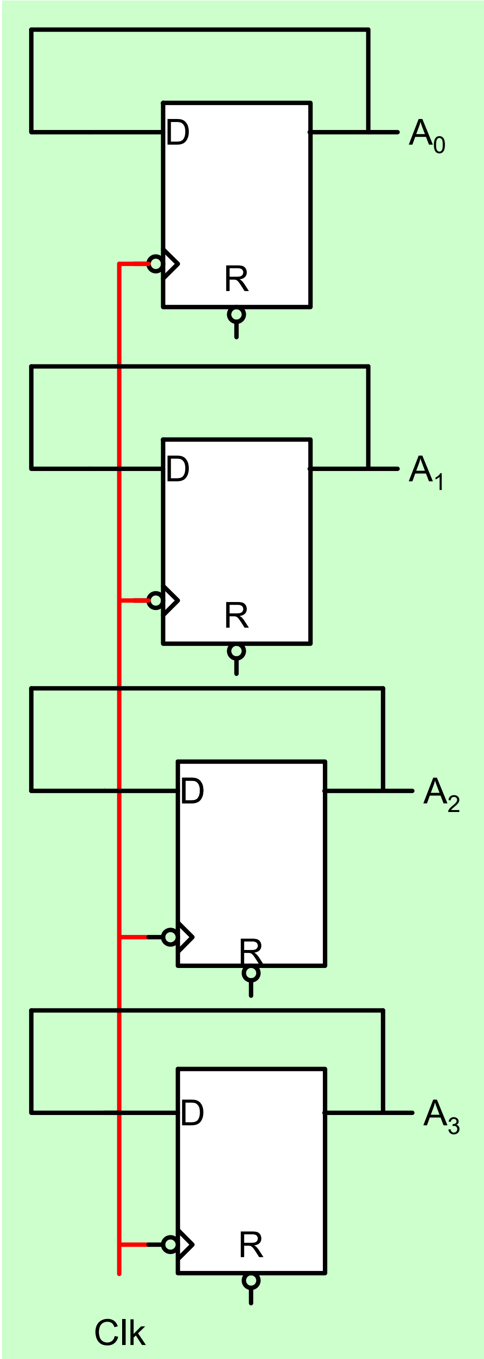


Load = 0

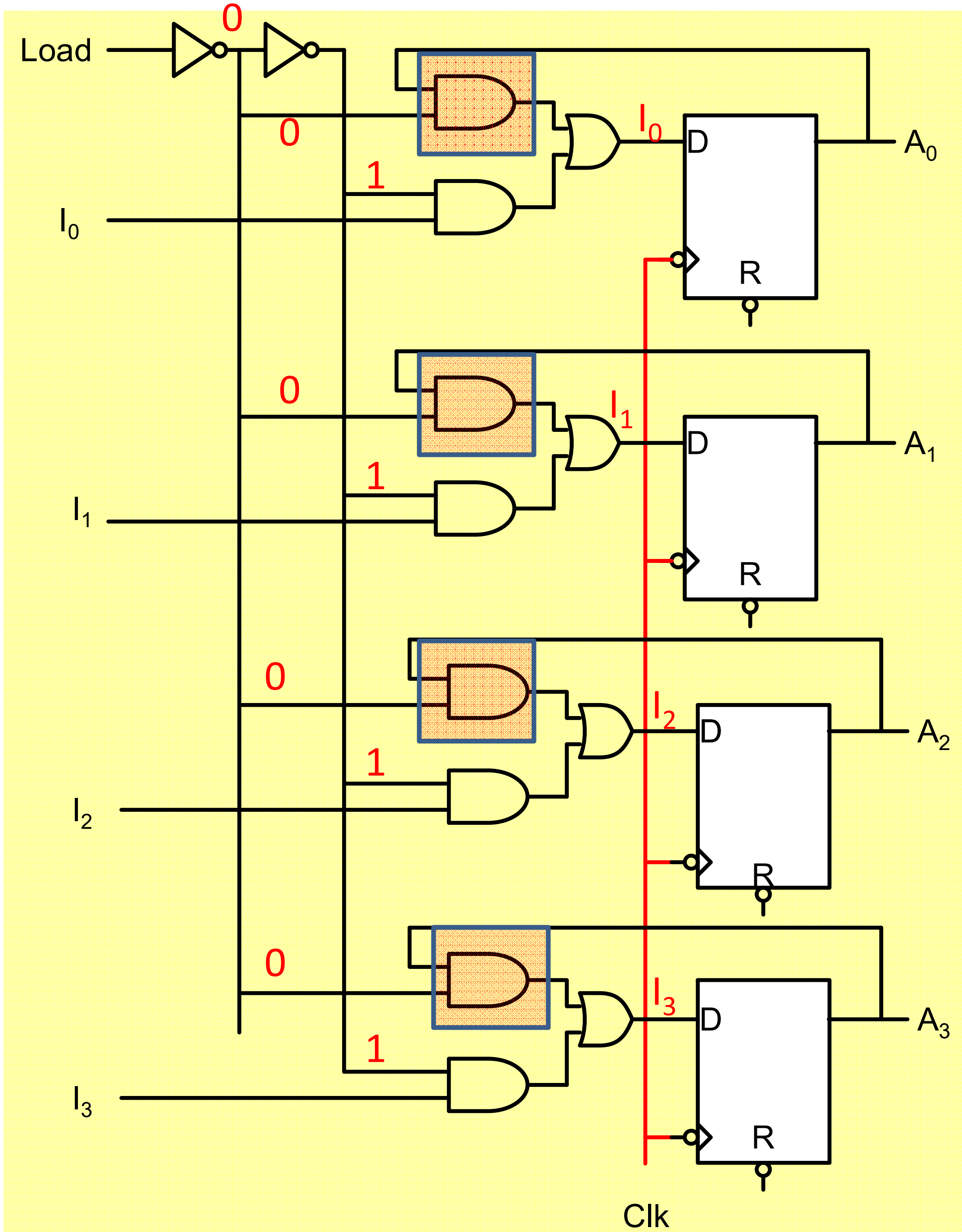


0
Load
1
0
A_0
A_0
D
R
A_0
0
I_0
A_1
A_1
D
R
A_1
0
I_1
A_2
A_2
D
R
A_2
0
I_2
A_3
A_3
D
R
A_3
0
I_3
Clk

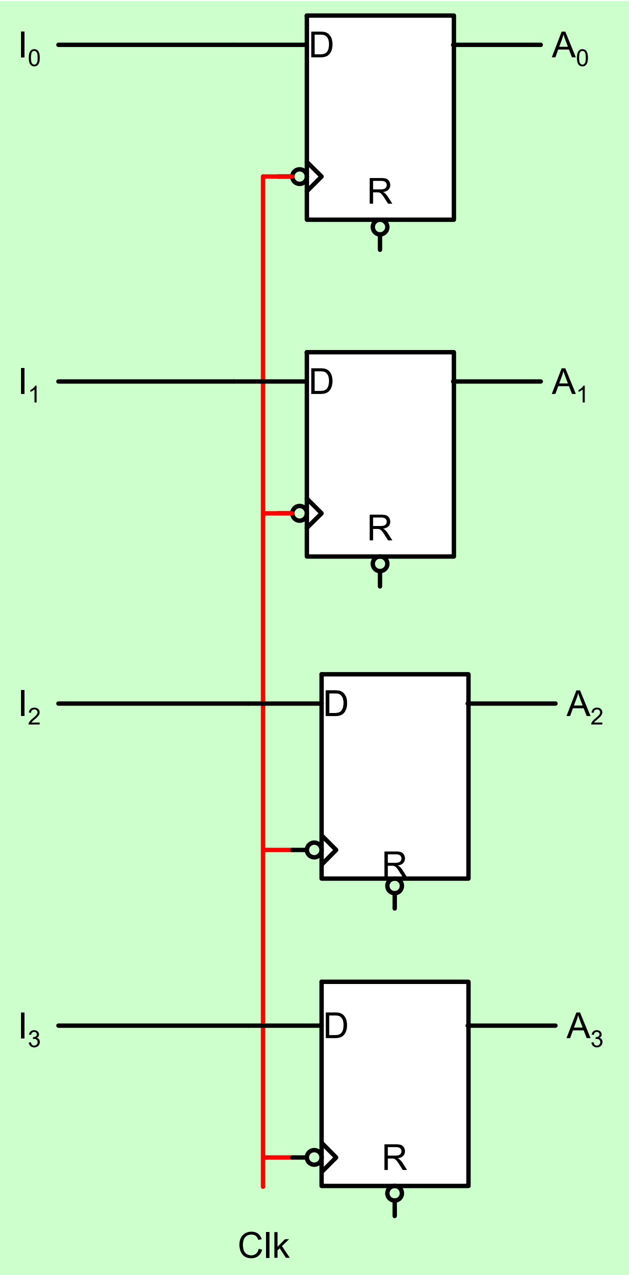
Equivalent Circuit



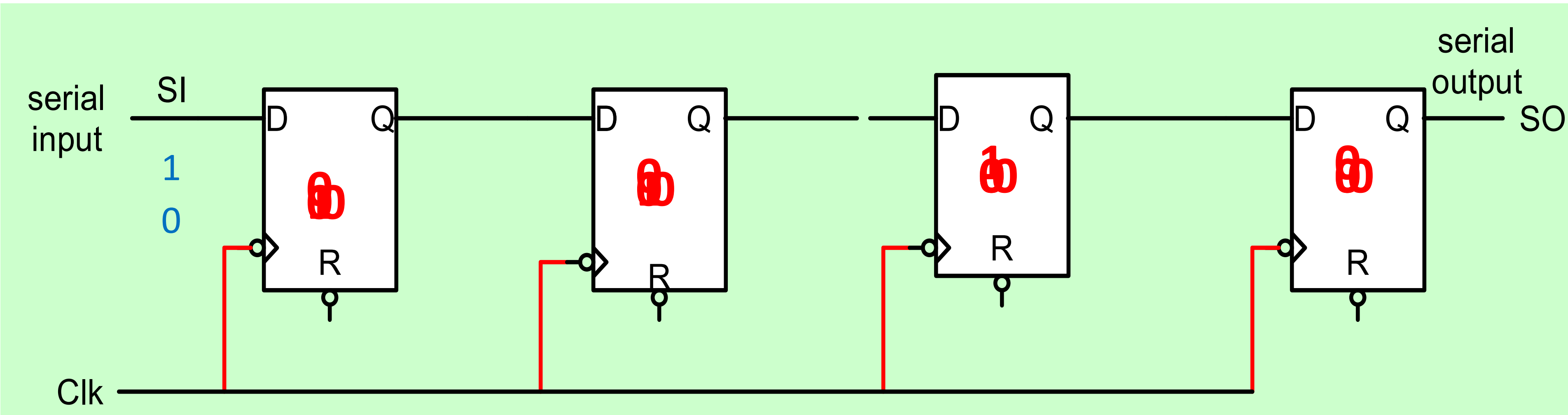
Load = 1



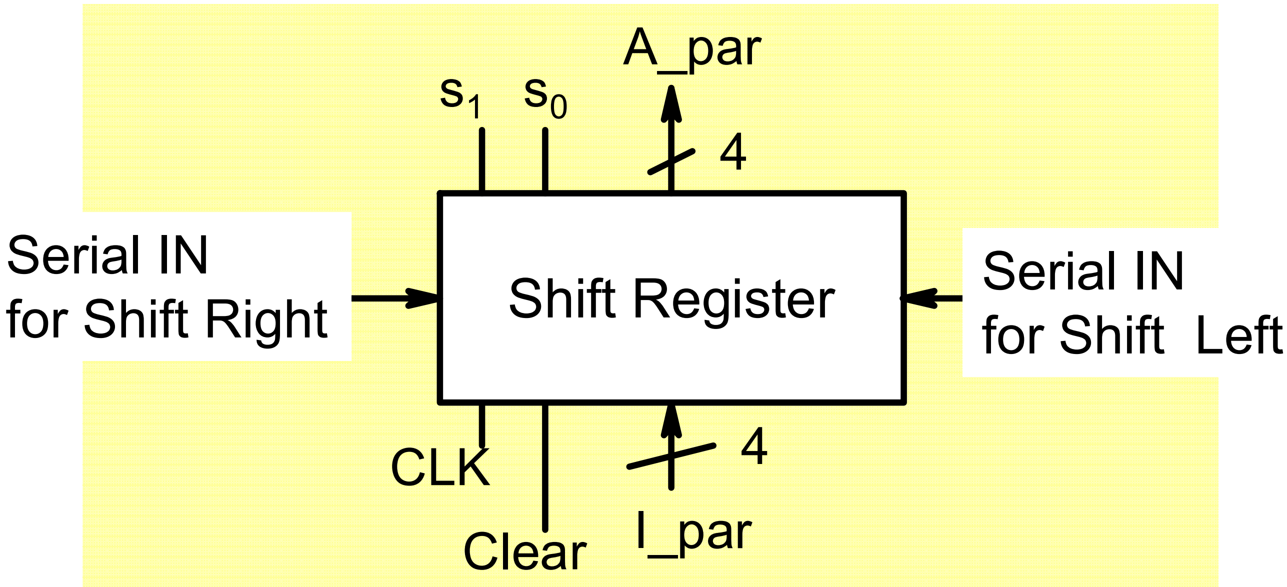
Equivalent Circuit



4-bit Shift Register

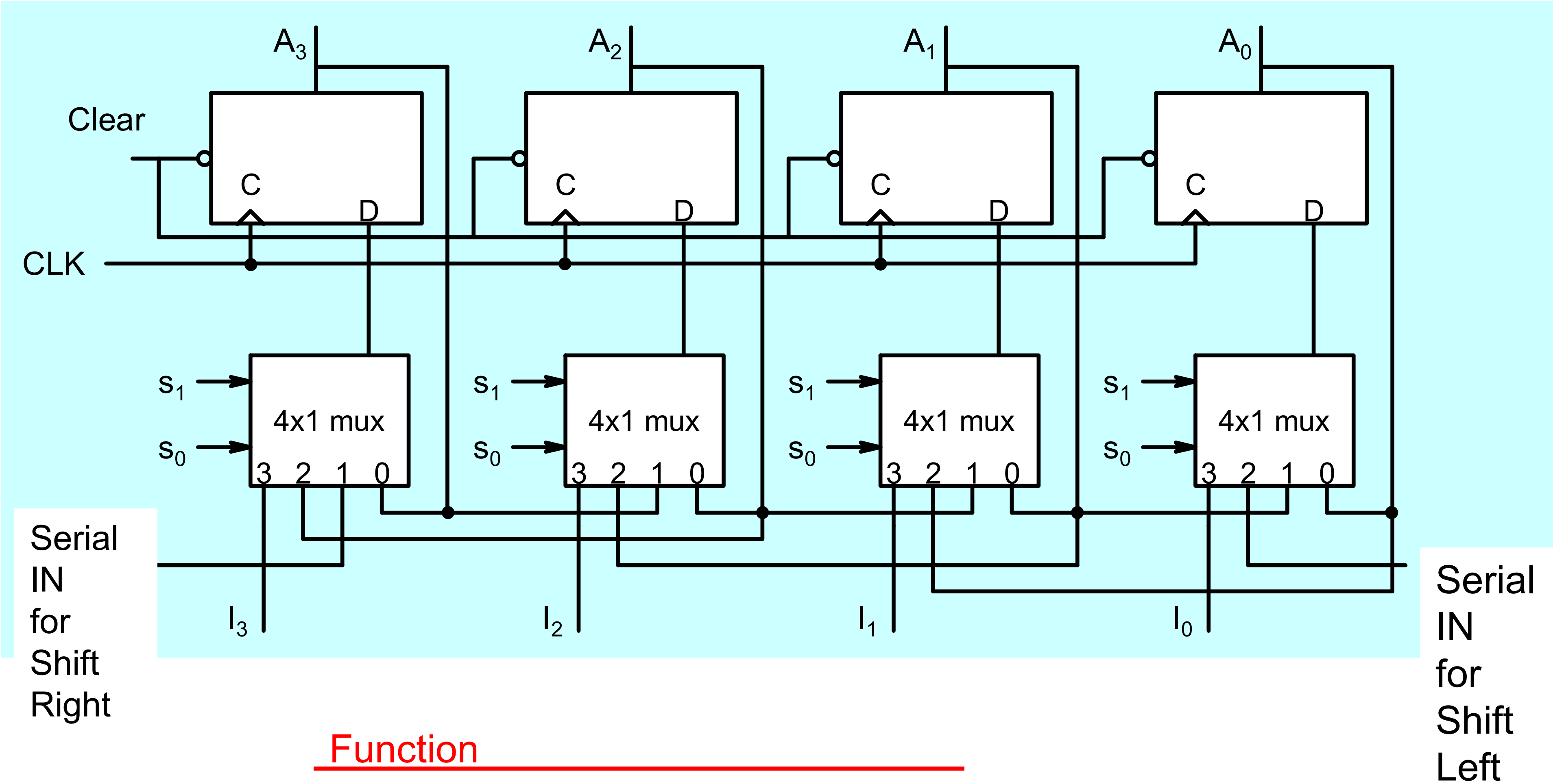


4-bit universal Register



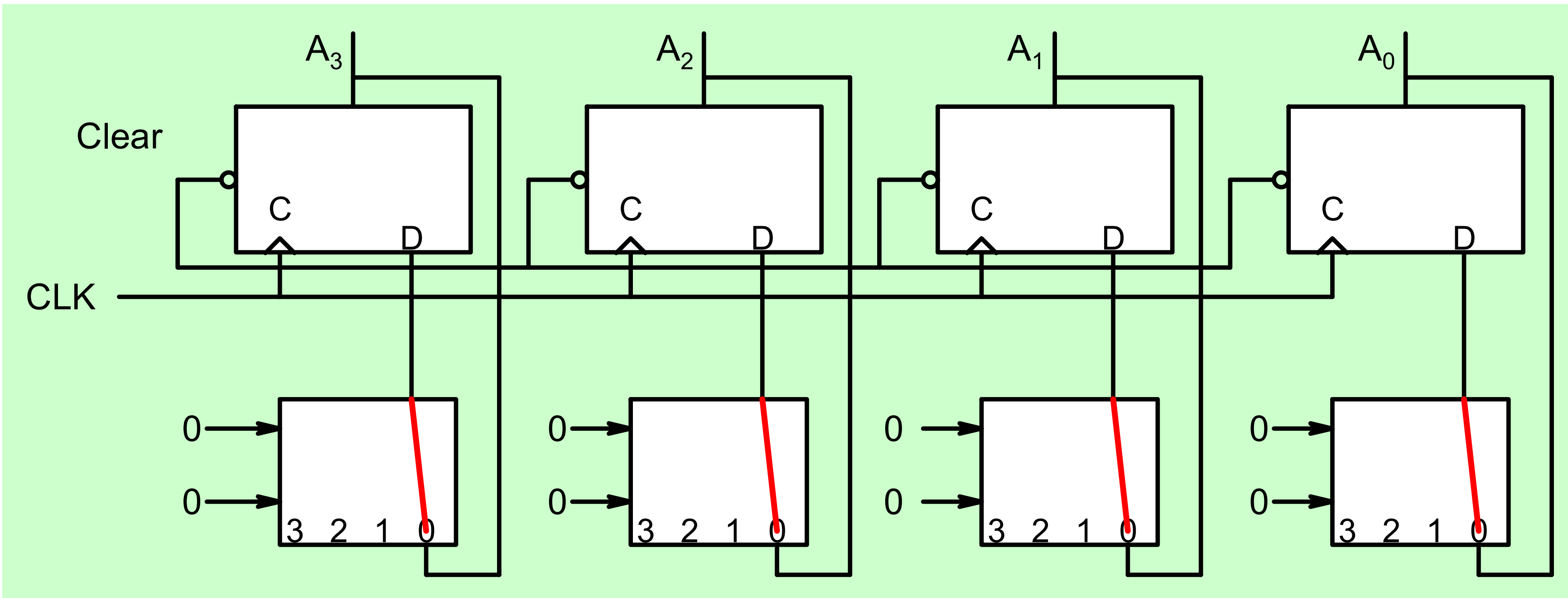
Function

S_1	S_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel Load



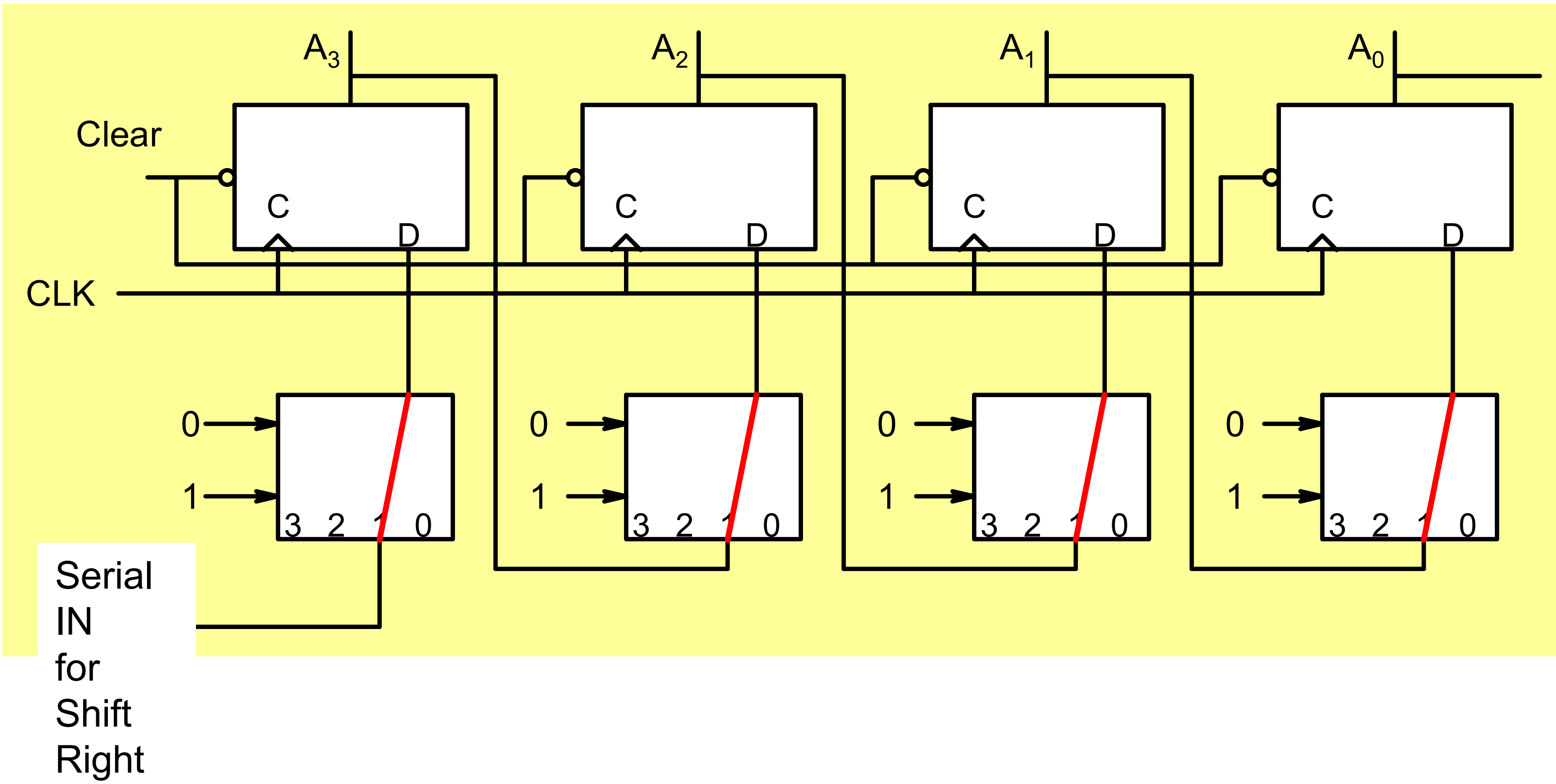
Function

S_1	S_0	Register Operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel Load

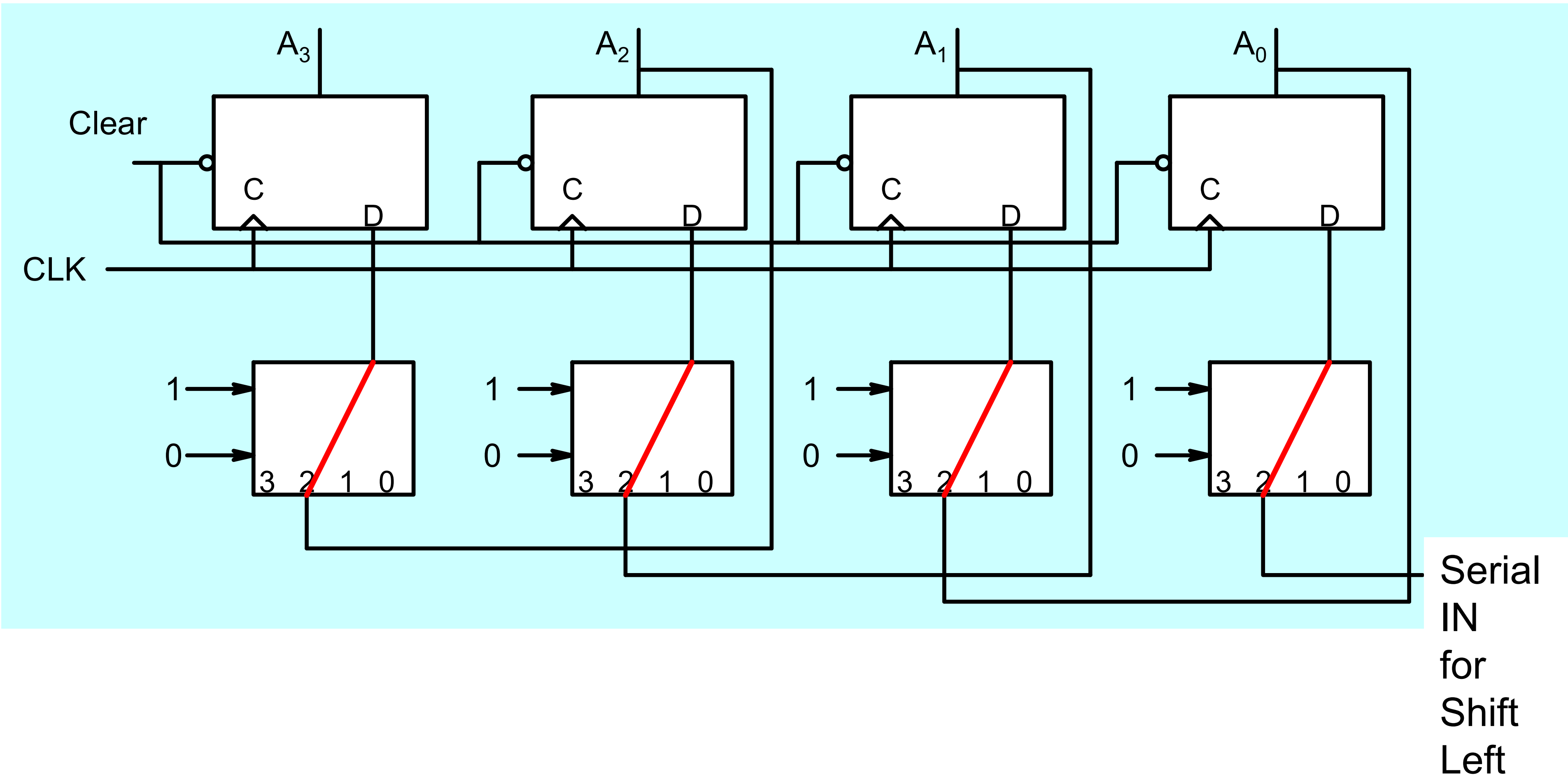


The register maintains its state

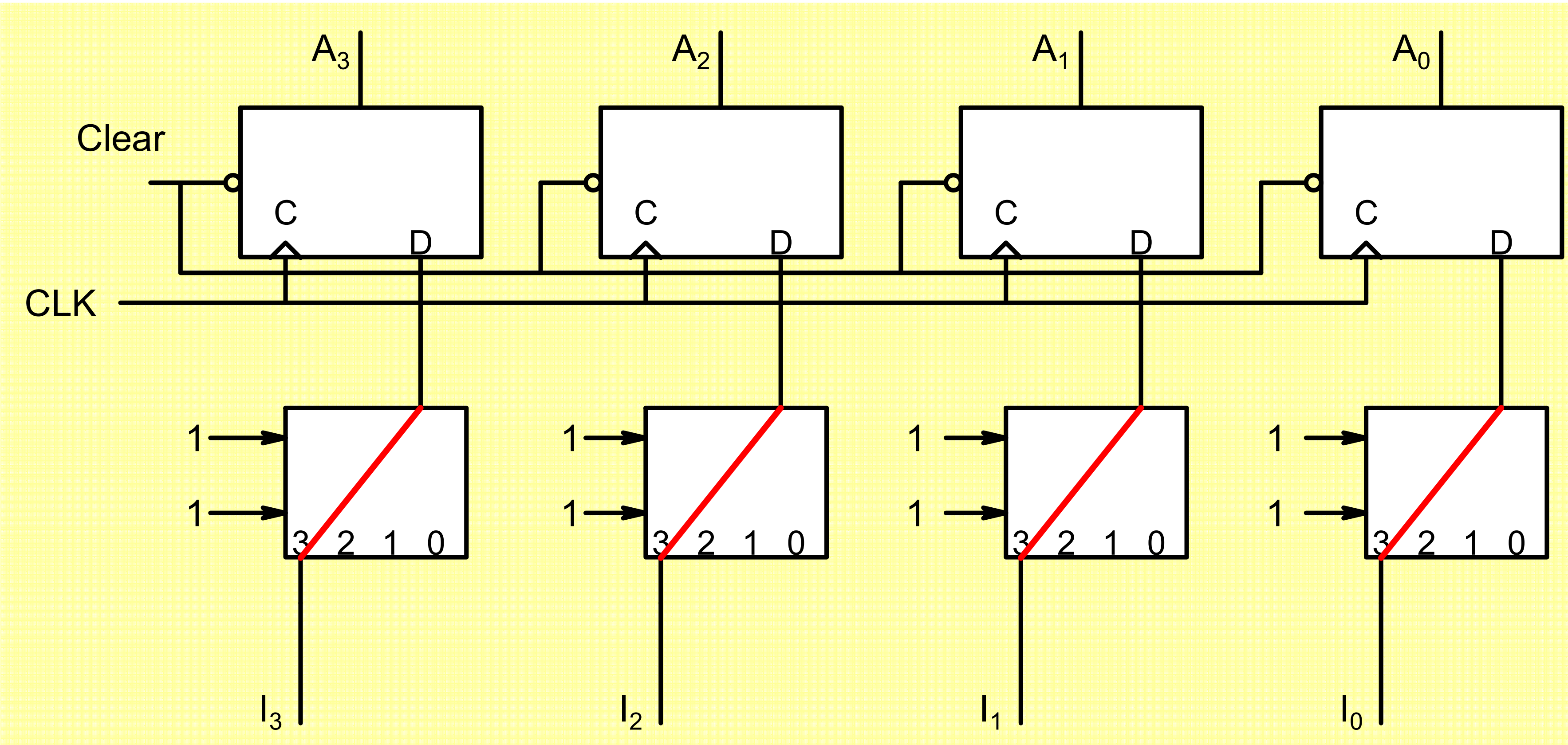
$S_1\ S_0$ = 01: Shift right



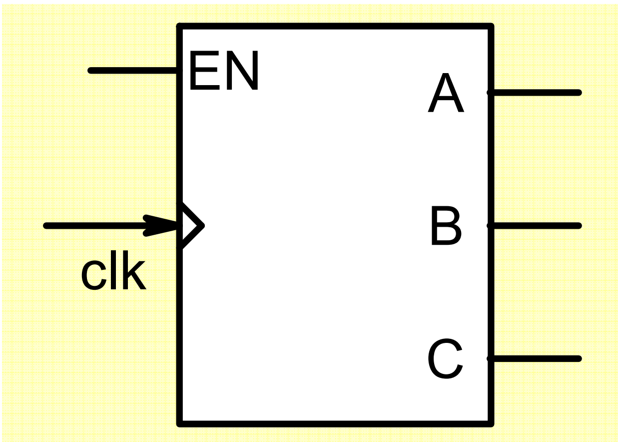
S_1 S_0 = 10: Shift left

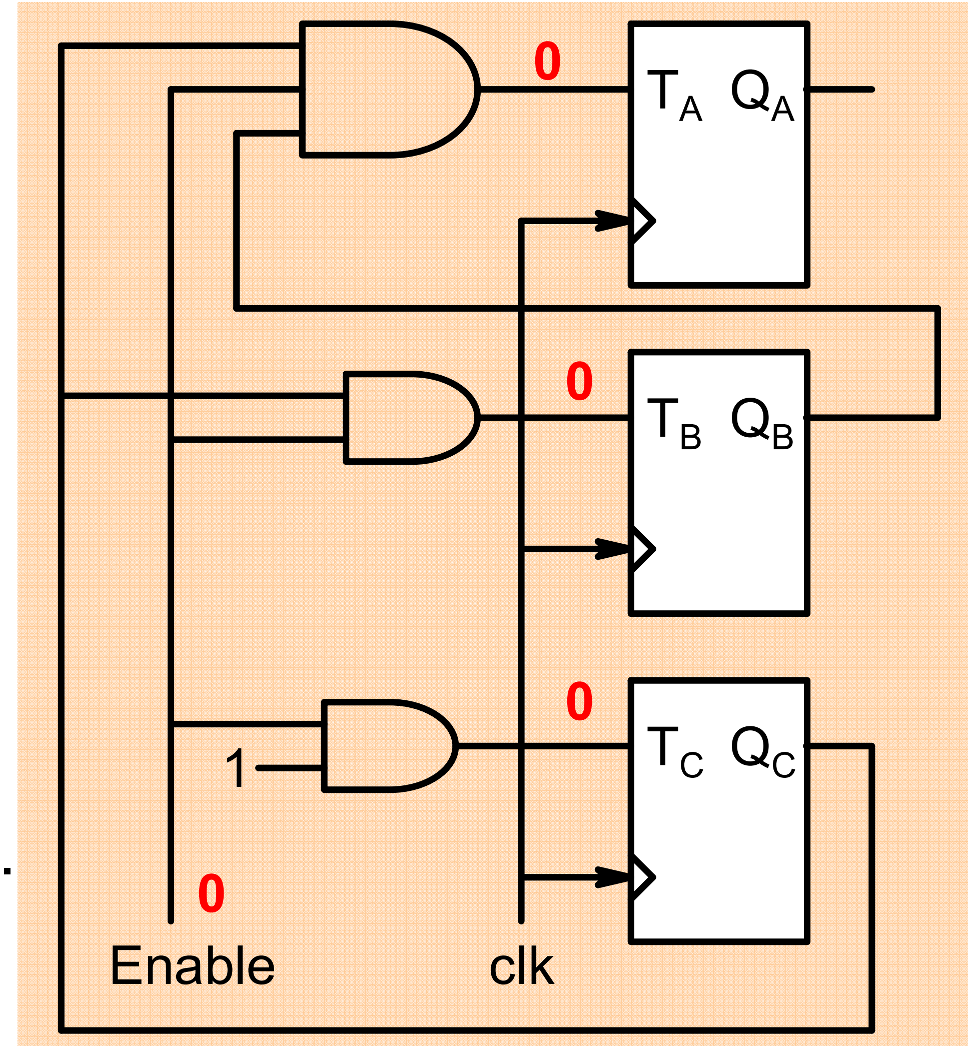


$S_1\ S_0 = 11$: Parallel Load



Counter with Enable



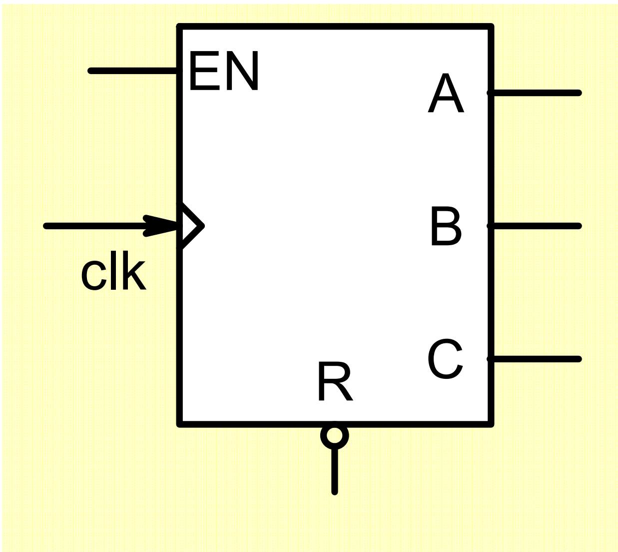


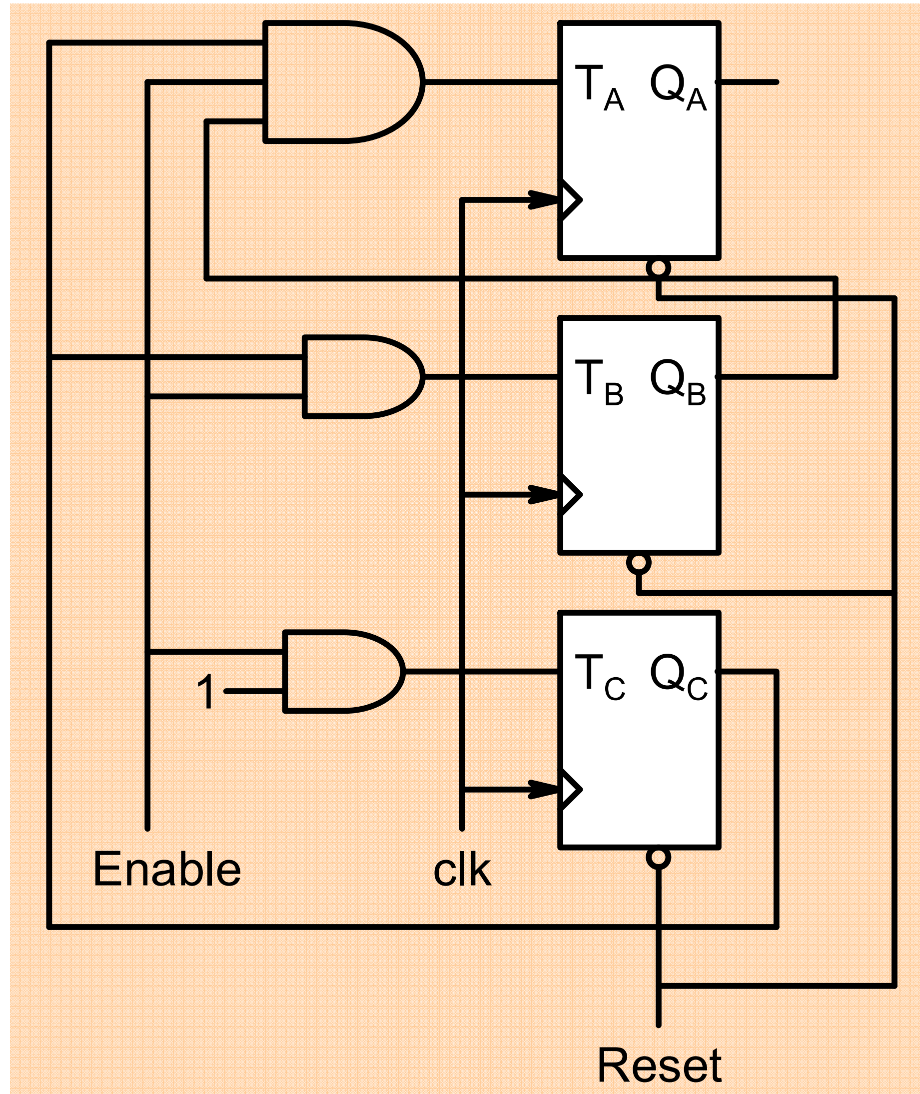
Counter is in Hold state.

$$T_A = B.C \ ; \ T_B = C \ ; \ T_C = 1$$

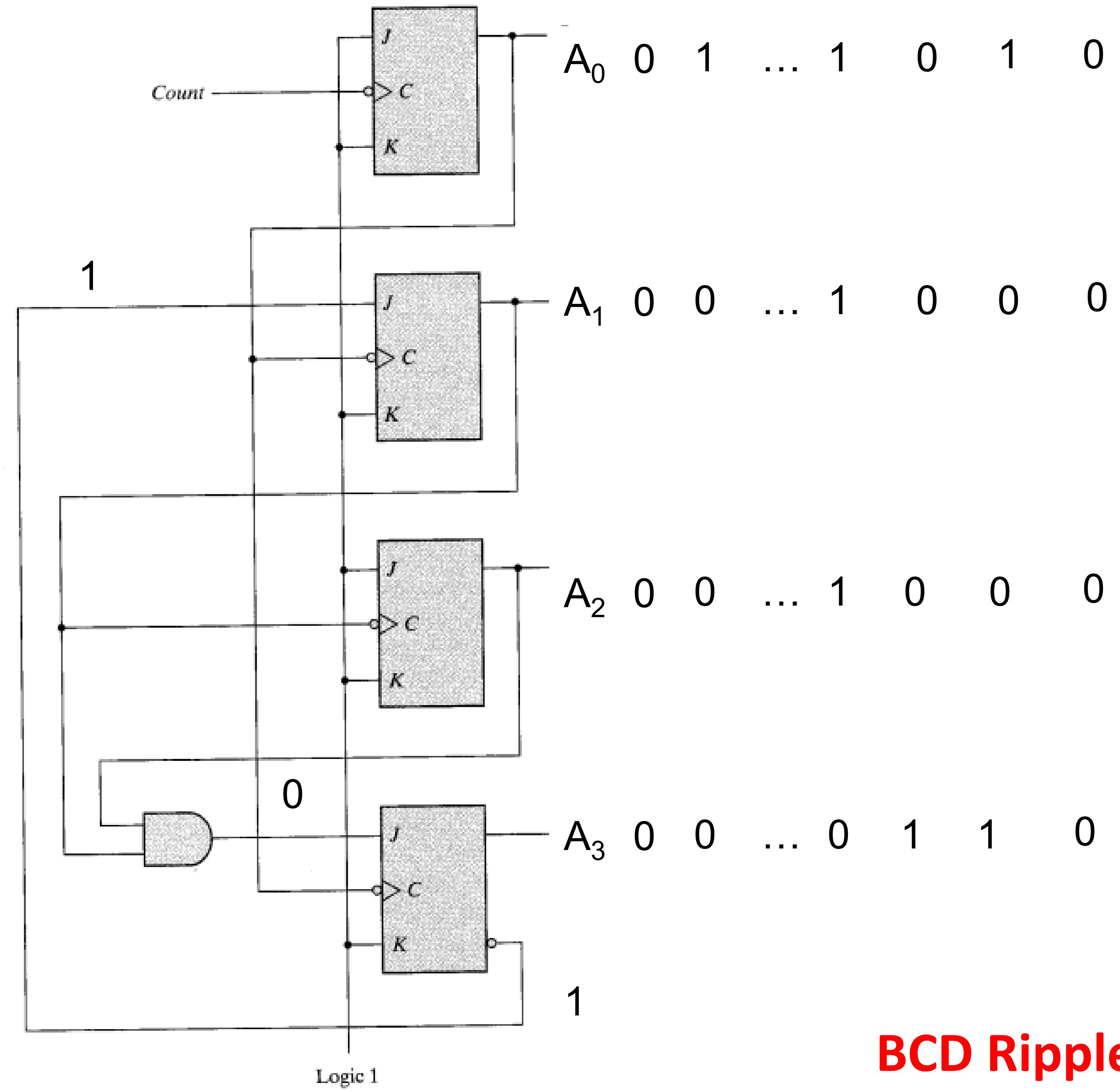
When Enable = 1, the counter begins the count.

Counter with Asynchronous Reset





When Reset = 1, the counter begins the count.



BCD Ripple Counter