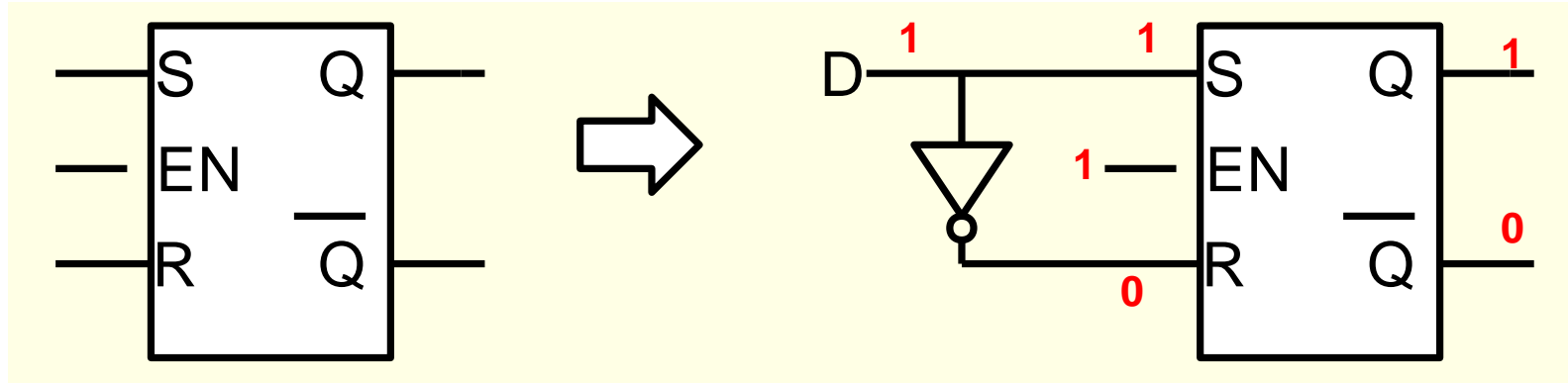
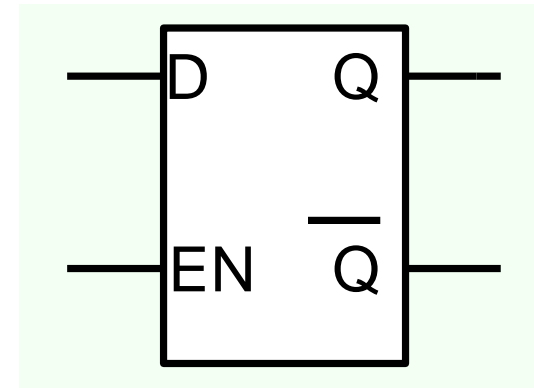


D latch

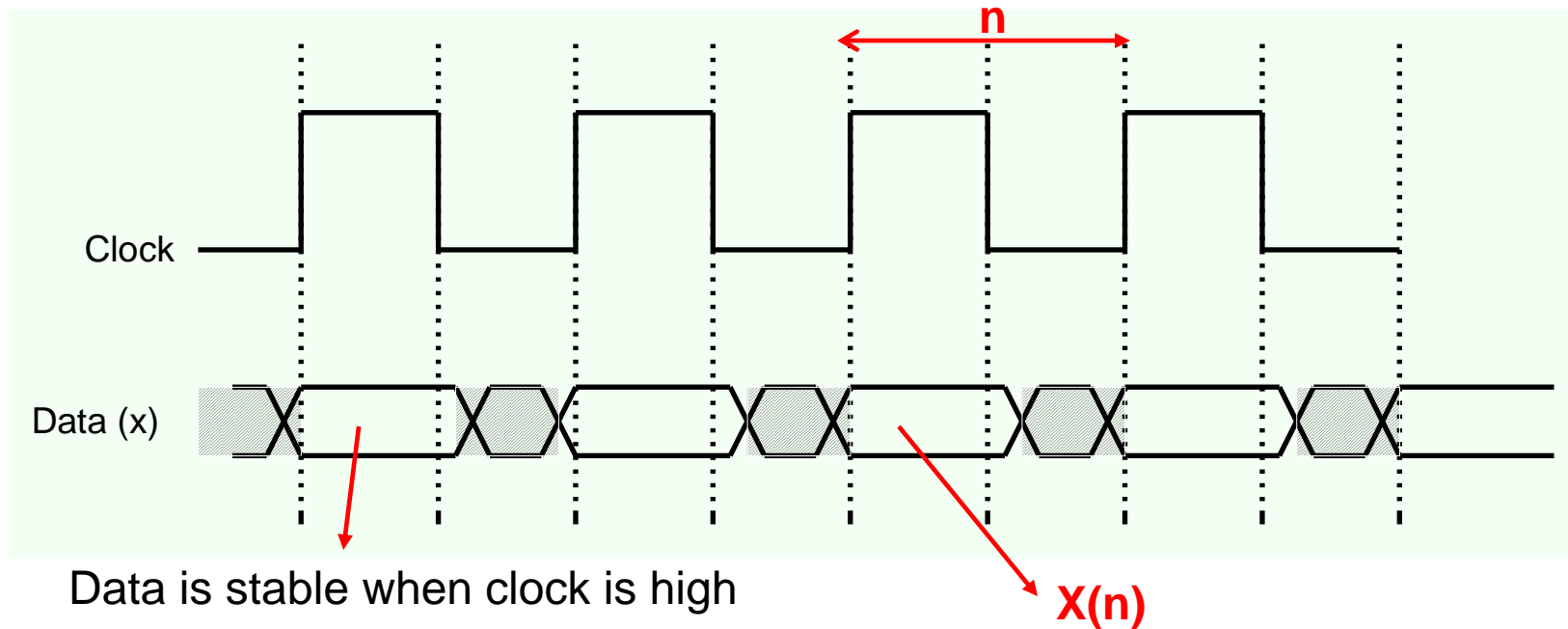
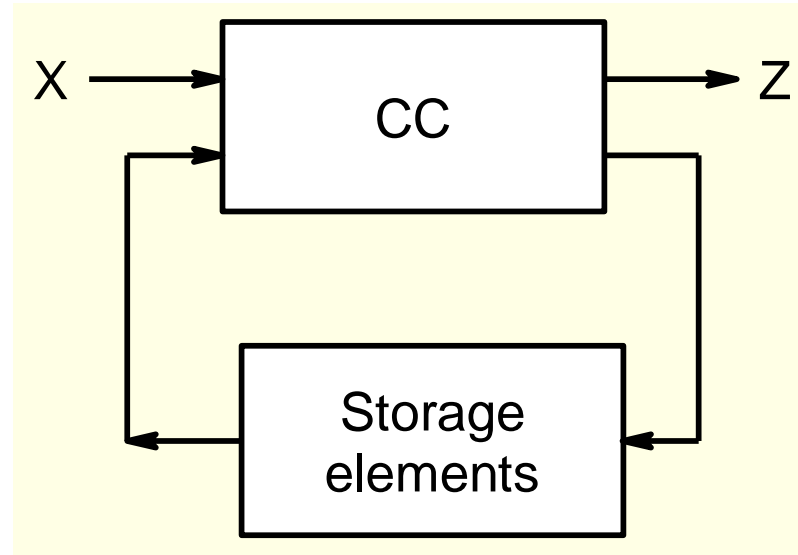


Enable	S	R	Q	\overline{Q}	State
0	x	x	Q	\overline{Q}	Hold
1	1	0	1	0	Set
1	0	1	0	1	Reset
1	0	0	Q	\overline{Q}	Hold
1	1	1	0	0	Invalid

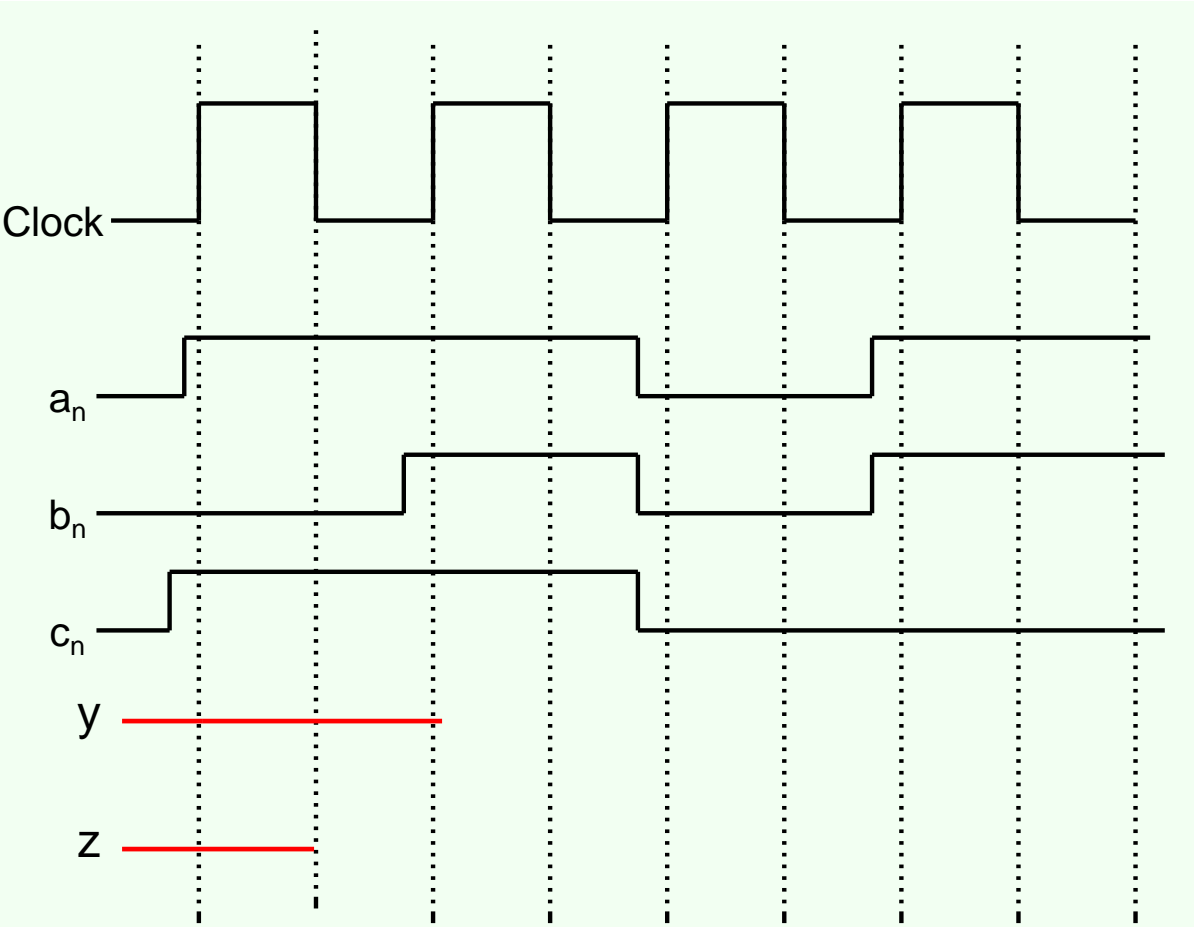
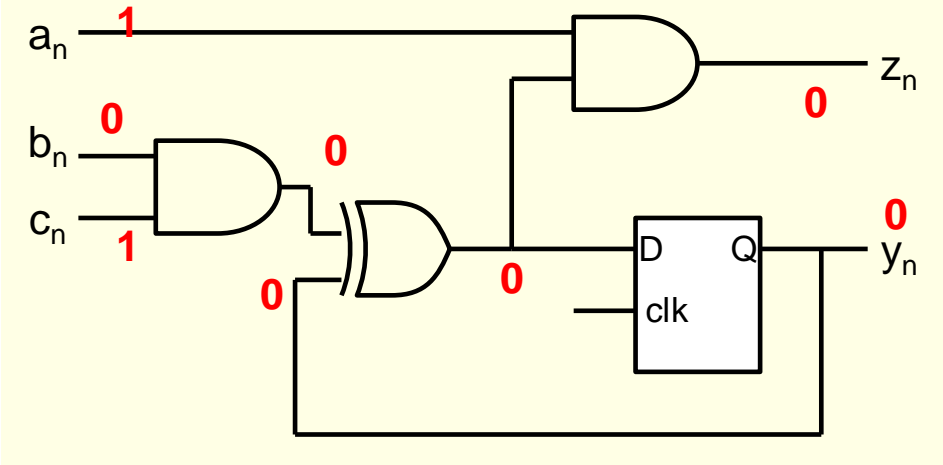


If $EN = 1$ then $Q = D$ otherwise the latch is in Hold state

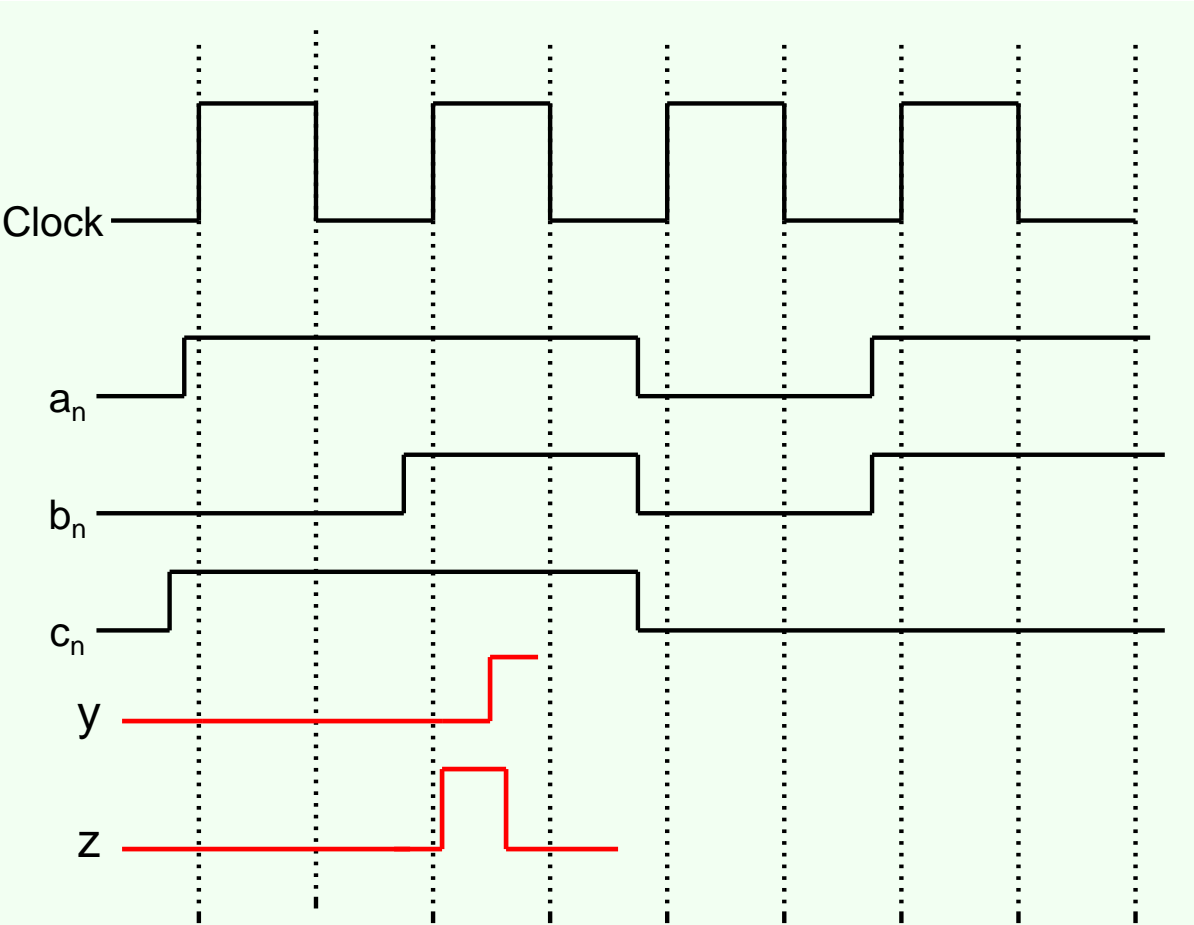
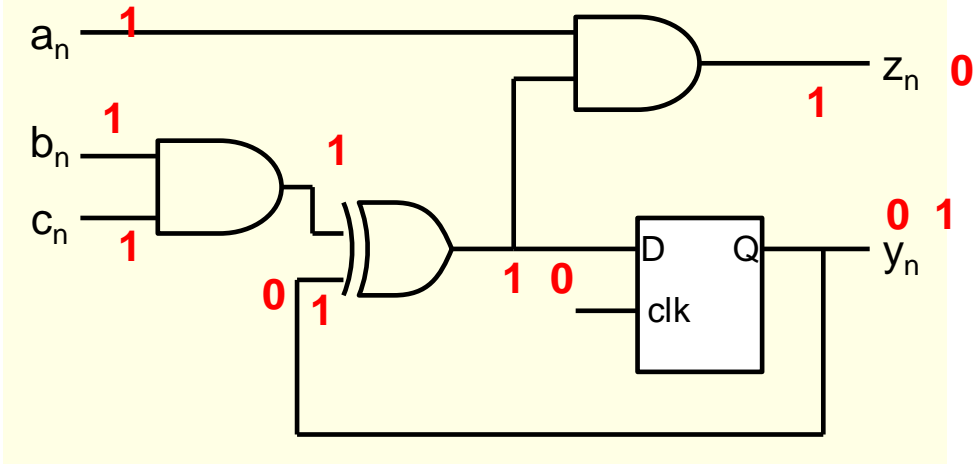
Synchronous Sequential Circuits



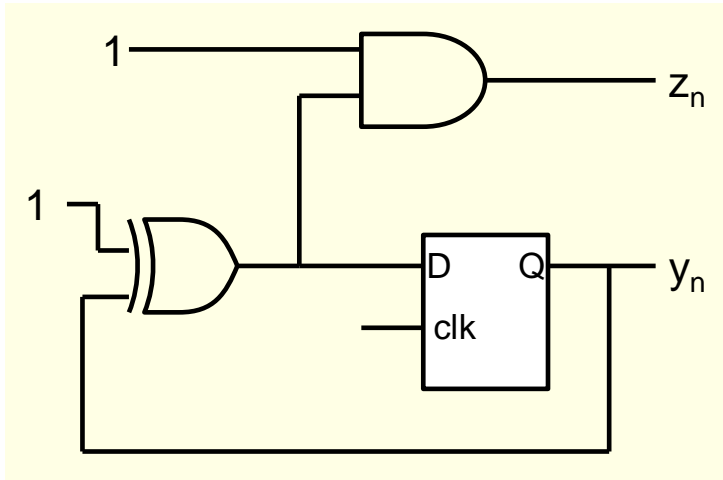
Example



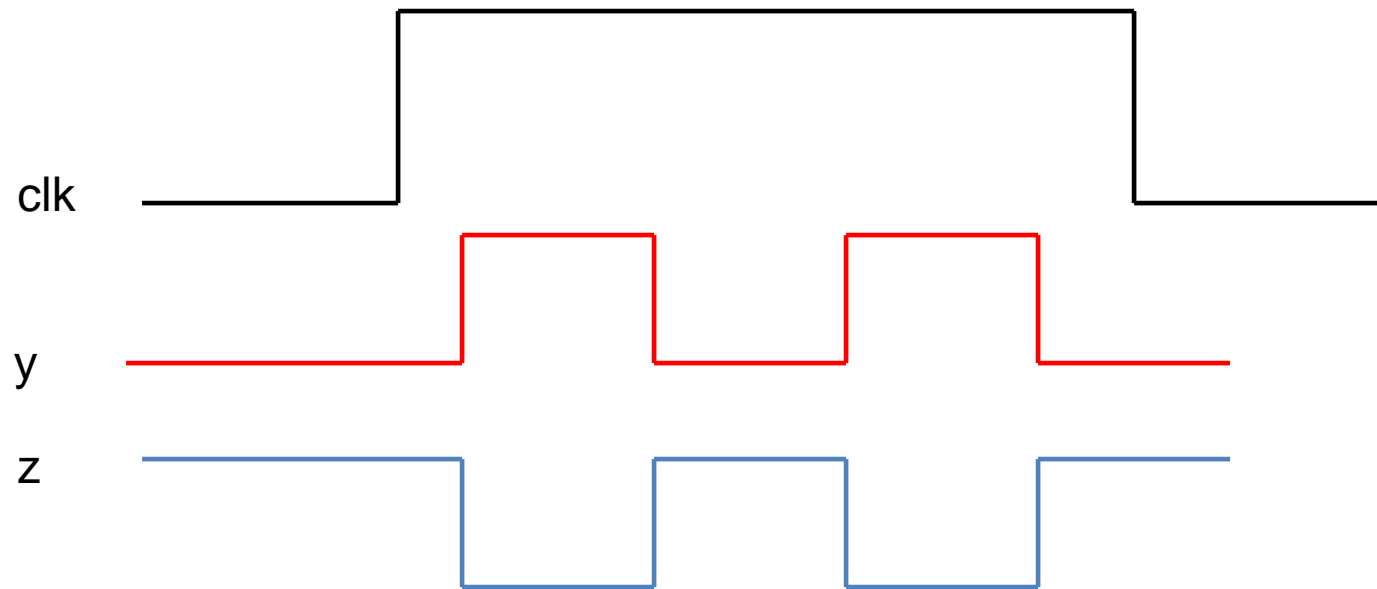
Example



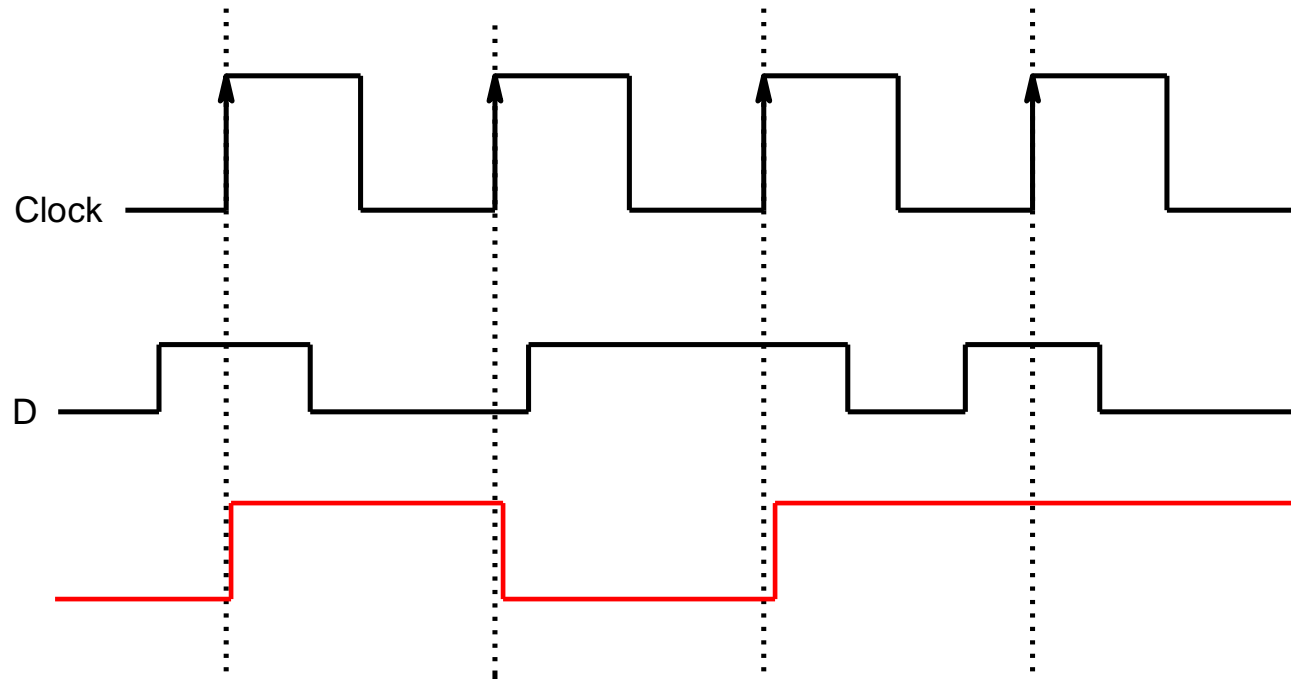
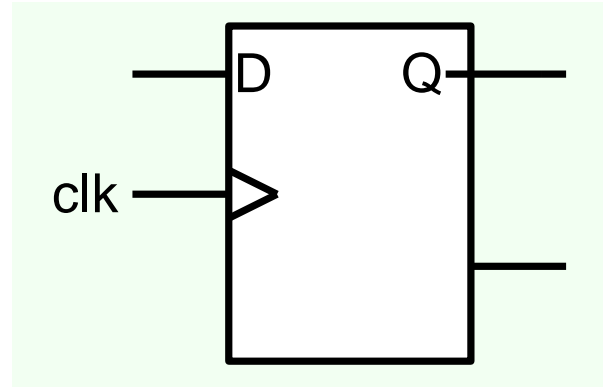
Problem with Latch



Circuits are designed with the idea there would be single change in output or memory state in single clock cycle.

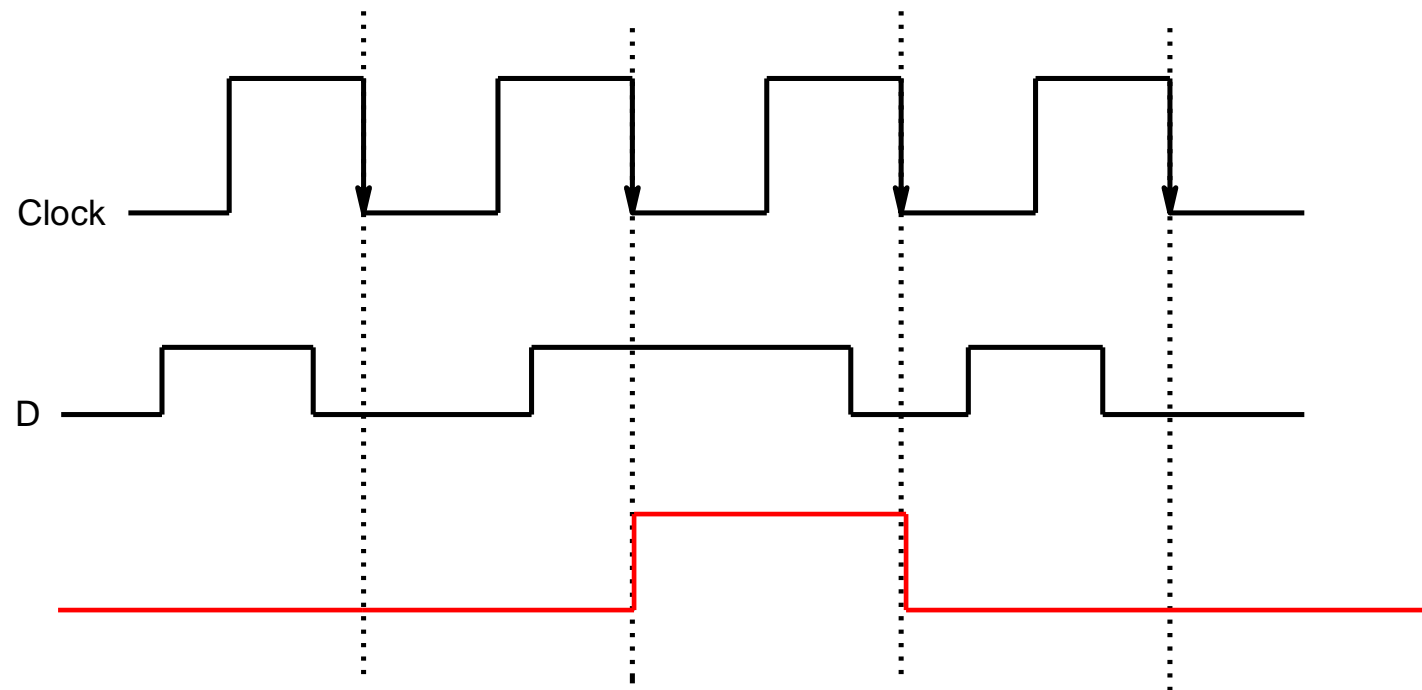
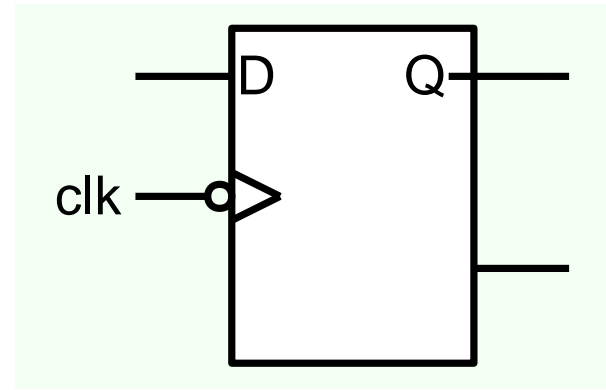


Edge Triggered Latch or Flip-flop

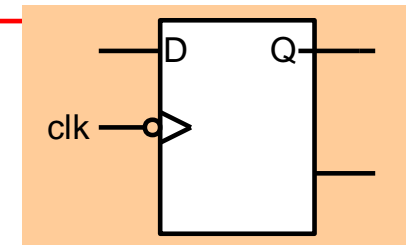
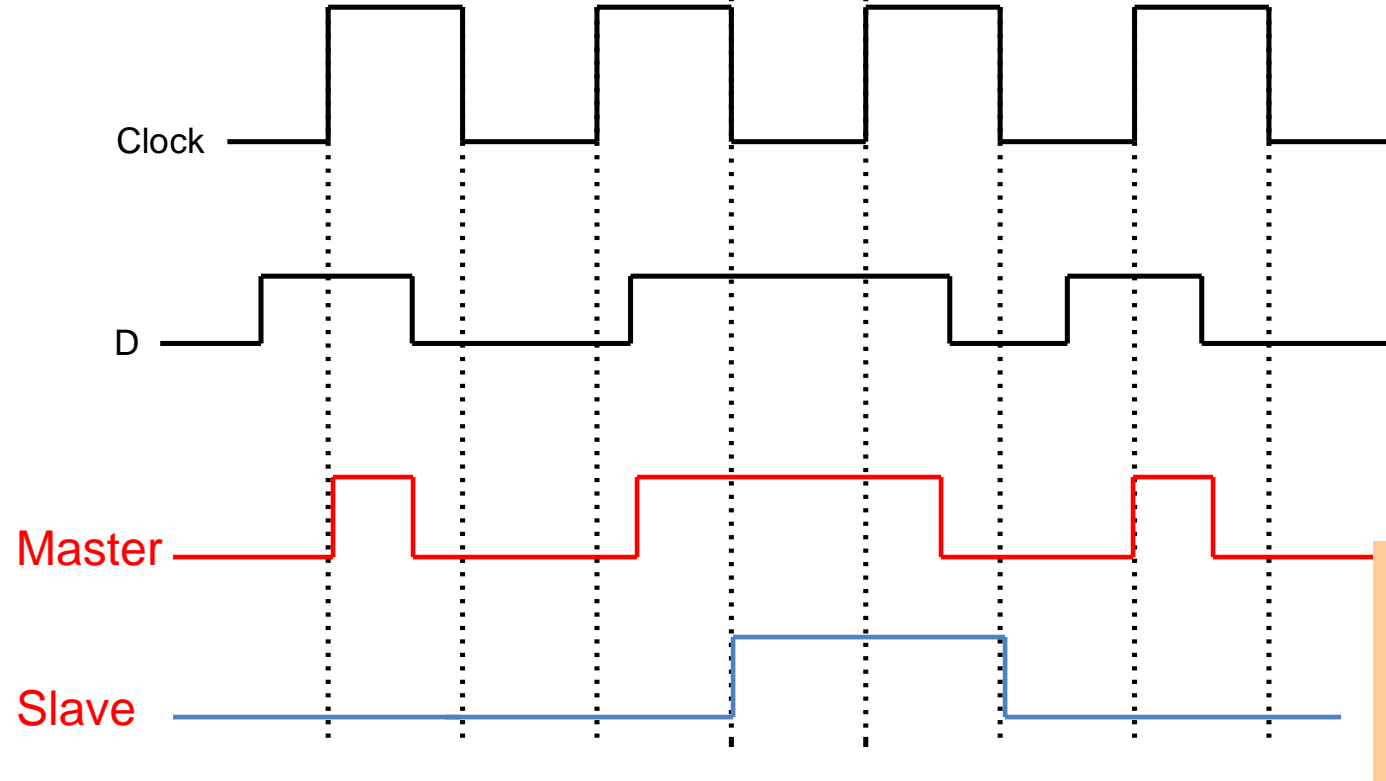
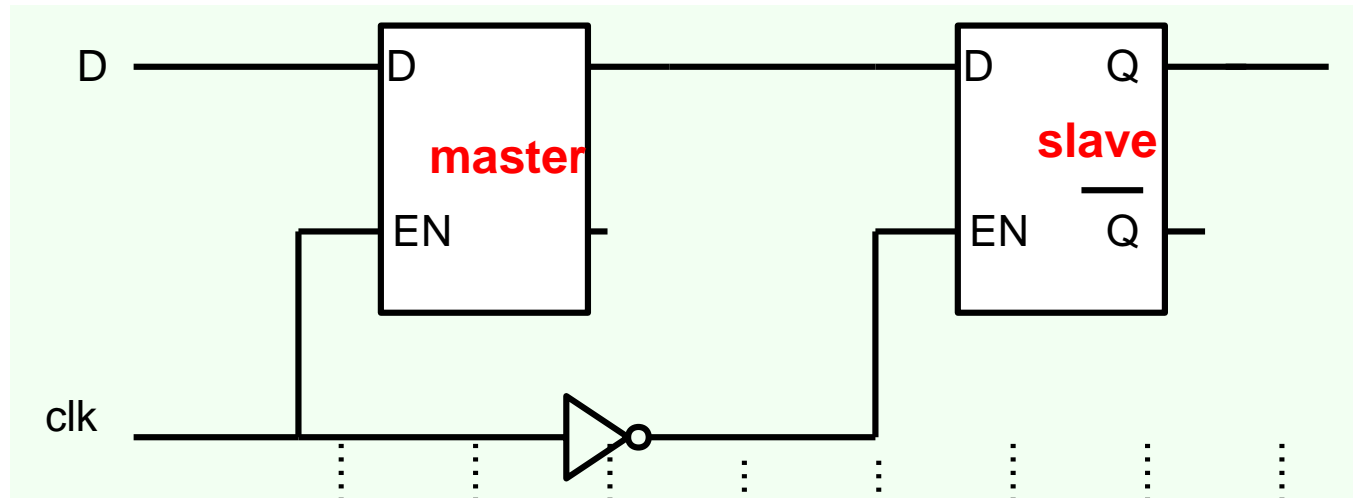


Positive edge triggered flipflop

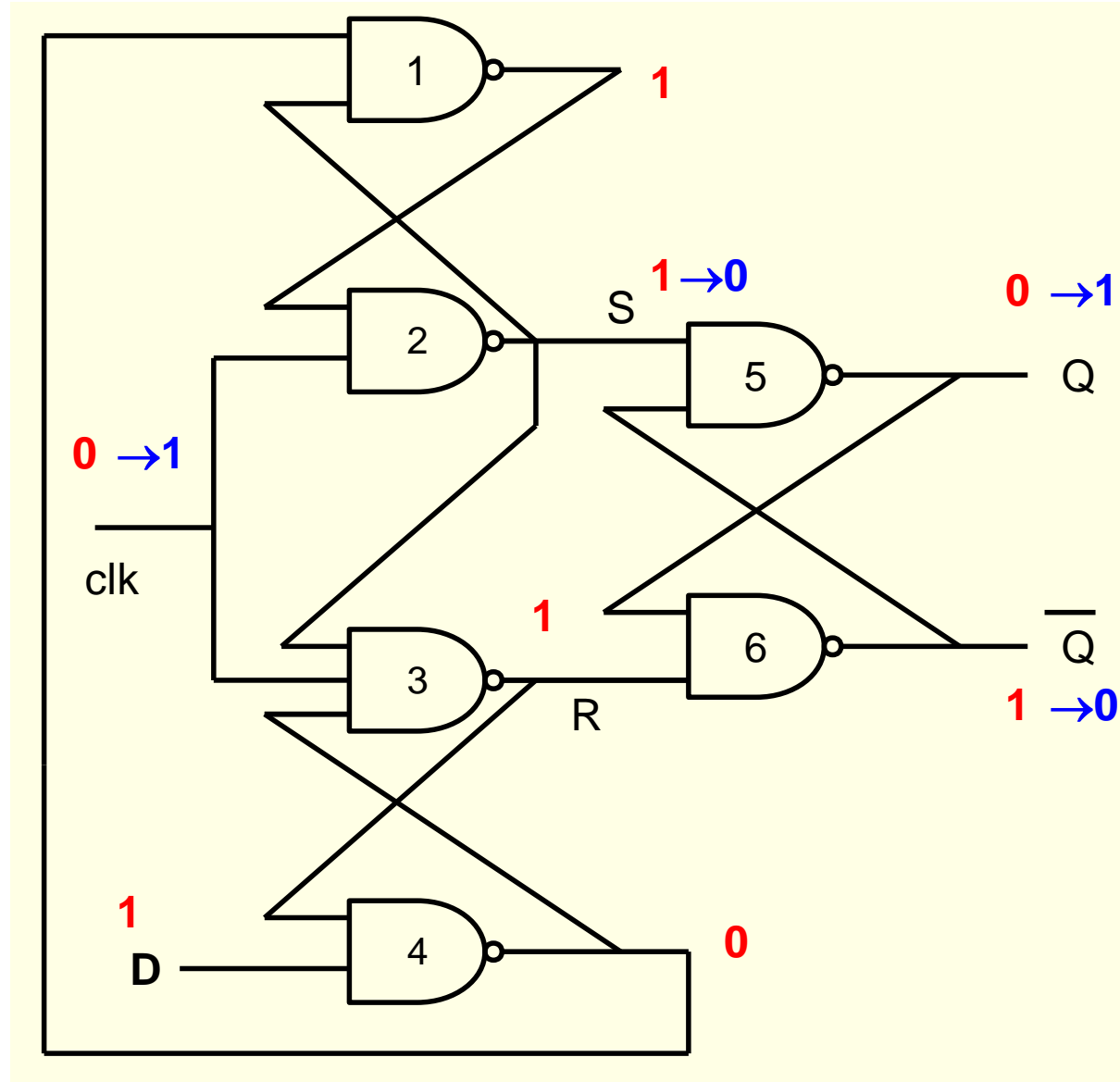
Negative Edge Triggered Latch or Flip-flop



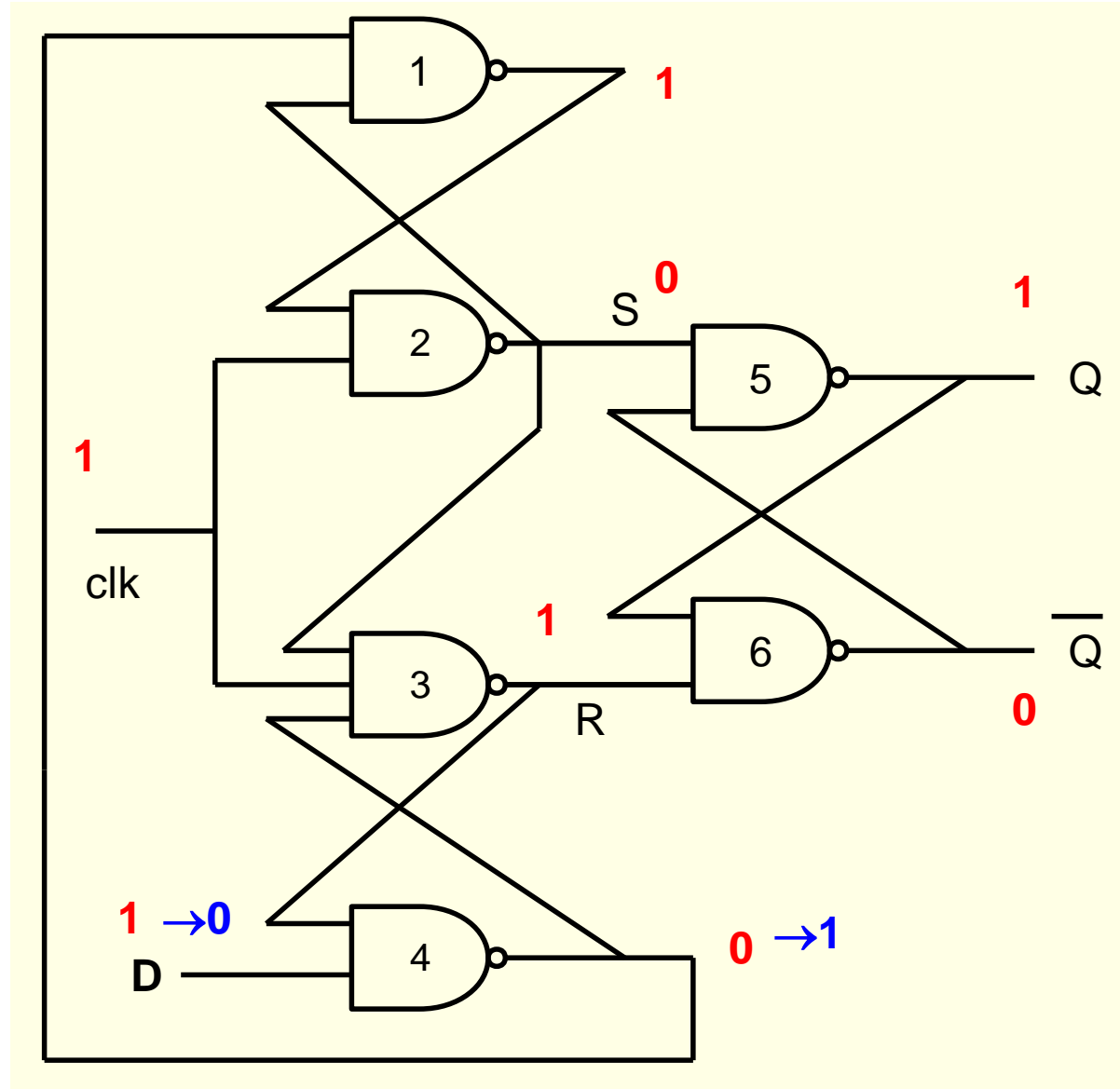
Master-Slave D Flip-flop



Positive edge triggered Flip-flop



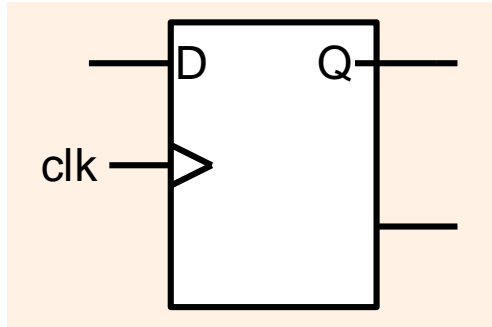
Positive edge triggered Flip-flop



A change in input has no effect if it occurs after the clock edge

Characteristic table

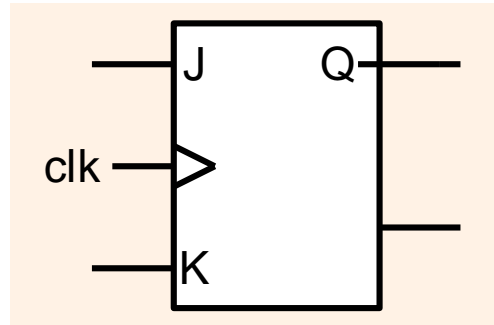
Given a input and the present state of the flip-flop, what is the next state of the flip-flop



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

$$Q(t+1) = D$$

JK Flip-flop

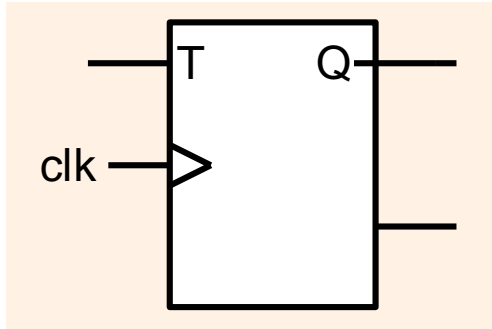


Inputs J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

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

→ Characteristic equation

Toggle or T Flip-flop



Inputs (T)	Q(t+1)
0	Q(t)
1	$\overline{Q(t)}$

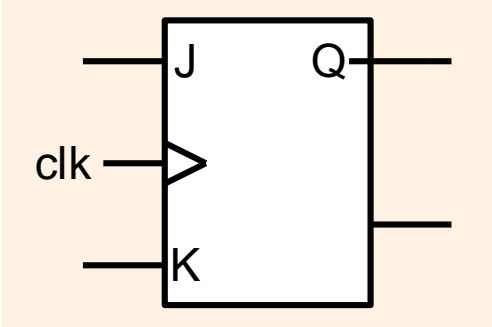
$$Q(t+1) = \overline{Q(t)}.T + Q(t).\overline{T}$$

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

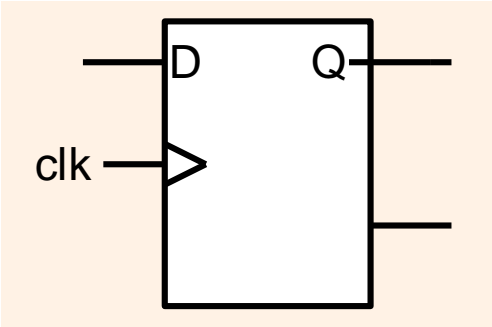
Excitation Table



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

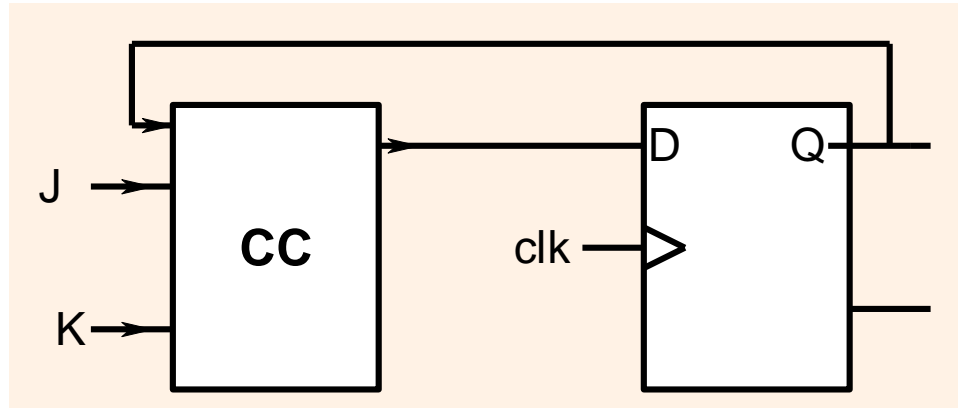


D	Q(t+1)
0	0
1	1

Inputs

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

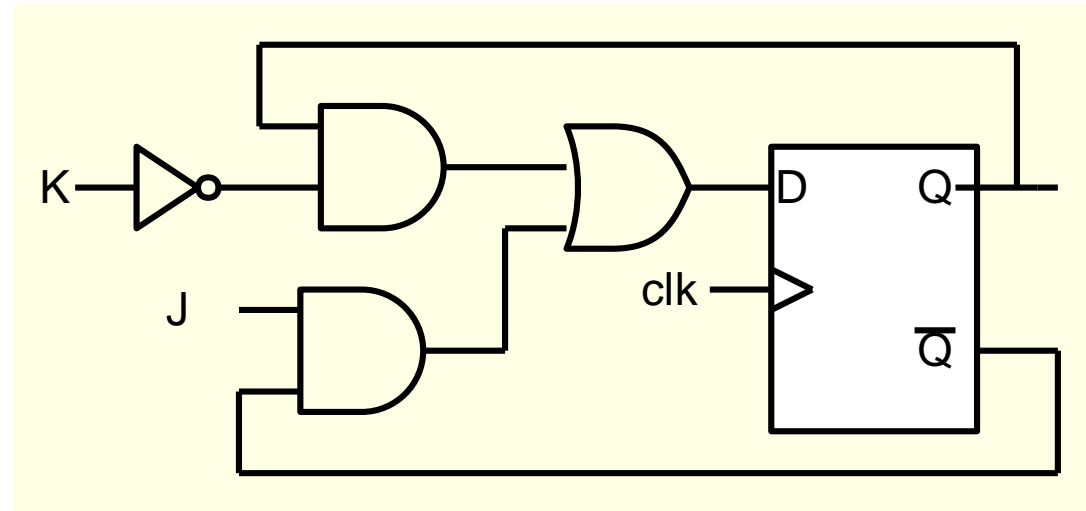
Convert a D FF to JK FF



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

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

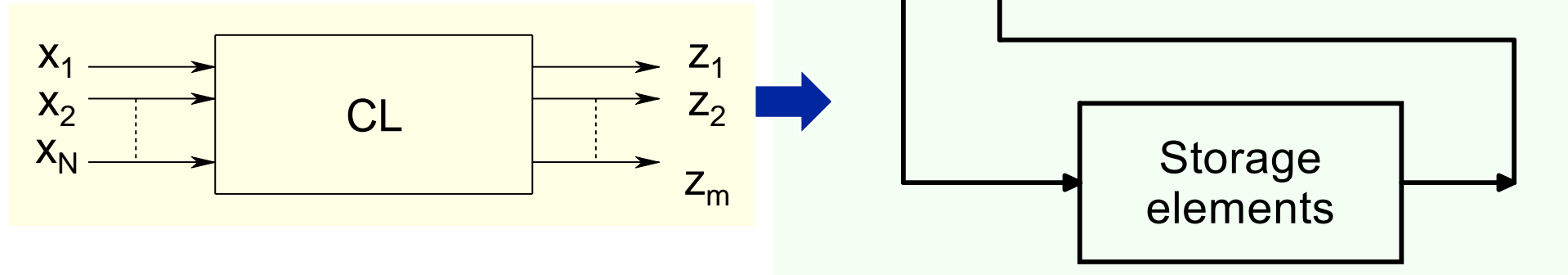
$$D = \overline{Q}.J + Q.\overline{K}$$



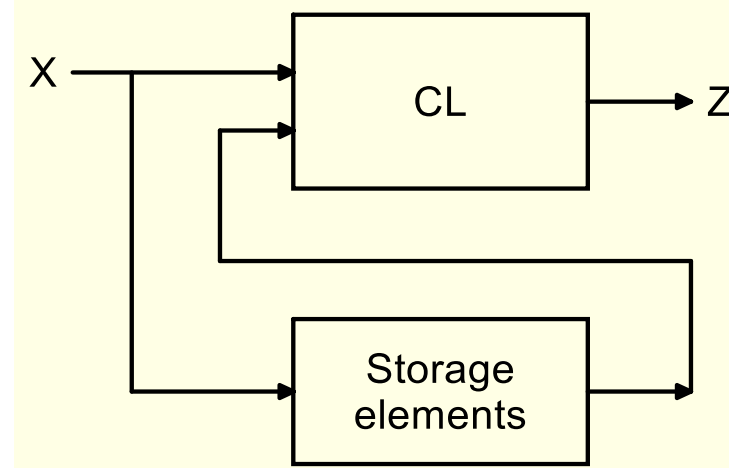
Limitations of Combinational logic:

decisions can only be based on the present value of inputs

- A more general purpose decision making machine should be able to make decisions based on past values of inputs as well.



Limitations:



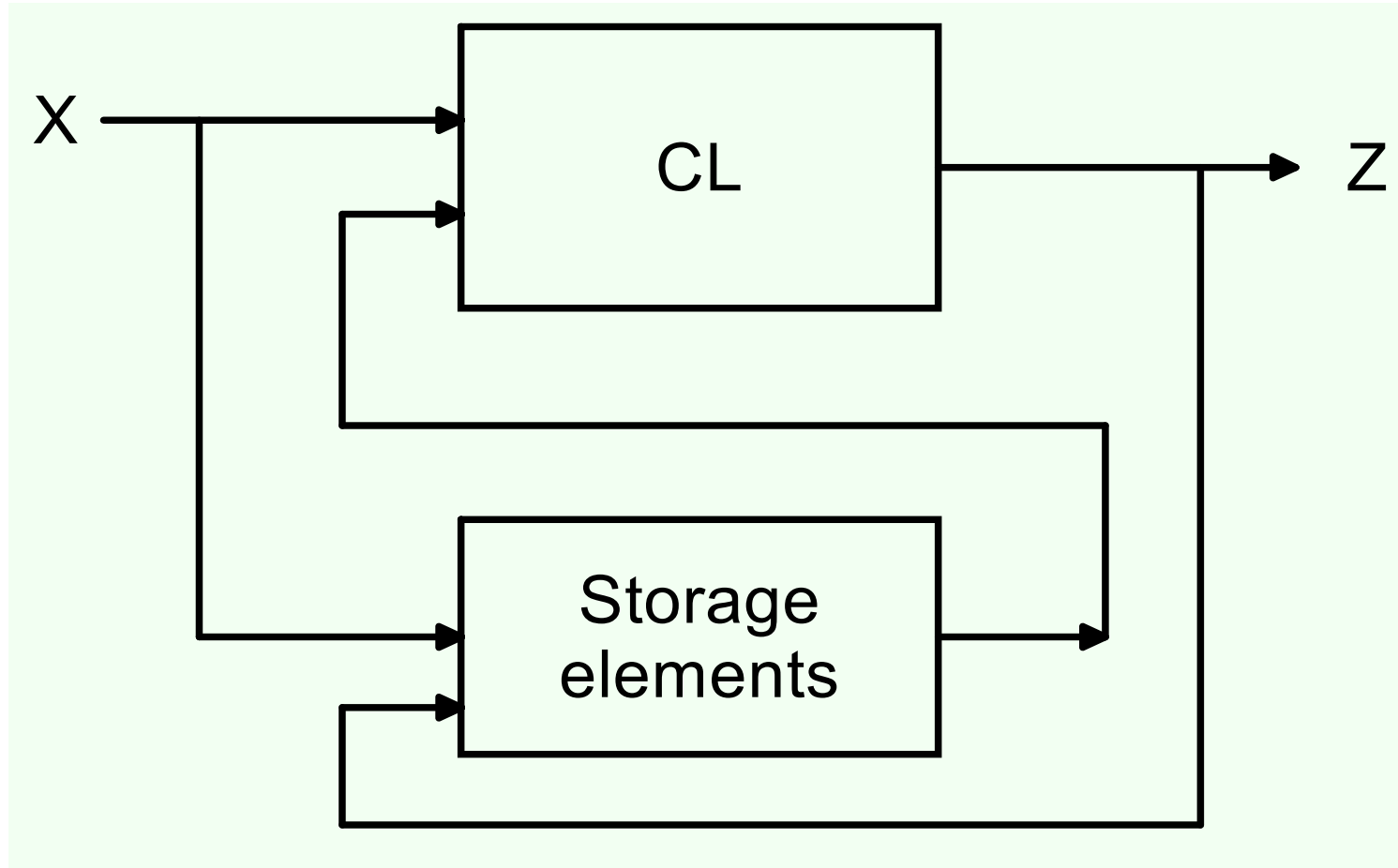
$$z[n] = x[n] \text{ .OR. } z[n-1]$$

$$z[n] = x[n] \text{ .OR. } x[n-1] \text{ .OR. } x[n-2] \text{ } x[0]$$

Requires infinite memory !

⇒ Make provision for storage of past values of Outputs as well

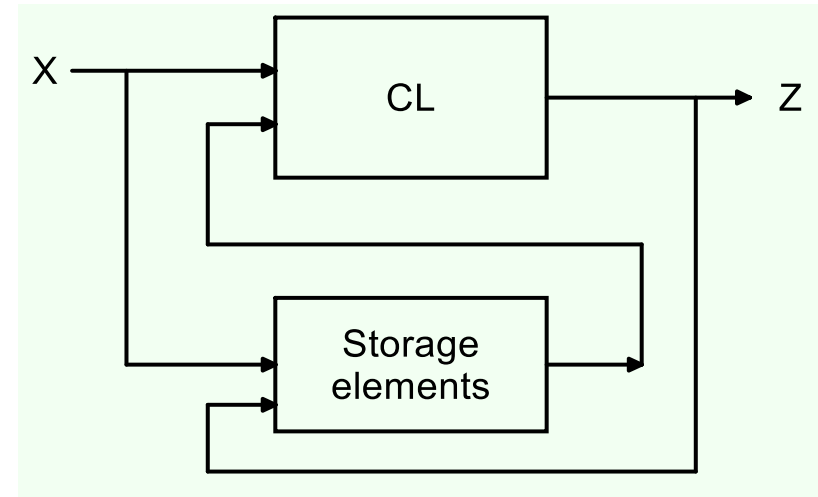
Improved System:



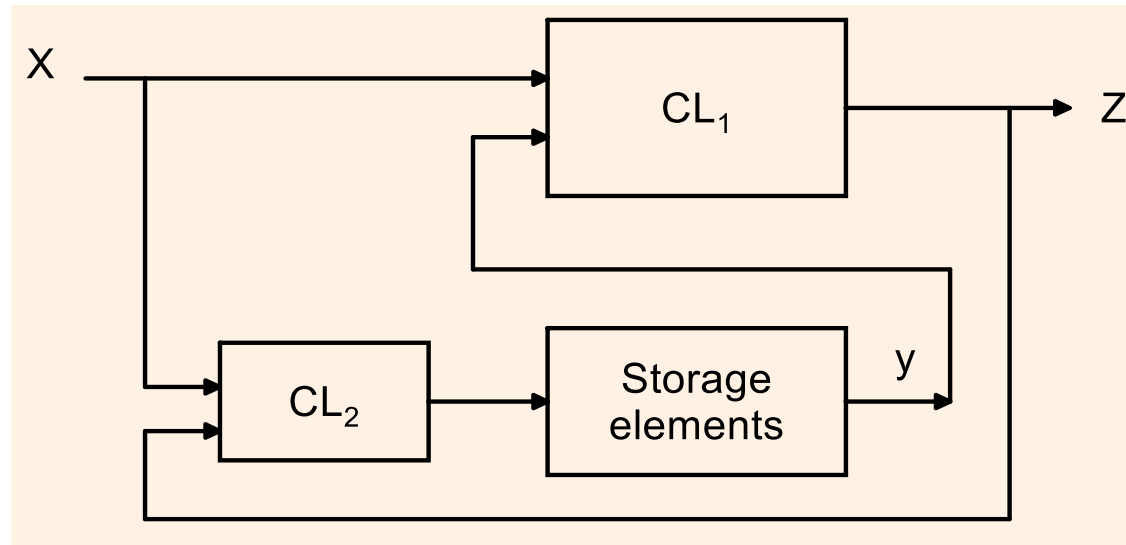
Limitations:

$$z[n] = x[n] \text{ .OR. } (x[n-1] \text{ .AND. } z[n-1])$$

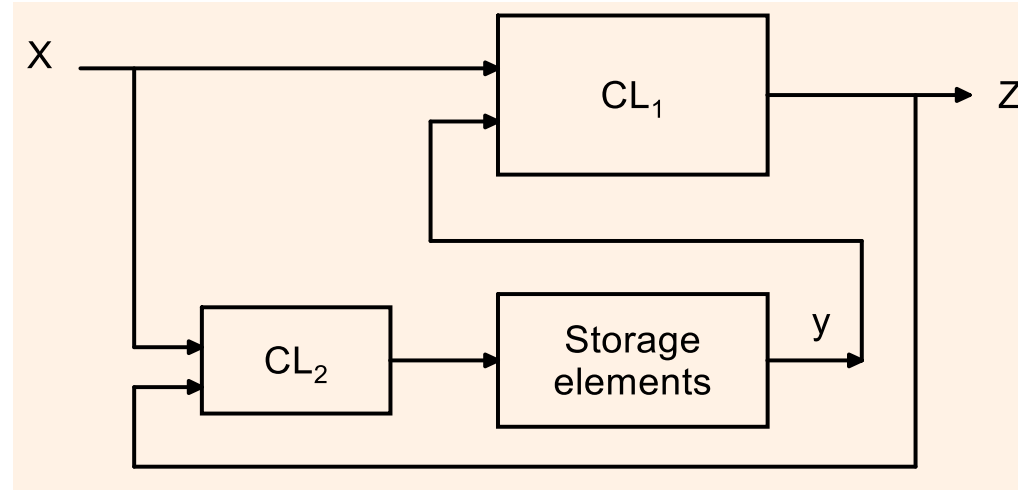
Requires two storage elements



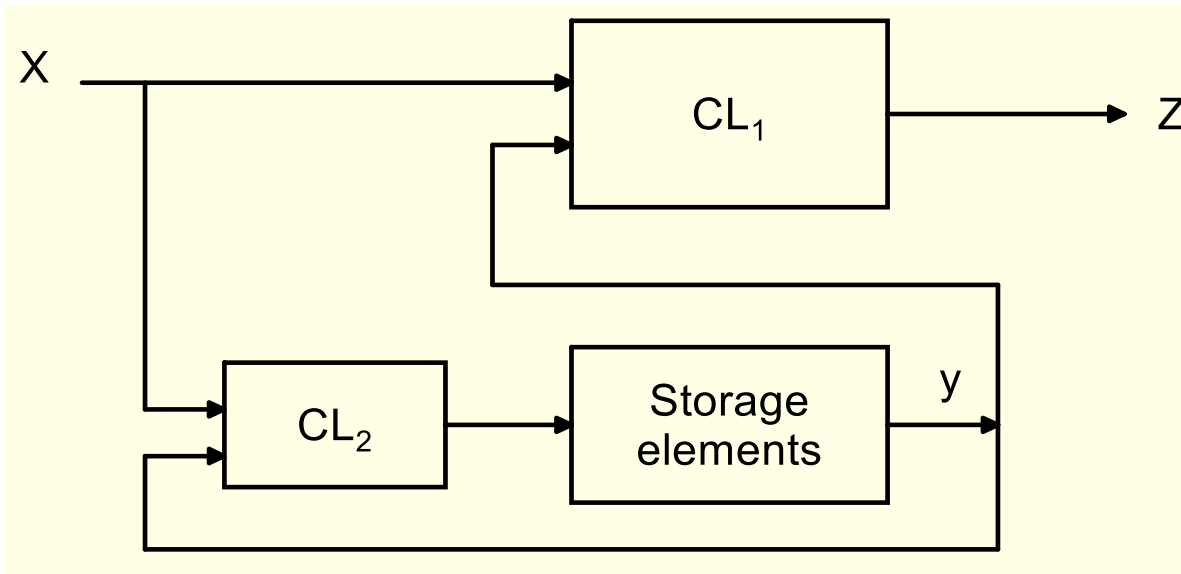
However, by defining a new variable $y[n] = x[n-1].\text{AND.} z[n-1]$, we can use only one storage element.

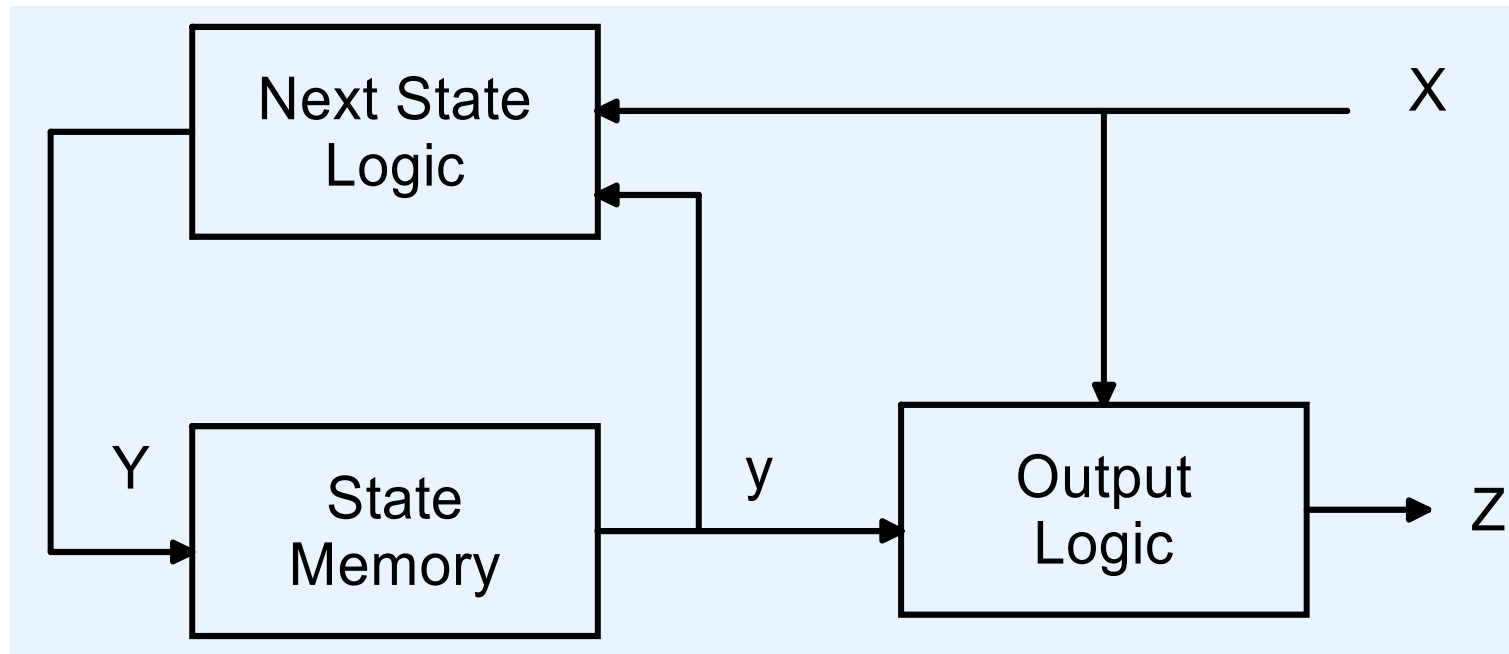
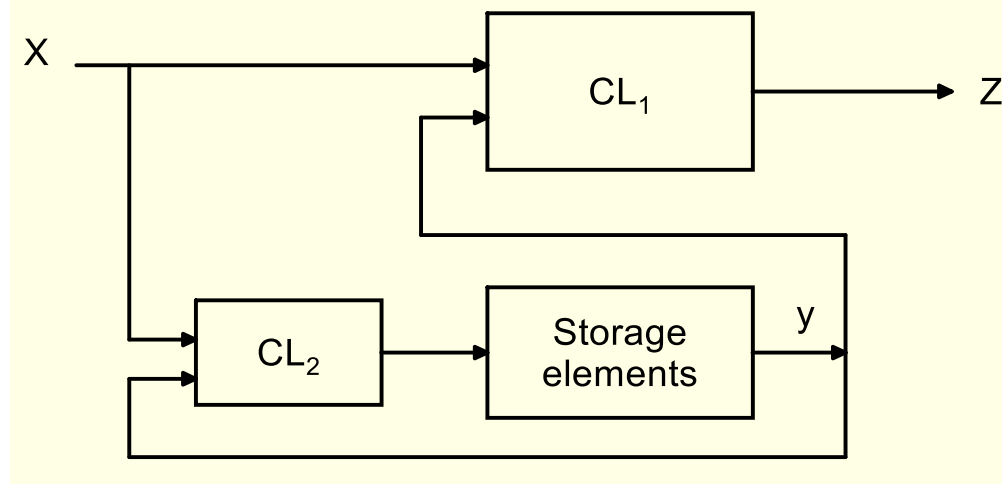


Improved Decision Making Machine:



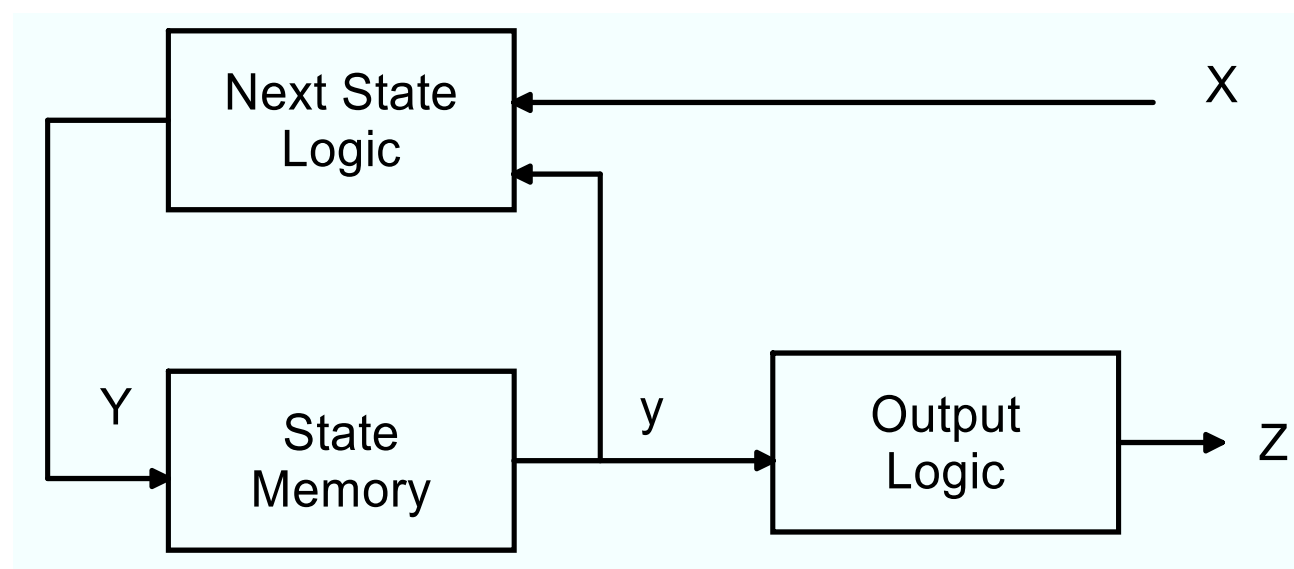
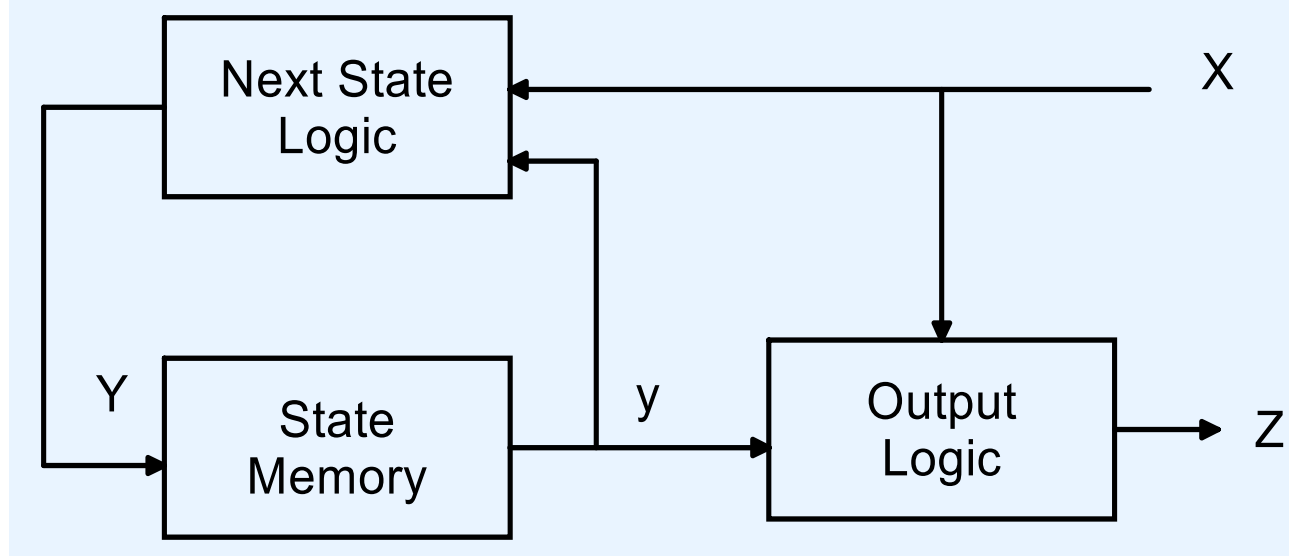
Since Output Z is a function of Y and X





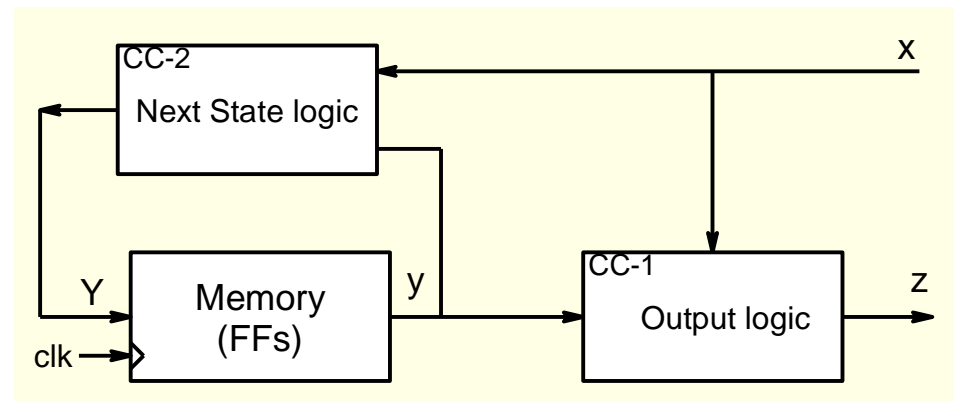
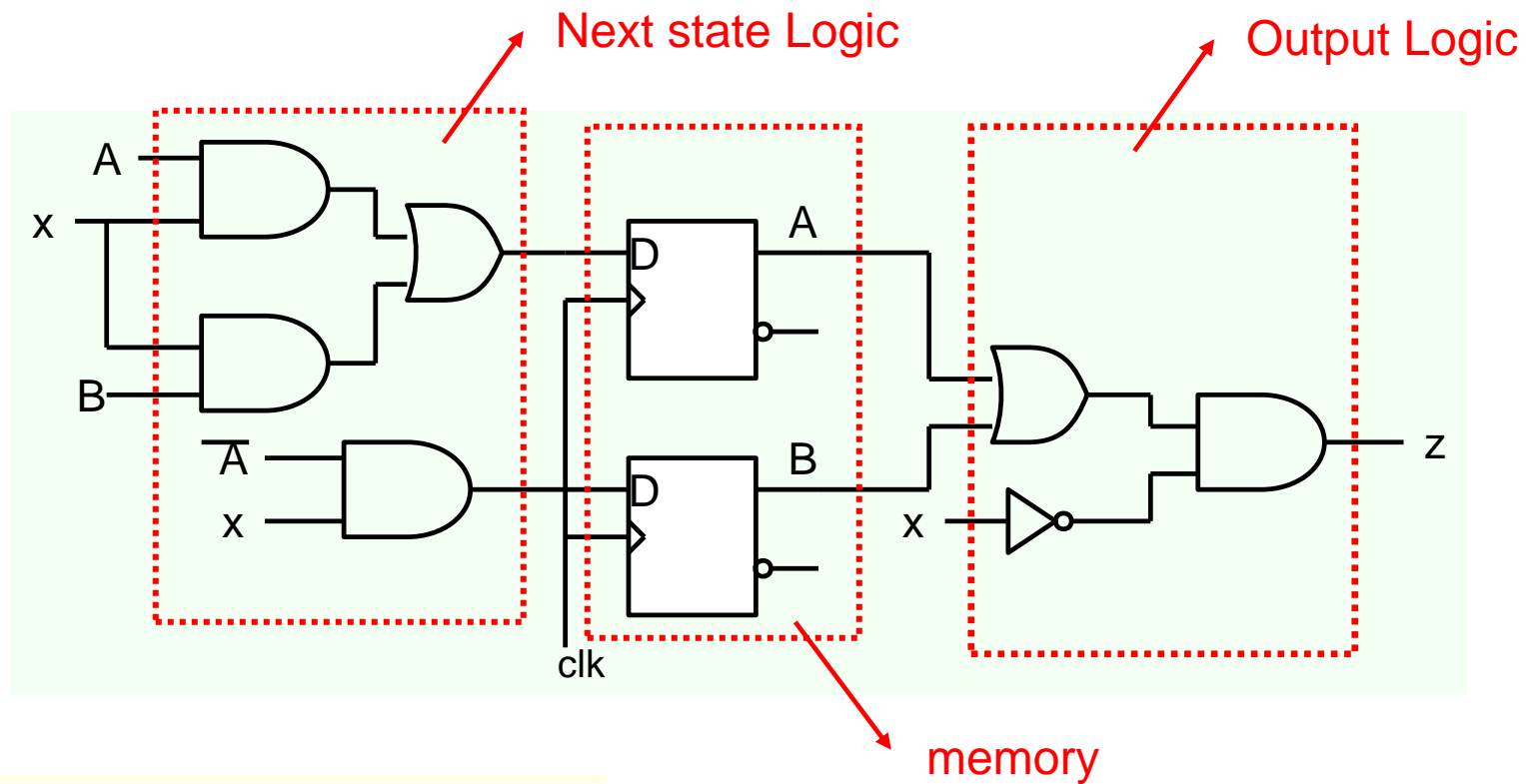
Mealy Sequential Machine

y : Present State
 Y : Next State

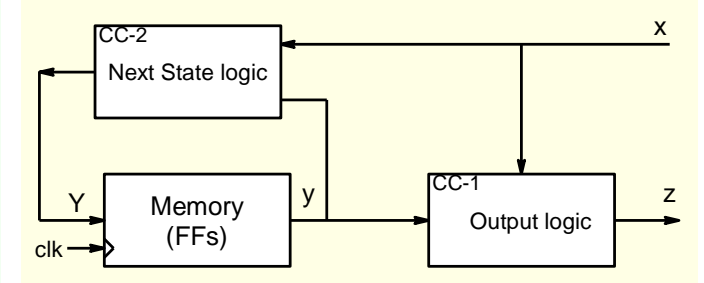
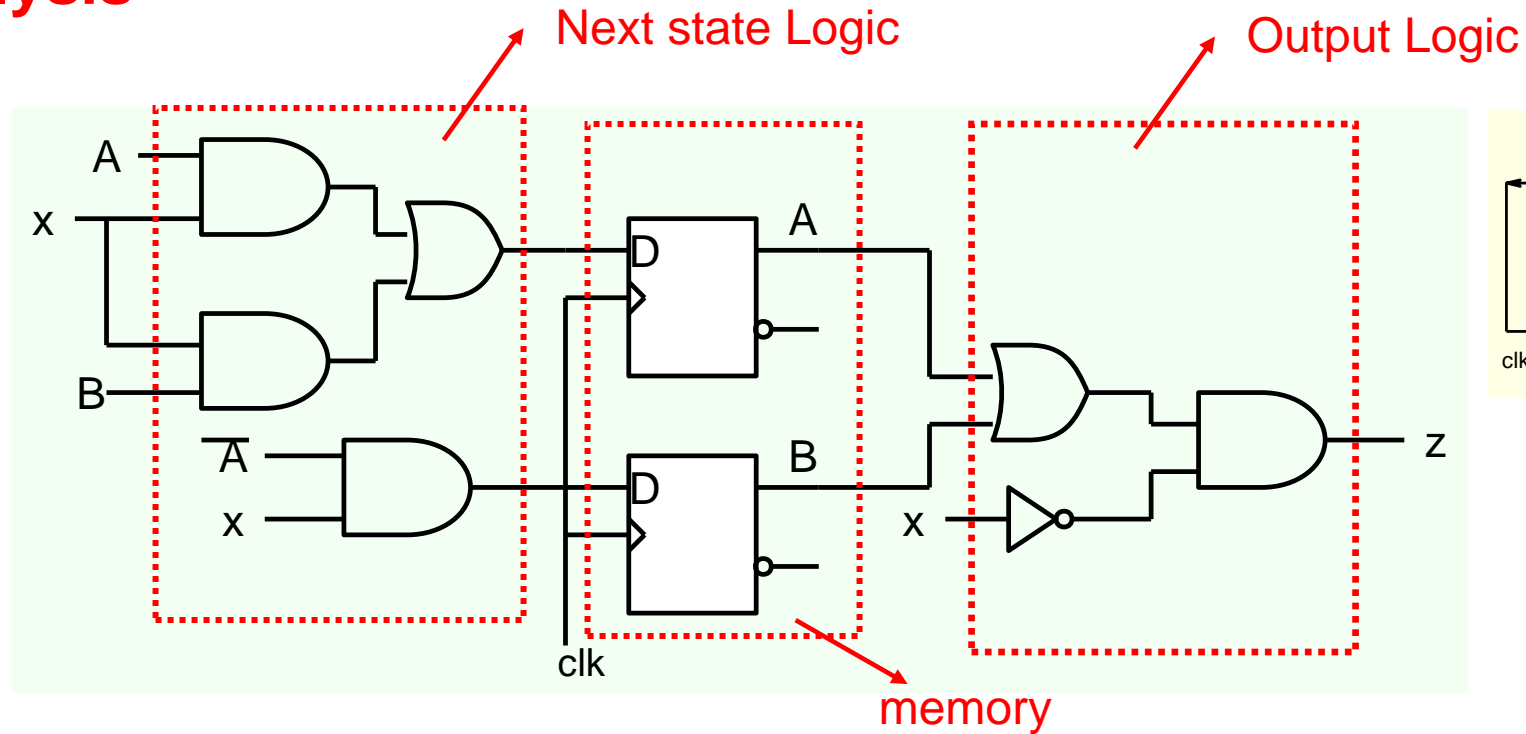


Moore Sequential Machine

Example of a synchronous sequential circuit



Analysis



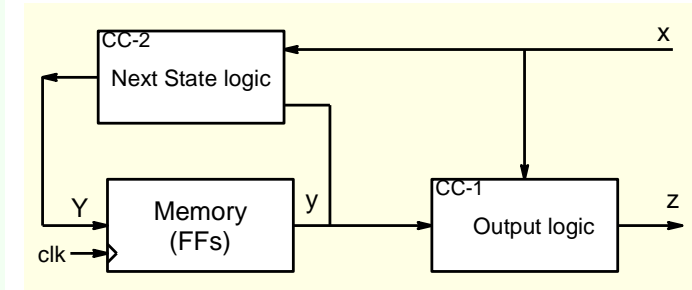
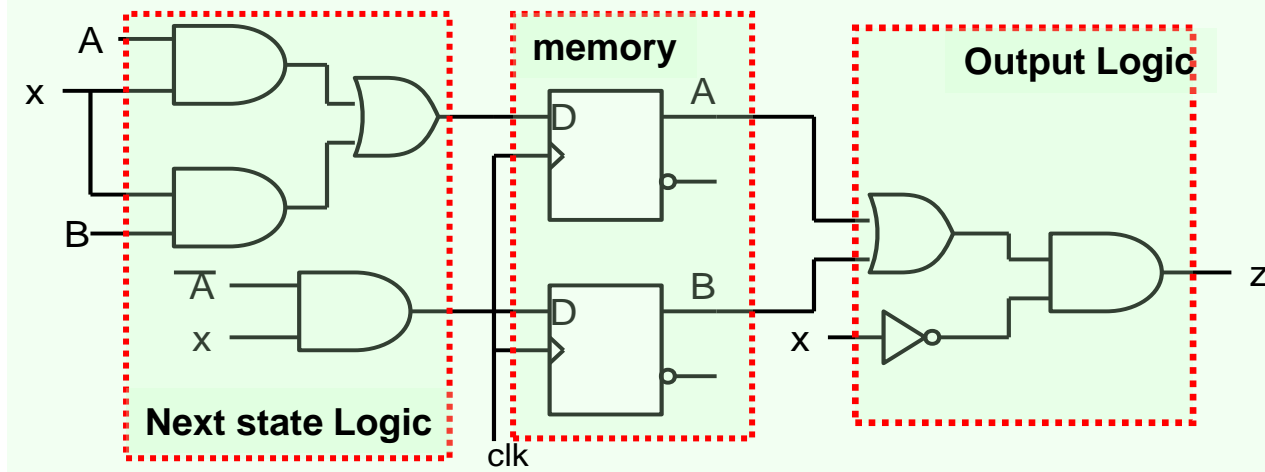
The dependence of output z on input x depends on the state of the memory (A, B)

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



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

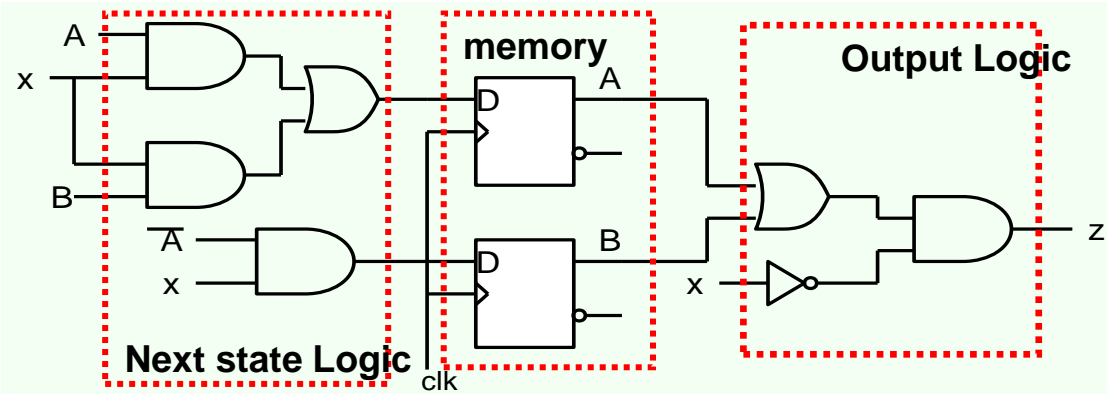
$$A(t+1) = A(t).x + B(t).x$$

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

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

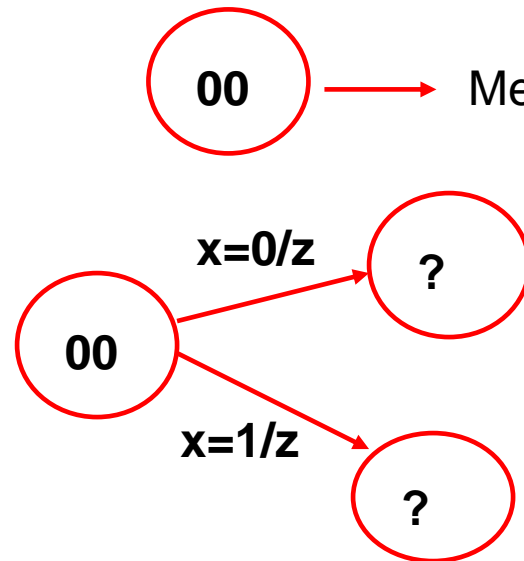
State Transition Table

Present State		Input	Next State		Output
A	B	x	A	B	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		Input	Next State		Output
A	B	x	A	B	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

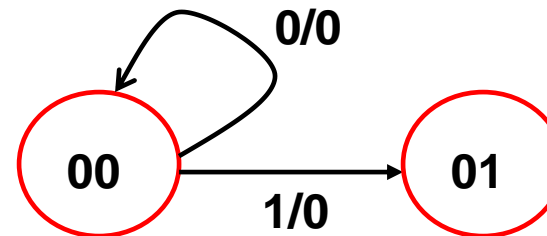


00

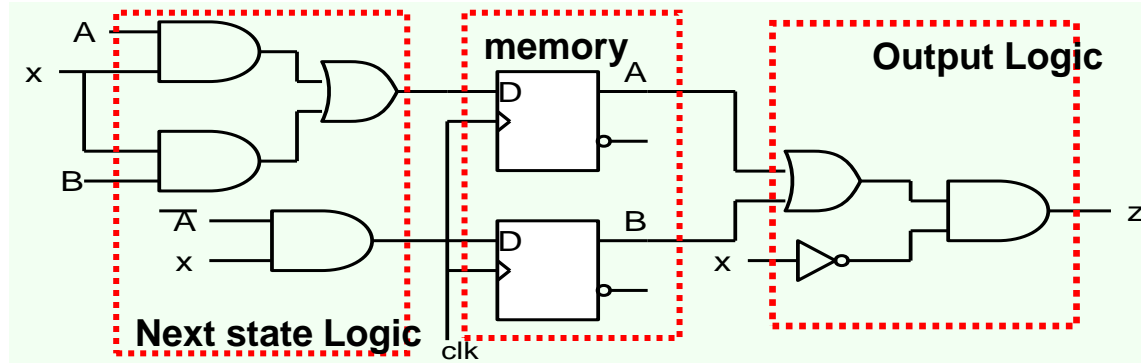
Memory state in which FF A& B have output values 00

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.



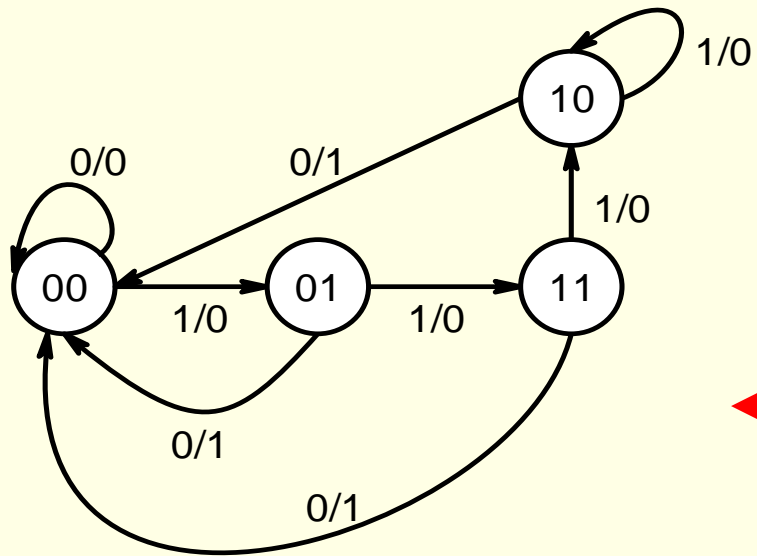
Analysis of Sequential Circuits



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

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

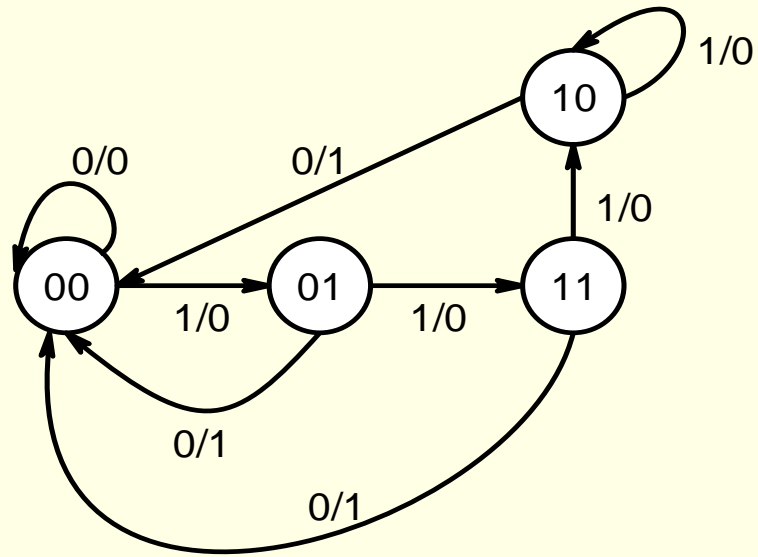
$$z = (A + B). \overline{x}$$



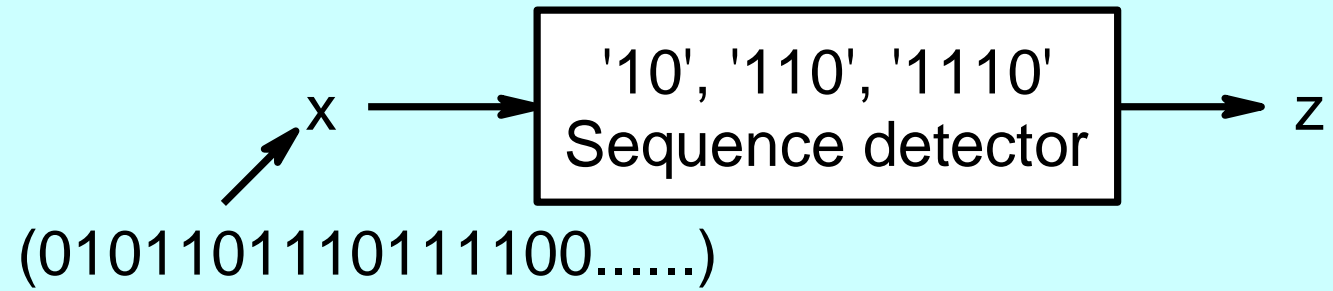
State transition Graph

State Transition Table

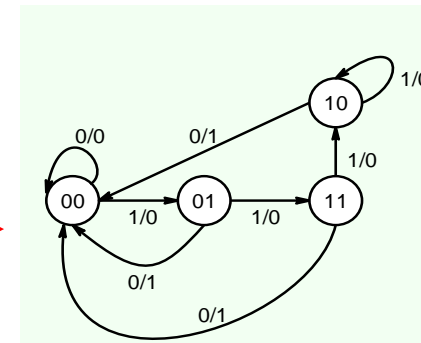
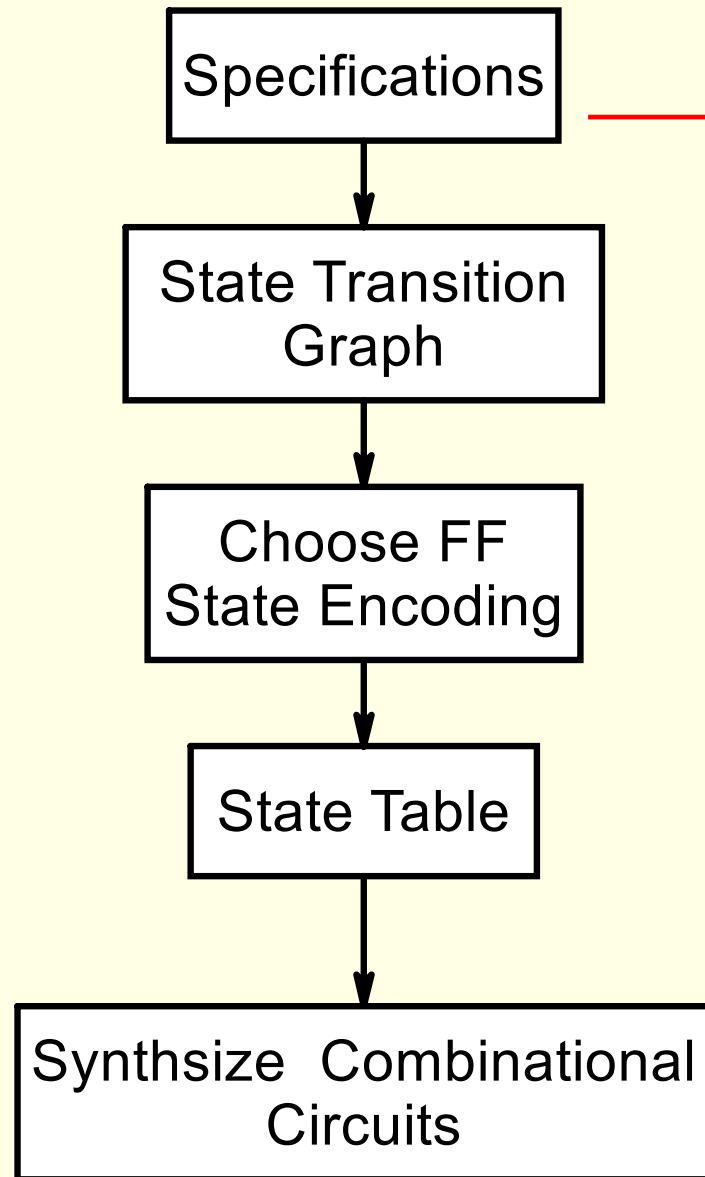
Present State		Input	Next State		Output
A	B	x	A	B	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



State transition Graph

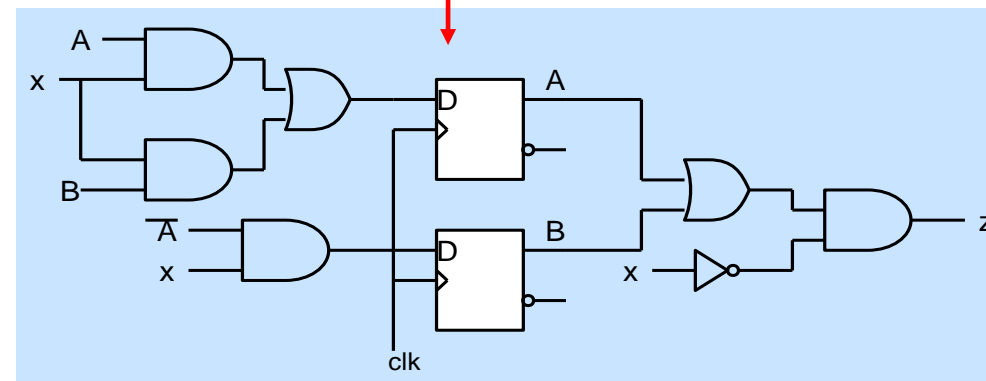


Design of Sequential Circuits

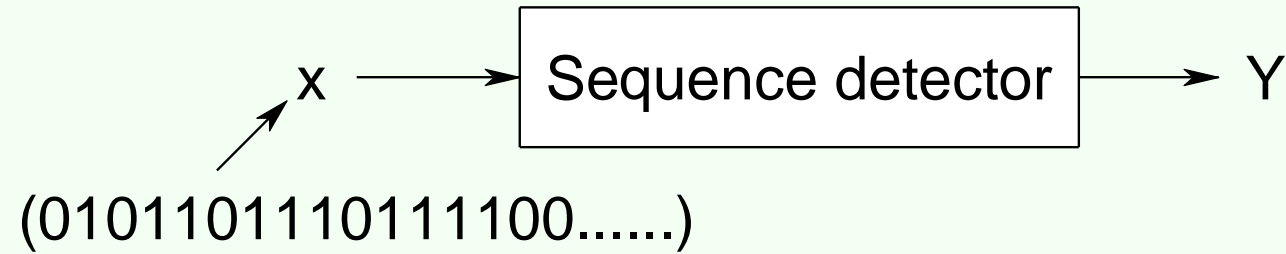


State Transition Table

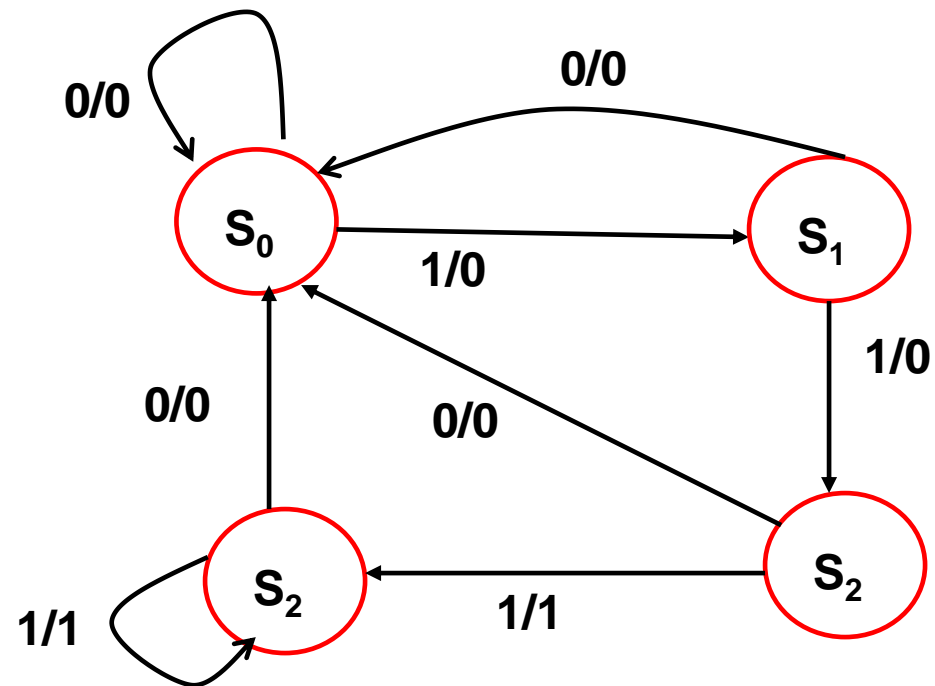
Present State		Input	Next State		Output
A	B	x	A	B	z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	1	0	0



System specification to State Transition Graph

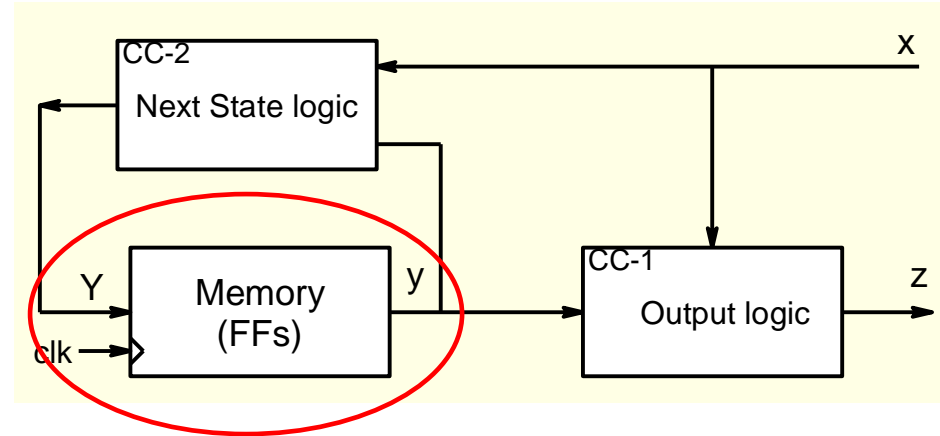
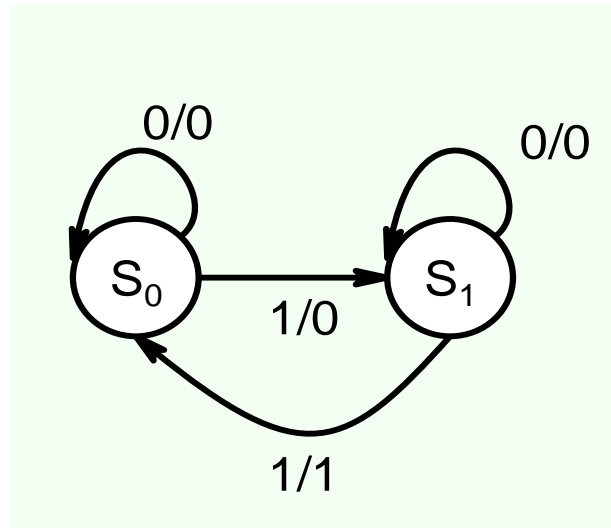


Detect 3 or more consecutive 1's in the input stream



Conversion of State transition graph to a circuit

Example-1



3 blocks need to be designed

1. How many FFs do we need?

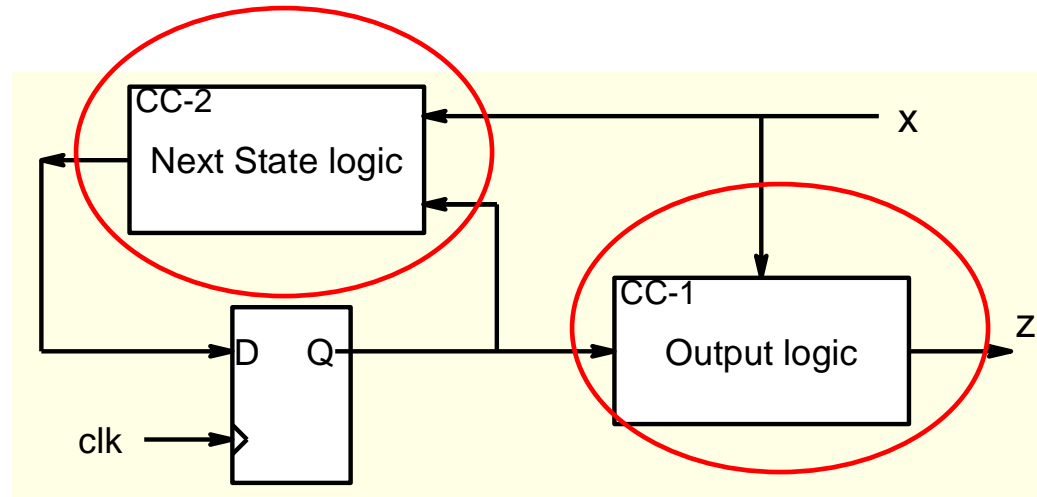
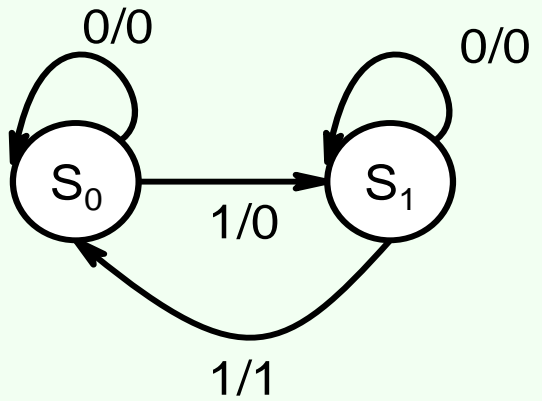
N FFs can represent 2^N states so Minimum is 1

2. Which FF do we choose?

Say D FF

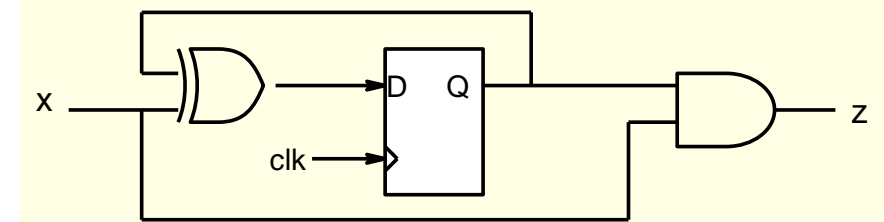
3. How are the states encoded?

Say FF output $Q=0$ represents S_0 and $Q=1$ represents S_1 state



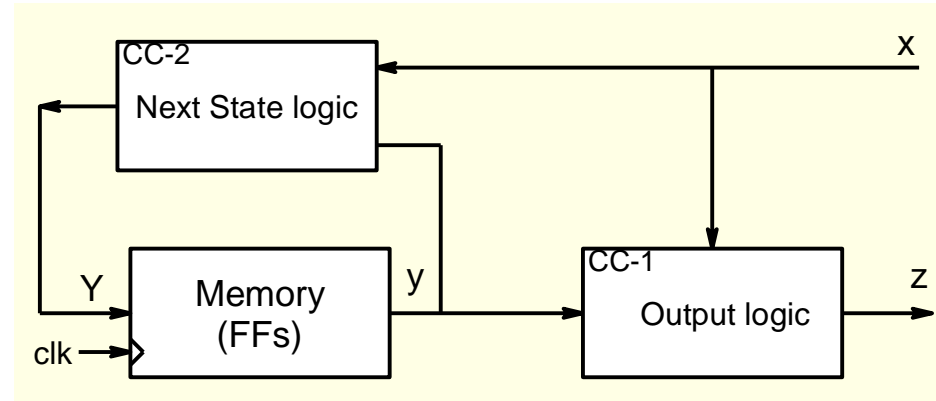
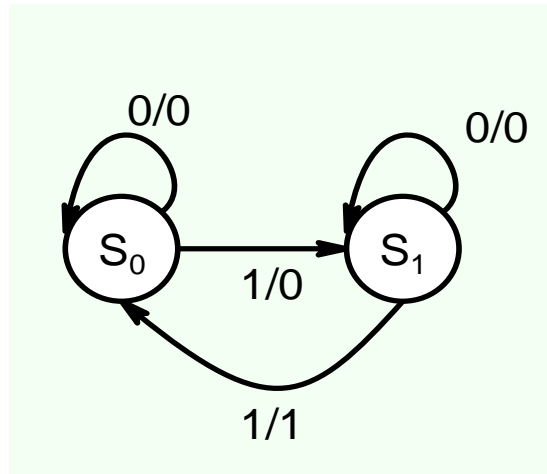
State Transition Table

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

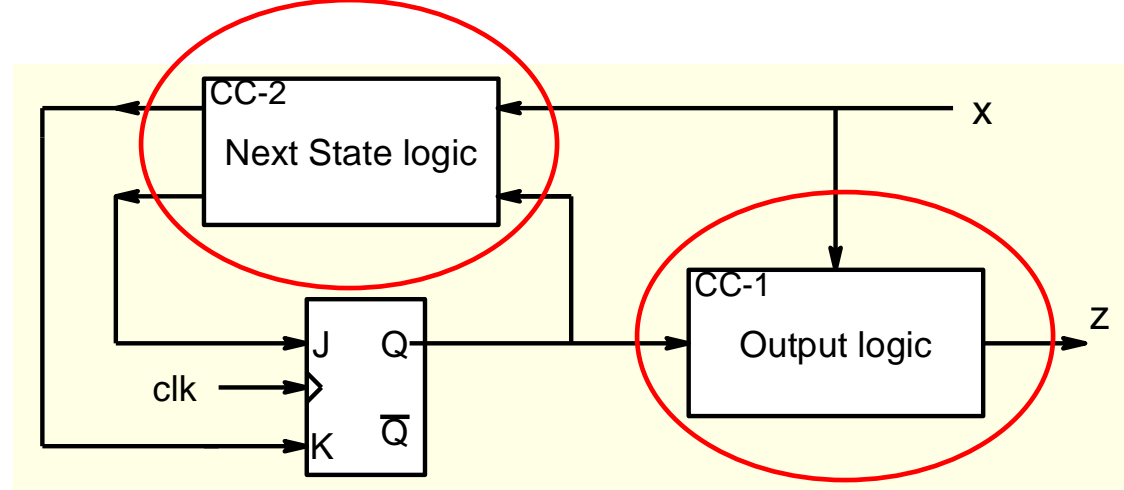
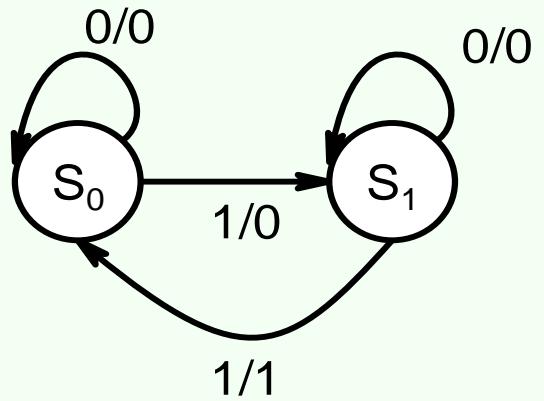


$$D = \overline{Q}.x + Q.\overline{x} \quad ; \quad z = Q.x$$

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**

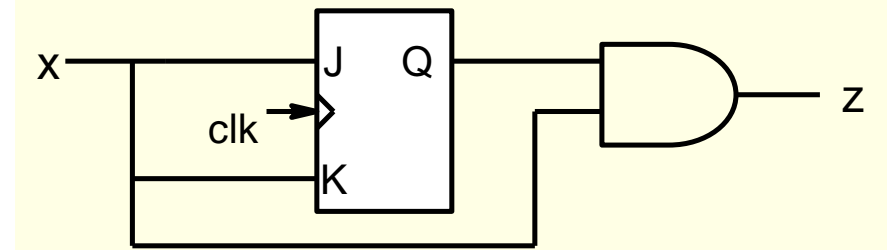


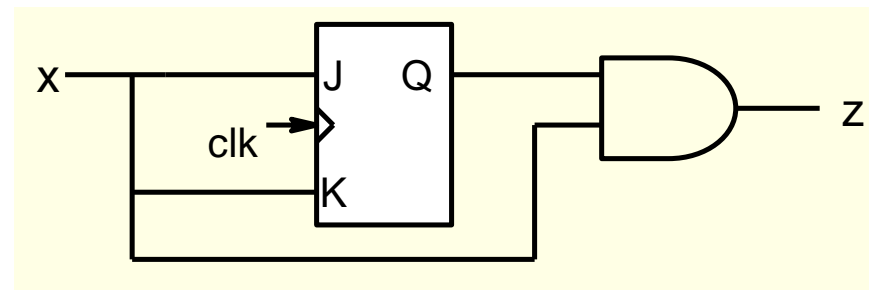
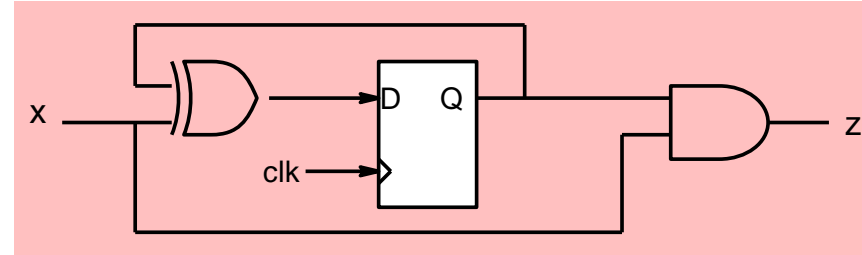
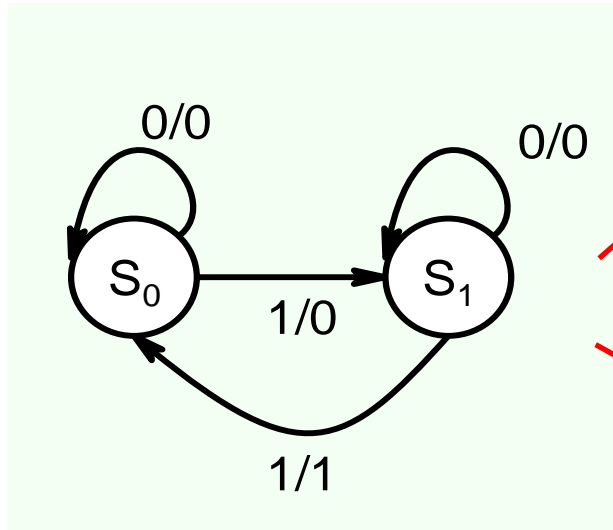
State Transition Table

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

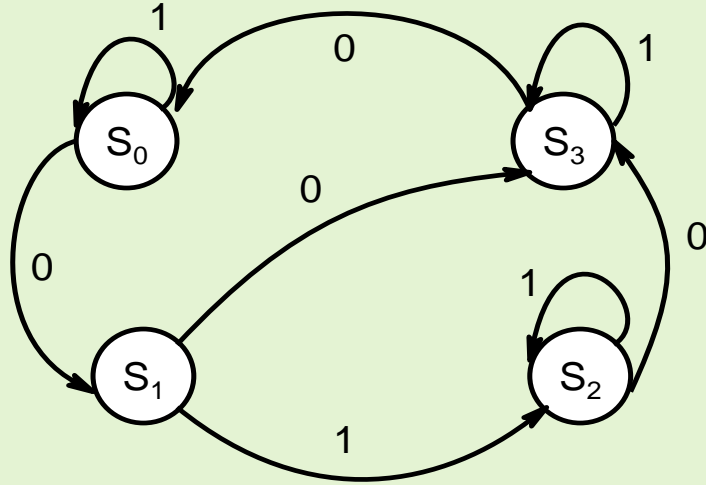
$$J = x ; K = x; z = Q \cdot x$$

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



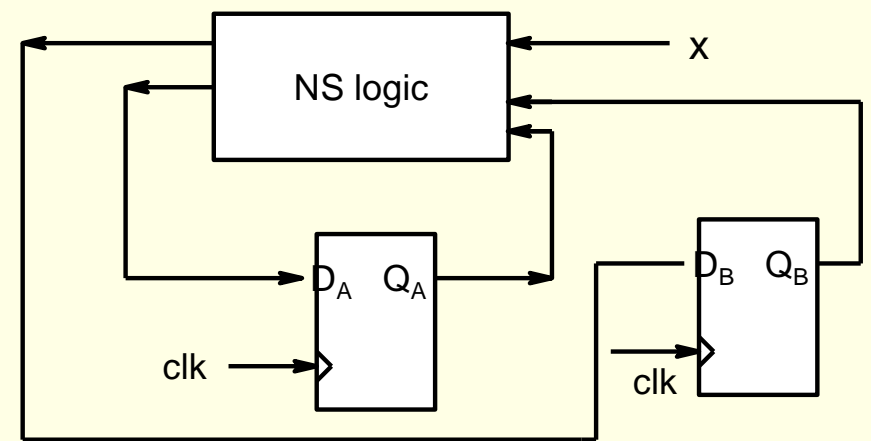


Example-3



State	FF O/P	
	A	B
S ₀	0	0
S ₁	0	1
S ₂	1	0
S ₃	1	1

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



Present State		Input x	Next State		D _A	D _B
A	B		A	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

$$D_A = \bar{A}B + xB + A\bar{B}$$

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

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

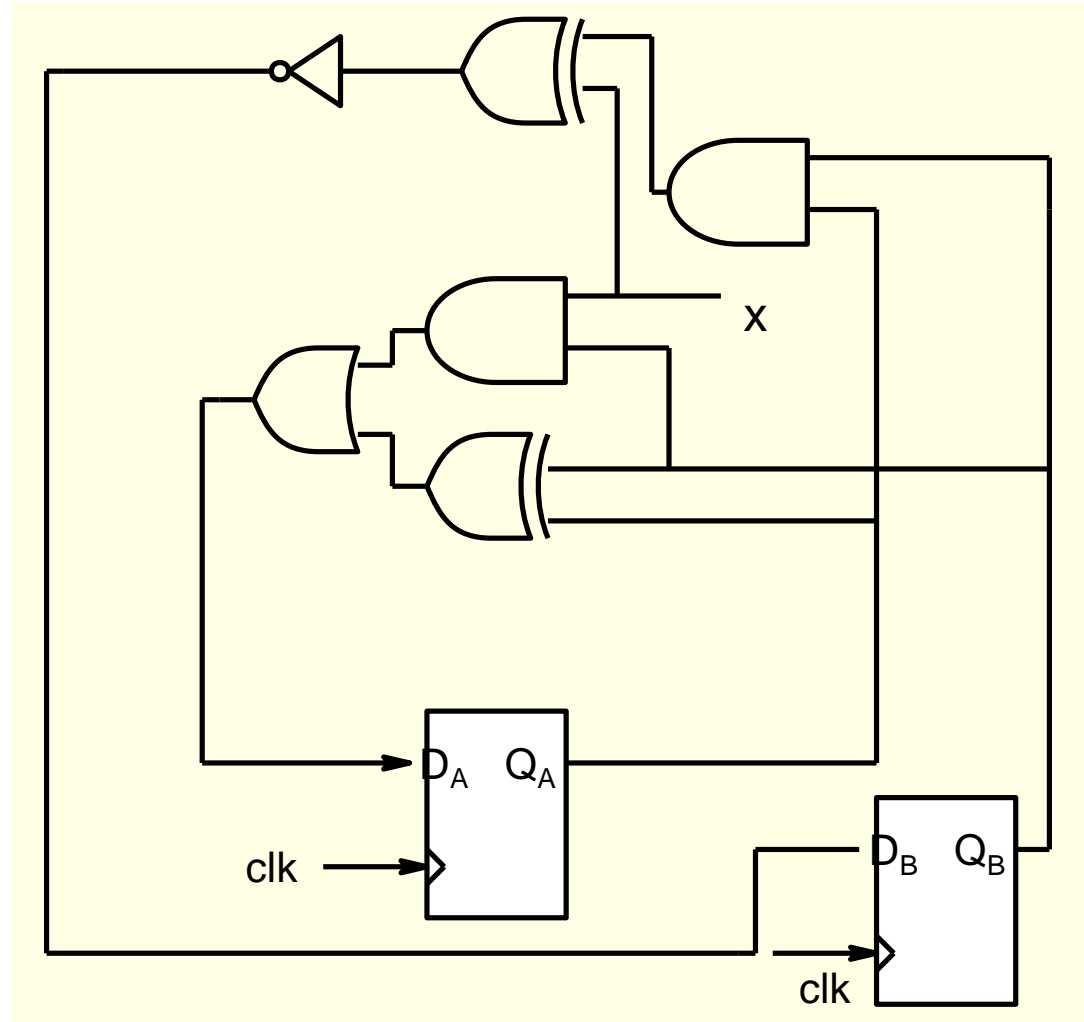
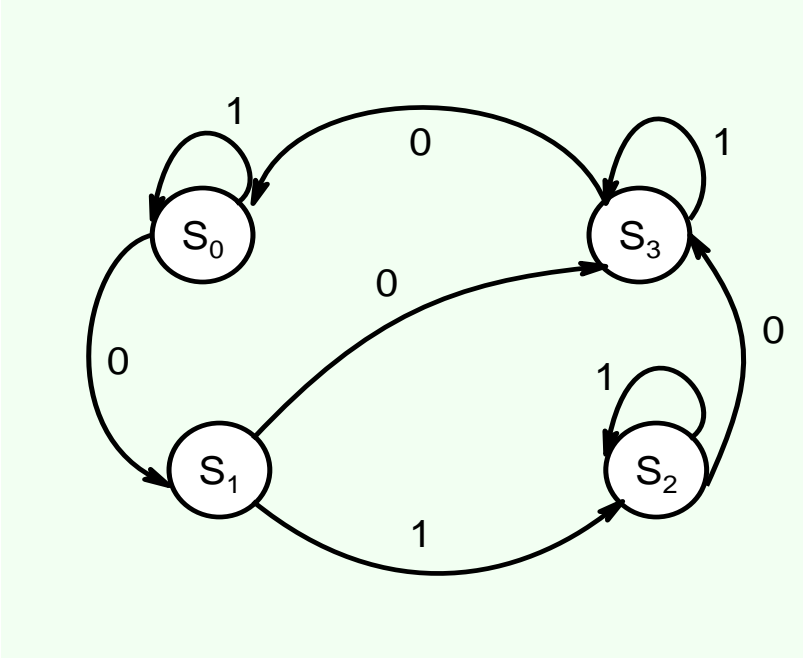
$$D_B = \bar{x}.\bar{A} + \bar{x}.\bar{B} + x.A.B$$

$$= \bar{x}.(\bar{A} + \bar{B}) + x.A.B$$

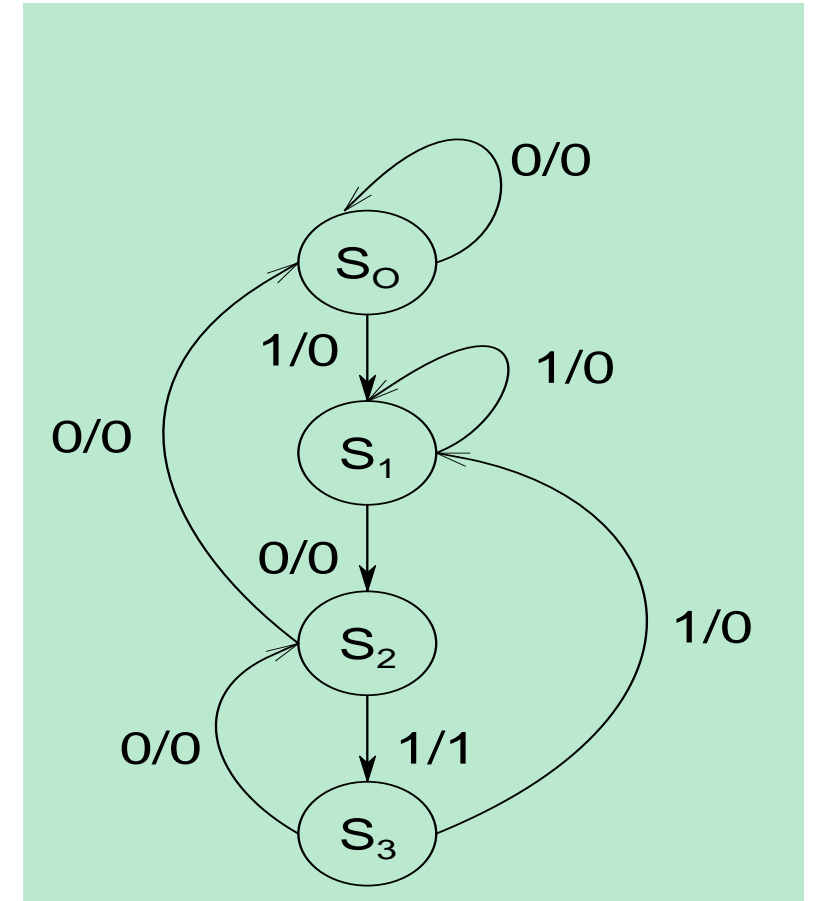
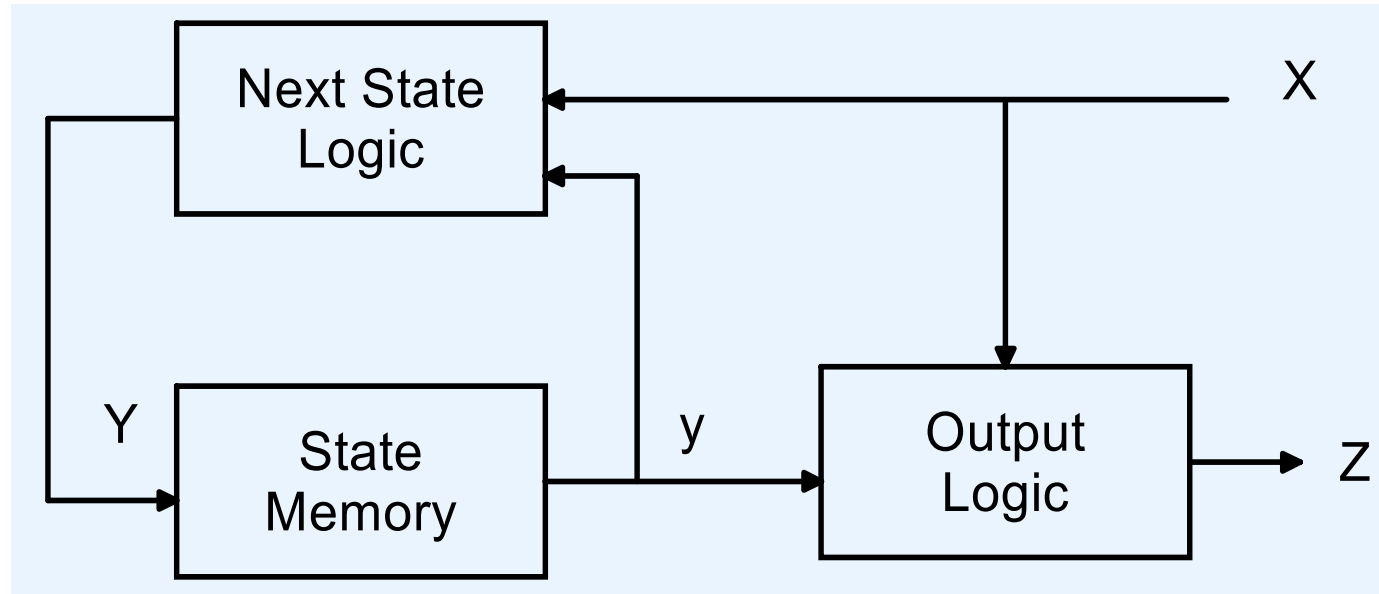
$$= \bar{x}.\overline{AB} + x.AB = \overline{x \oplus AB}$$

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

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



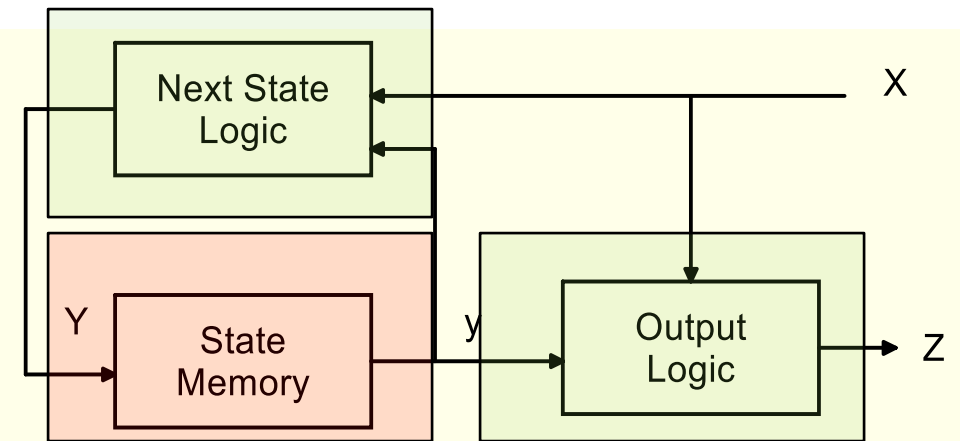
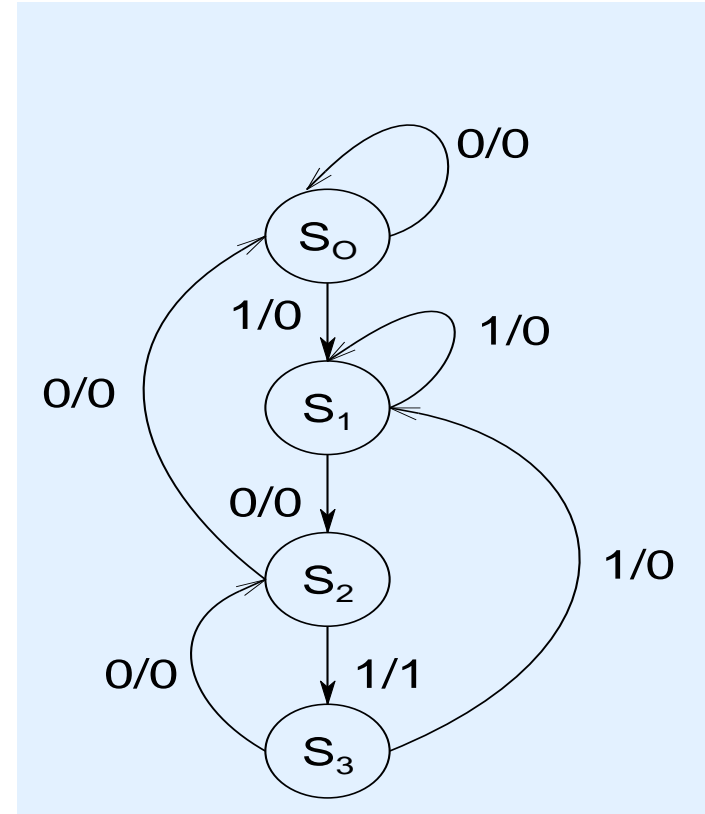
Sequential Circuits: Summary



- we need to know how the system goes from one state to another in response to the inputs and how the outputs respond to these changes

Synthesis involves the following tasks:

- Number of storage elements
- State Encoding
- Choosing a flipflop type to implement states
- Synthesize next state logic
- Synthesize output logic



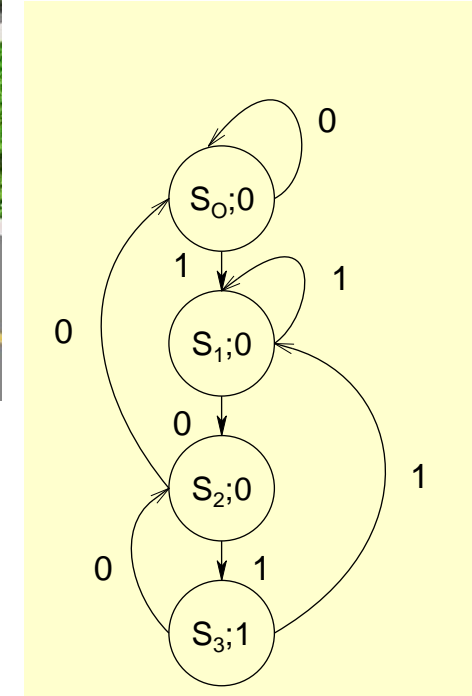
Examples of FSM



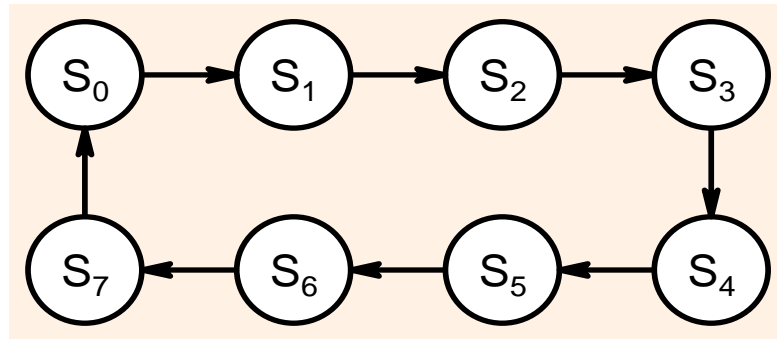
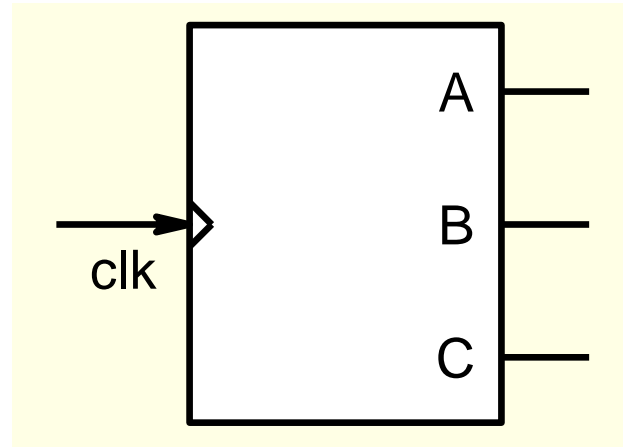
Vending machine



Traffic light controller



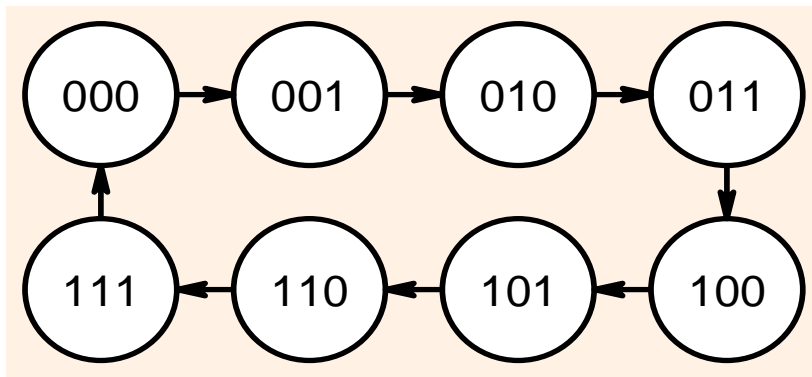
Counters



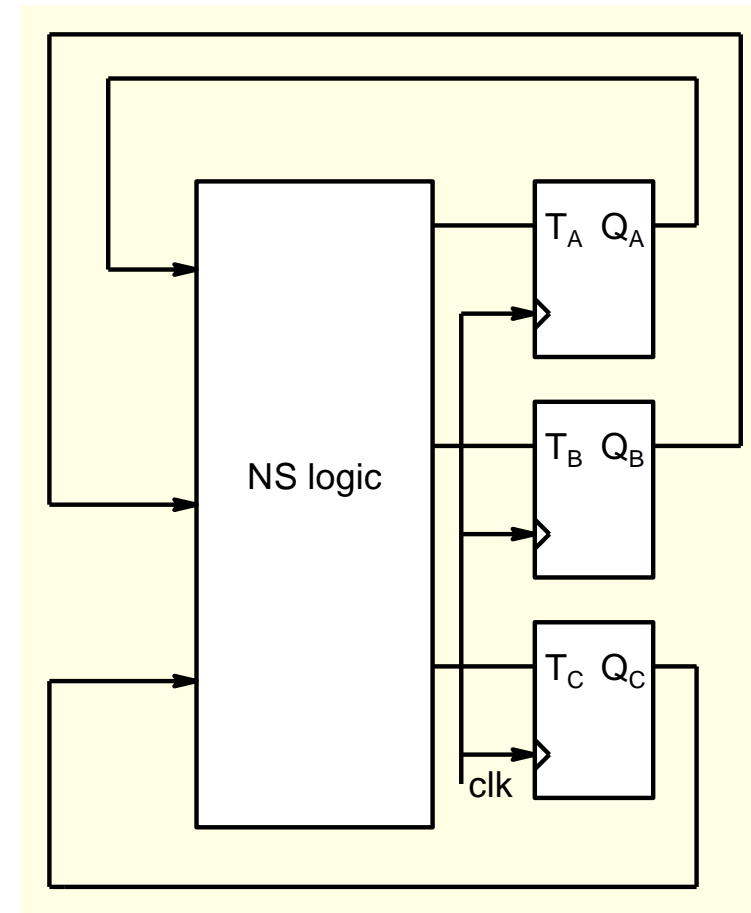
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

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.

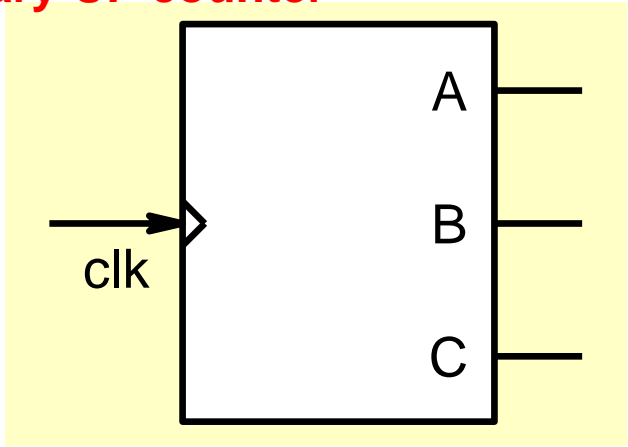


PS			NS					
A	B	C	A	B	C	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

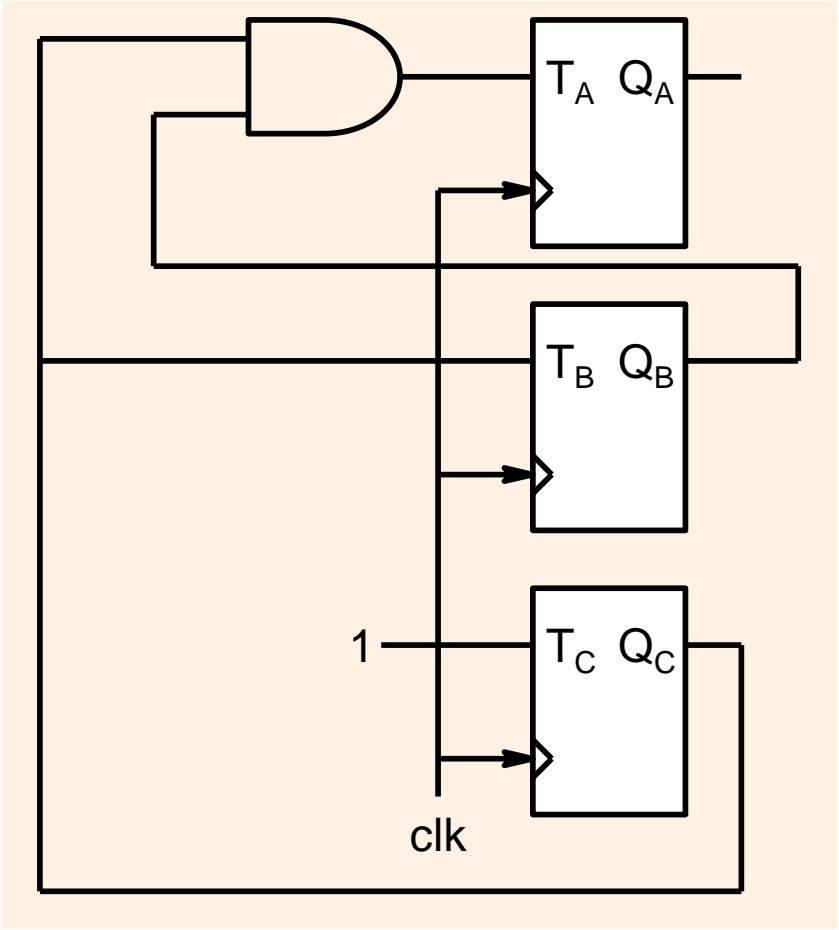


$$T_A = B.C ; T_B = C ; T_C = 1$$

Binary UP counter



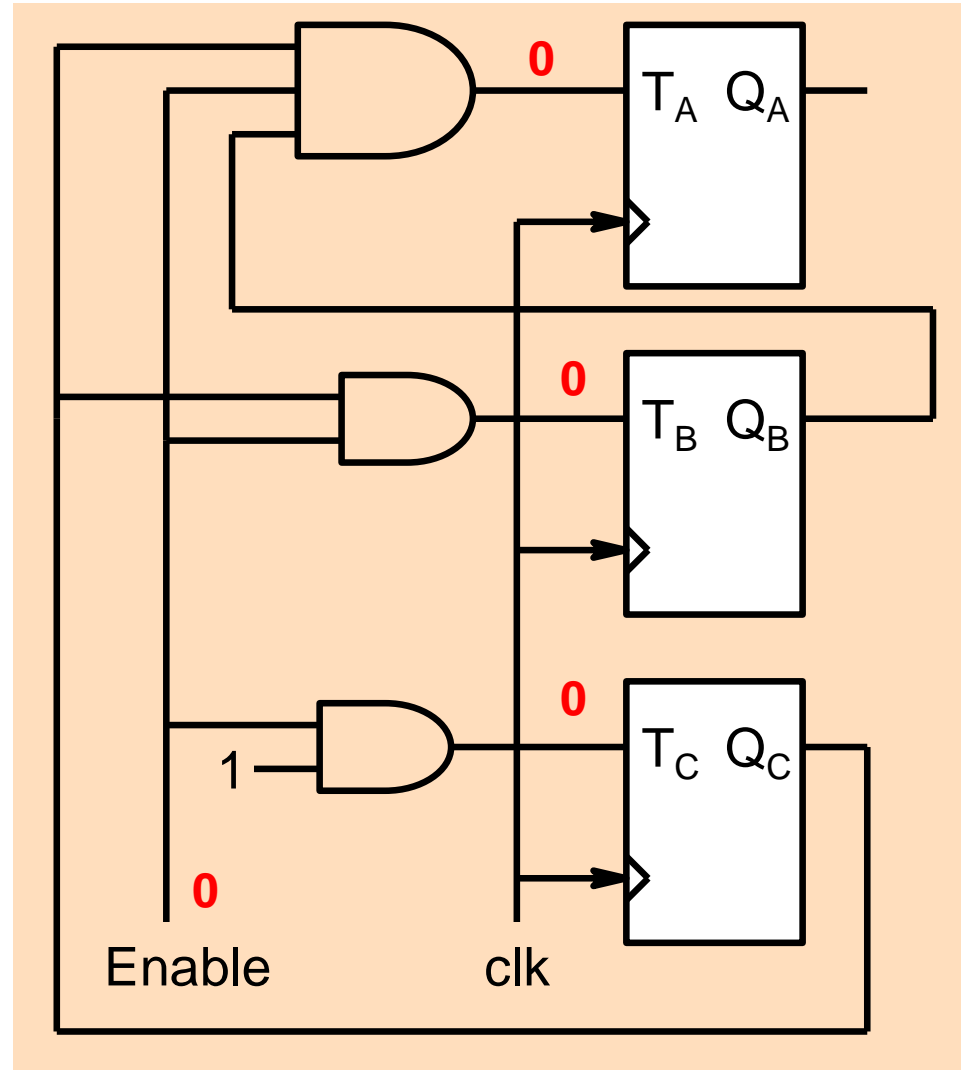
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



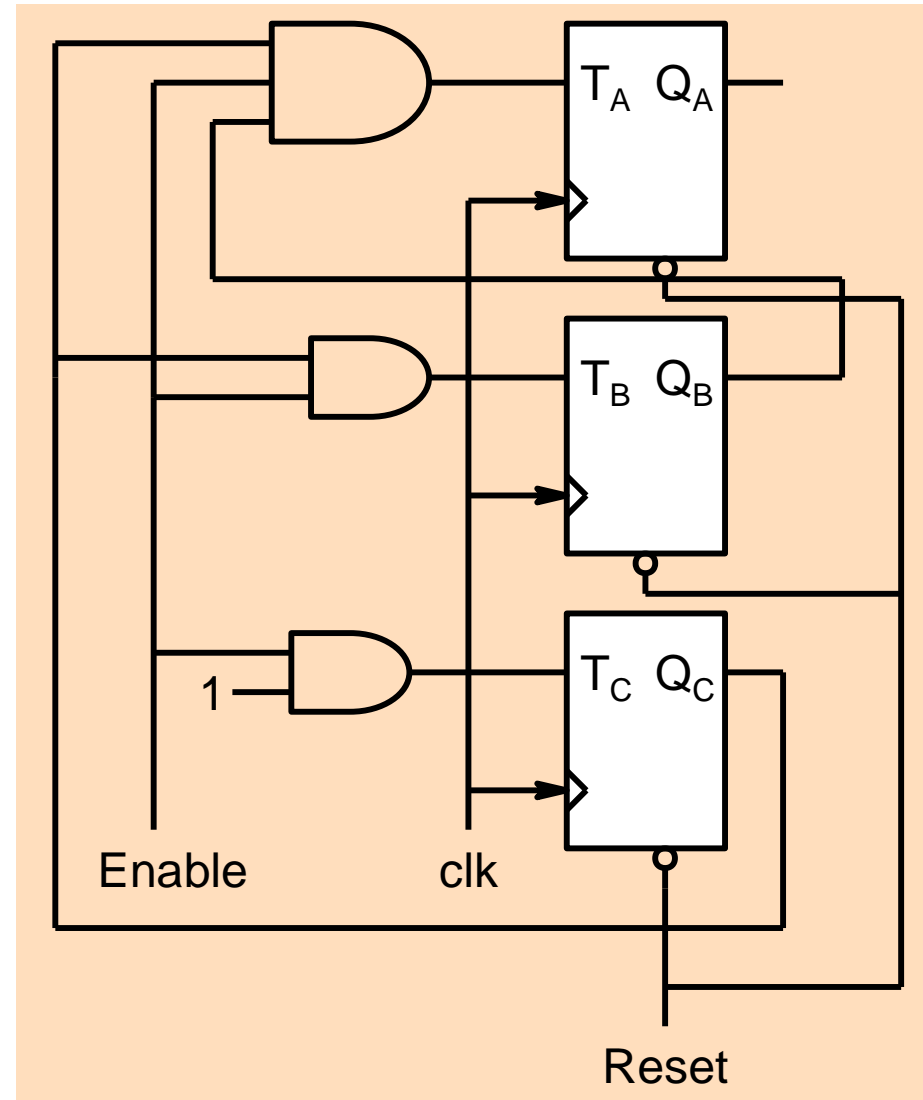
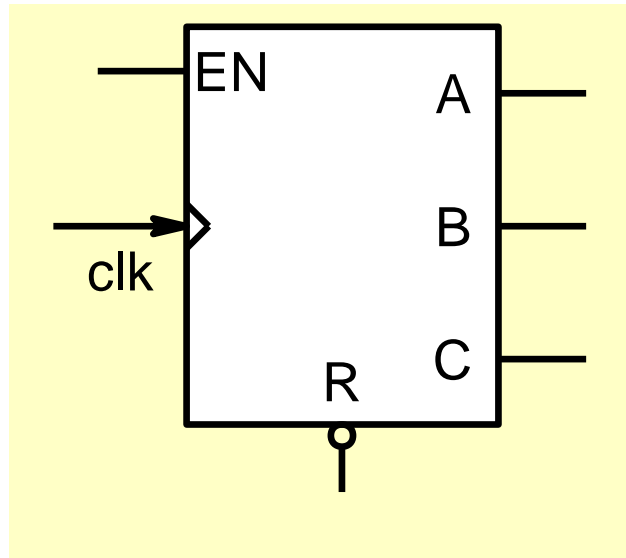
$T_A = B.C ; T_B = C ; T_C = 1$

The diagram shows a rectangular block representing a 3-bit shift register. On the left side, there are two inputs: 'EN' (Enable) and 'clk' (clock). The 'EN' input is a simple line, while the 'clk' input is a line ending in a triangle symbol. On the right side, there are three outputs labeled 'A', 'B', and 'C', each with a corresponding output line.

When Enable = 1, the counter begins the count.



Counter with Asynchronous Reset

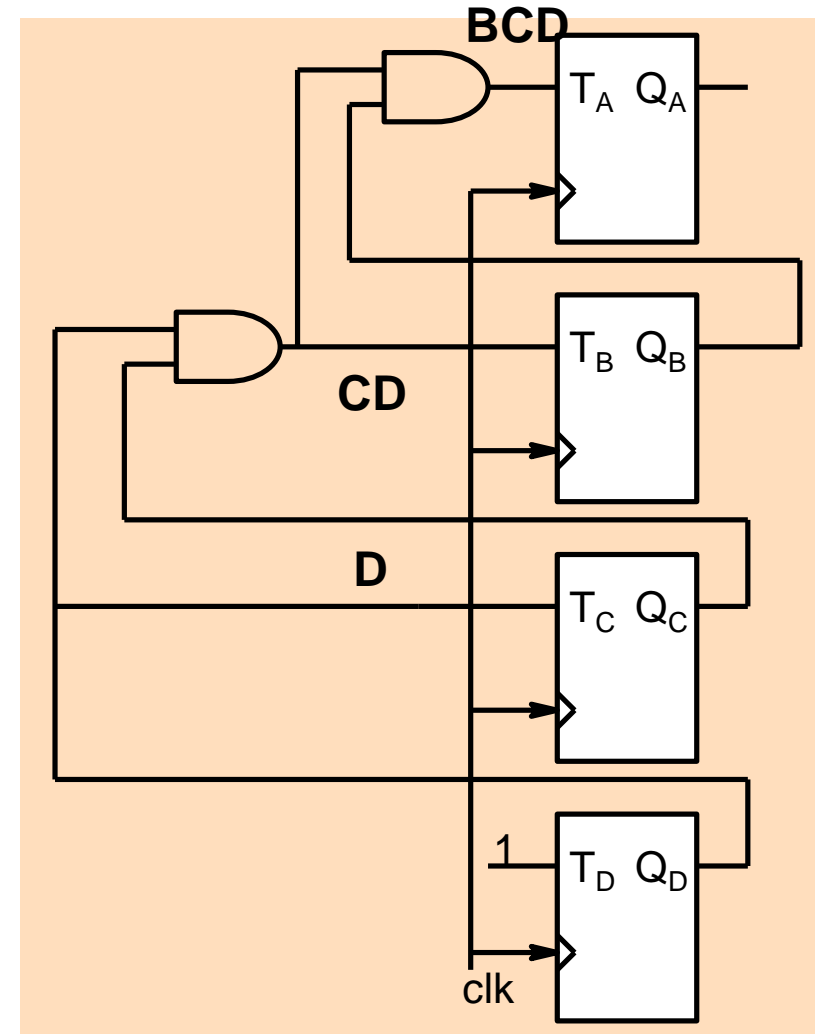


When Enable = 1, the counter begins the count.

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
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
- B toggles only when D is 1
- C 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

1000

0 1 1 1

0 1 1 0

0 1 0 1

0 1 0 0

0 0 1 1

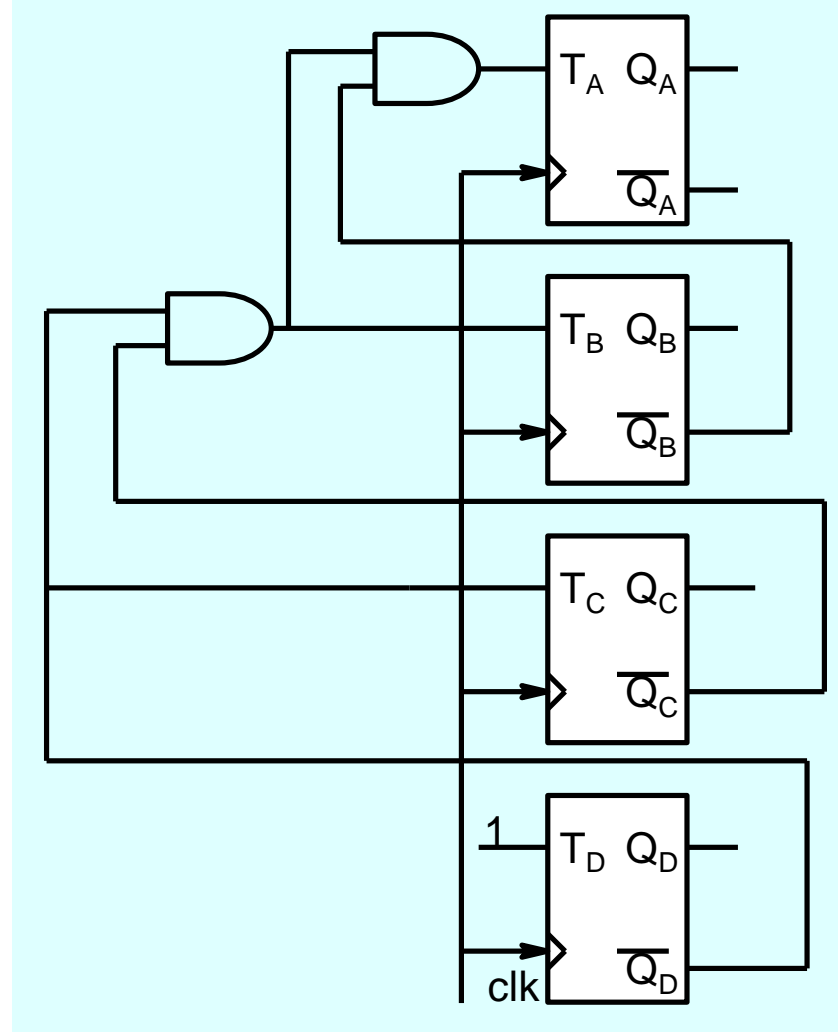
0 0 1 0

0 0 0 1

0 0 0 0

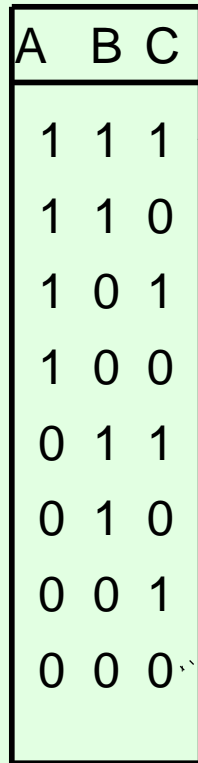
1 1 1 1

- 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



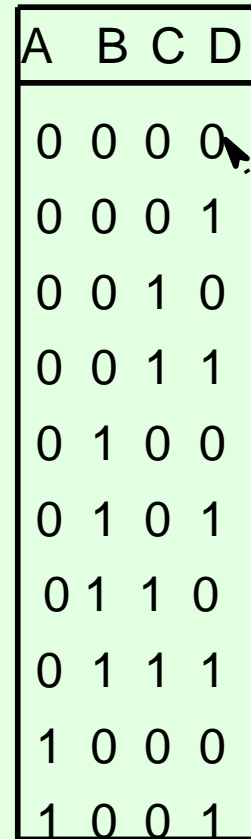
Counters

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



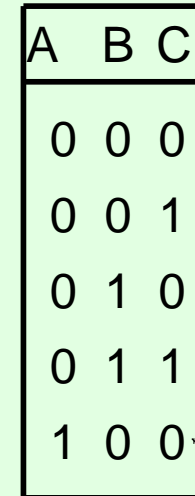
Binary down counter

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



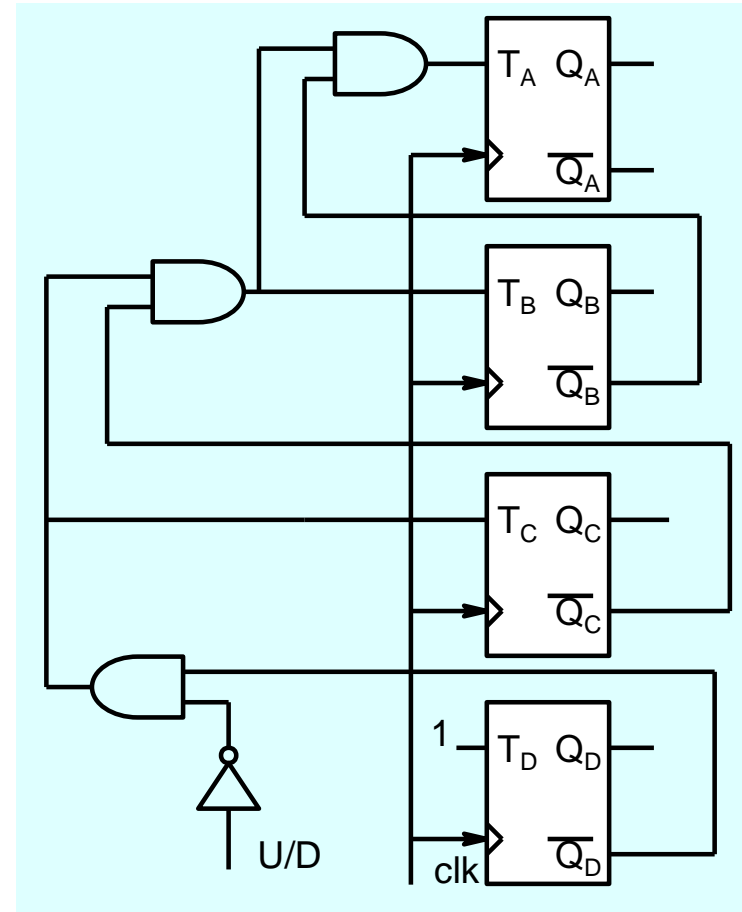
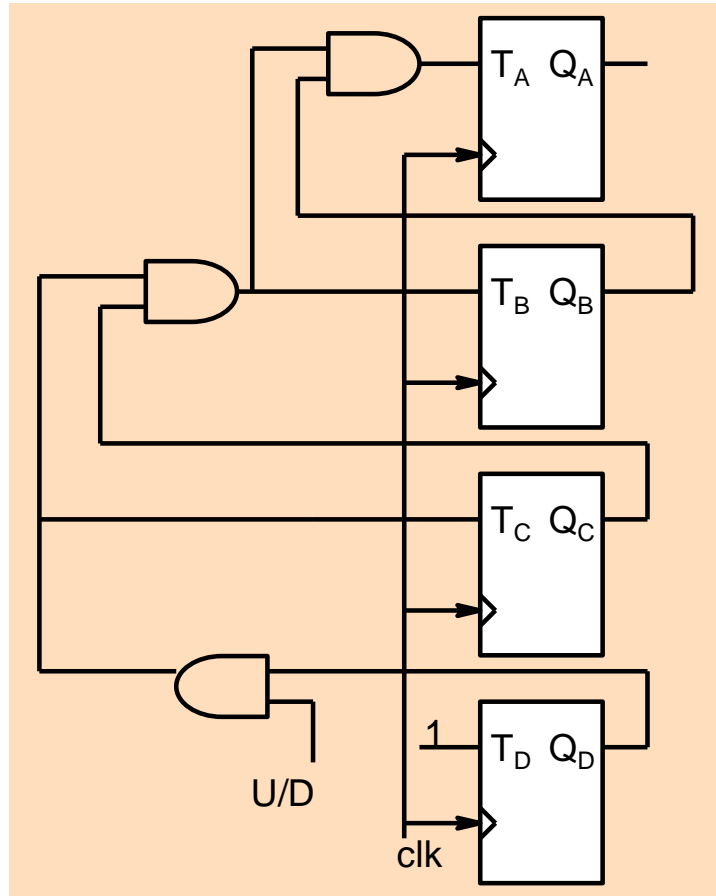
Decade counter
Modulo-10 Counter

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



Modulo-5 Counter

4-bit Up-Down Counter



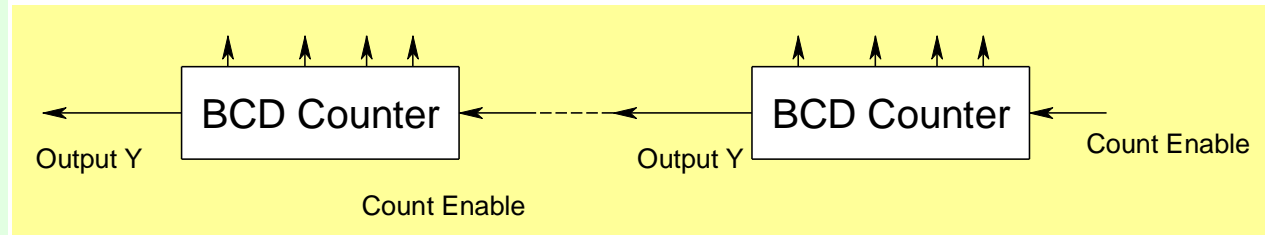
Merging of the two structures gives an Up/down counter

BCD Counter

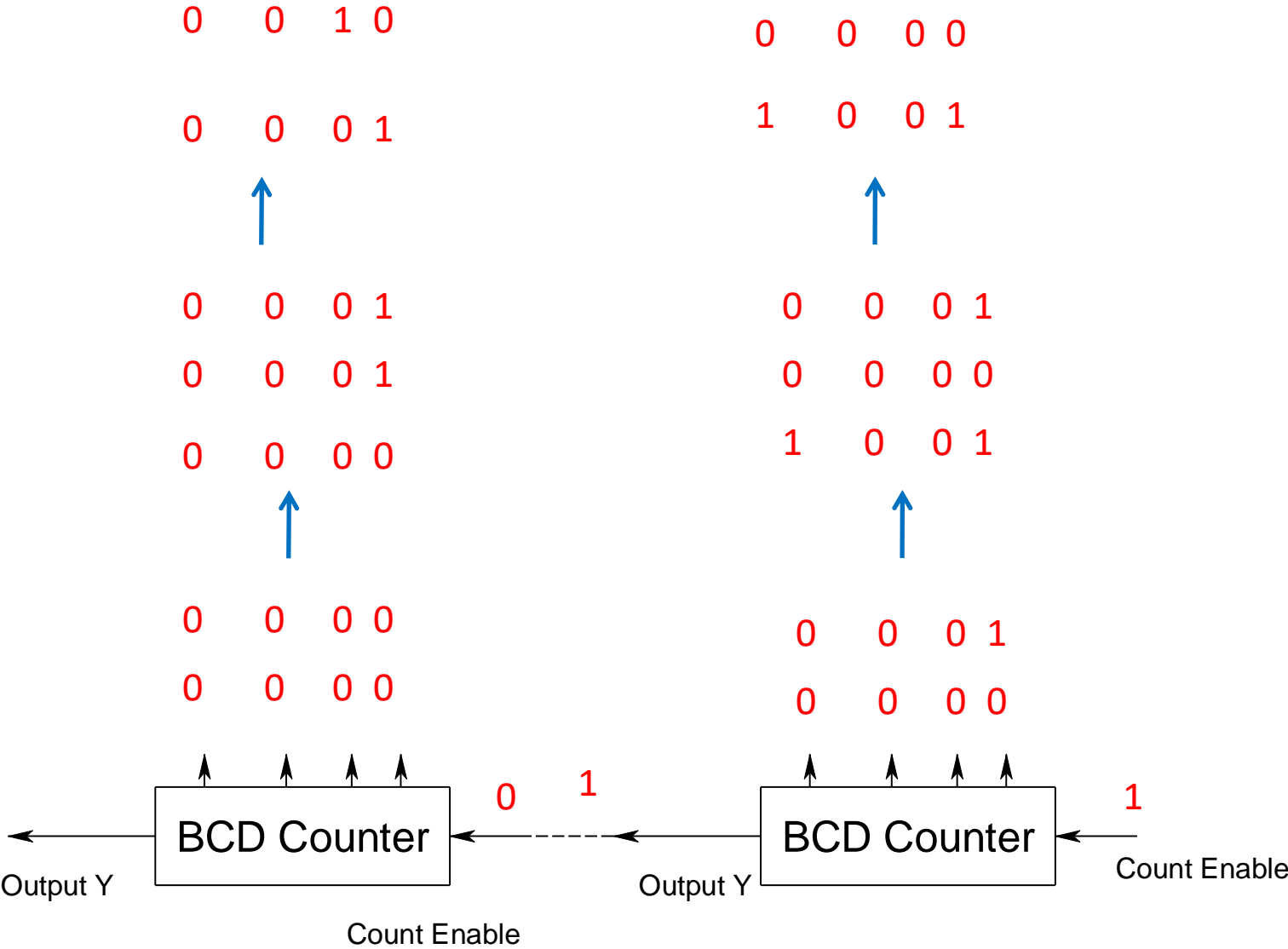
Binary Coded Decimal (BCD): each decimal digit is coded as a 4-bit binary number

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	1
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			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

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

$$J_A = B$$

Counter with Unused States

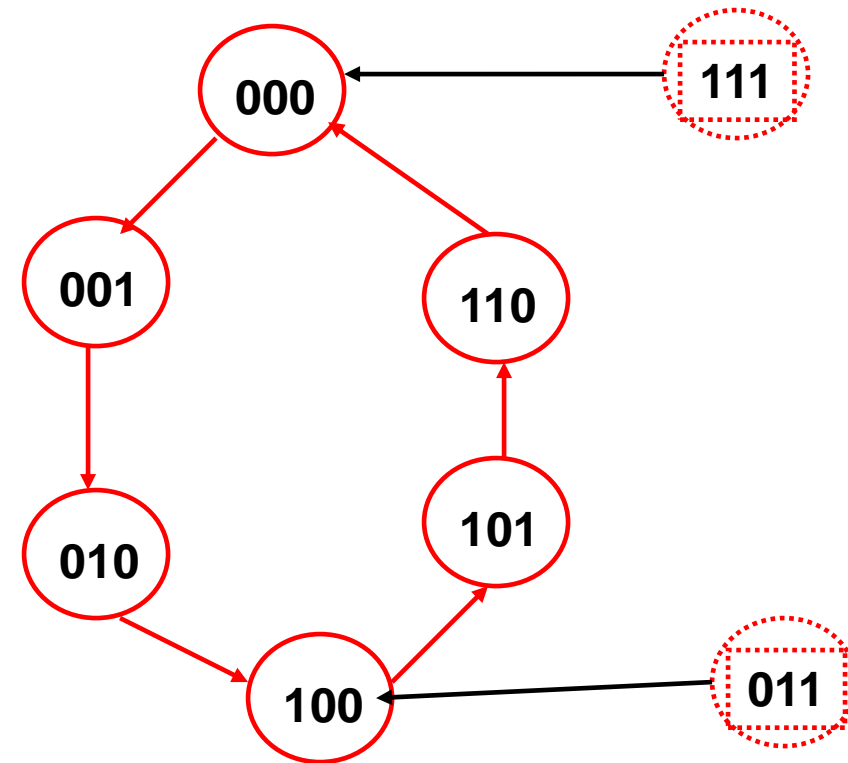
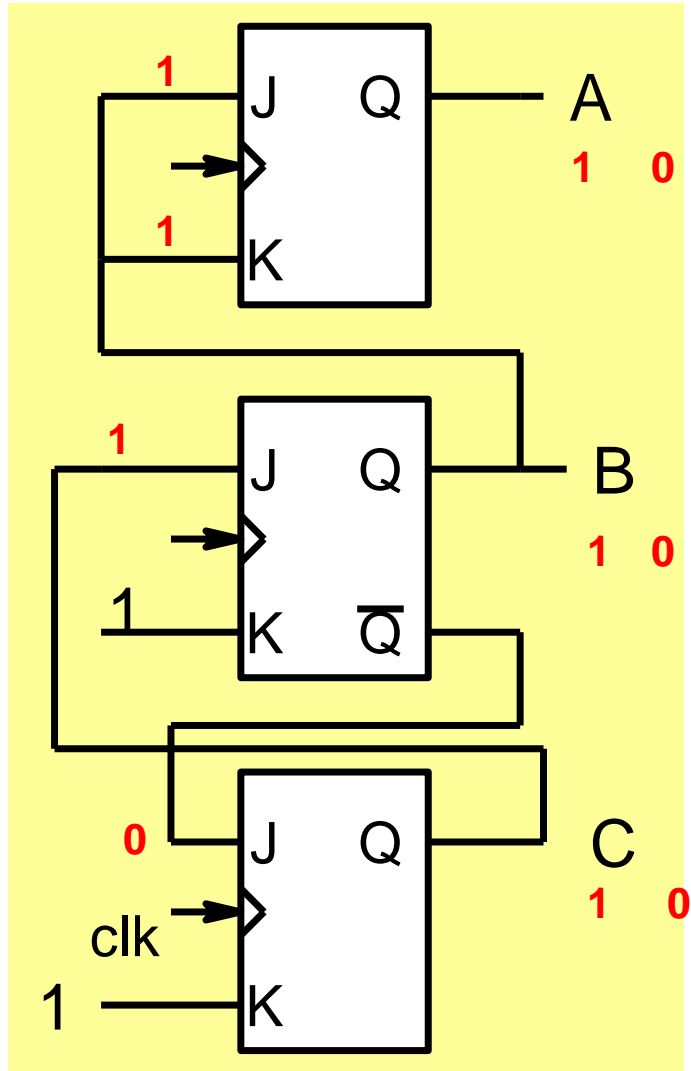
PS			NS			J_A K_A		J_B K_B		J_C K_C	
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

$$J_A = B \quad K_A = B$$

$$J_B = C \quad K_B = 1$$

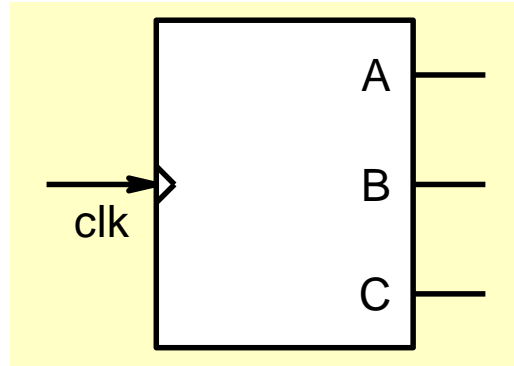
$$J_C = \overline{B} \quad K_C = 1$$

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

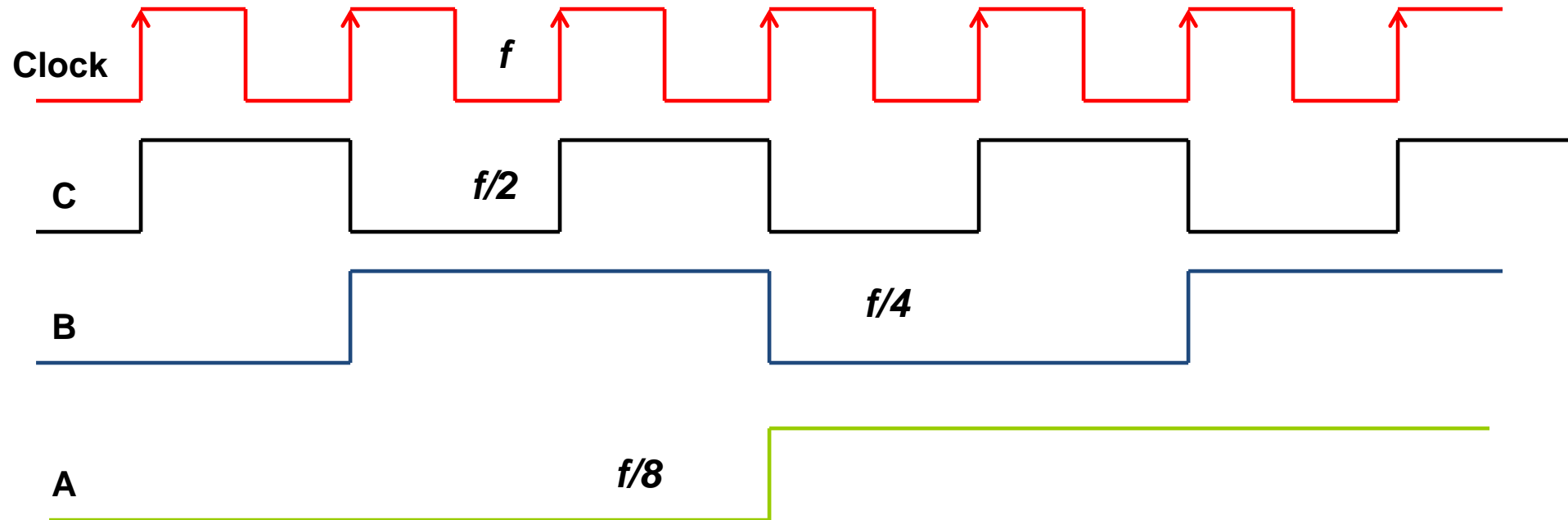


We can see that if by chance the counter goes into unused states 111 or 011, then after a clock cycle it enters one of the used states.

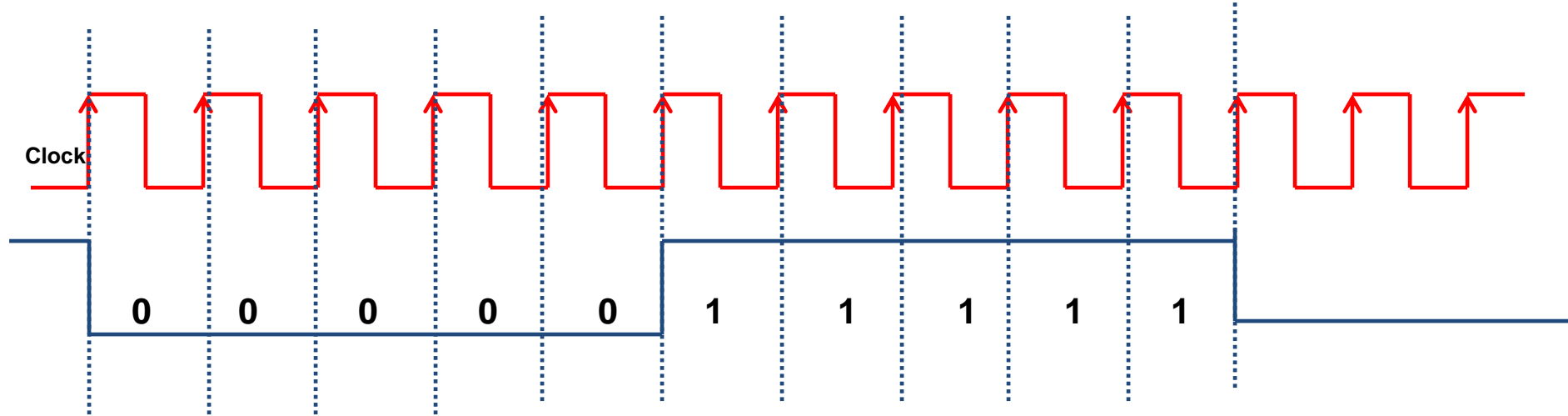
Counter as frequency divider



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

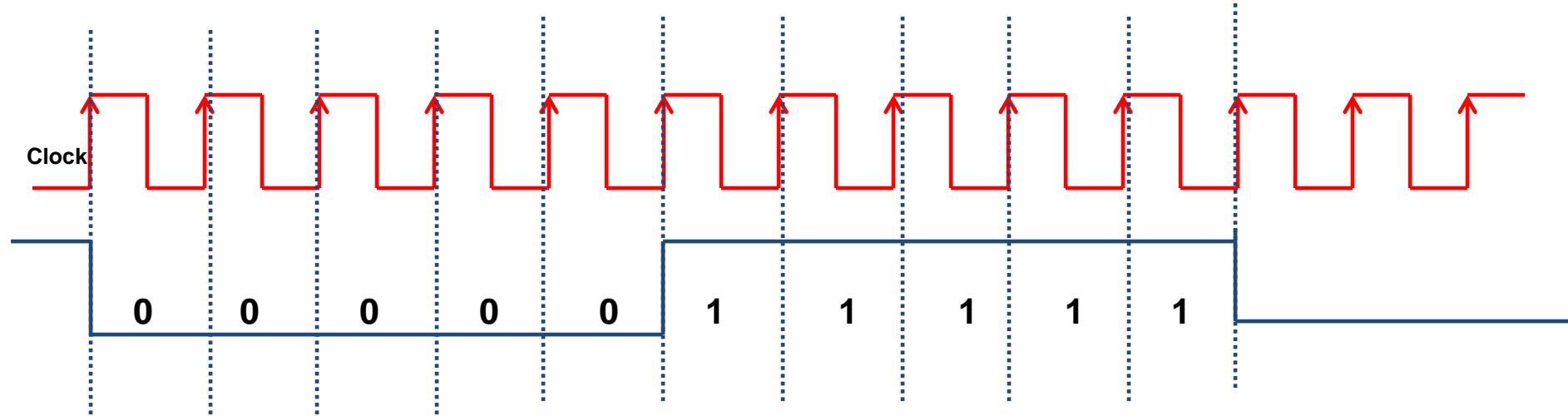


A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

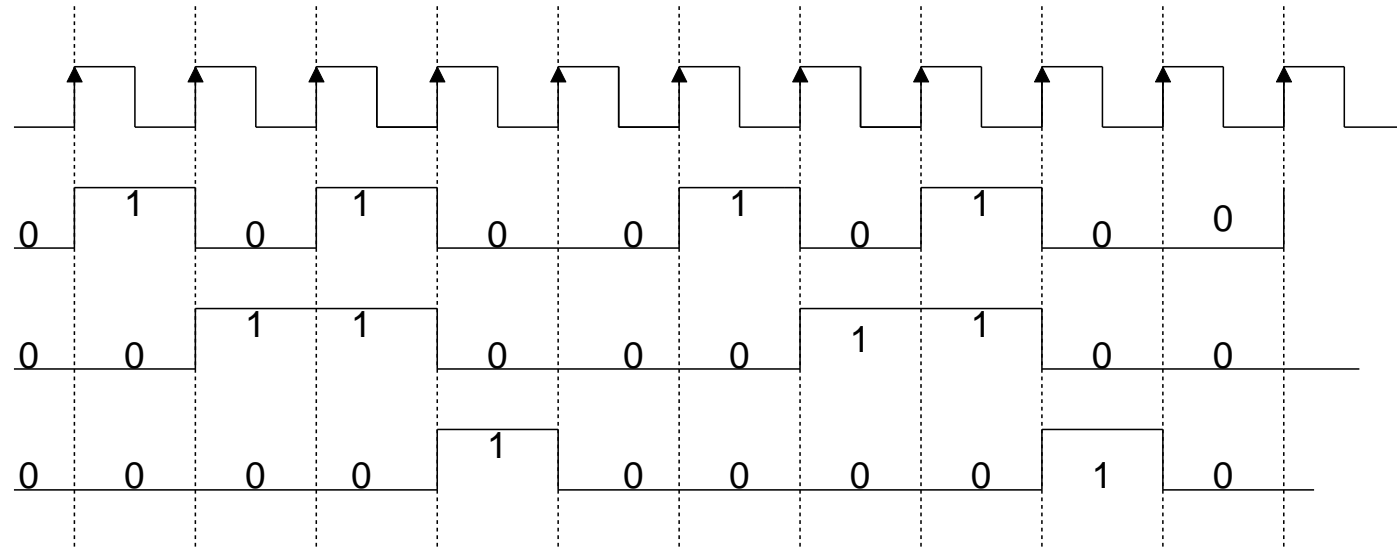
This will have a frequency of 1KHz but it will not have the same waveform

Example

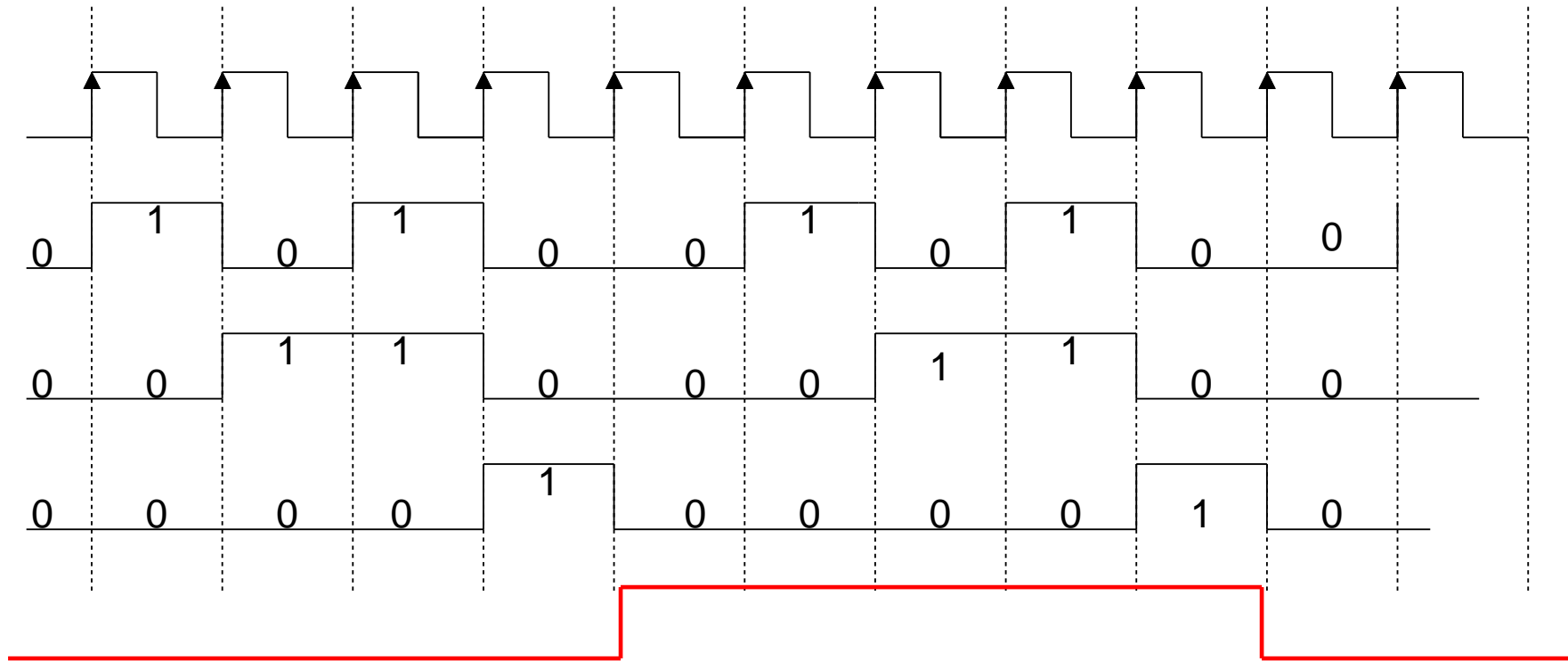
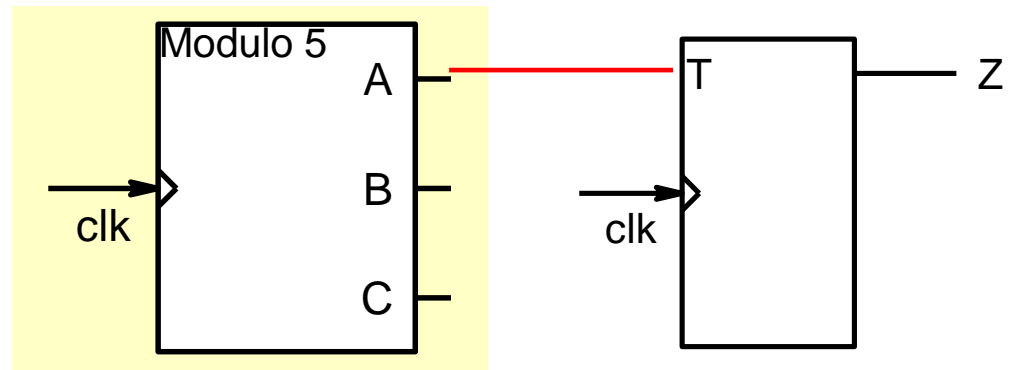
From a frequency of 10KHz, generate a signal of frequency 1KHz

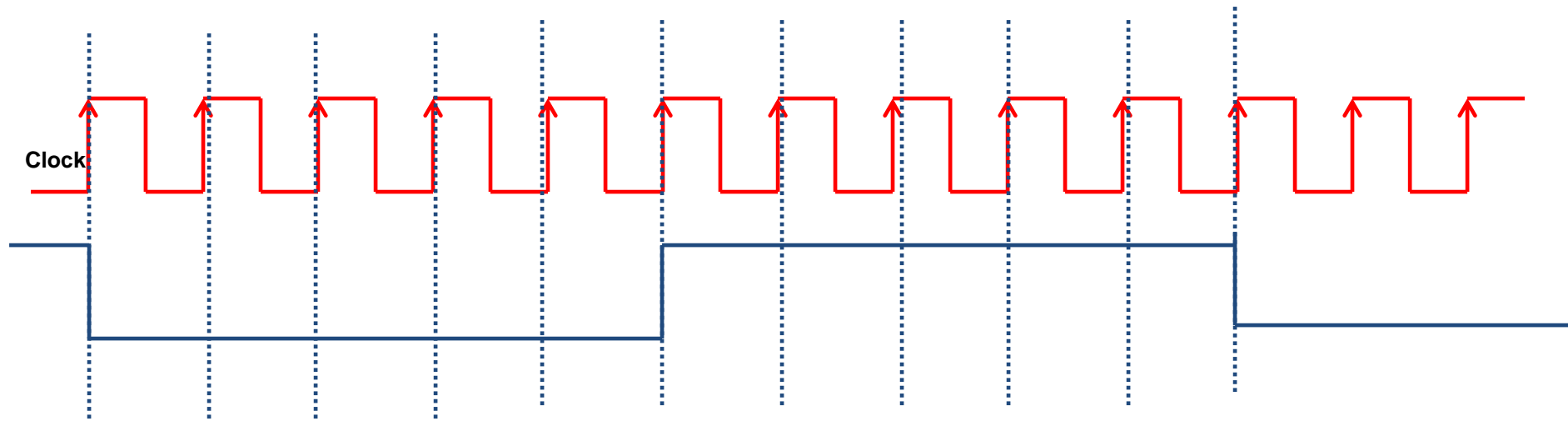


A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0



The idea is to generate a divide by 5 counter first and then divide it again by 2 to get the required waveform

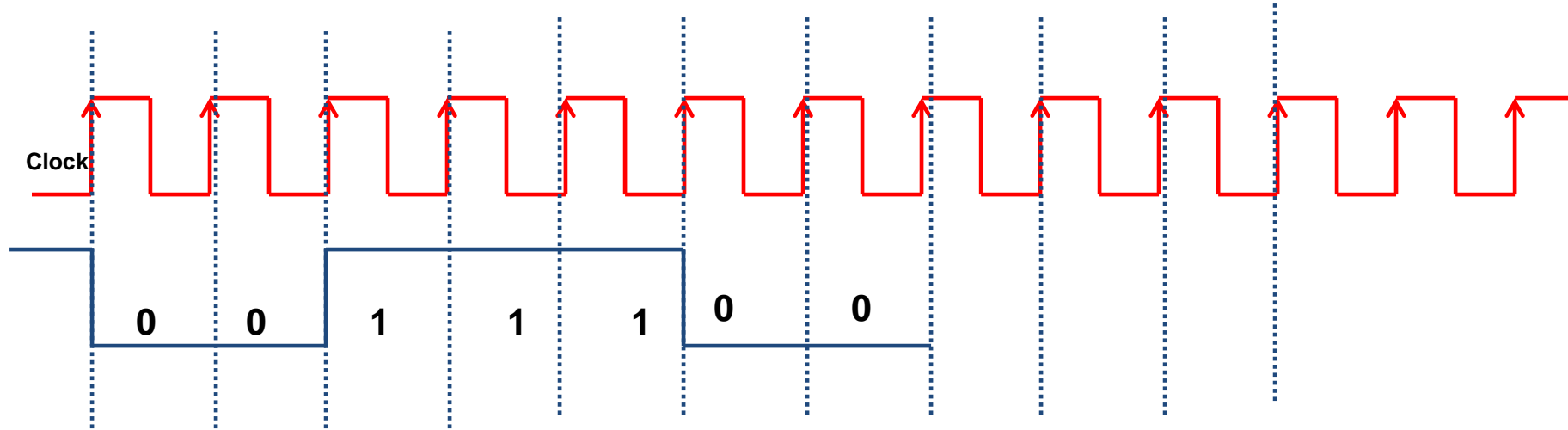




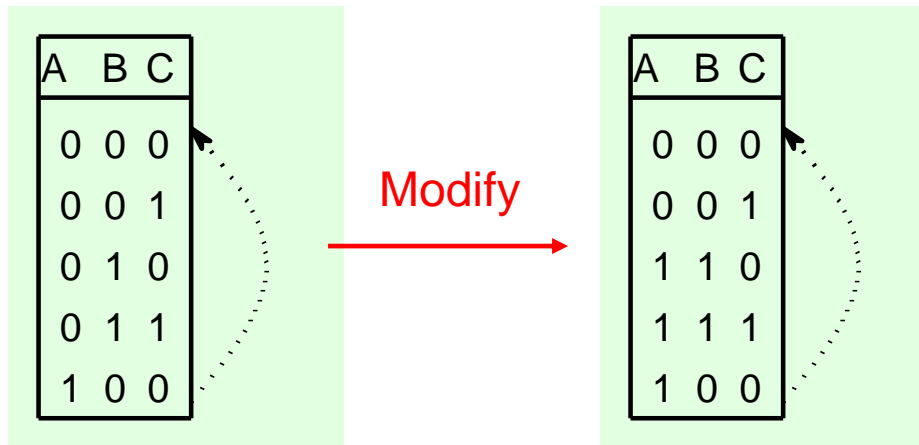
Alternatively design a divide by 10 counter with the following states

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

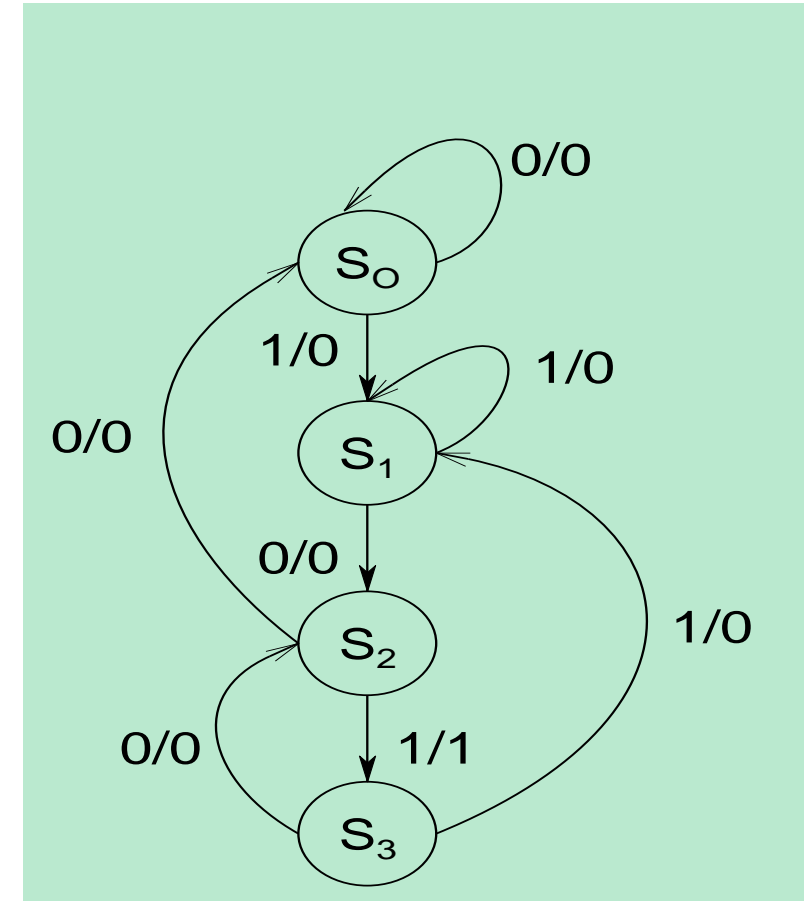
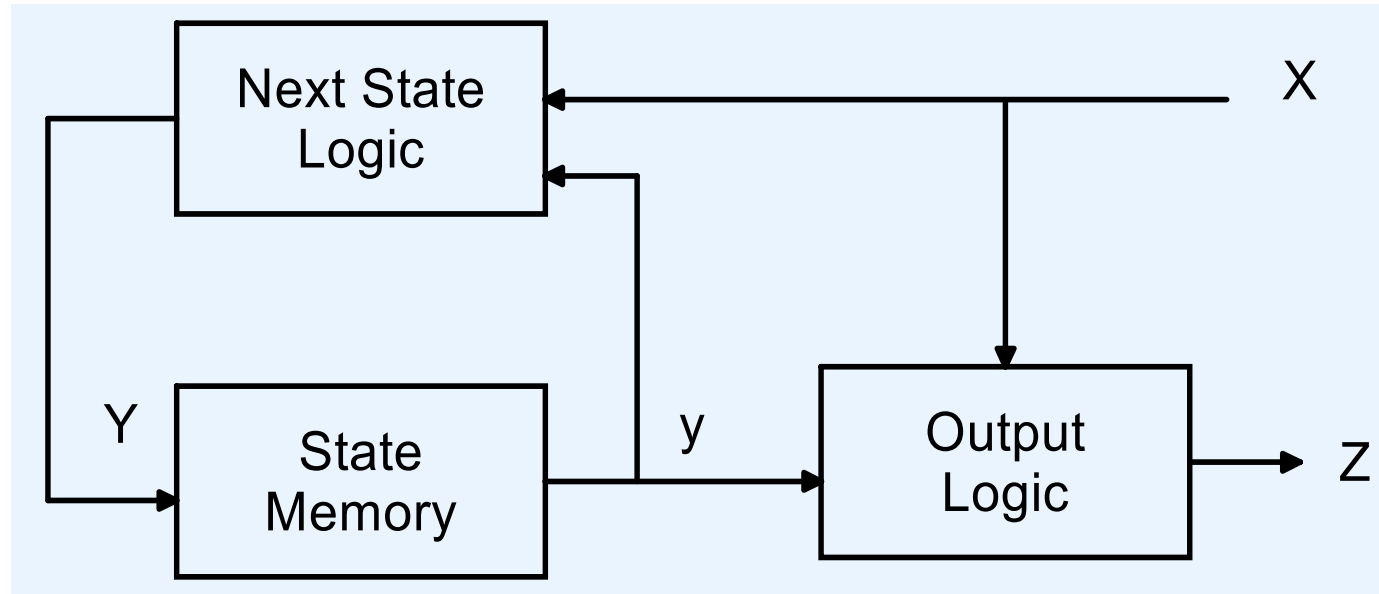


A divide by 5 counter is required that has 5 states.



A will give the required waveform.

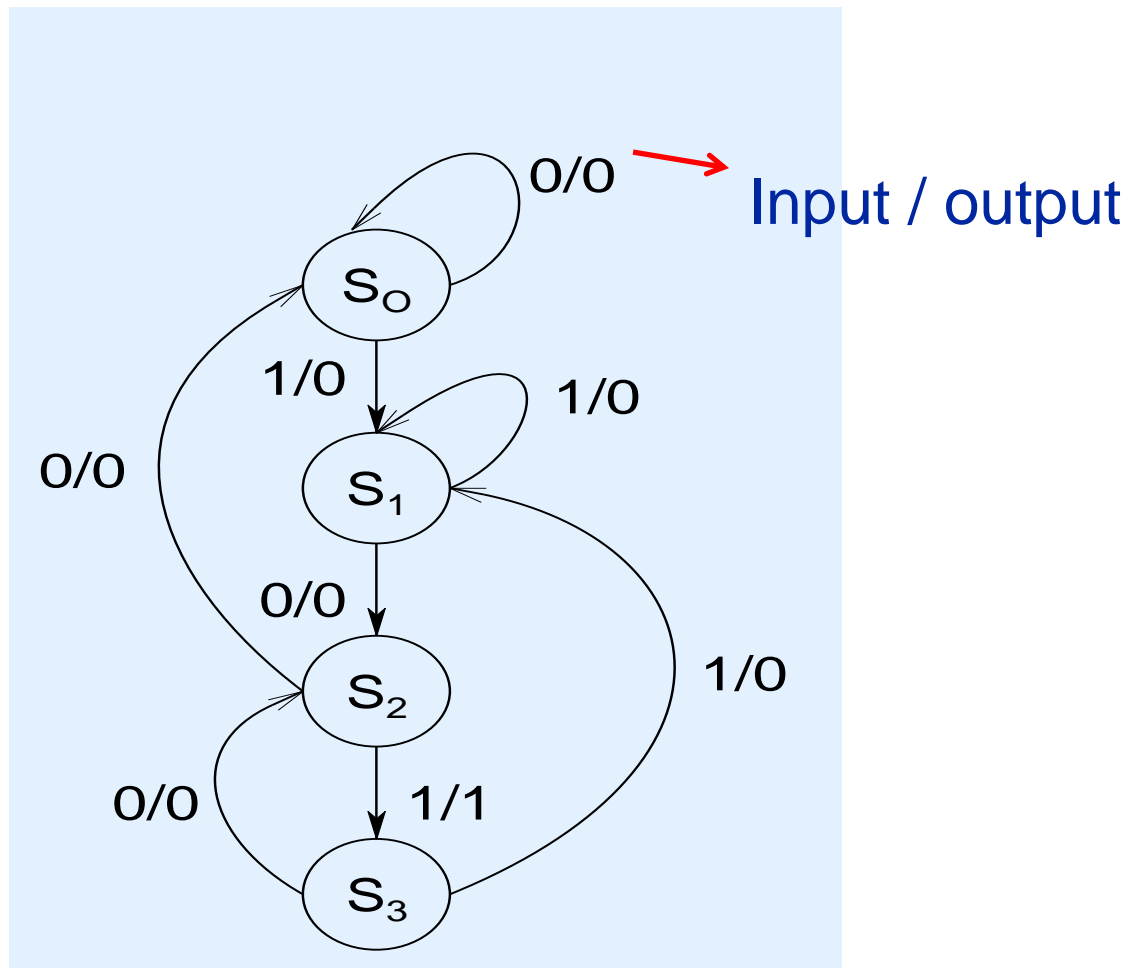
Synthesis of Sequential Machine



- we need to know how the system goes from one state to another in response to the inputs and how the outputs respond to these changes

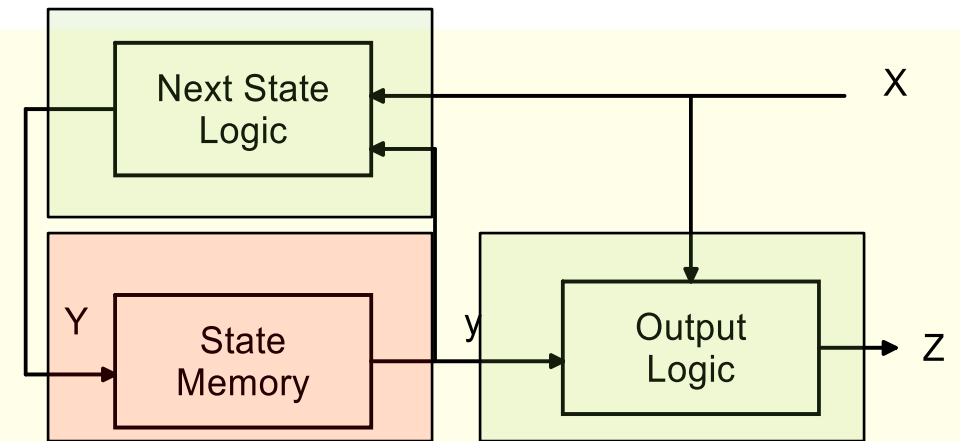
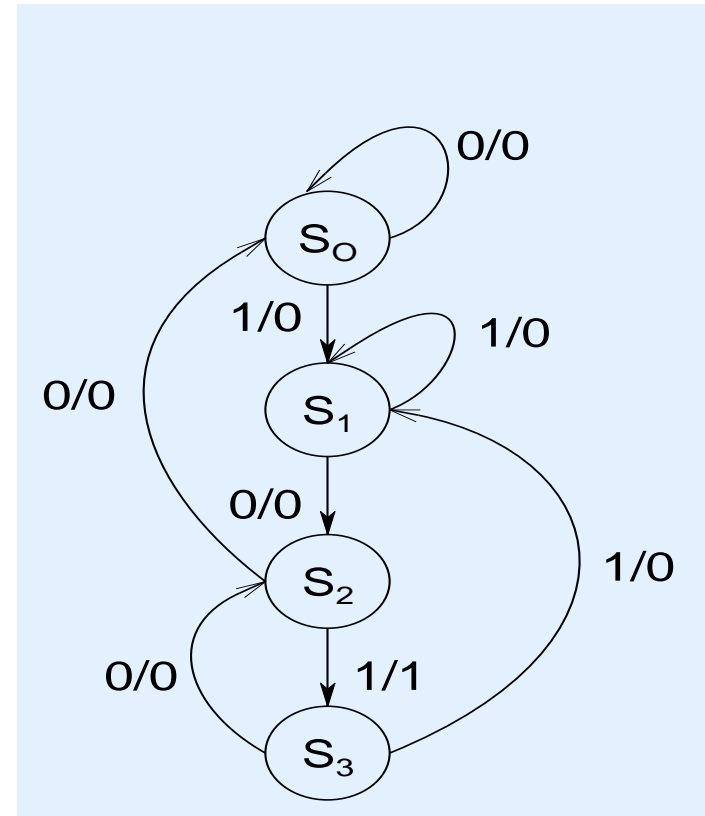
- we need to know how the system goes from one state to another in response to the inputs and how the outputs respond to these changes

State Transition Graph

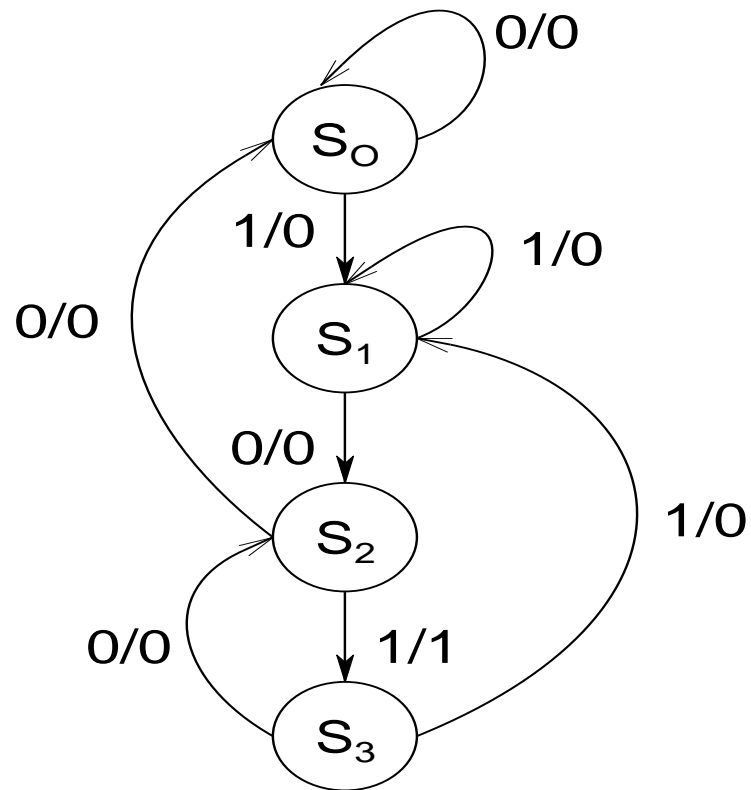
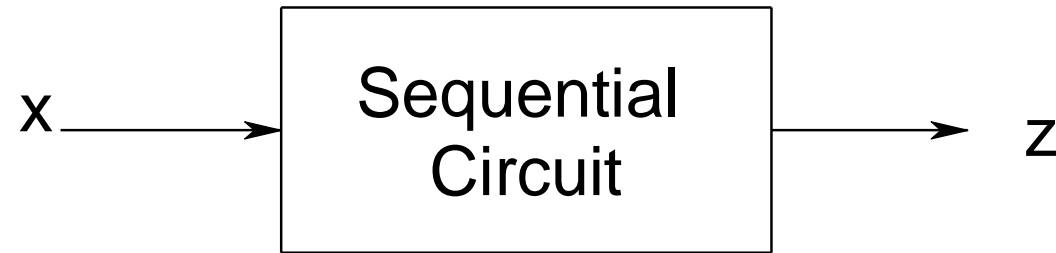


Synthesis involves the following tasks:

- Number of storage elements
- State Encoding
- Choosing a flipflop type to implement states
- Synthesize next state logic
- Synthesize output logic



Example : Detect pattern “101” in the incoming bit Stream.



State Minimization

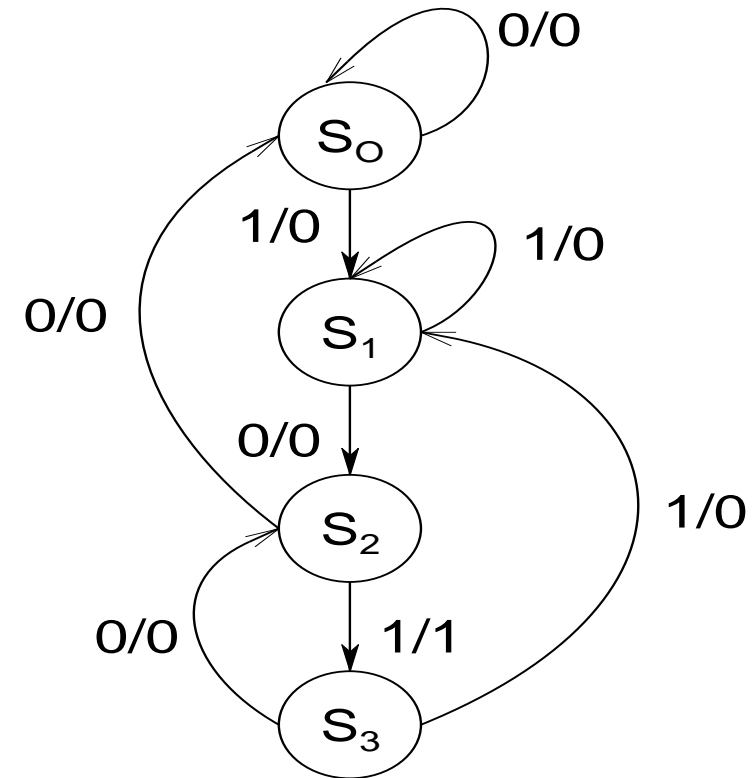
Find equivalent states and eliminate redundancy

Equivalent States : those that have
the **same**

output behavior and

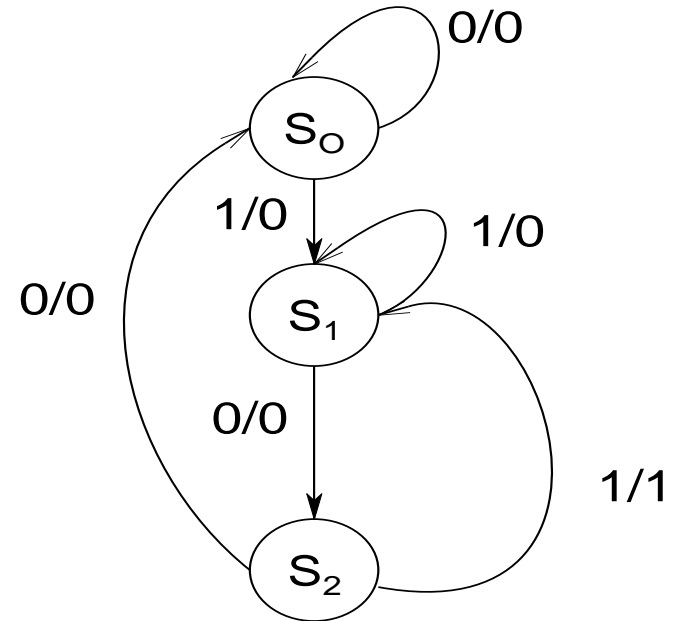
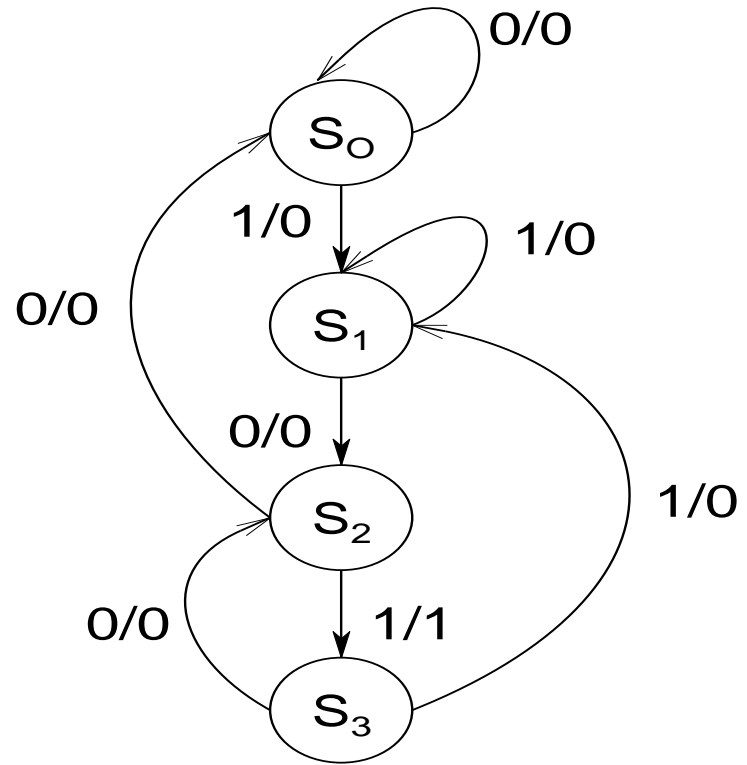
have the same state transitions

for all input combinations.



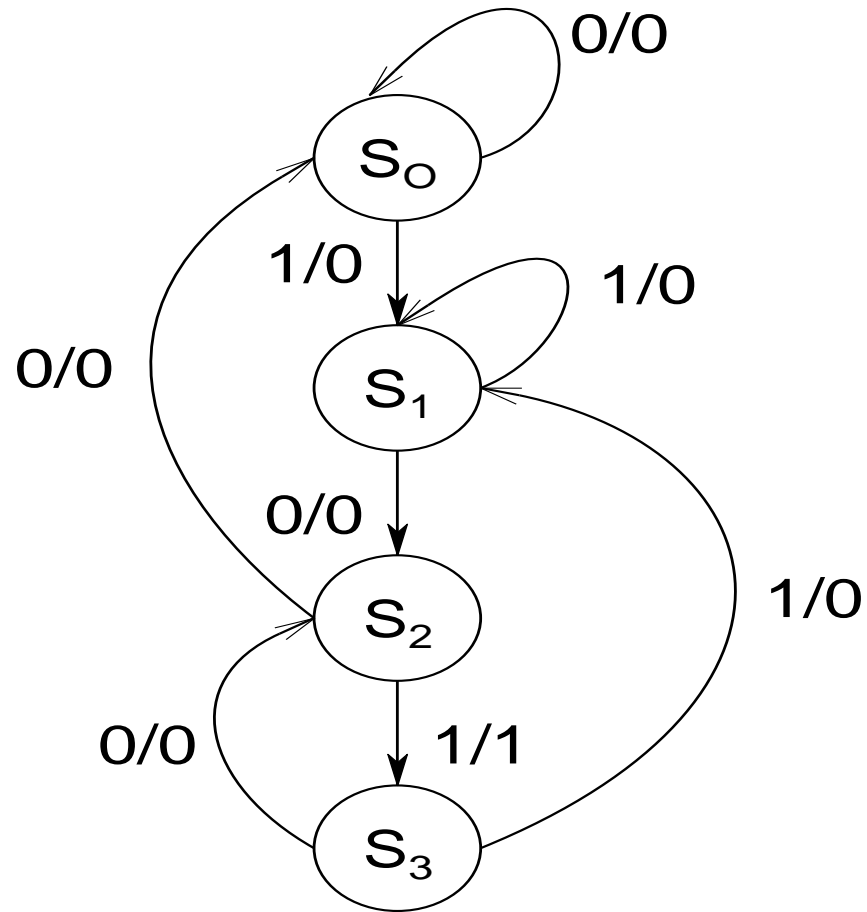
State S_3 and S_1 are equivalent

State Minimization



State Encoding

Attach a bit pattern to the symbolic states



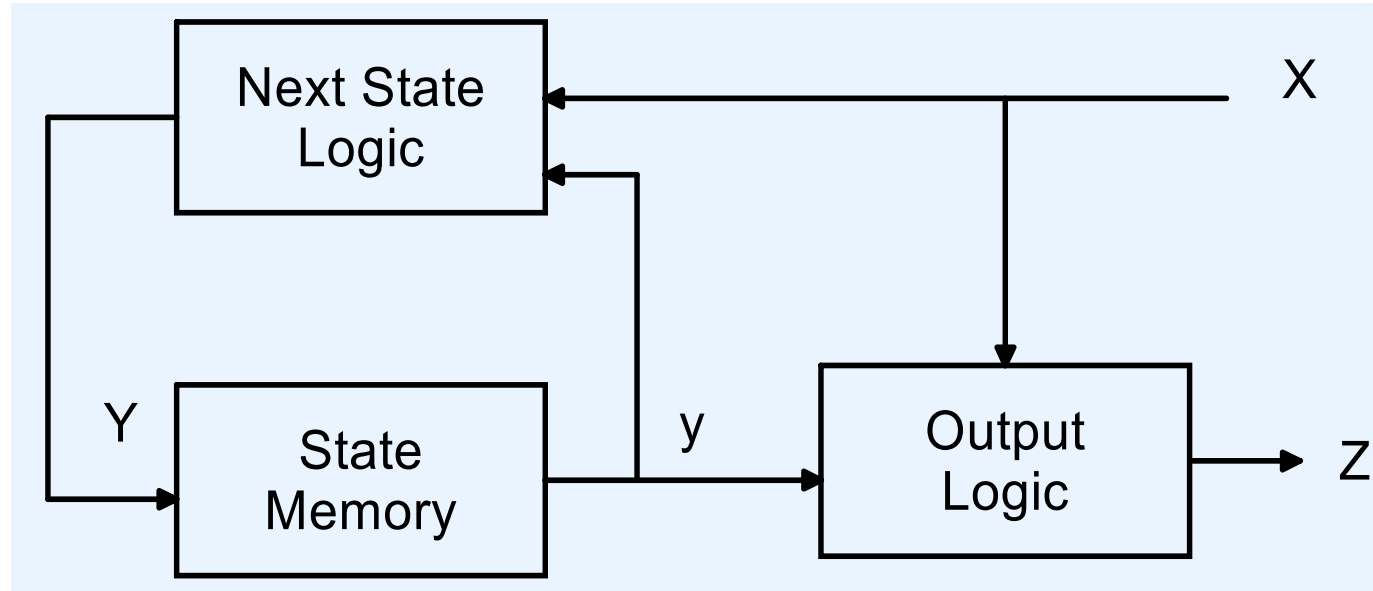
First Question : How many bits?

$$\geq \log_2(\# \text{ of states})$$



2bits?

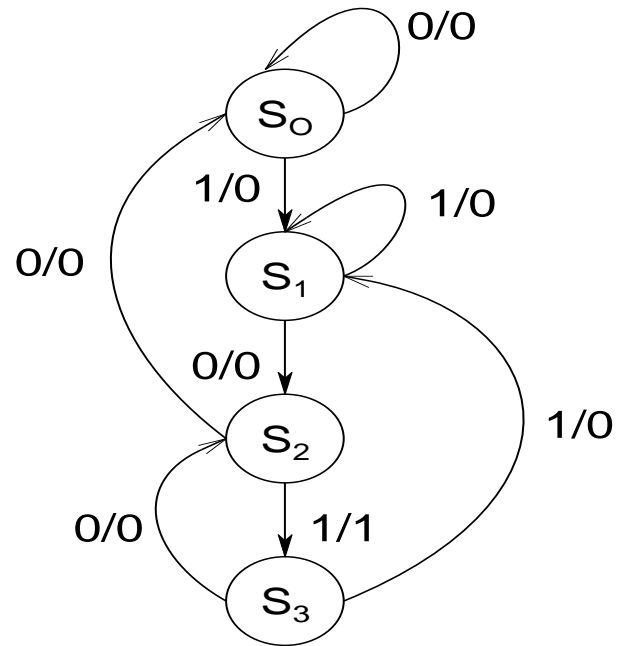
Remember that the goal is not to minimize no. of bits but overall **circuit complexity**



Minimization of no. of bits minimizes only storage but not necessarily combinational logic complexity

State Encoding

Given binary code, how should the code be Assigned to different states?



I

State	Code
S_0	01
S_1	10
S_2	11
S_3	00

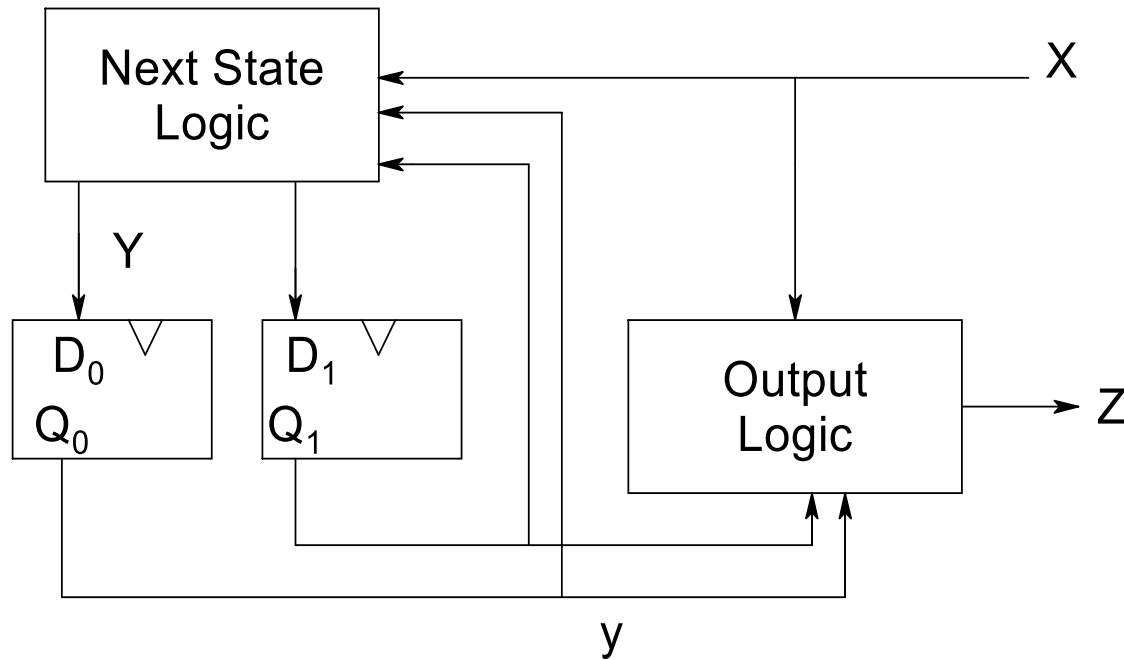
II

State	Code
S_0	00
S_1	01
S_2	10
S_3	11

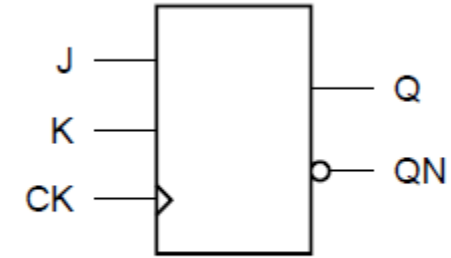
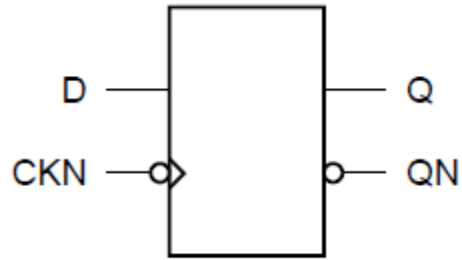
Sequential circuit design : Choose flipflop type

A D FF allows easier implementation

While a JK FF may result in simpler circuit



D vs. JK: area comparison



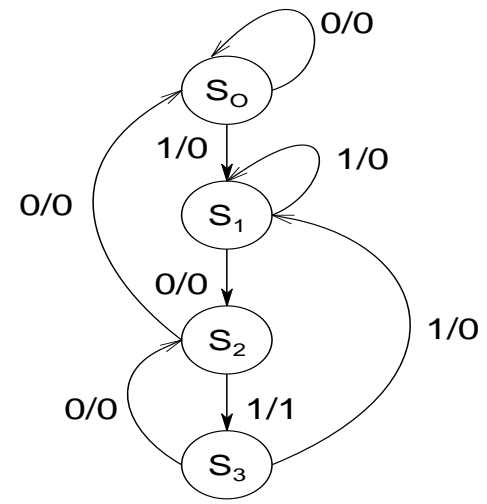
Cell Size

Drive Strength	Height (μm)	Width (μm)
DFFNXL	5.0	11.2
DFFNX1	5.0	11.2
DFFNX2	5.0	13.9
DFFNX4	5.0	15.8

Drive Strength	Height (μm)	Width (μm)
JKFFXL	5.0	13.9
JKFFX1	5.0	13.9
JKFFX2	5.0	16.5
JKFFX4	5.04	19.80

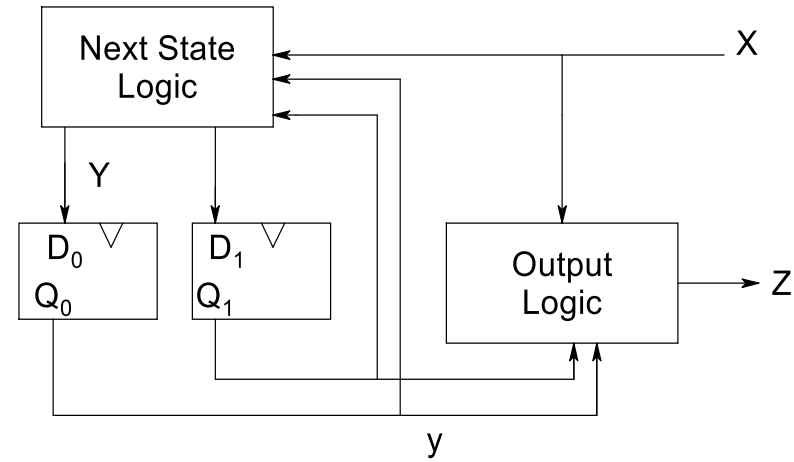
24% larger

Sequential Circuit Design : Truth tables



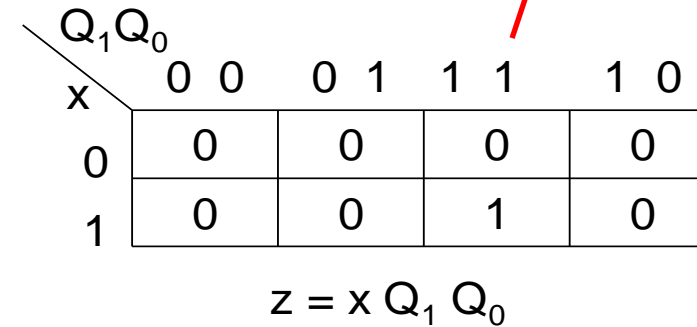
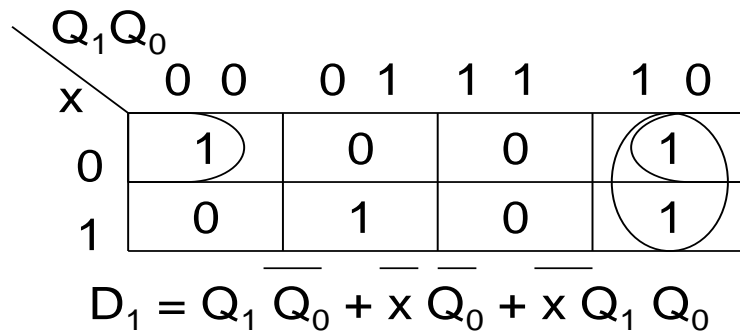
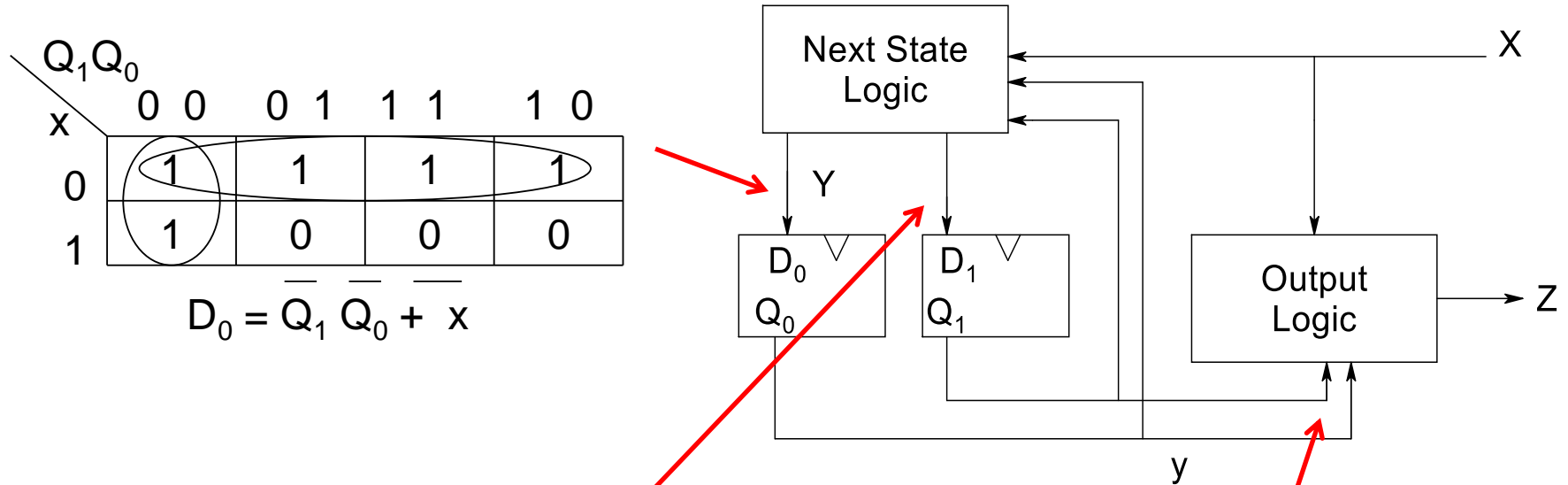
I

State	Q_1Q_0
S_0	01
S_1	10
S_2	11
S_3	00

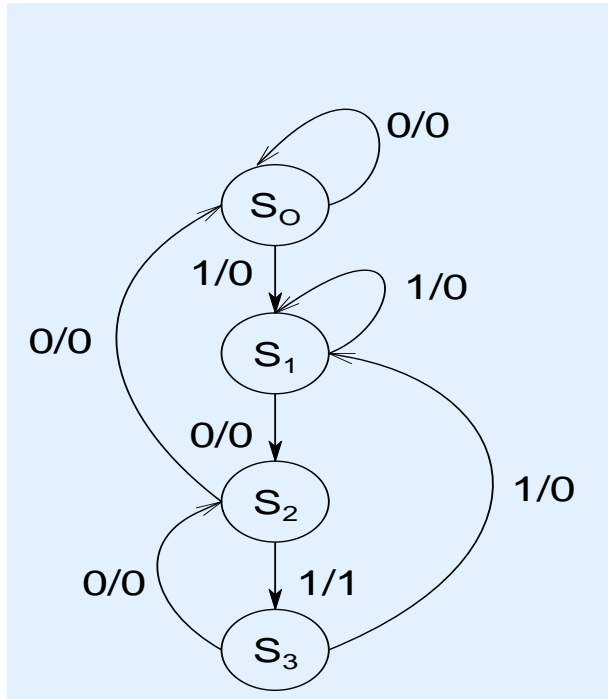
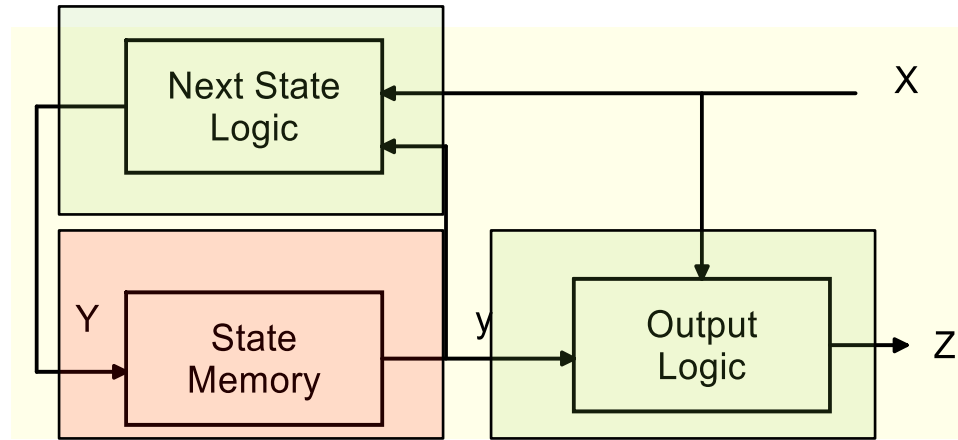


i/p (x)	P. State	N. State	o/p	i/p FF
x	Q_1Q_0	Q_1Q_0	z	D_1D_0
0	0 1	0 1	0	0 1
1	0 1	1 0	0	1 0
0	1 0	1 1	0	1 1
1	1 0	1 0	0	1 0
0	1 1	0 1	0	0 1
1	1 1	0 0	1	0 0
0	0 0	1 1	0	1 1
1	0 0	0 1	0	0 1

Sequential Circuit Design : Logic minimization

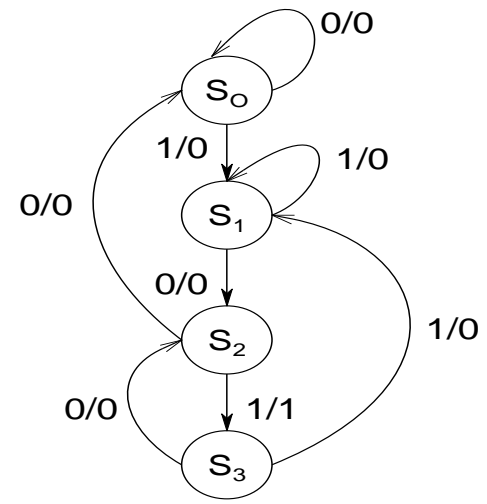


FSM Synthesis



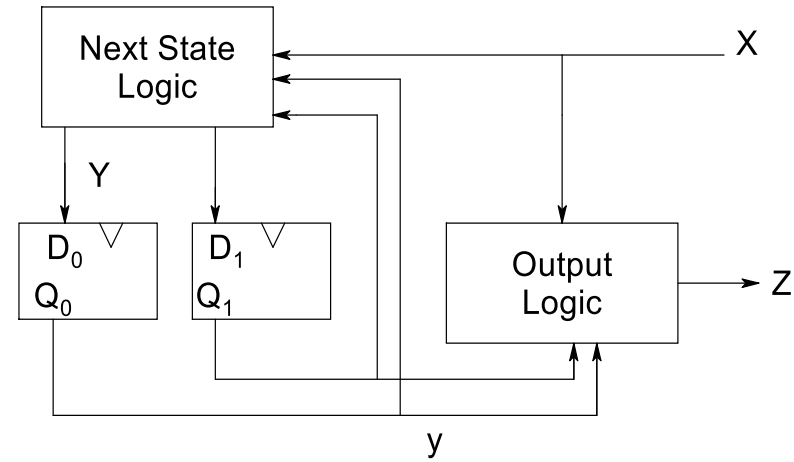
- State minimization
- State Encoding
- Choosing a flipflop type to implement states
- Synthesize next state logic
- Synthesize output logic

Sequential Circuit Design : Truth tables



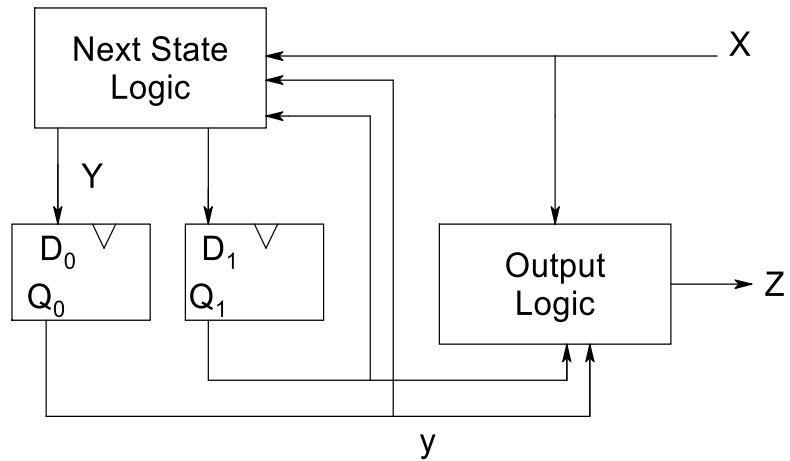
I

State	Q_1Q_0
S_0	01
S_1	10
S_2	11
S_3	00



i/p (x)	P. State	N. State	o/p	i/p FF
x	Q_1Q_0	Q_1Q_0	z	D_1D_0
0	0 1	0 1	0	0 1
1	0 1	1 0	0	1 0
0	1 0	1 1	0	1 1
1	1 0	1 0	0	1 0
0	1 1	0 1	0	0 1
1	1 1	0 0	1	0 0
0	0 0	1 1	0	1 1
1	0 0	0 1	0	0 1

Sequential Circuit Design :combinational logic synthesis



i/p (x)	P. State	N. State	o/p	i/p FF
x	Q_1Q_0	Q_1Q_0	z	D_1D_0
0	0 1	0 1	0	0 1
1	0 1	1 0	0	1 0
0	1 0	1 1	0	1 1
1	1 0	1 0	0	1 0
0	1 1	0 1	0	0 1
1	1 1	0 0	1	0 0
0	0 0	1 1	0	1 1
1	0 0	0 1	0	0 1

Q_1Q_0	0 0	0 1	1 1	1 0
x				
0	1	0	0	1
1	0	1	0	1

$$D_1 = Q_1 \bar{Q}_0 + x \bar{Q}_0 + x \bar{Q}_1 Q_0$$

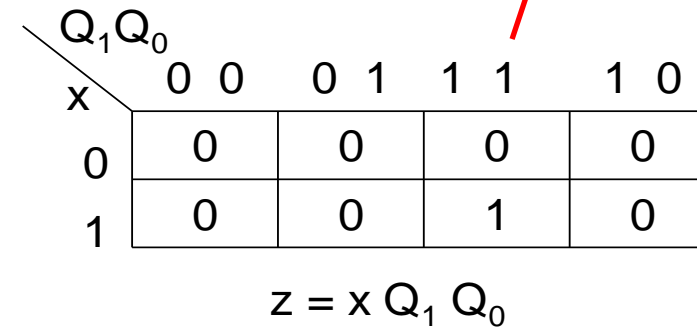
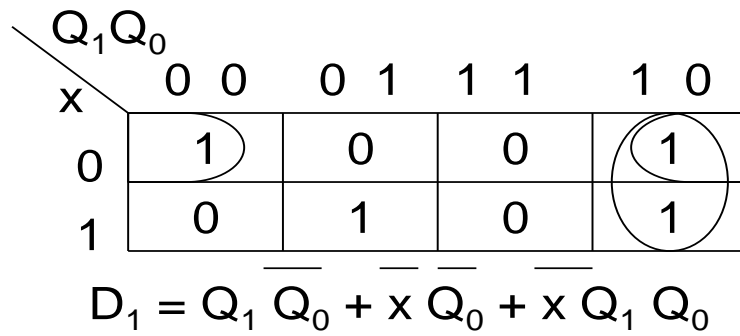
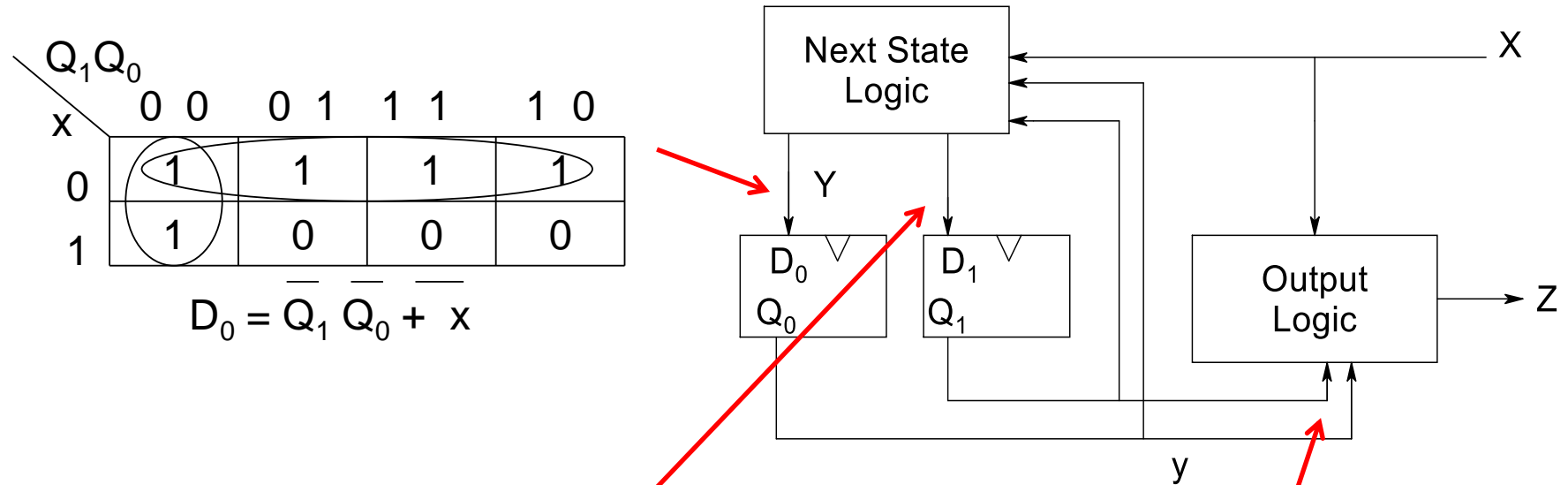
Q_1Q_0	0 0	0 1	1 1	1 0
x				
0	1	1	1	1
1	1	0	0	0

$$D_0 = \bar{Q}_1 \bar{Q}_0 + \bar{x}$$

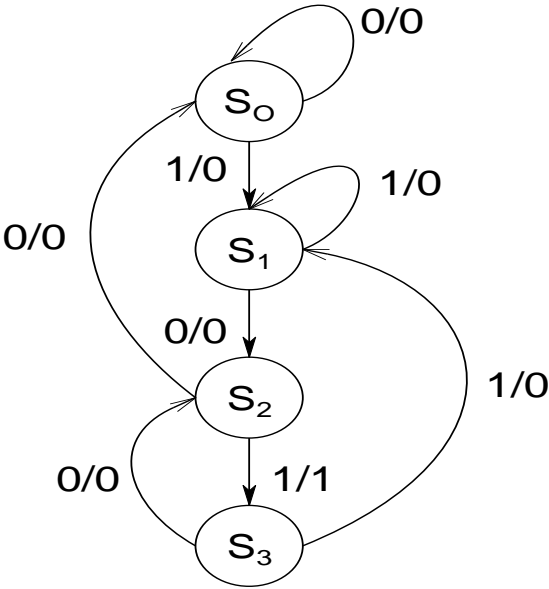
Q_1Q_0	0 0	0 1	1 1	1 0
x				
0	0	0	0	0
1	0	0	1	0

$$z = x Q_1 Q_0$$

Sequential Circuit Design :



Sequential Circuit Design : State coding scheme-II



II

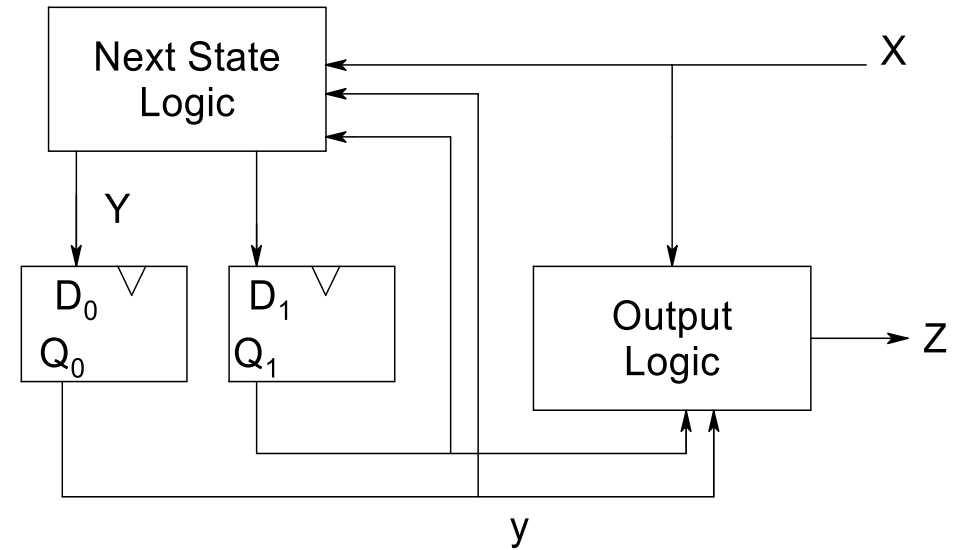
State	Q_1Q_0
S_0	00
S_1	01
S_2	10
S_3	11

i/p (x)	P. State	N. State	o/p	i/p FF
x	Q_1Q_0	Q_1Q_0	z	D_1D_0
0	0 0	0 0	0	0 0
1	0 0	0 1	0	0 1
0	0 1	1 0	0	1 0
1	0 1	0 1	0	0 1
0	1 0	0 0	0	0 0
1	1 0	1 1	1	1 1
0	1 1	1 0	0	1 0
1	1 1	0 1	0	0 1

Sequential Circuit Design : Logic minimization

$Q_1 Q_0$					
x	0 0	0 1	1 1	1 0	
0	0	0	0	0	
1	1	1	1	1	

$$D_0 = x$$



$Q_1 Q_0$					
x	0 0	0 1	1 1	1 0	
0	0	1	1	0	
1	0	0	0	1	

$$D_1 = x \bar{Q}_0 + x \bar{Q}_1 Q_0$$

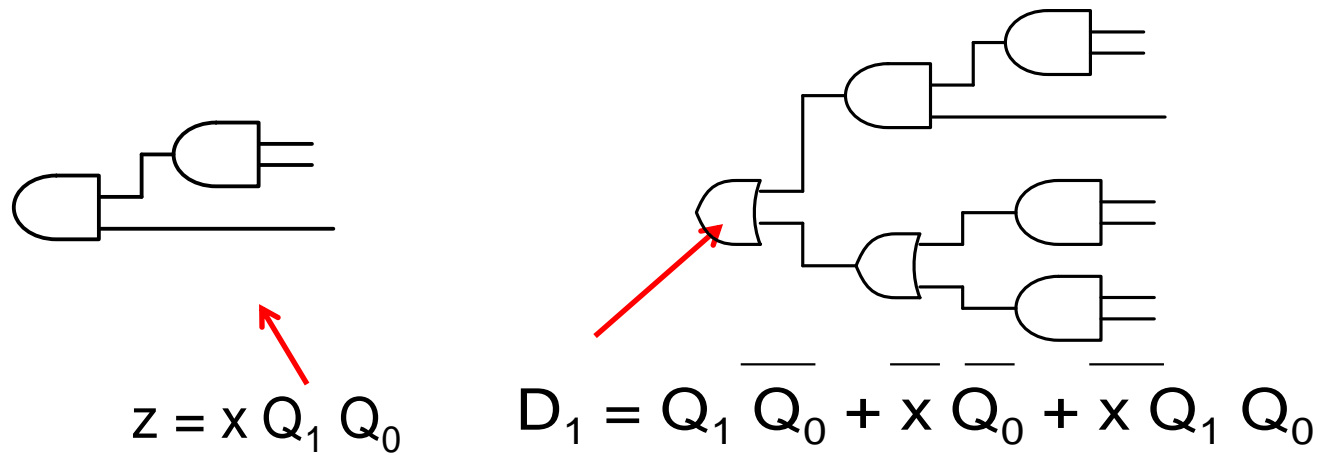
$Q_1 Q_0$					
x	0 0	0 1	1 1	1 0	
0	0	0	0	0	
1	0	0	0	1	

$$z = x \bar{Q}_1 \bar{Q}_0$$

Comparison

I

State	$Q_1 Q_0$
S_0	01
S_1	10
S_2	11
S_3	00



$$D_0 = \overline{Q_1} \overline{Q_0} + x$$

$$\text{Cost} = 2+6+2=10$$

II

State	$Q_1 Q_0$
S_0	00
S_1	01
S_2	10
S_3	11

$$z = x Q_1 \overline{Q_0} \quad D_1 = \overline{x} Q_0 + x Q_1 \overline{Q_0}$$

$$D_0 = x$$

$$\text{Cost} = 2+4+0=6$$