Unit 6

Sequential Logic

Content:

- Flip-flops,
- Triggering of flip-flops
- Design procedure
- Design with state equations and state reduction table.

Introduction:

- Output at any instant of time depends not only on the present inputs but also on past/previous outputs.
- Consists of combinational circuit to which memory elements are connected to form a feedback path.

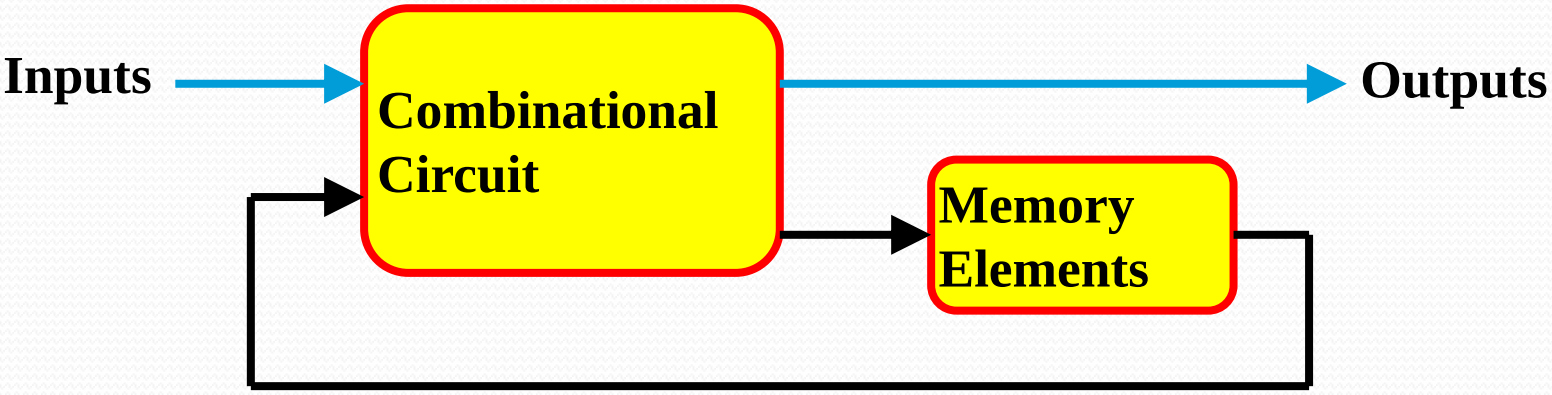


Fig: Block diagram of sequential circuit

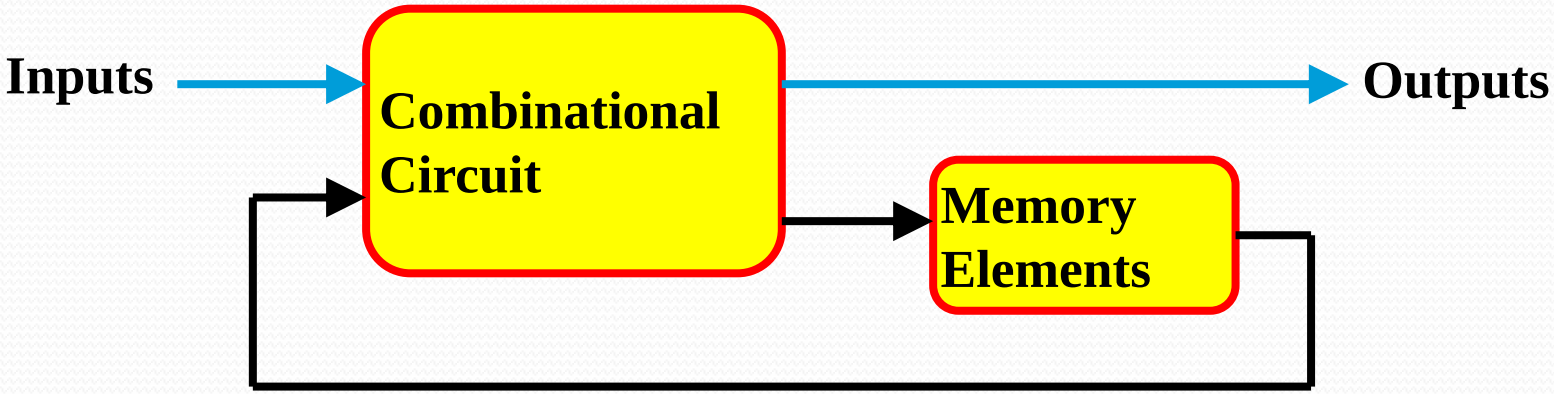
- Memory elements are devices capable of storing binary information within them. The binary information stored in the memory elements at any given time defines the *state* of the sequential circuit.

Types of sequential circuits:

- ***Asynchronous (Unclocked) sequential circuit*:**

➢ A system whose behavior depends upon the order in which its input signals change and can be affected at any instant of time.

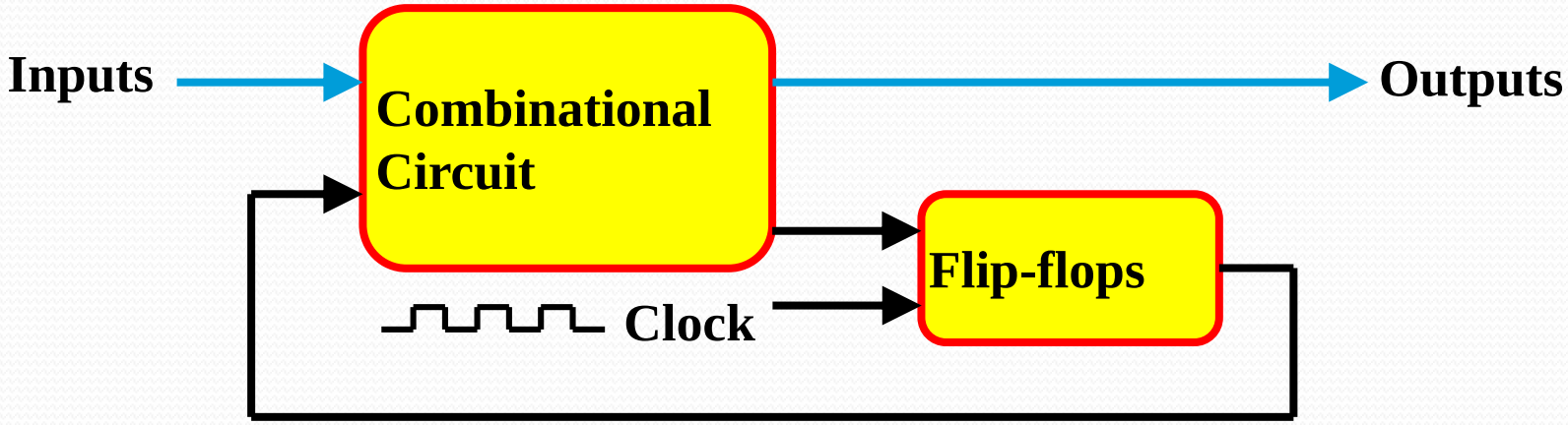
➢ The memory elements commonly used in asynchronous sequential circuits are time-delay devices.



- ***Synchronous (Clocked) Sequential Circuit*:**

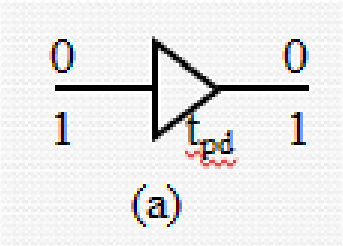
➢ A system whose behavior can be defined from the knowledge of its signals at discrete instants of time.

➢ Synchronous sequential circuits that use clock pulses in the inputs of memory elements are called ***clocked sequential circuits.***

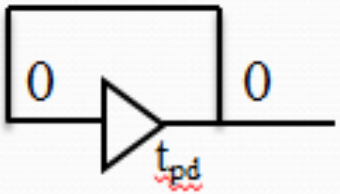


Information storage in digital system

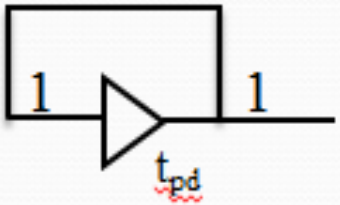
Fig (a) shows a buffer which has a propagation delay t_{pd} and can store information for time t_{pd} since buffer input at time t reaches to its output at time t_{pd}. But, in general, we wish to store information for an indefinite time that is typically much longer than the time delay of one or even many gates. This stored value is to be changed at arbitrary times based on the inputs applied to the circuit and should not depend on the specific time delay of a gate.



(a)



(b)

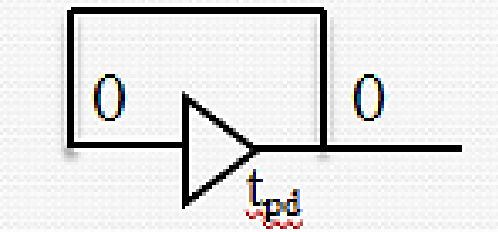


(c)

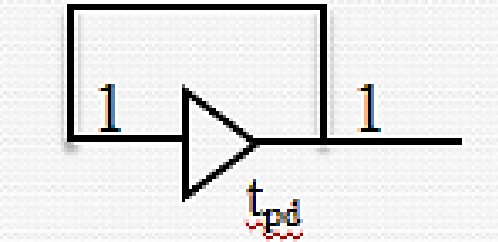
In Fig (b) we have output of buffer connected to its input making a feedback path. This time input to buffer has been 0 for at least time t_{pd}. Then the output produced by the buffer will be 0 at time $t + t_{pd}$. This output is applied to the input so that the output will also be 0 at time $t + 2\ t_{pd}$. This relationship between input and output holds for all t, so the 0 will be stored indefinitely.

A buffer is usually implemented by using two inverters, as shown in Fig (d). The signal is inverted twice, i.e. (X')' = X

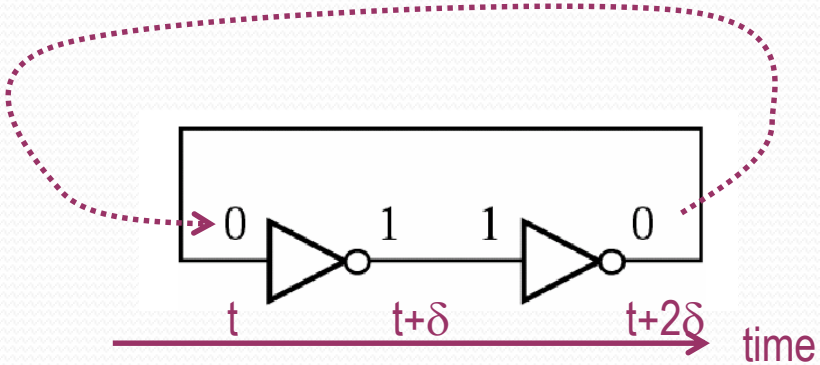
In fact, this example is an illustration of one of the most popular methods of implementing storage in computer memories. With inverters there is no way to change the information stored. By replacing the inverters with NOR or NAND gates, the information can be changed. Asynchronous storage circuits called latches are made in this manner.

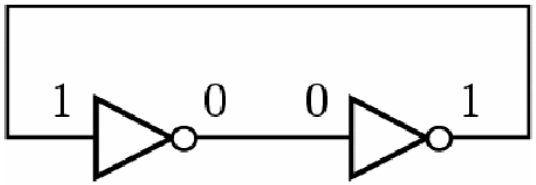


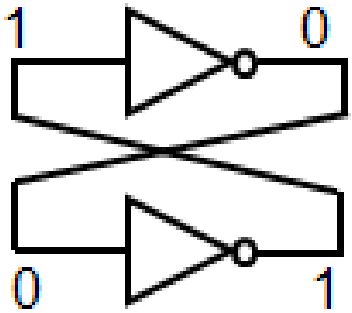
(b)



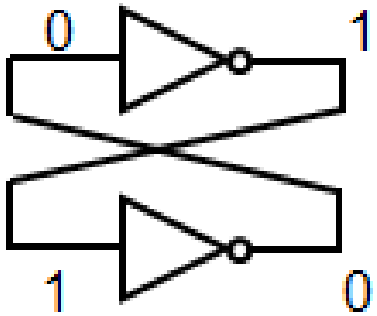
(c)







0-state



1-state

Fig (d)

Memory Devices:

- ***Latches*** A *latch* is a memory element whose excitation signals control the state of the device. A latch has two stages *set* and *reset*. *Set* stage sets the output to 1. *Reset* stage set the output to 0.

- ***Flip-flops*** A *flip-flop* is a memory device that has clock signals control the state of the device. They flip from one state to another and then flop back.

Flip-Flops

- The latch with the additional control input is called Flip-Flops
- The additional control input is either clock or enable input.
- The memory elements used in clocked sequential circuits are called *Flip-Flops.*
- These circuits are binary cells capable of storing one bit of information.
- A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it.
- Flip Flops are of different types depending on how their inputs and clock pulses causes transition between two states. They are
    i. RS Flip Flop
    ii. JK Flip Flop
    iii. D Flip Flop
    iv. T Flip Flop

Basic flip-flop circuit (RS Latch)

- Using two NAND gates or two NOR gate.
- The cross-coupled connection from the output of one gate to the input of the other gate constitutes a feedback path.
- Each flip-flop has two outputs: Q and Q'(the normal and complement outputs, respectively), and two inputs: *set(S)* and *reset(R).*

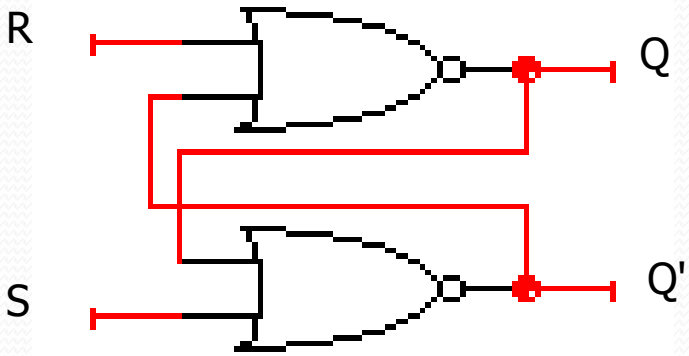
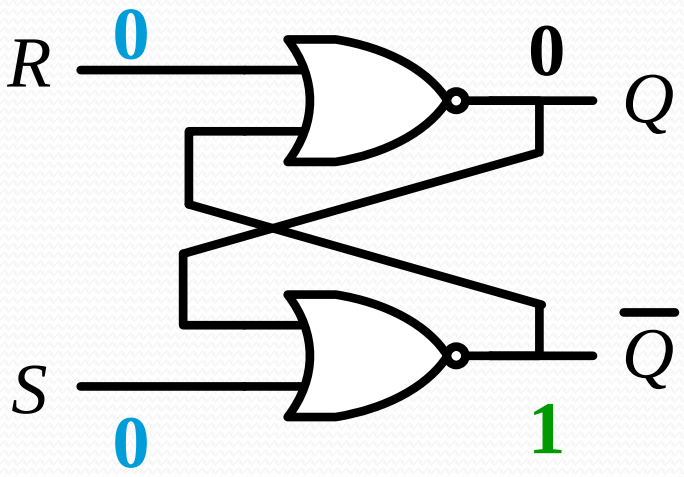


Fig: Logic Diagram of Basic flip-flop circuit with NOR gates

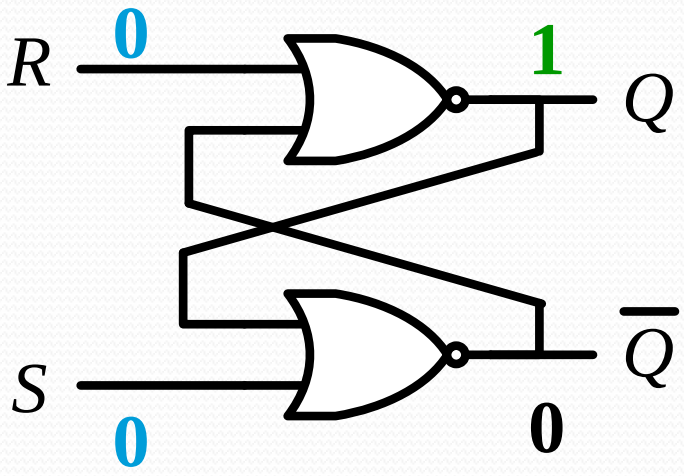
SR or RS Latch



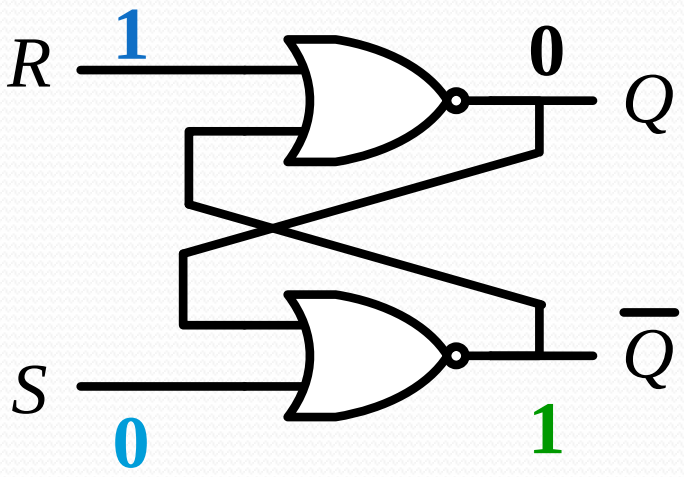
S	R	Q_0	Q	Q'
0	0	0	0	1

$Q = Q_0$

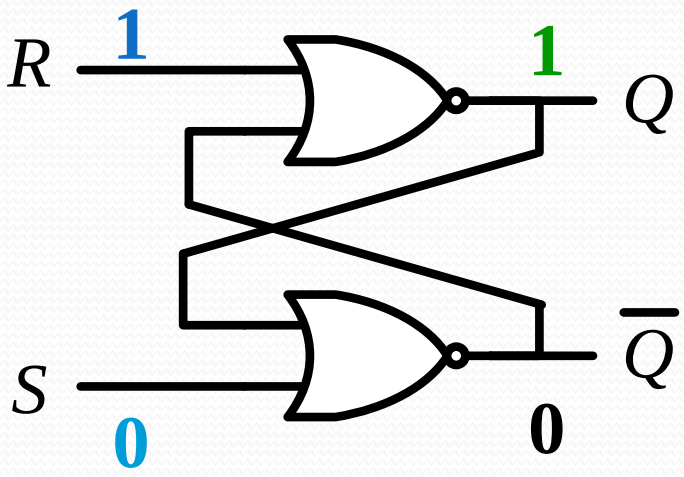
Initial Value



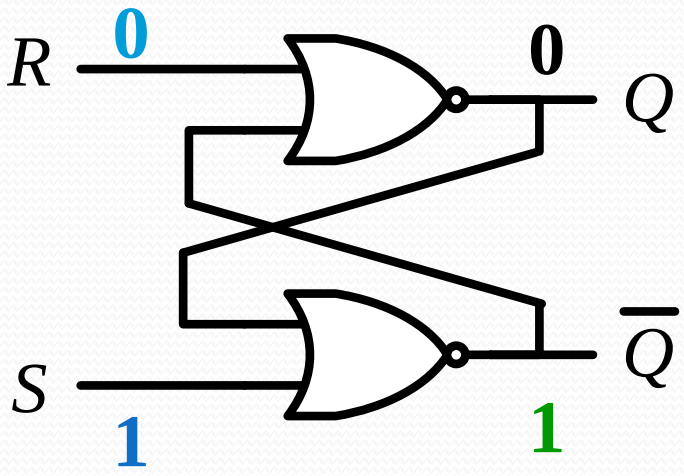
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	$Q = Q_0$



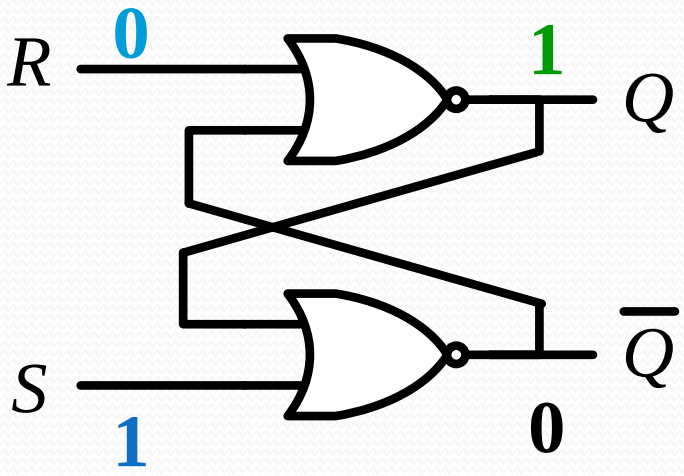
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$



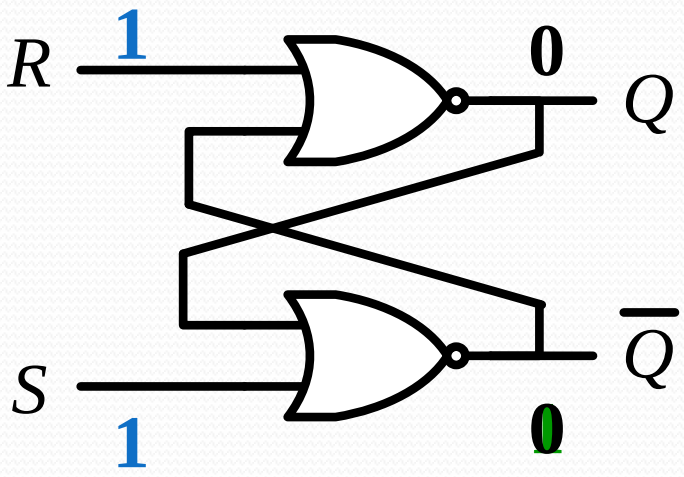
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	$Q = 0$



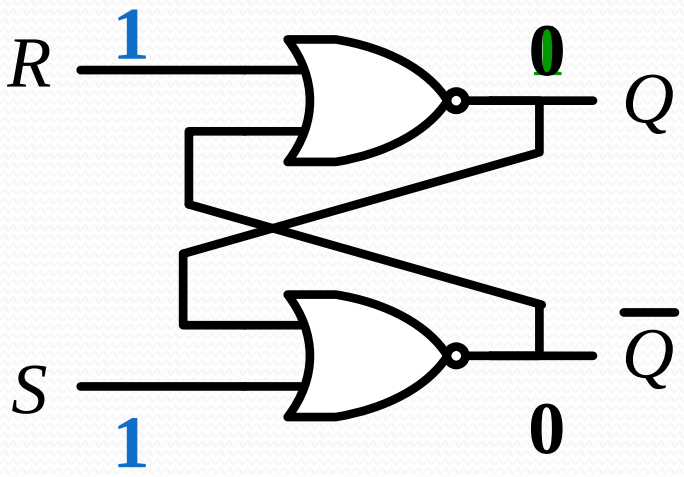
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$



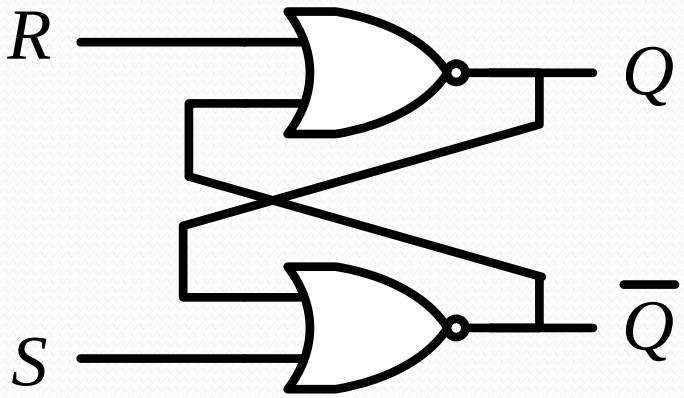
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	$Q = 1$



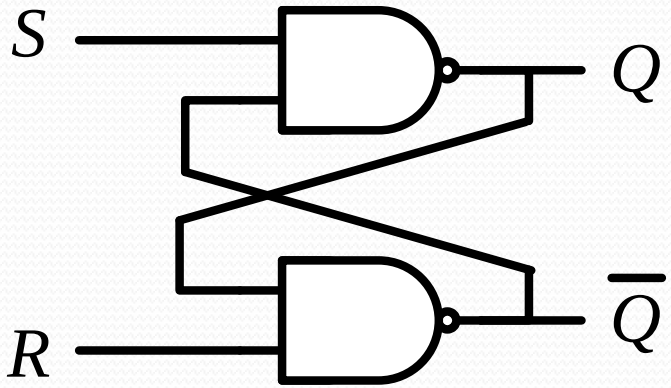
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$



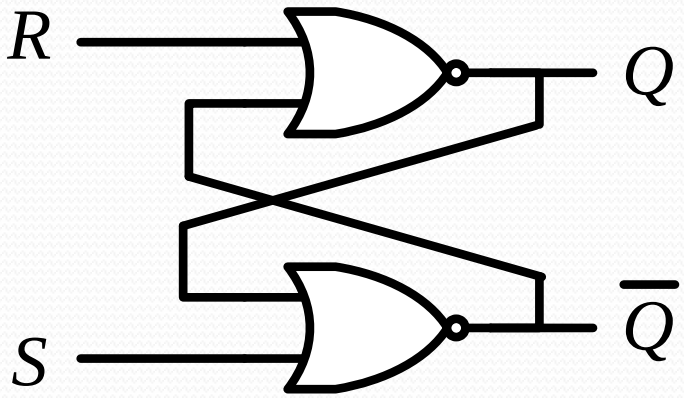
S	R	Q_0	Q	Q’	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q’$
1	1	1	0	0	$Q = Q’$



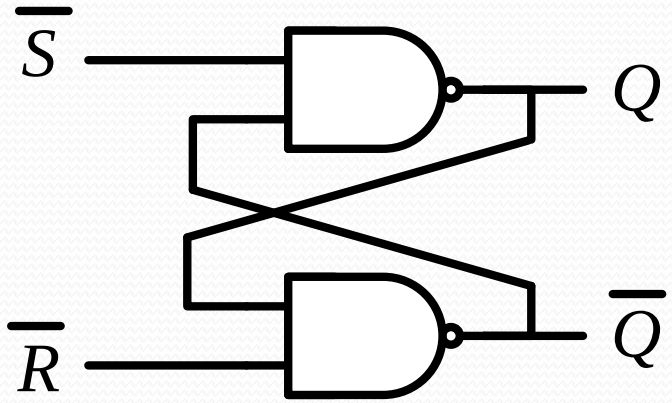
S	R	Q	
0	0	Q_0	No change
0	1	0	Reset
1	0	1	Set
1	1	Q=Q'=0	Invalid



S	R	Q	
0	0	Q=Q'=1	Invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_0	No change



S R	Q	
0 0	Q_0	No change
0 1	0	Reset
1 0	1	Set
1 1	Q=Q'=0	Invalid



S'R'	Q	
0 0	Q=Q'=1	Invalid
0 1	1	Set
1 0	0	Reset
1 1	Q_0	No change

R	S	Q_{t+1}	Remarks
0	0	Q_t	No Change(Hold State)
0	1	1	**Set State**
1	0	0	**Reset State**
1	1	?	Indeterminate(Unstable State)

Where,

Q_t= Present Output and Q_{t+1} = Next Output

Thus,

A flip-flop has two useful states.

- **Set state**: When Q = 1 and Q' = 0, (or 1-state or Set State)
- **Reset state**: When Q = 0 and Q' = 1, (or 0-state or Reset or Clear State)

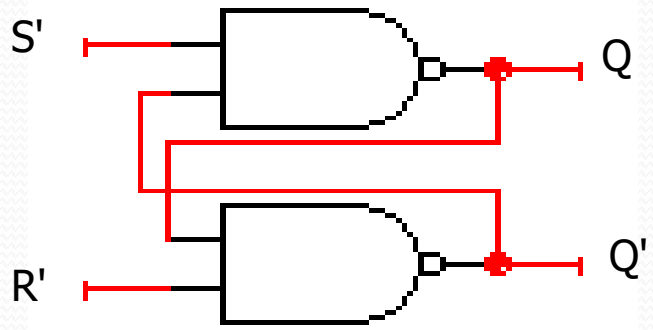


Fig: Logic Diagram of Basic flip-flop circuit with NAND gates

R'	S'	R	S	Q_{t+1}	Remarks
1	1	0	0	Q_t	No Change(Hold)
1	0	0	1	1	**Set**
0	1	1	0	0	**Reset**
0	0	1	1	?	(Unstable)

Where,

Q_t= Present Output and Q_{t+1} = Next Output

The operation of the basic flip-flop can be modified by providing an additional control input that determines when the state of the circuit is to be changed. This fact arises 4 common types of flip-flops and are discussed in what follows:

1. Clocked RS Flip-Flop

It consists of a basic RS flip-flop circuit and two additional NAND gates along with clock pulse (C) input. The pulse input acts as an enable signal for the other two inputs.

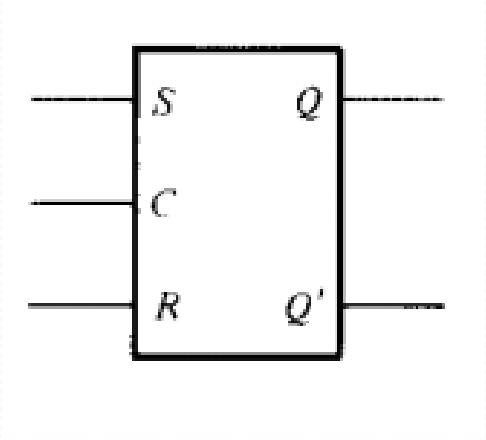
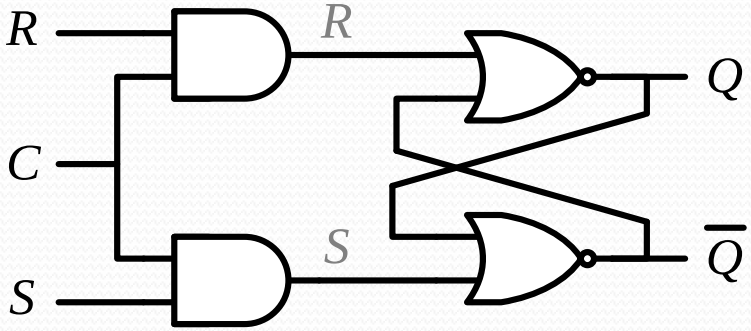
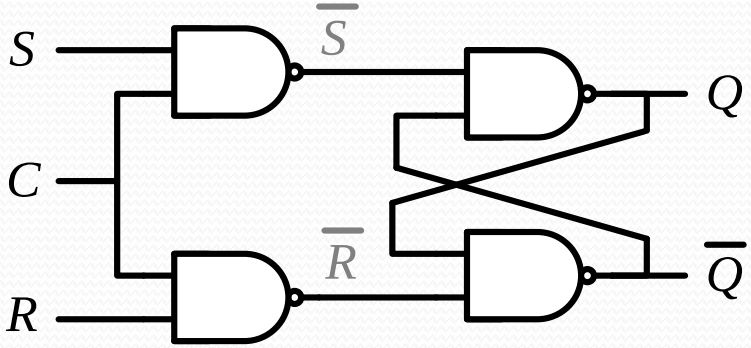


Fig: Logic Symbol

- ***SR* Flip Flop with Control Input or Clocked RS Flip-Flop**





C S R	Q	
0 x x	Q_0	No change
1 0 0	Q_0	No change
1 0 1	0	Reset
1 1 0	1	Set
1 1 1	Q=Q’	Invalid

- When *CP* returns to 0, the circuit remains in its previous state.
- When the pulse input goes to 1, information from the S or *R* input is allowed to reach the output.
- Set state: S = 1, *R* = 0, and *CP* = 1.
- Reset state: S = 0, *R* = 1, and *CP* = 1.
- When *CP* = 1 and both the S and *R* inputs are equal to 0, the state of the circuit does not change.

From K-map of R and S, we get

Characteristics Equation: $\mathbf{Q_{t+1} = S + Q_t R'}$

Or

When CP=1 :

Q_t	R	S	Q_{t+1}	Remarks
0	0	0	0	No Change
0	0	1	1	Set
0	1	0	0	Reset
0	1	1	?	Indeterminate
1	0	0	1	No Change
1	0	1	1	Set
1	1	0	0	Reset
1	1	1	?	Indeterminate

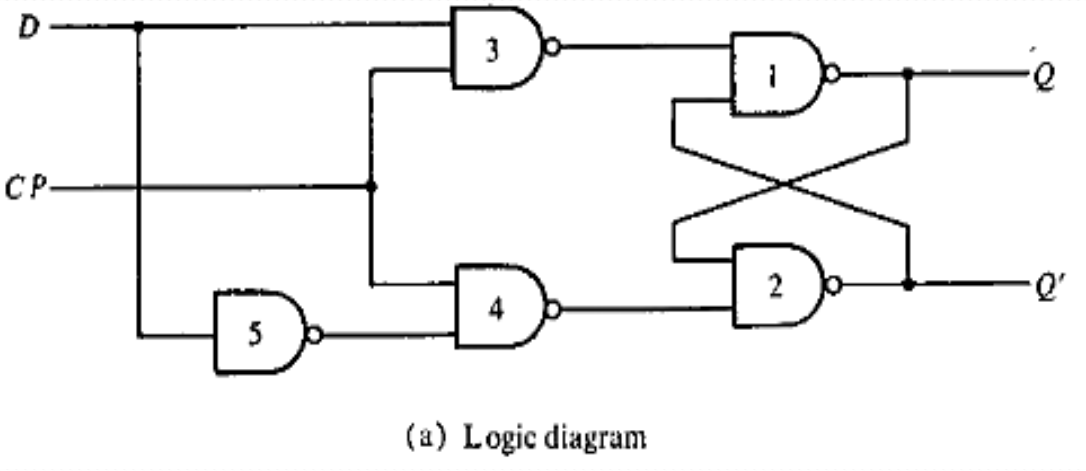
From K-map

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

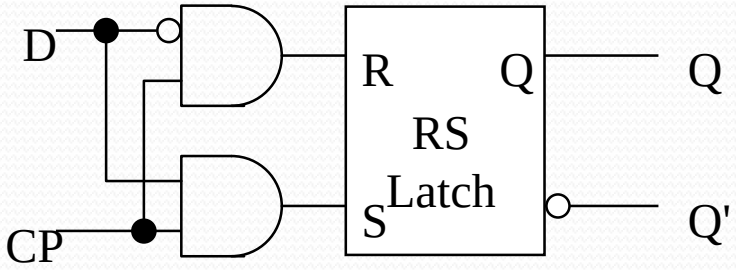
Note: Indeterminate State is as Don’t Care Condition

2. Clocked D Flip-Flop (D= Data)

One way to eliminate the undesirable condition of the indeterminate state in the *RS* flip-flop is to ensure that inputs S and *R* are never equal to 1 at the same time. This is done in the *D* flip-flop. The *D* flip-flop has only two inputs: *D* and *CP*. The *D* input goes directly to the S input and its complement is applied to the *R* input.



(a) Logic diagram



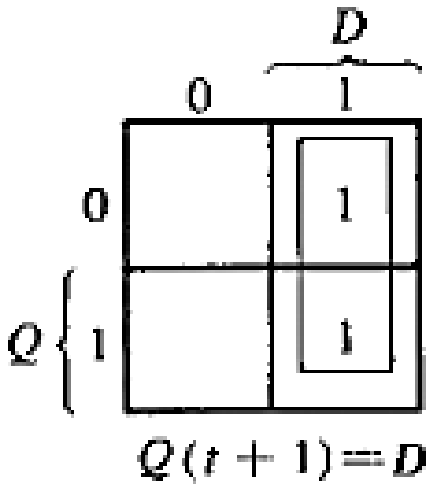
- When *CP* or C = 0, the circuit remains in its previous state.
- The *D* input is sampled when *CP* = 1.

➢ If *D* is 1, *Q* = 1, placing the circuit in the **set state**.

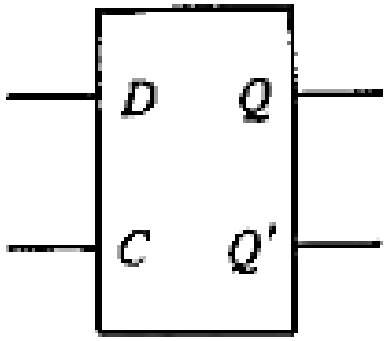
➢ If *D* is 0, Q= 0 and the circuit switches to the **clear state**.

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

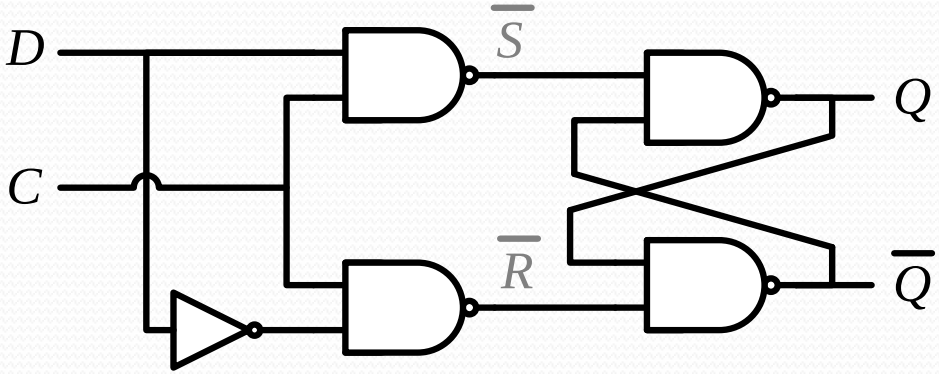
(b) Characteristic table



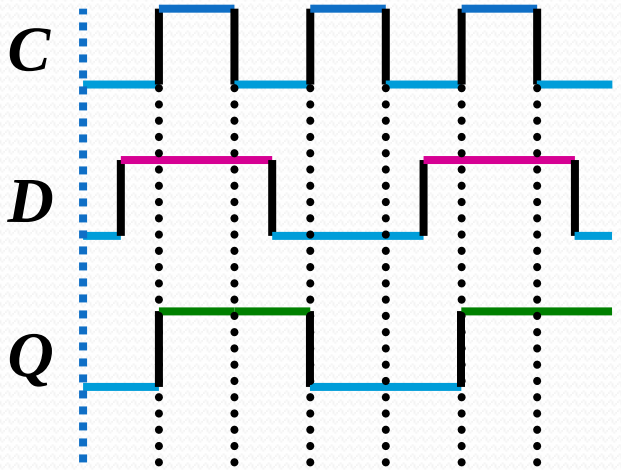
(c) Characteristic equation



(d) Graphic symbol



Timing Diagram



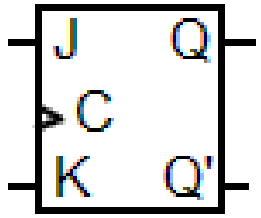
C D	Q	
0 x	Q_0	No change
1 0	0	Reset
1 1	1	Set

3. JK flip-flop/Clocked JK Flip-Flop

- A *JK* flip-flop is a refinement of the *RS* flip-flop in that the indeterminate state of the *RS* type is defined in the *JK* type.
- Inputs *J* and *K* behave like inputs *S* and *R* to set and reset the flip-flop, respectively.
- When both inputs *J* and *K* are equal to 1, the flip-flop switches to its complement state(toggle state), that is, if $Q = 1$, it switches to $Q = 0$, and vice versa.
- When $C=0$, the circuit remains in its previous state.
- When $C= 1$.

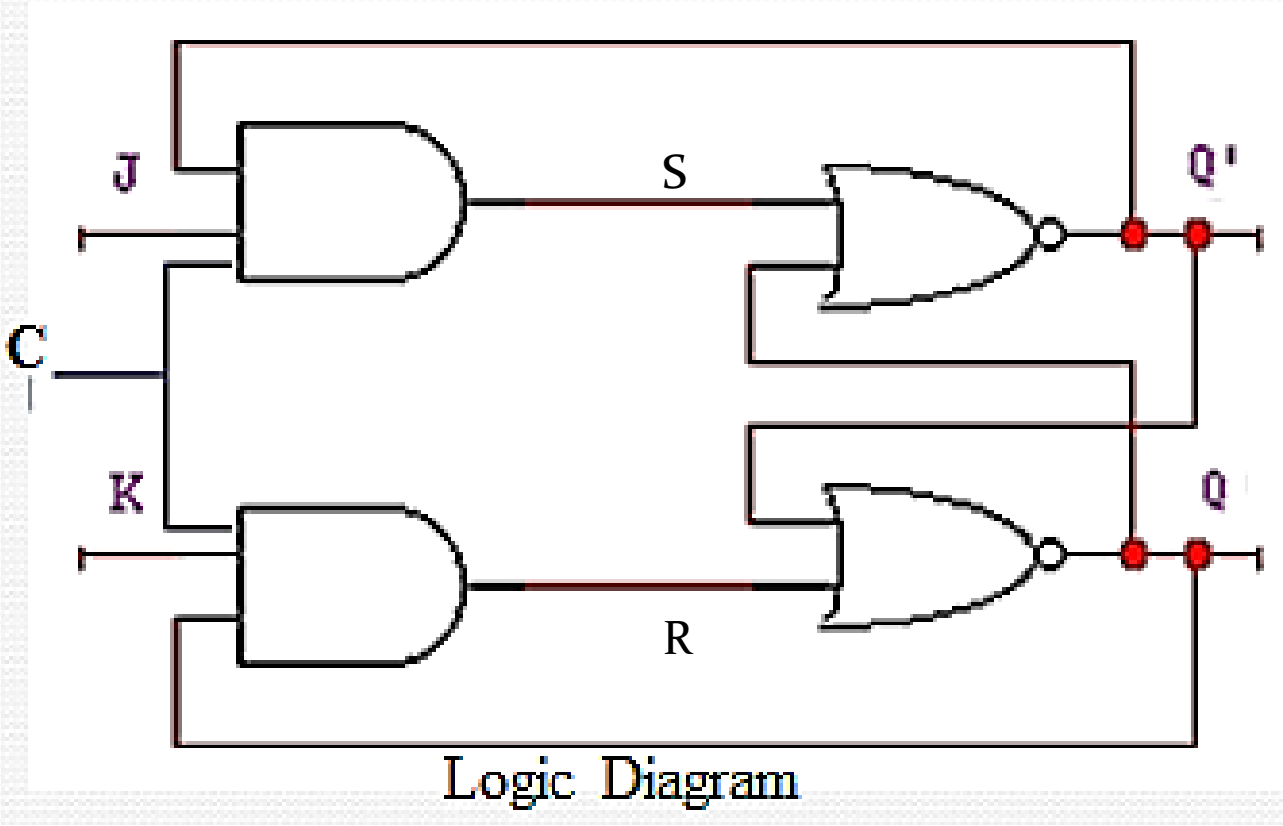
➢ If $J=1$, $Q = 1$, placing the circuit in the **set state**.

➢ If $K=0$, $Q= 0$ and the circuit switches to the **clear state**.



Graphical Symbol

Because of the feedback connection in the *JK* flip-flop, a *C* pulse that remains in the 1 state while both *J* and *K* are equal to 1 will cause the output to complement again and repeat complementing until the pulse goes back to 0.

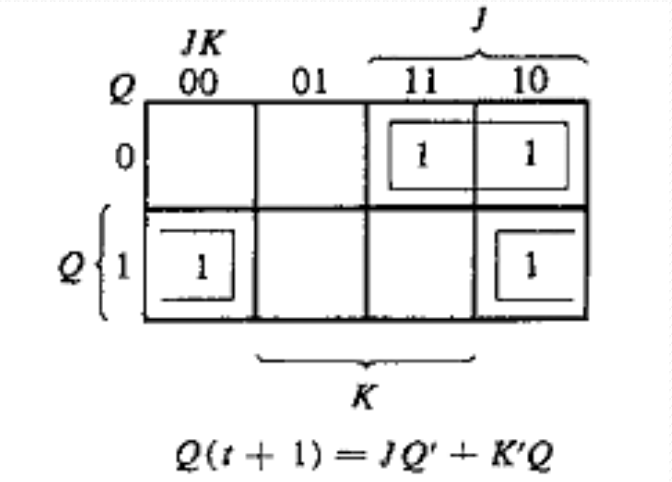


Logic Diagram

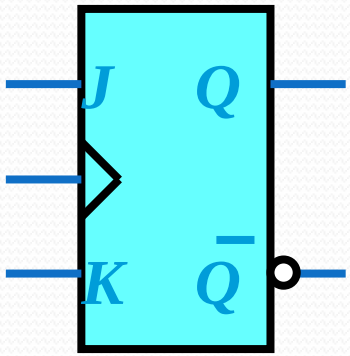
Clocked JK Flip-Flop

J	*K*	*Q*(*t*)	*Q*(*t*+1)	
0	0	0	0	Hold
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

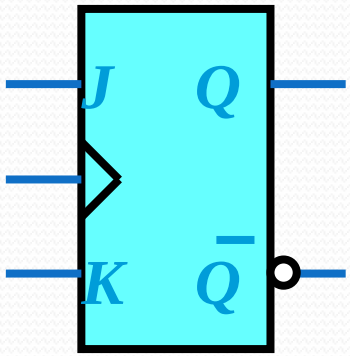
Characteristic Table



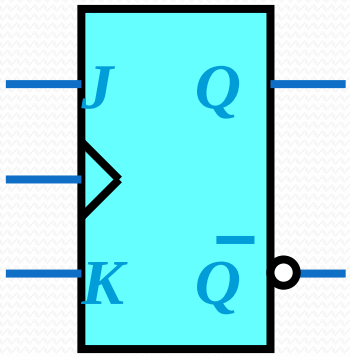
Characteristic Equation



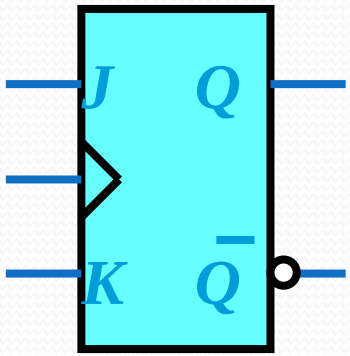
J	K	Q(t)	Q(t+1)	
0	0	0	0	No change
0	0	1	1	
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



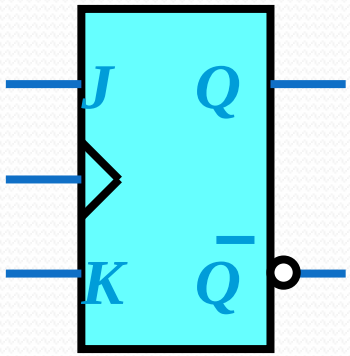
J	*K*	*Q*(*t*)	*Q*(*t*+1)	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0		
1	0	1		
1	1	0		
1	1	1		



J	*K*	*Q*(*t*)	*Q*(*t*+1)	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0		
1	1	1		



J	*K*	*Q*(*t*)	*Q*(*t*+1)	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	



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

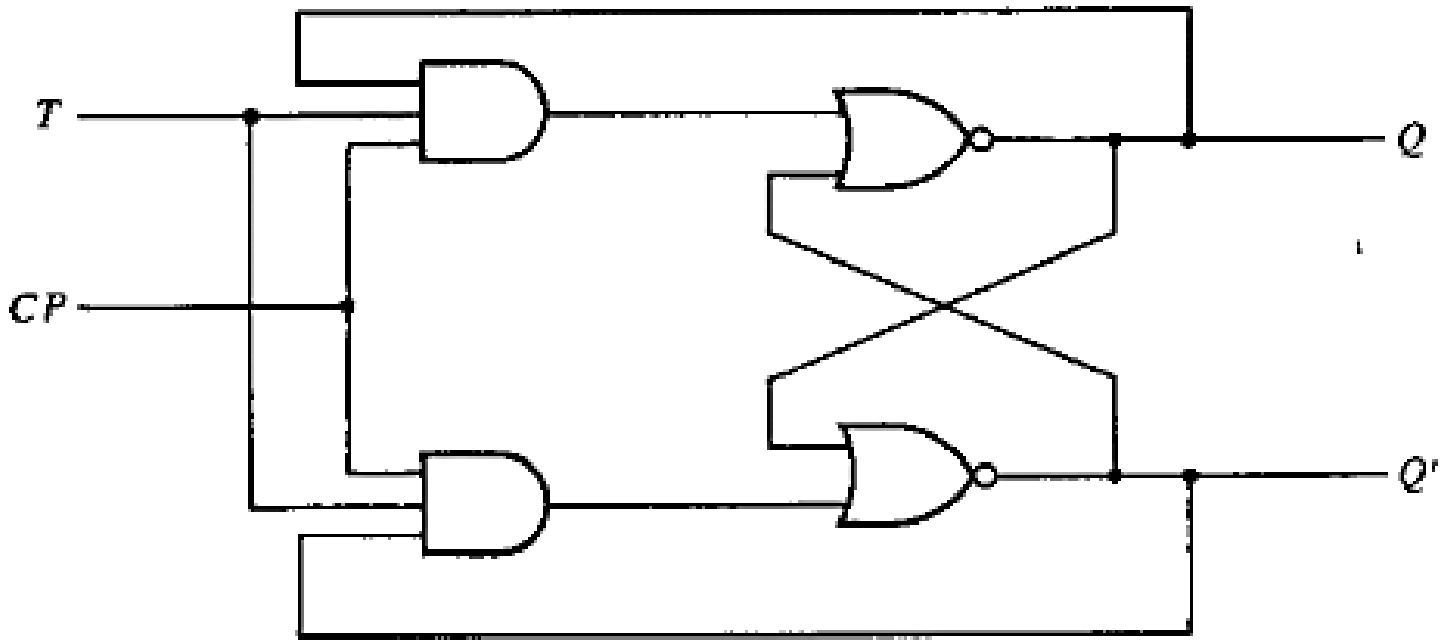
$$Q(t+1) = JQ' + K'Q$$

4. Clocked T Flip-Flop / T Flip-Flop

The *T* flip-flop is a single-input version of the *JK* flip-flop and is obtained from the *JK* flip-flop when both inputs are tied together. The designation *T* comes from the ability of the flip-flop to "toggle," or complement, its state. Regardless of the present state, the flip-flop complements its output when the clock pulse occurs while input *T* is 1. The characteristic table and characteristic equation show that:

- When $T = 0$, $Q(t + 1) = Q$, that is, the next state is the same as the present state and no change occurs.
- When $T = 1$, then $Q(t + 1) = Q'$, and the state of the flip-flop is complemented.

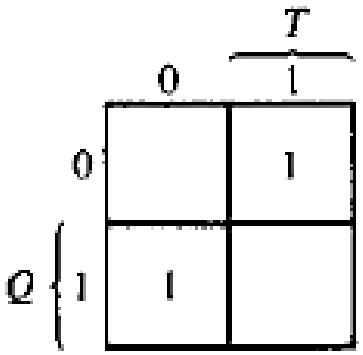
T	$Q(t)$	$Q(t+1)$	
0	0	0	Hold
0	1	1	
1	0	1	Toggle
1	1	0	



(a) Logic diagram

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

(b) Characteristic table

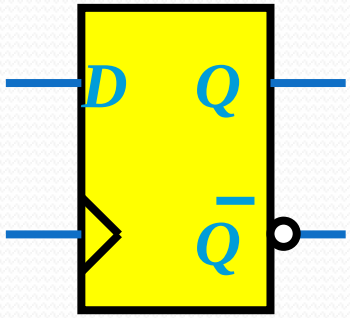


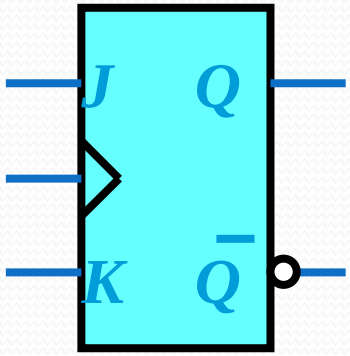
$Q(t+1) = TQ' + T'Q$

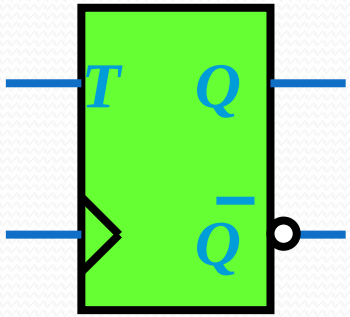
(c) Characteristic equation

Summary:

Graphical Symbols of Flip-Flop







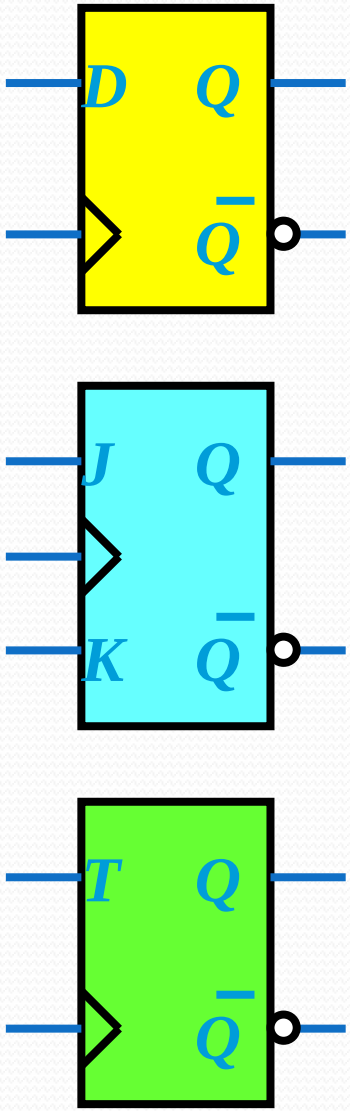
Characteristics Table

D	$Q(t+1)$	
0	0	Reset
1	1	Set

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Toggle

T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Toggle

Graphical Symbols of Flip-Flop



Characteristics Table

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

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$

Characteristics Equation

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

$$Q(t+1) = JQ' + K'Q$$

$$Q(t+1) = T \oplus Q$$

- **Triggering of Flip-Flops**

The state of a flip-flop is switched by a momentary change in the input signal. This momentary change is called a *trigger* and the transition it causes is said to trigger the flip-flop.

Clocked flip-flops are triggered by *pulses*. A pulse starts from an initial value of 0, goes momentarily to 1, and after a short time, returns to its initial 0 value.

A trigger is a control signal used to initiate an action.

In gated latches, the trigger is enable line.

There are two forms of trigger:

1. Level trigger (High or low)
2. Edge trigger (+ve or –ve going transition)

A clock pulse may be either positive or negative.

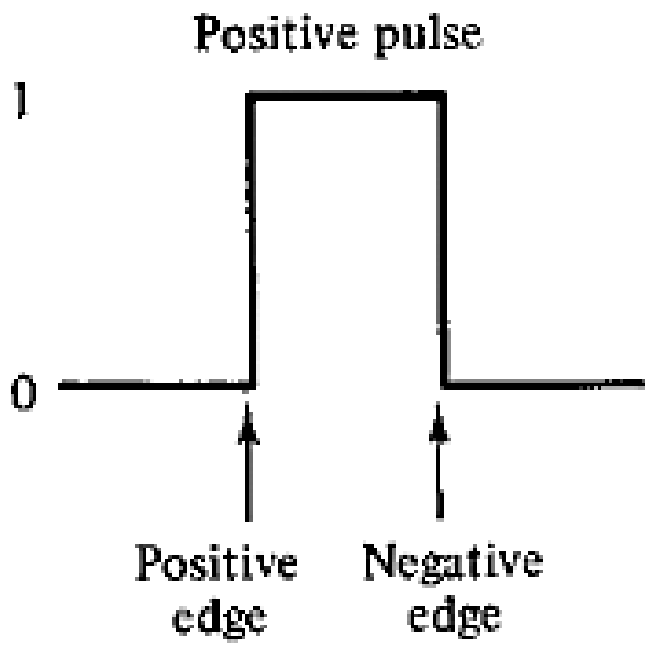
- A positive clock source remains at 0 during the interval between pulses and goes to 1 during the occurrence of a pulse. The pulse goes through two signal transitions: from 0 to 1 and the return from 1 to 0. As shown in Fig. below, the positive transition is defined as the *positive edge* and the negative transition as the *negative edge*.

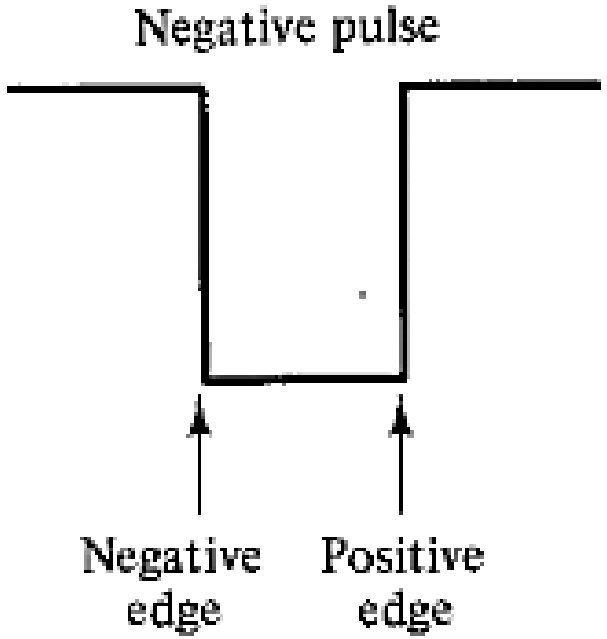
Positive edge-triggering

Inputs sampled on rising edge; outputs change after rising edge

Negative edge-triggering

Inputs sampled on falling edge; outputs change after falling edge





Positive Edge or Leading edge triggering

Negative Edge or Trailing edge triggering

- The output of one flip-flop cannot be applied to the inputs of another flip-flop when both are triggered by the same clock pulse. However, if we can make the flip-flop respond to the positive- (or negative-) edge transition only, instead of the entire pulse duration, then the multiple-transition problem can be eliminated.
- **Edge triggering** is achieved by using a master-slave or edge triggered flip-flop.
- **Reducing the gating time: EDGE TRIGGERED FLIP FLOPS**

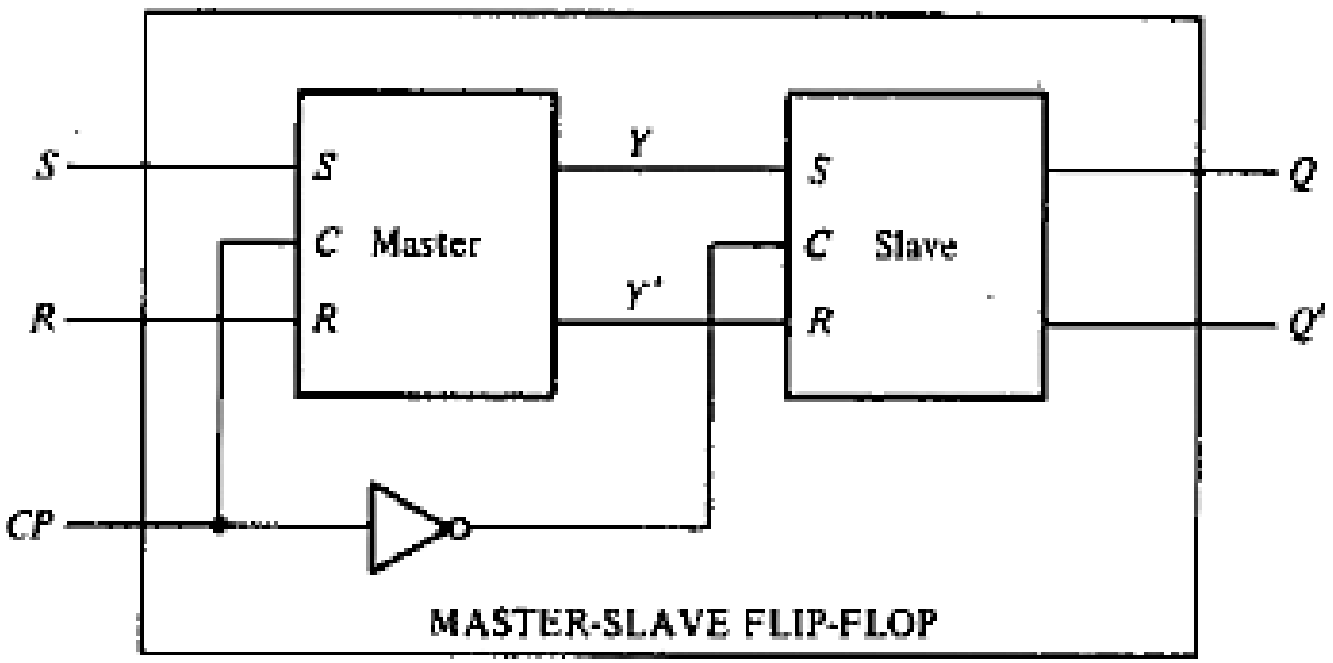
  To even further protect the flip flops from glitches, the gating time (the time during which the input signals affect the output signals) can be reduced by making the circuit sensitive only when the clock signal makes transitions from either high to low or vice versa. This is known as *edge triggering*.

- **Master-slave Flip-Flop**

A master-slave flip-flop is constructed from two separate flip-flops. One circuit serves as a **master** and the other as a **slave**, and the overall circuit is referred to as a *master slave flip-flop*.

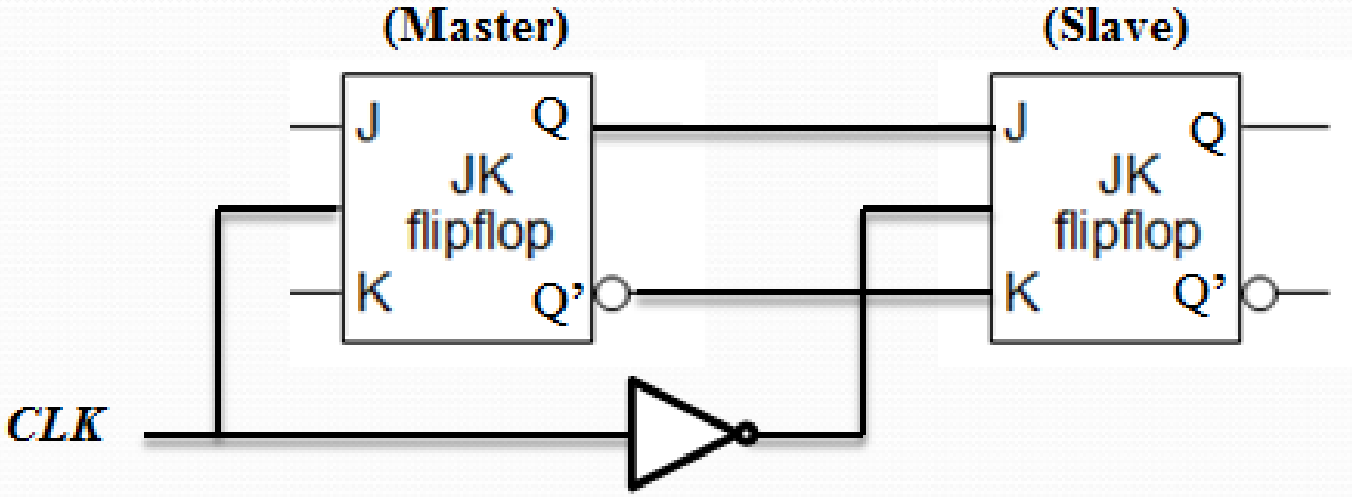
- RS Master-Slave Flip-Flop
- JK Master-Slave Flip-Flop
- D Master-Slave Flip-Flop

- **RS Master-Slave Flip-Flop**



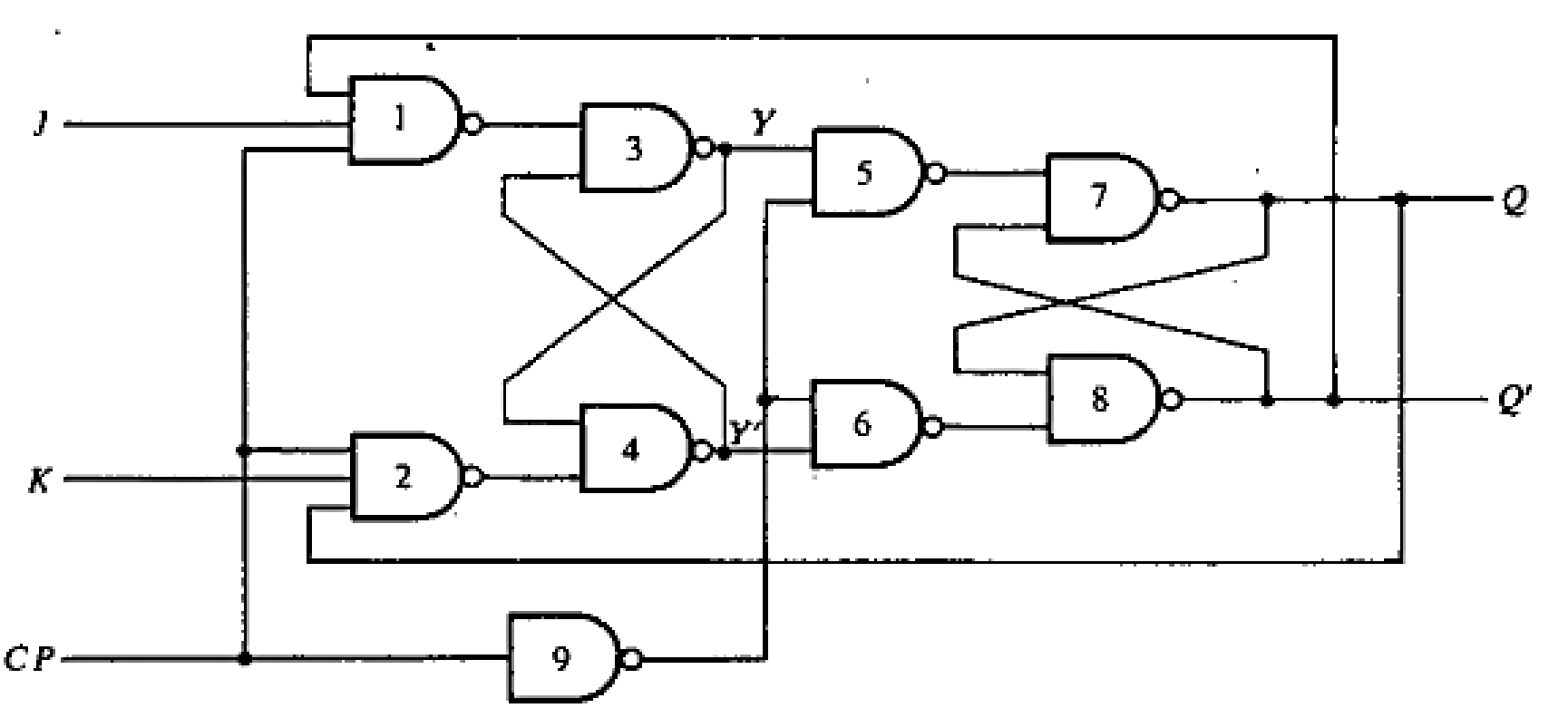
- The first flip-flop, called the master, is driven by the positive edge of the clock pulse and acts according to its RS inputs, but the slave does not respond.
- The second flip-flop called the slave, is driven by the negative edge of the clock pulse.

- **JK Master-Slave Flip-Flop**



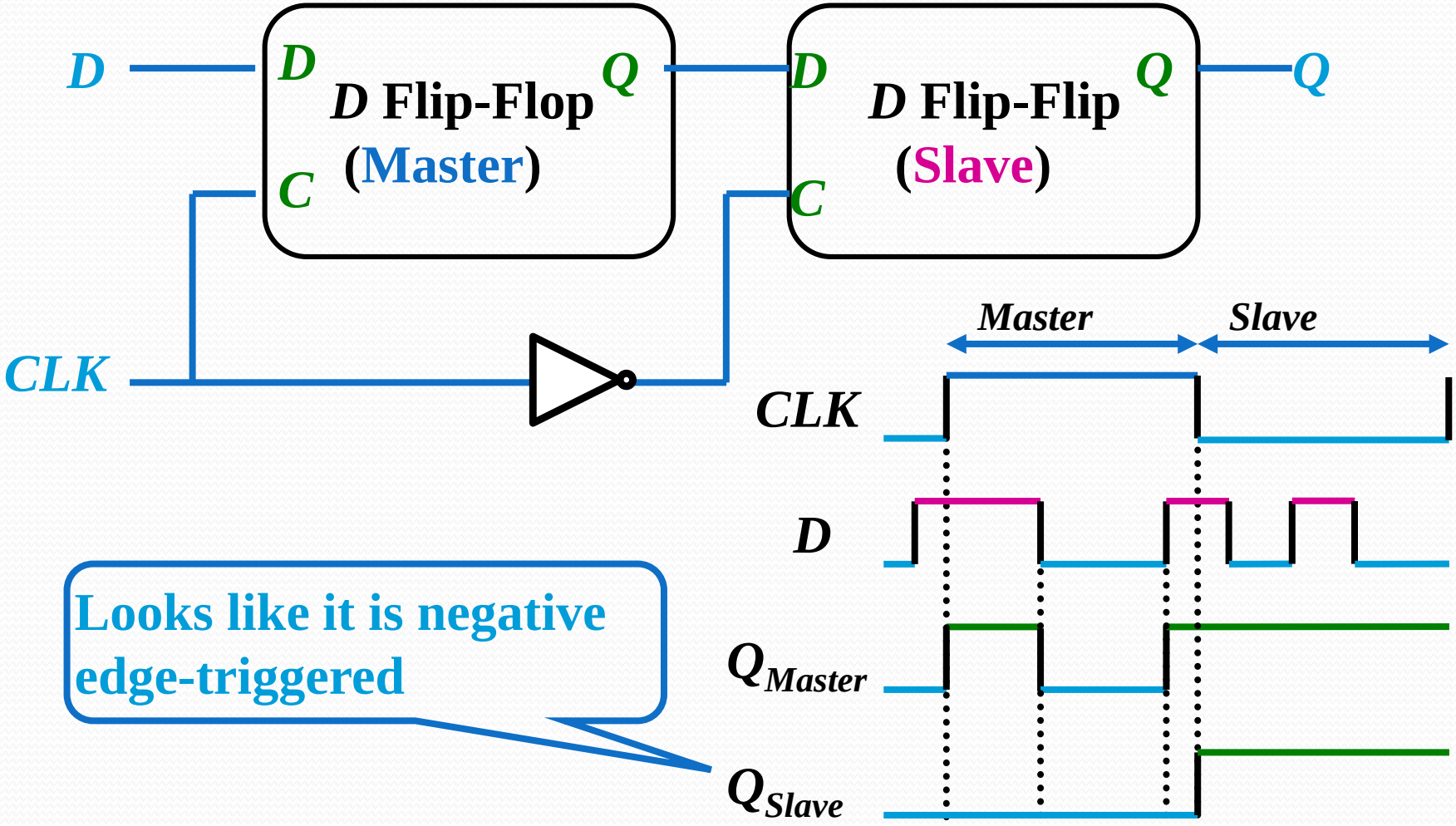
- The first flip-flop, called the master, is driven by the positive edge of the clock pulse and acts according to its J-K inputs, but the slave does not respond.
- The second flip-flop called the slave, is driven by the negative edge of the clock pulse.

- A master-slave J-K flip-flop constructed using NAND gates is shown in the figure

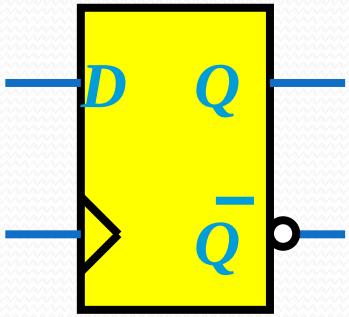


Note: The slave flip-flop is a clocked *RS* type

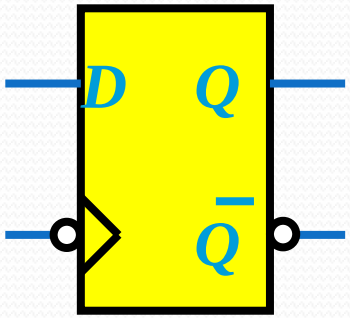
- **Master-Slave *D* Flip-Flop**



- **Edge-Triggered *D* Flip-Flop**



Positive Edge Triggered D Flip-Flop

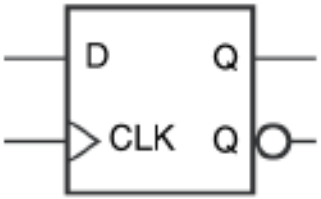


Negative Edge Triggered D Flip-Flop

Functional Table

D	CLK	Q	QN
0	↑	0	1
1	↑	1	0
x	0	last Q	last QN
x	1	last Q	last QN

Characteristics Table



$Q^+ = D$

Characteristics Equation

Truth Table

DQ	Q^+
0 0	0
0 1	0
1 0	1
1 1	1

D	Q^+
0	0
1	1

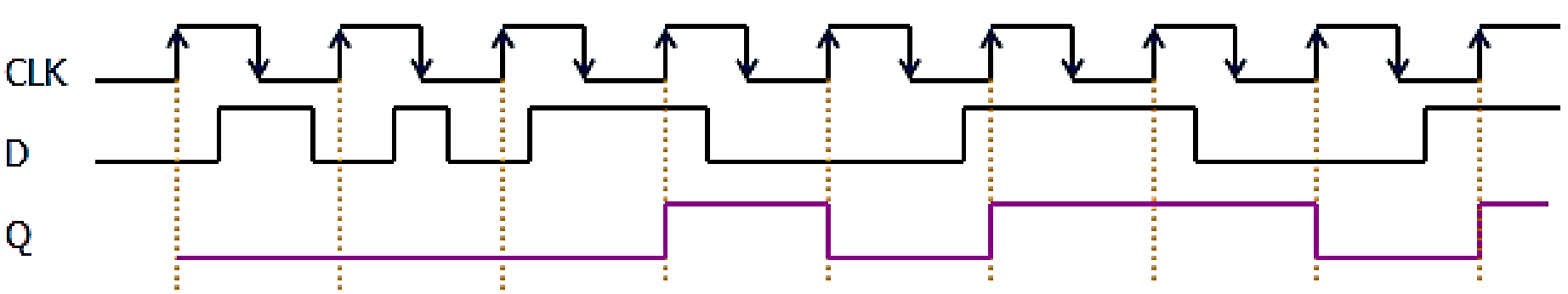
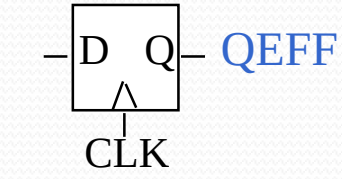
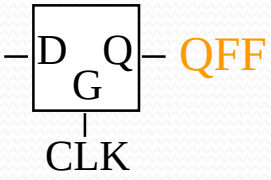


Fig: Timing Diagram

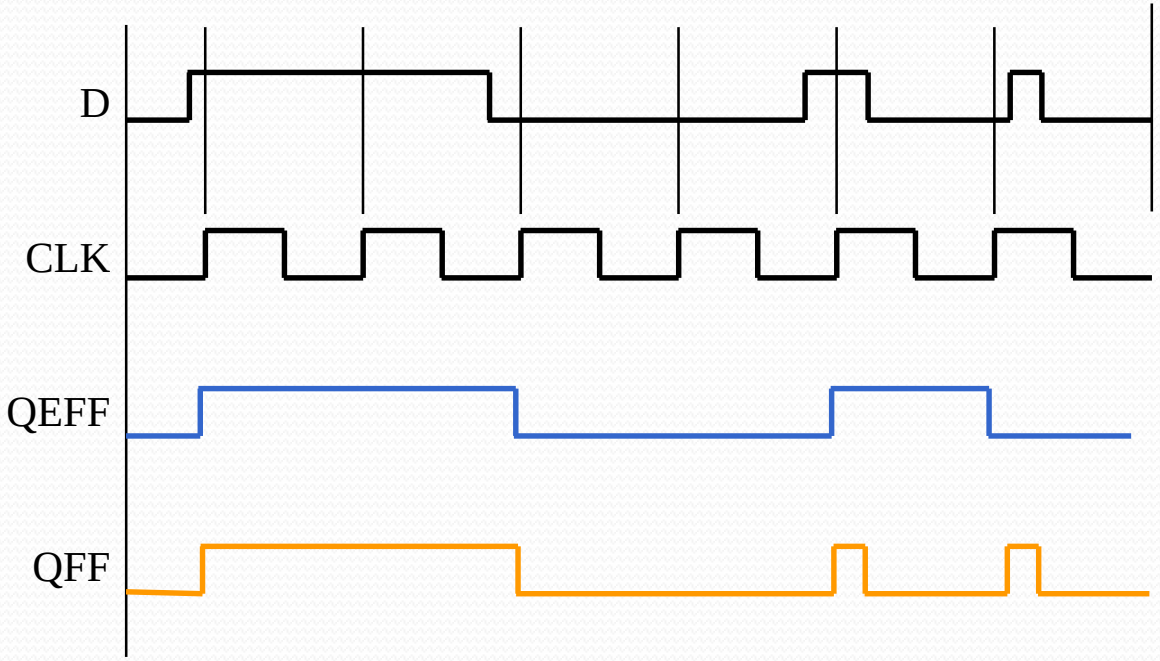
Comparison of Edge Triggered And Level Sensitive Flip-flop



Positive edge-triggered Flip-Flop

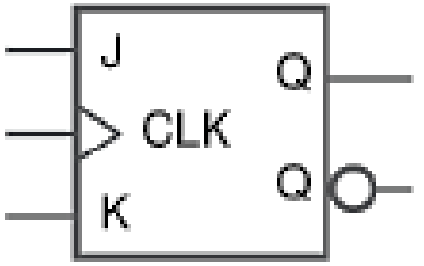


(level-sensitive) Flip-Flop



behavior is the same unless input changes while the clock is high

- **JK Flip Flop (rising edge triggered)**



J	K	CLK	Q	QN
x	x	0	last Q	last QN
x	x	1	last Q	last QN
0	0	↓	last Q	last QN
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	last QN	last Q

Functional Table

JKQ	Q^+
000	0
001	1
010	0
011	0
100	1
101	1
110	1
111	0

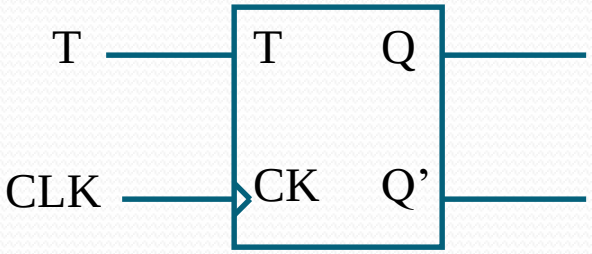
Truth Table

J K	Q^+
0 0	Q
0 1	0
1 0	1
1 1	Q’

More Compact Truth Table

Next state equation: $Q^+ = JQ' + K'Q$

- **Toggle Flip Flop (rising edge triggered)**

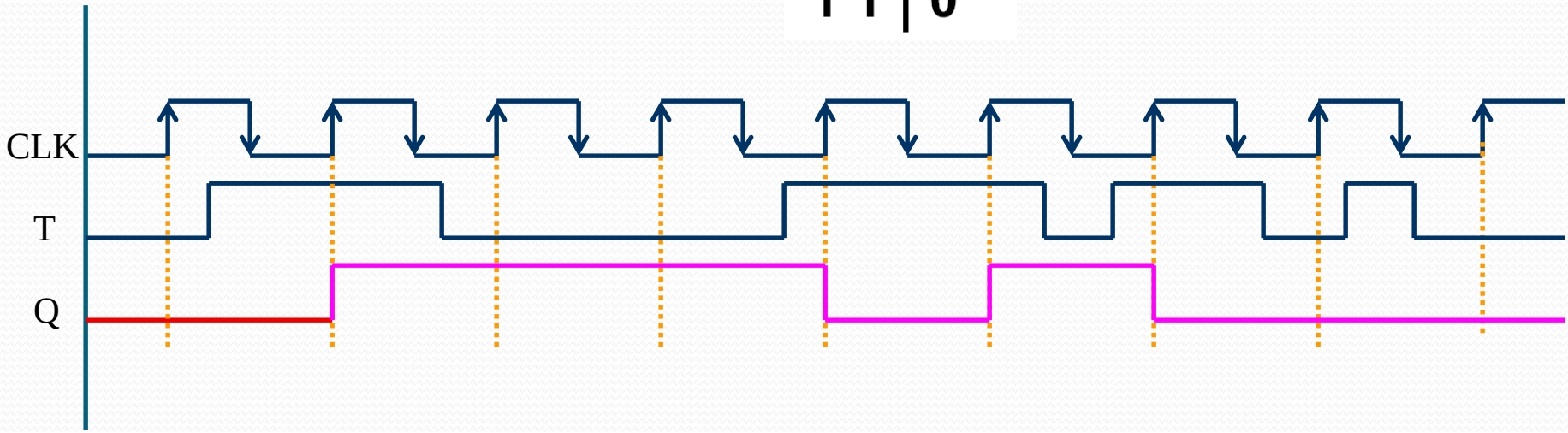


Truth Table

TQ	Q^+
0 0	0
0 1	1
1 0	1
1 1	0

More compact Truth Table

T	Q^+
0	Q
1	Q'

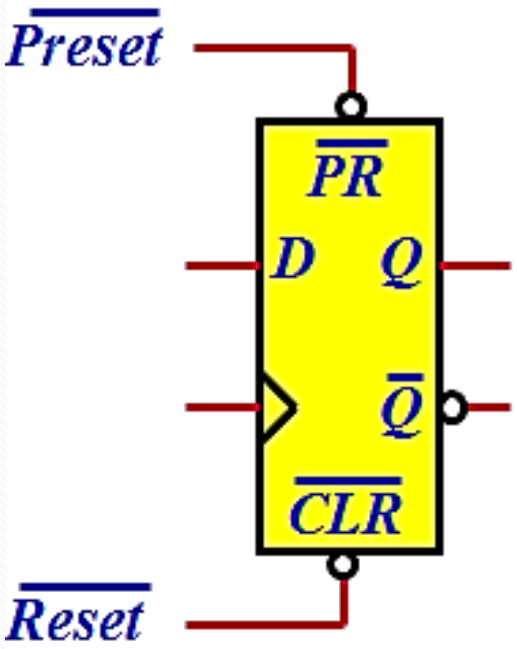


$$Q^+ = T'Q + TQ' = Q \oplus T$$

- **Direct Inputs**

Flip-flops available in IC packages sometimes provide special inputs for setting or clearing the flip-flop asynchronously. These inputs are usually called *direct preset* and *direct clear.* They affect the flip-flop on a positive (or negative) value of the input signal without the need for a clock pulse. These inputs are useful for bringing all flip-flops to an initial state prior to their clocked operation.

Example: After power is turned on in a digital system, the states of its flip-flops are indeterminate. A *clear(reset)* switch clears all the flip-flops to an initial cleared state and a *start(preset)* switch begins the system's clocked operation. The clear switch must clear all flip-flops asynchronously without the need for a pulse.



PR'	CLR'	D	CLK	Q(t+1)
1	0	x	x	0
0	1	x	x	1
1	1	0	↑	0
1	1	1	↑	1

Fig: Positive Edge Triggered D Flip-Flop with asynchronous Preset and Clear

Analysis of Clocked Sequential Circuits

- The behavior of a sequential circuit is determined from the inputs, the outputs, and the state of its flip-flops. The outputs and the next state are both a function of the inputs and the present state.
- The **analysis of a sequential circuit** consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states.
- It is also possible to write Boolean expressions that describe the behavior of the sequential circuit.

A logic diagram is recognized as a clocked sequential circuit if it includes flip-flops. The flip-flops may be of any type and the logic diagram may or may not include combinational circuit gates.

Example:

An example of a clocked sequential circuit is shown in Fig. below. The circuit consists of two *D* flip-flops *A* and *B*, an input *x*, and an output *y*.

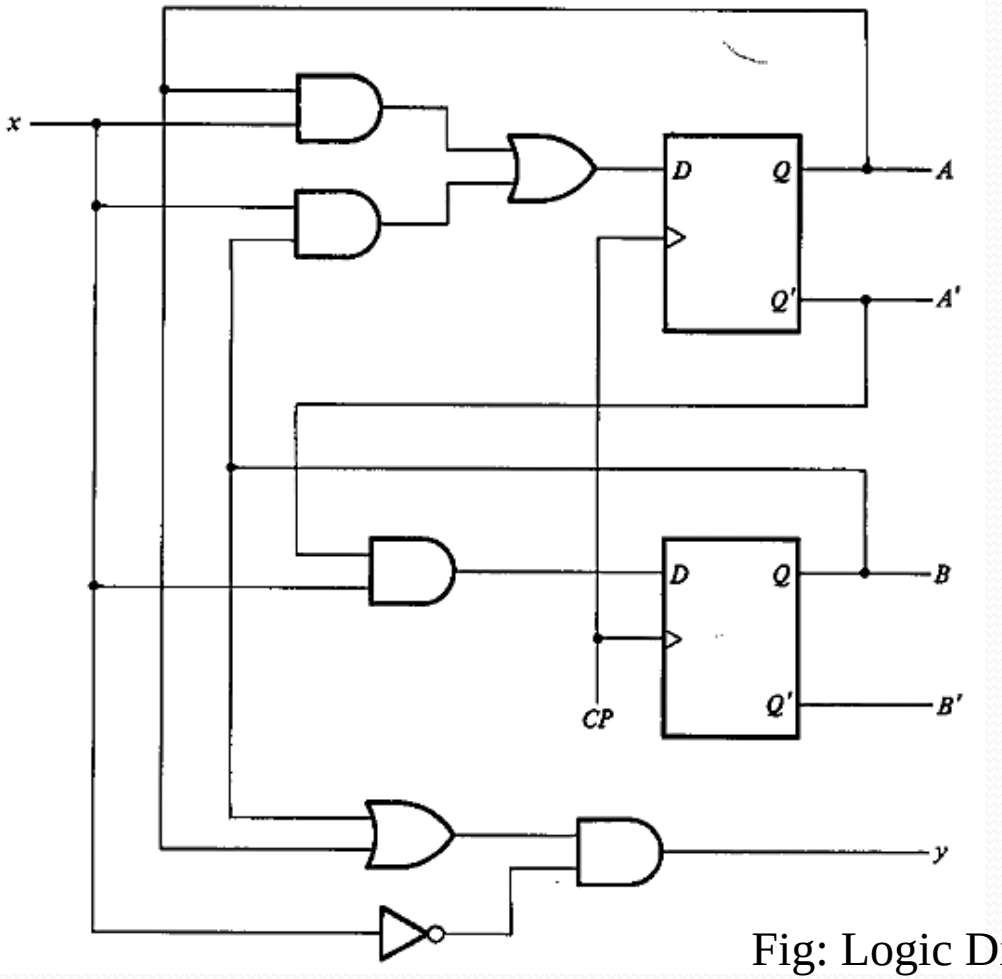


Fig: Logic Diagram

- **State Equations / State Transition Equations**

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation denotes the next state of the flip-flop and the right side of the equation is a Boolean expression that specifies the present state and input conditions that make the next state equal to 1.

In above example, *D* inputs determine the flip-flop's next state, so it is possible to write a set of next state equations for the circuit:

$$A(t+1) = A(t)x(t) + B(t)x(t)$$
$$B(t+1) = A'(t)x(t)$$

Can be written conveniently as:

$$A(t+1) = Ax + Bx$$
$$B(t+1) = A'x$$

Similarly, the present-state value of the output *y* can be expressed algebraically as follows:

$$y(t) = [A(t) + B(t)]x'(t)$$

Removing the symbol *(t)* for the present state, we obtain the output Boolean function:

$$y = (A + B)x'$$

- **State Table / State Transition Table**

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a *state table*.

The state table for the example circuit above is shown in the table below.

The table consists of four sections:

- *Present state:* shows the states of flip-flops *A* and *B* at any given time *t*
- *Input:* gives a value of *x* for each possible present state
- *Next state:* shows the states of the flip-flops one clock period later at time *t* + 1.
- *Output:* gives the value of *y* for each present state.

The derivation of a state table consists of first listing all possible binary combinations of present state and inputs.
Next state and output column is derived from the state equations.

Present State		Input	Next State		Output
A	B	x	A	B	y
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 Equations:

- $A(t+1) = (A+B)x$
- $B(t+1) = A'x$
- $y = (A+B)x'$

This table can alternatively be represented as:

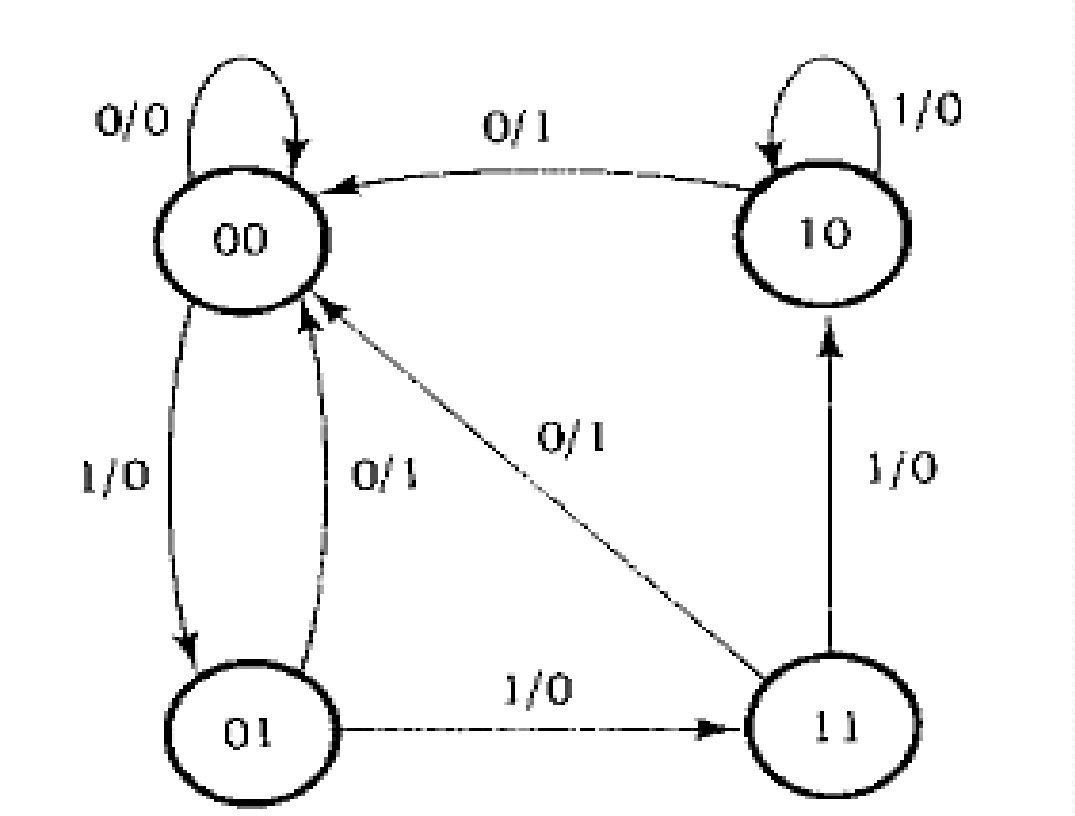
State Equations:

- $A(t+1) = (A+B)x$
- $B(t+1) = A'x$
- $y = (A+B)x'$

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

- **State Diagram / State Transition Diagram**

The information available in a state table can be represented graphically in a **state diagram**. In this type of diagram, a state is represented by a circle, and the transition between states is indicated by directed lines connecting the circles.

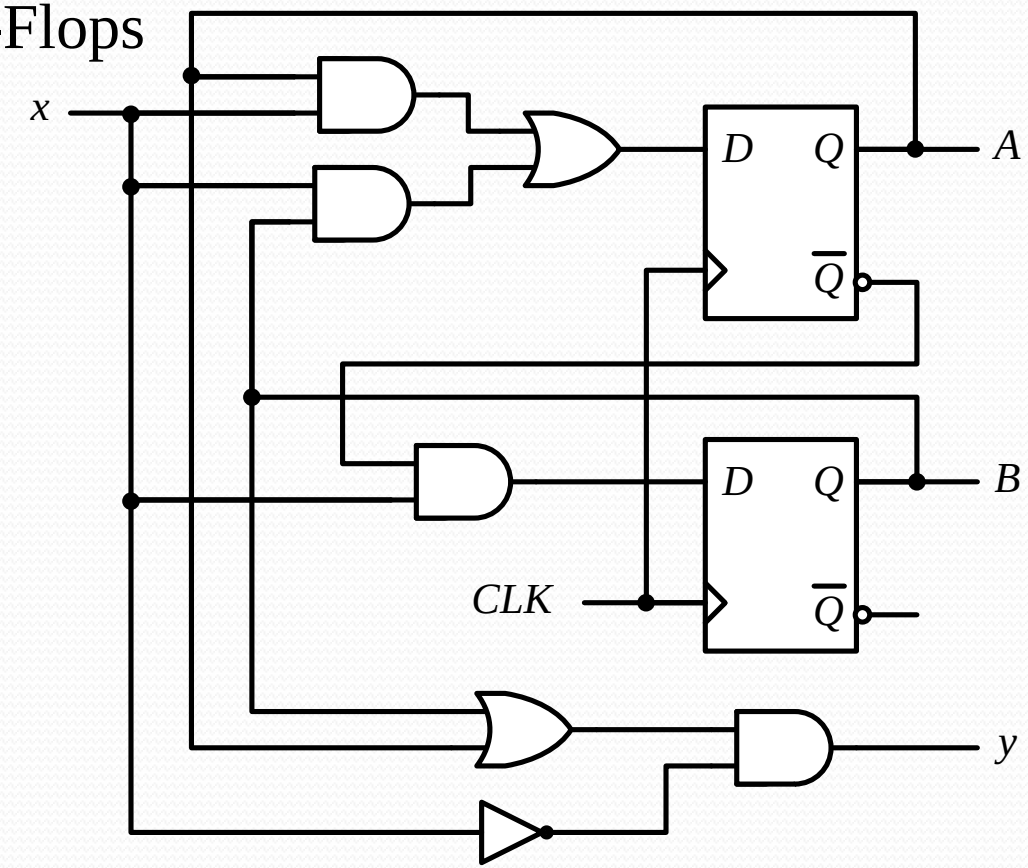


Analysis of Clocked Sequential Circuits

- The State
  - State = Values of all Flip-Flops

Example

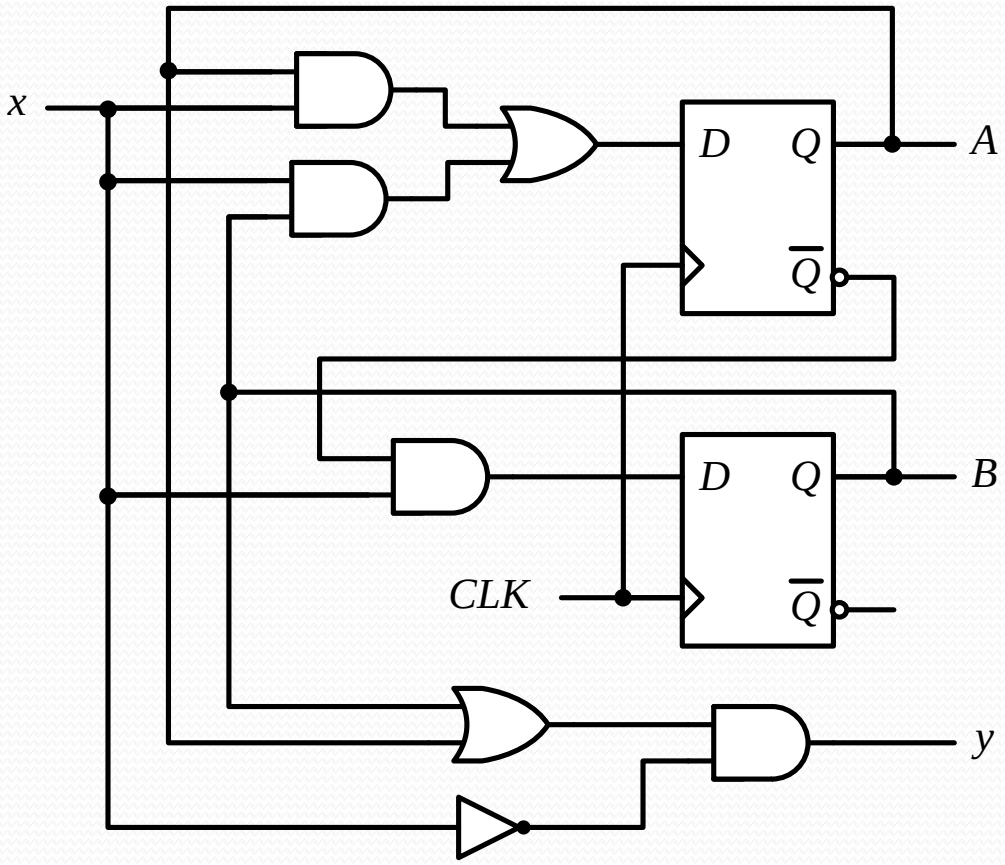
$A\ B = 0\ 0$



Analysis of Clocked Sequential Circuits

- State Equations

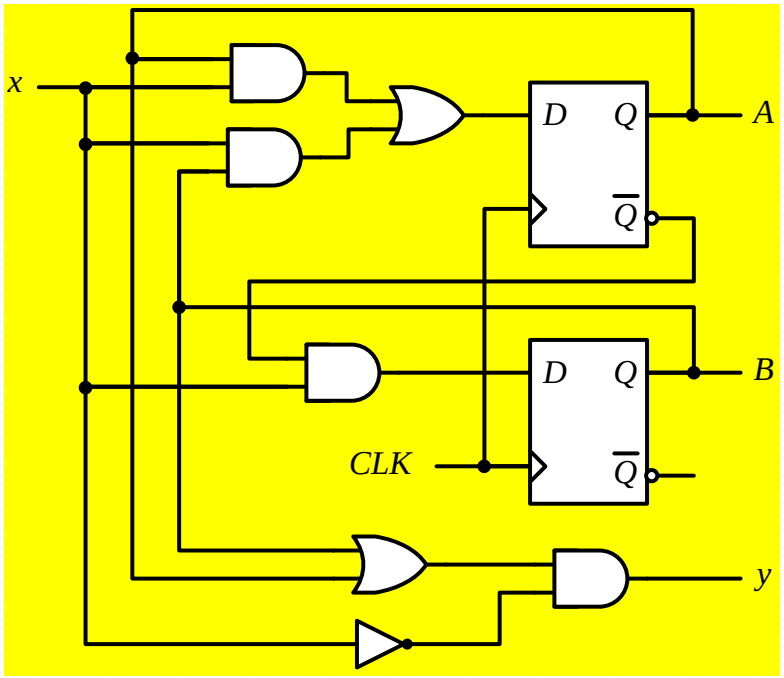
$$A(t+1) = D_A$$
$$= A(t)\,x(t) + B(t)\,x(t)$$
$$= A\,x + B\,x$$
$$B(t+1) = D_B$$
$$= A'(t)\,x(t)$$
$$= A'x$$
$$y(t) = [A(t) + B(t)]\,x'(t)$$
$$= (A + B)\,x'$$



Analysis of Clocked Sequential Circuits

- State Table (Transition Table)

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
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
			t+1		
			t		*t*



$$A(t+1) = A\,x + B\,x$$

$$B(t+1) = A'\,x$$

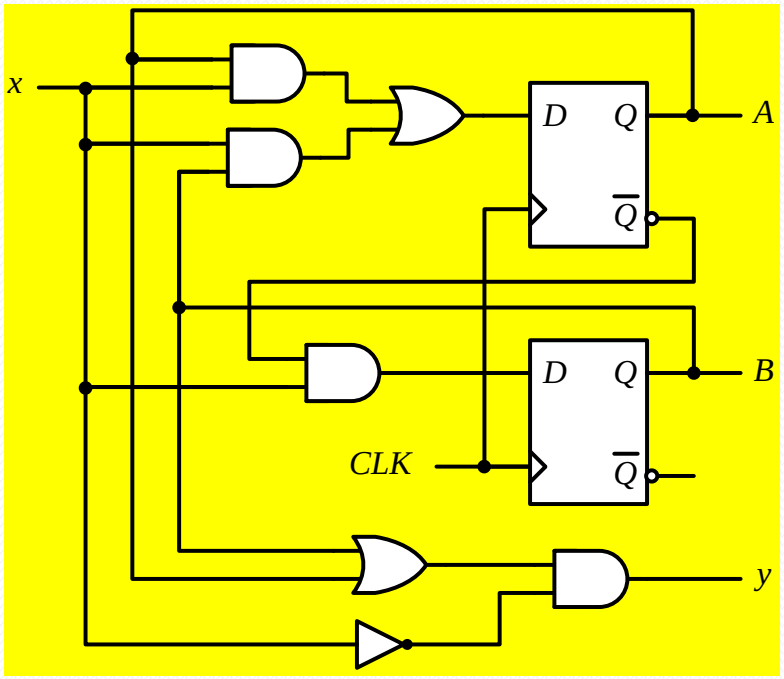
$$y(t) = (A + B)\,x'$$

Analysis of Clocked Sequential Circuits

- State Table (Transition Table)

Present State	Next State				Output	
	$x = 0$		$x = 1$		$x = 0$	$x = 1$
A B	A	B	A	B	y	y
0 0	0	0	0	1	0	0
0 1	0	0	1	1	1	0
1 0	0	0	1	0	1	0
1 1	0	0	1	0	1	0

t $t+1$ t



$$A(t+1) = A\,x + B\,x$$

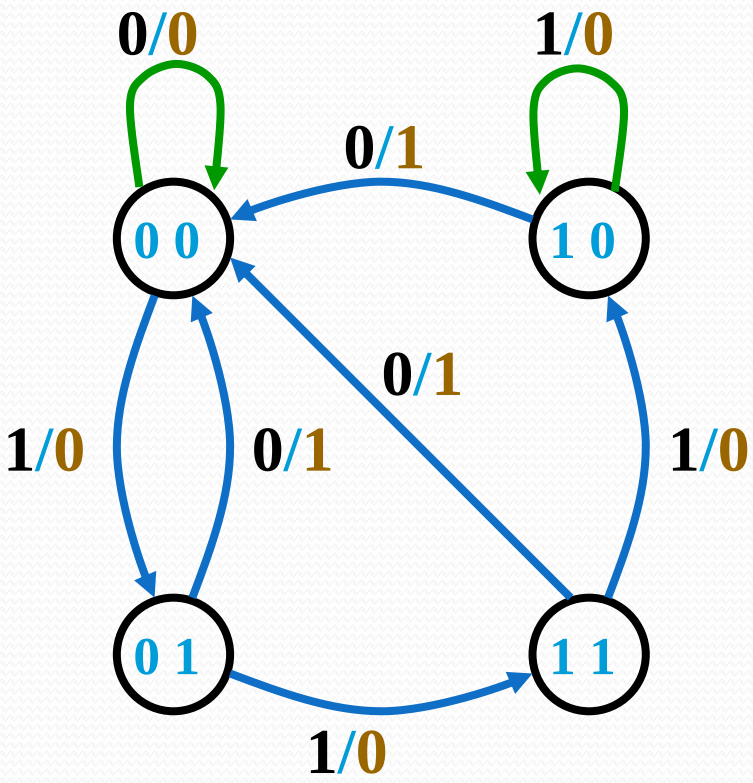
$$B(t+1) = A'\,x$$

$$y(t) = (A + B)\,x'$$

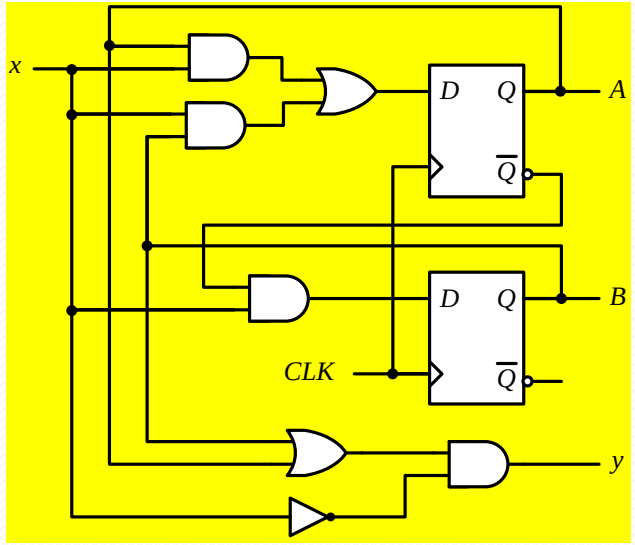
Analysis of Clocked Sequential Circuits

- State Diagram

Mealy State Diagram

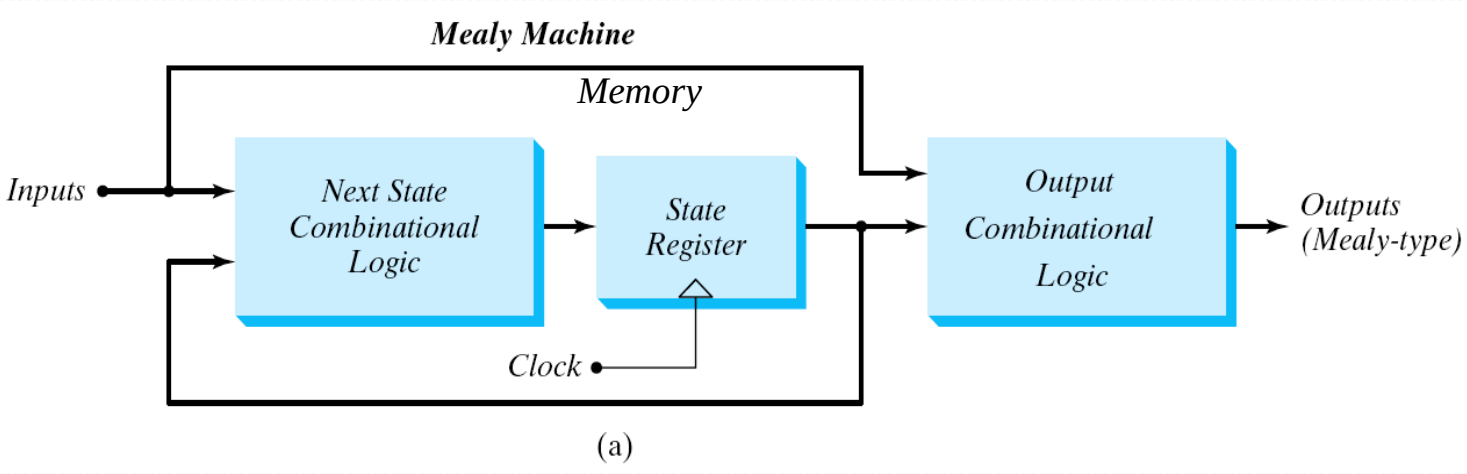


Present State	Next State				Output	
	$x = 0$		$x = 1$		$x = 0$	$x = 1$
A B	A	B	A	B	y	y
0 0	0	0	0	1	0	0
0 1	0	0	1	1	1	0
1 0	0	0	1	0	1	0
1 1	0	0	1	0	1	0

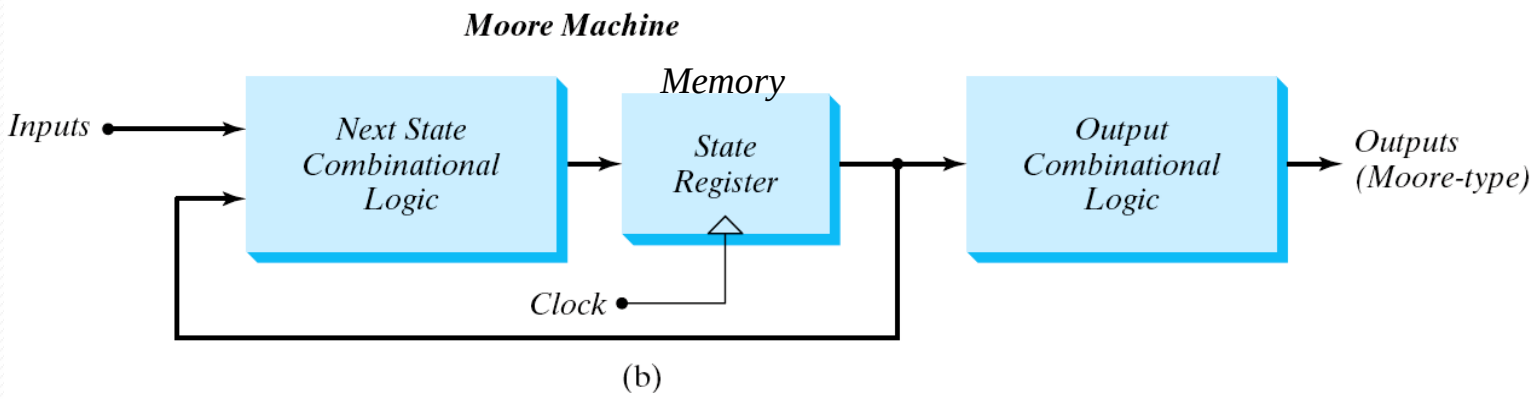


Mealy and Moore Models for state diagram:

- The Mealy model: the outputs are functions of both the present state and inputs.



- The Moore model: the outputs are functions of the present state only.



Mealy

Present State	I/P	Next State	O/P
A B	*x*	*A B*	*y*
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

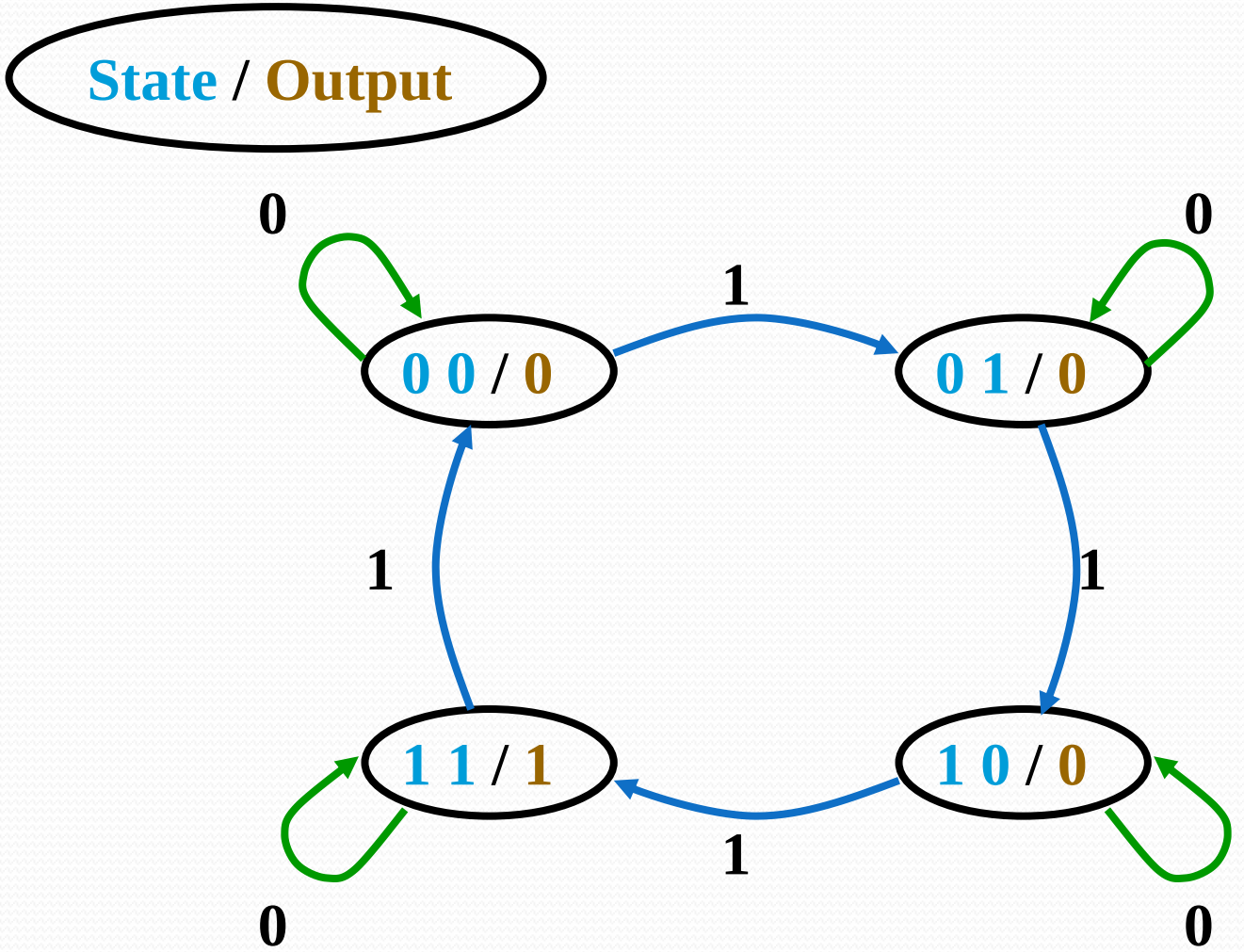
For the same state,
the output changes with the input

Moore

Present State	I/P	Next State	O/P
A B	*x*	*A B*	*y*
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	0 1	0
0 1	1	1 0	0
1 0	0	1 0	0
1 0	1	1 1	0
1 1	0	1 1	1
1 1	1	0 0	1

For the same state,
the output does not change with the input

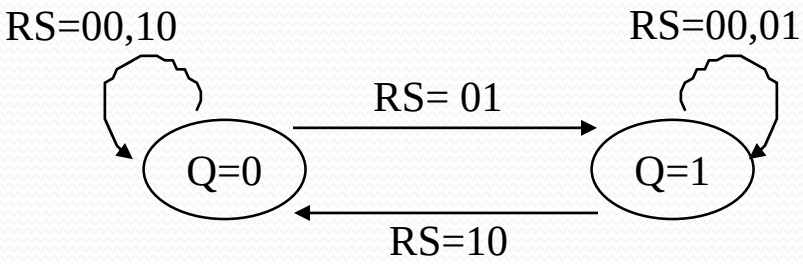
Moore State Diagram



RS Flip-Flop

PS	NS (Q^+)			
Q	SR 00	SR 01	SR 10	SR 11
0	0	0	1	x
1	1	0	1	x
Q	Q	0	1	x

State Table

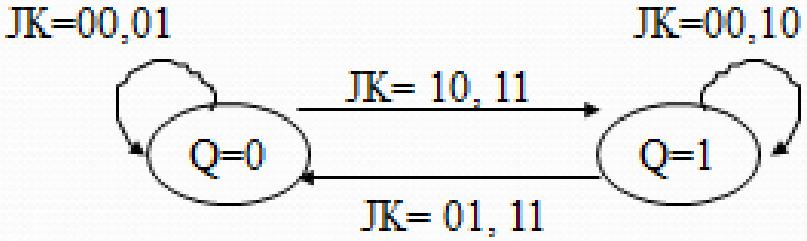


State Diagram

JK Flip-Flop

PS	NS (Q^+)			
Q	JK 00	JK 01	JK 10	JK 11
0	0	0	1	1
1	1	0	1	0
Q	Q	0	1	Q

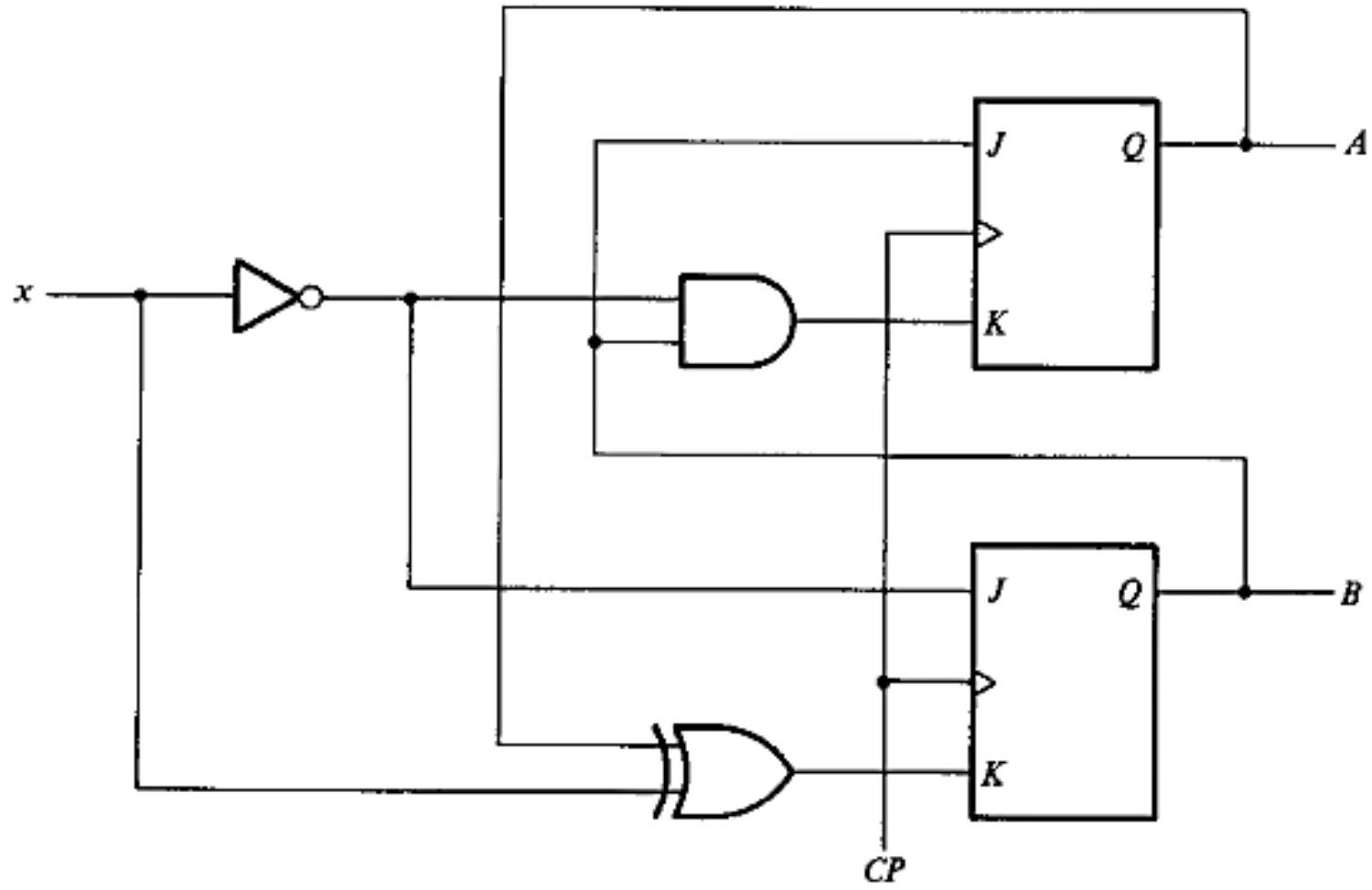
State Table



State Diagram

Assignment: Analyze the following sequential circuit

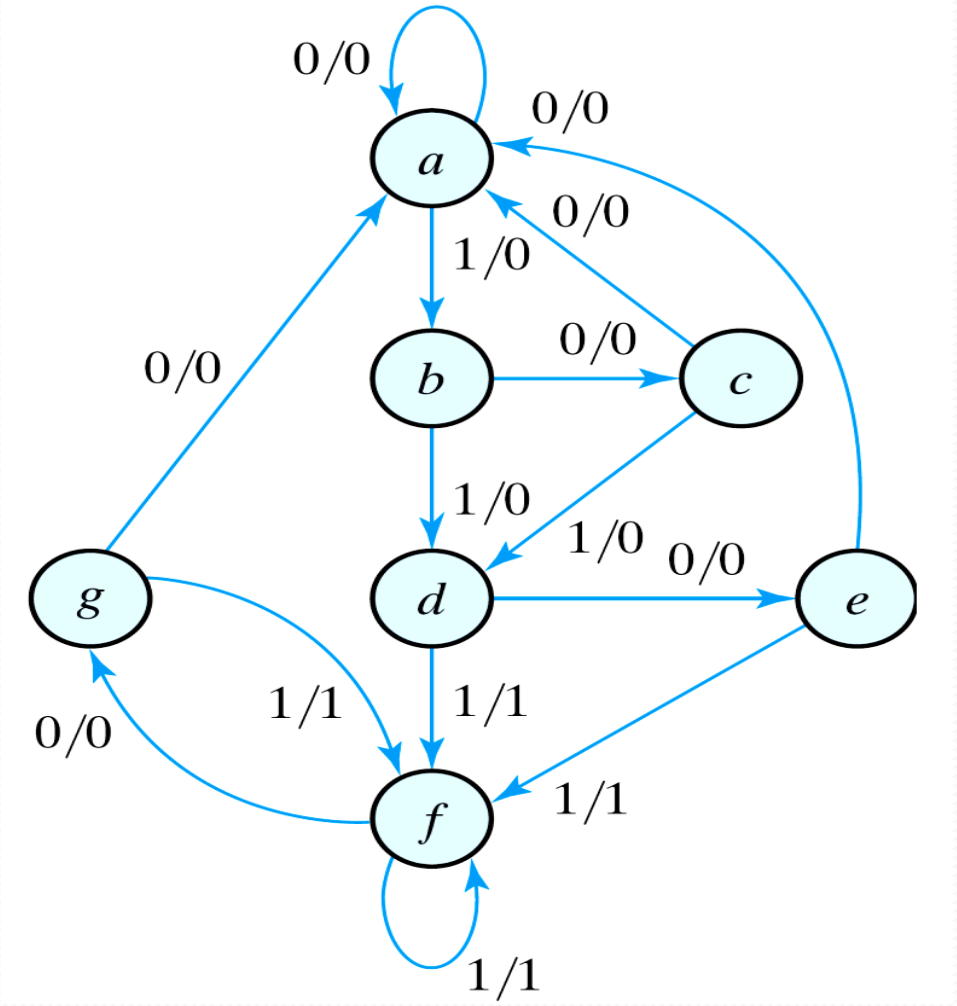
(State Equation, State Table and State Diagram)



State Reduction

- The analysis of sequential circuits starts from a circuit diagram and culminates in a state table or diagram.
- The design of a sequential circuit starts from a set of specifications and culminates in a logic diagram.
- Any design process must consider the problem of minimizing the cost of the final circuit (reduce the number of gates and flip-flops during the design).
- The reduction of the number of flip-flops in a sequential circuit is referred to as the *state-reduction* problem.
- State-reduction algorithms are concerned with procedures for reducing the number of states in a state-table while keeping the external input-output requirements unchanged.

Example: Consider a sequential circuit with following specification. States marked inside the circles are denoted by letter symbols instead of by their binary values.



Solution:

Consider the input sequence 01010110100 starting from the initial state *a*. Each input of 0 or 1 produces an output of 0 or 1 and causes the circuit to go to the next state. From the state diagram, we obtain the output and state sequence for the given input sequence as follows:

Present State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	
Next State	a	b	c	d	e	f	f	g	f	g	a	

Algorithm:

"Two states are said to be equivalent if,

- for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state.
- When two states are equivalent, one of them can be removed without altering the input output relationships."

Steps:

1. First, we need the state table (from state diagram above)

Table 5.6
State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

2. Reducing the state table

Equivalent State: $e = g$ (remove g);

Similarly then $d = f$ (remove f);

Table 5.7

Reducing the State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

3. Finally, reduced form is obtained.

Table 5.8
Reduced State Table

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Fig. Reduced State Table

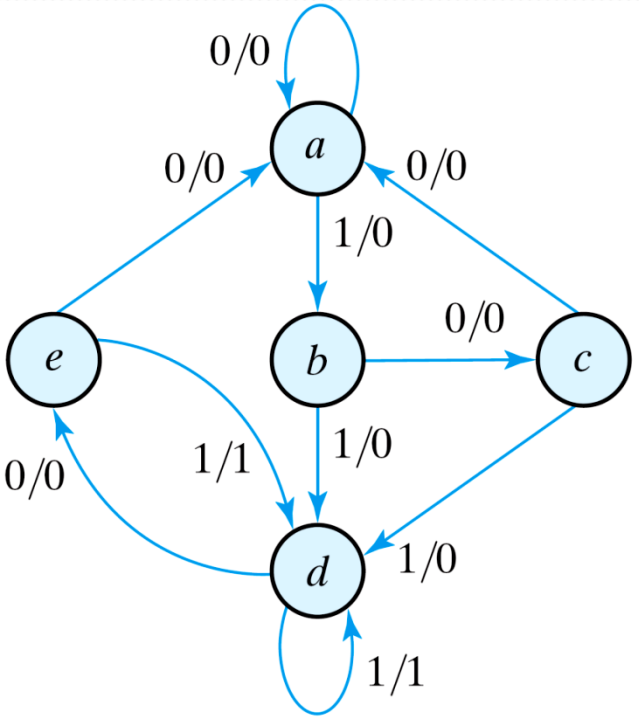


Fig. Reduced State diagram

- **State Assignment**
- The cost of the combinational-circuit part of a sequential circuit can be reduced by using the known simplification methods for combinational circuits.
- However, there is another factor, known as the *state assignment* problem that comes into play in minimizing the combinational gates.
- State-assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of the combinational circuit that drives the flip-flops.

Note: Any binary number assignment is satisfactory as long as each state is assigned a unique number.

Example: Consider a example shown in state reduction, 3 examples of possible binary assignments are shown in Table below for the five states of the reduced table.

Table 5.9
Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Excitation Tables:

A table that lists required inputs for a given change of state (Present to next-state) is called an *excitation table*.

The required input conditions for each transitions are derived from the information available in the characteristic table. The symbol X in the tables represents a don't-care condition, i.e., it does not matter whether the input is 1 or 0.

Flip-Flop Excitation Tables

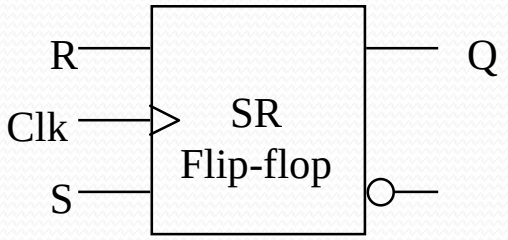
SR Flip-Flop

R	S	Q^+
0	0	Q
0	1	1
1	0	0
1	1	forbid

Transition Table

Q	Q^+	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

Excitation Table



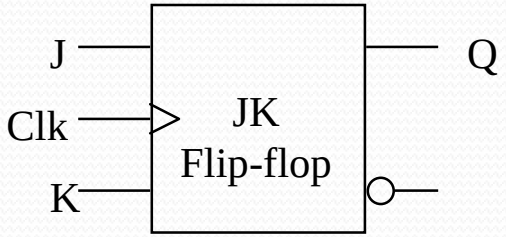
JK Flip-Flop

K	J	Q^+
0	0	Q
0	1	1
1	0	0
1	1	Q'

Transition Table

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table



Flip-Flop Excitation Tables

Useful Design Tool:

For each state-transition, the excitation table lists the required input combination(s)

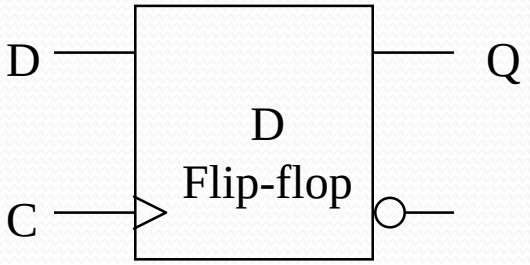
1. D FlipFlop

D	Q^+
0	0
1	1

Transition Table

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table



$Q^+ = D$

---

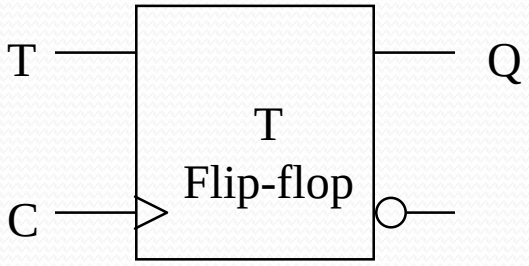
2. T FlipFlop

T	Q^+
0	Q
1	Q'

Transition Table

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table



$Q^+ = TQ'+T'Q$

Flip-Flop Excitation Tables

Present State	Next State	F.F. Input
$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

Present State	Next State	F.F. Input	
$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

0 0 (No change)
0 1 (Reset)

1 0 (Set)
1 1 (Toggle)

0 1 (Reset)
1 1 (Toggle)

0 0 (No change)
1 0 (Set)

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

Design Procedure

The design of a clocked sequential circuit starts from a set of specifications (state table) and ends in a logic diagram or a list of Boolean functions from which the logic diagram can be obtained.

Procedure:

- The **word description of the circuit behavior** to get a **state diagram**
- Obtain the **state table** and perform **State reduction** if necessary
- Assign binary values to the states
- Obtain the binary-coded **state table.**
- Determine the **number of flip-flops** needed and assign a letter symbol to each.
- Choose the type of flip-flops.
- Derive the simplified flip-flop input equations and output equations.
- Draw the **logic diagram**

Example: Design Procedure:

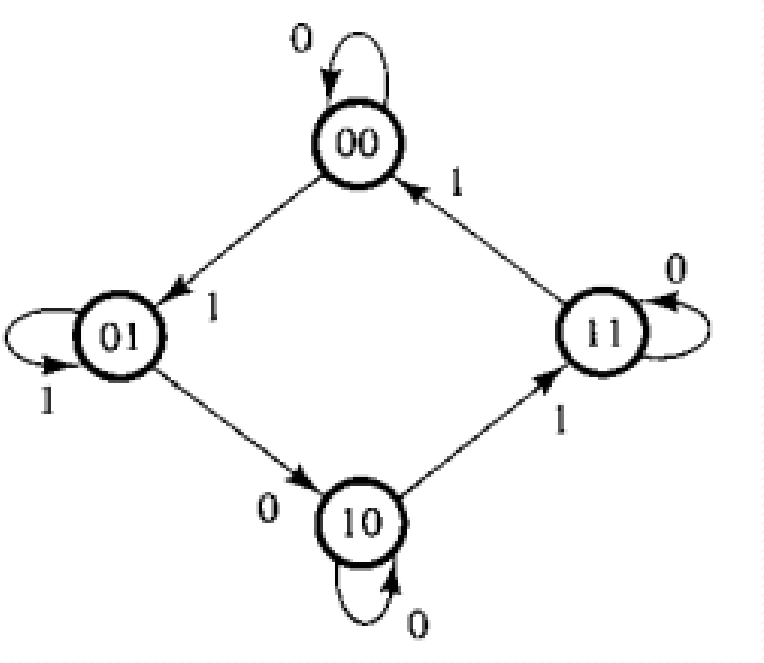


Fig: State-diagram for design example

Procedure step: (1), (2) and (3)

- The state diagram consists of four states with binary values already assigned.
- Directed lines contain single binary digit without a slash, we conclude that there is one input variable and no output variables. (The state of the flip-flops may be considered the outputs of the circuit).
- The two flip-flops needed to represent the four states are designated *A* and *B*.
- Let the input variable is designated *x*.

Procedure step: (4)

The state table for this circuit, derived from the state diagram. Note that there is no output section for this circuit.

State Table

Present State		Next State			
		$x = 0$		$x = 1$	
A	B	A	B	A	B
0	0	0	0	0	1
0	1	1	0	0	1
1	0	1	0	1	1
1	1	1	1	0	0

In the derivation of the excitation table, present state and input variables are arranged in the form of a truth table.

- Let JK type is used here. (Procedure Step: (5))

Since *JK* flip-flops are used we need columns for the J and *K* inputs of flip-flop *A* (denoted by J*A* and *KA)* and *B* (denoted by *JB* and *KB).*

Inputs of Combinational Circuit			Next State		Outputs of Combinational Circuit			
Present State		Input			Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	*JA*	*KA*	*JB*	*KB*
0	0	0	0	0	0	*X*	0	*X*
0	0	1	0	1	0	*X*	1	*X*
0	1	0	1	0	1	*X*	*X*	1
0	1	1	0	1	0	*X*	*X*	0
1	0	0	1	0	*X*	0	0	*X*
1	0	1	1	1	*X*	0	1	*X*
1	1	0	1	1	*X*	0	*X*	0
1	1	1	0	0	*X*	1	*X*	1

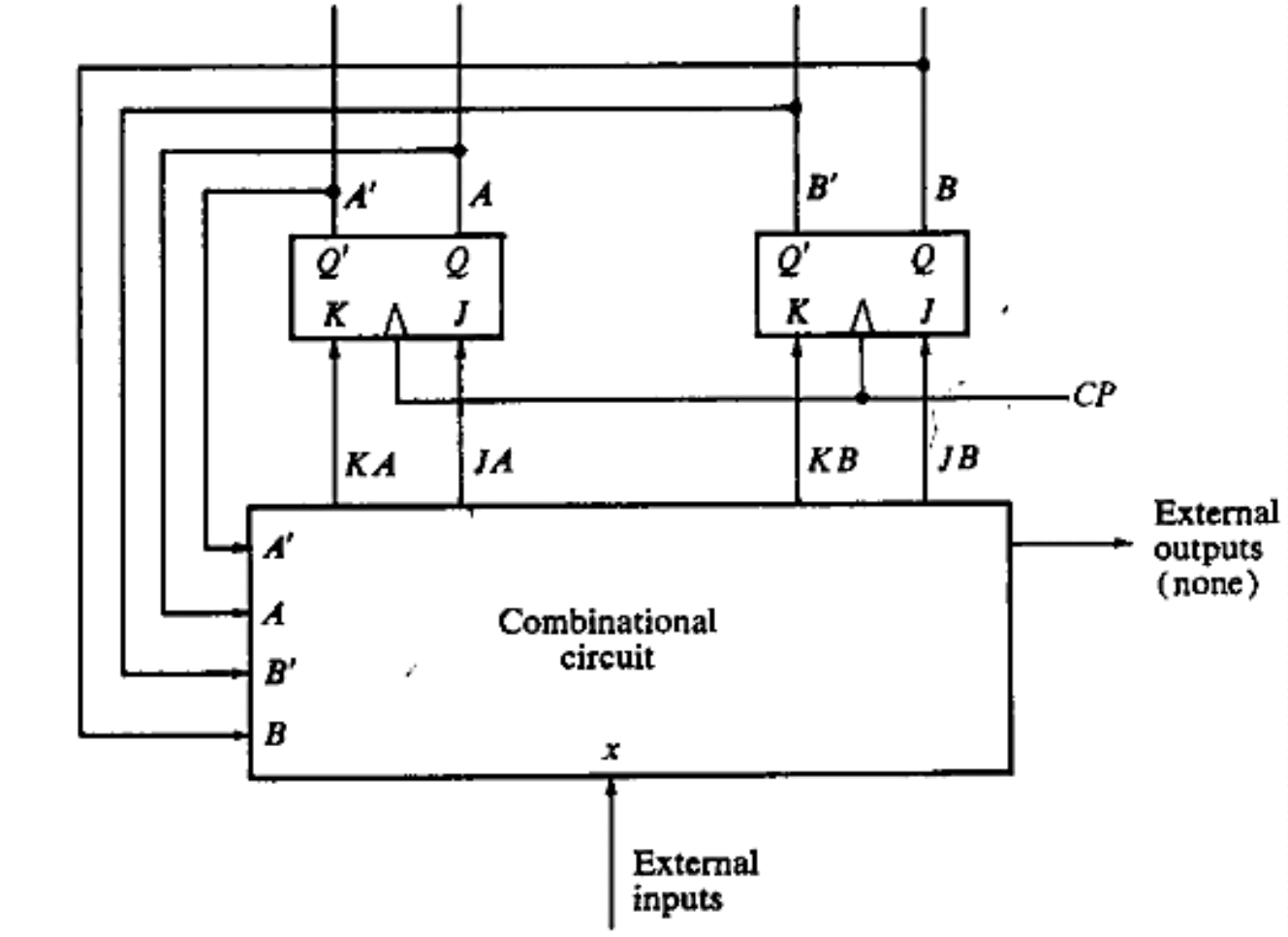


Fig: Block diagram of sequential circuit

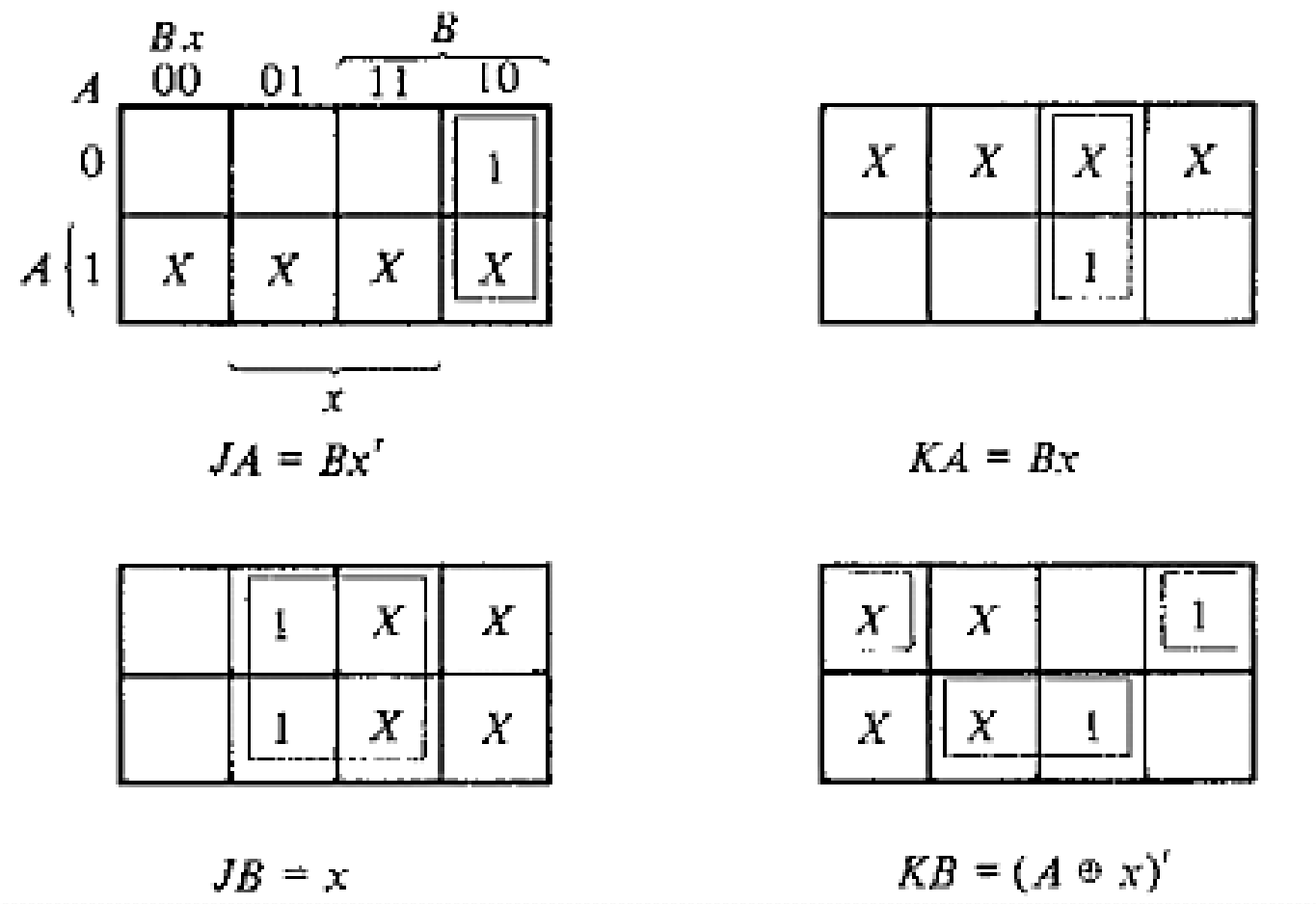


Fig: K- Maps for combinational circuit

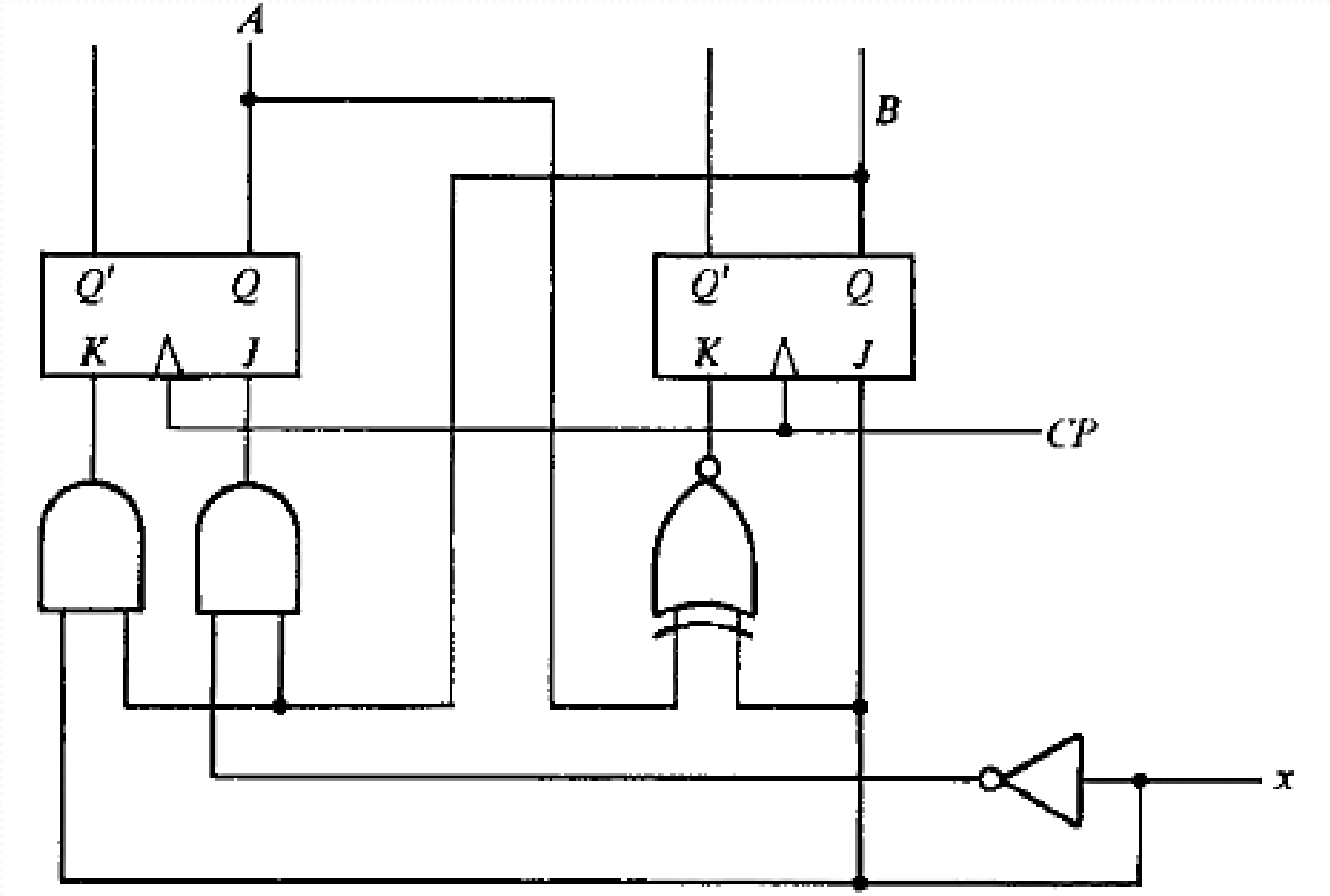
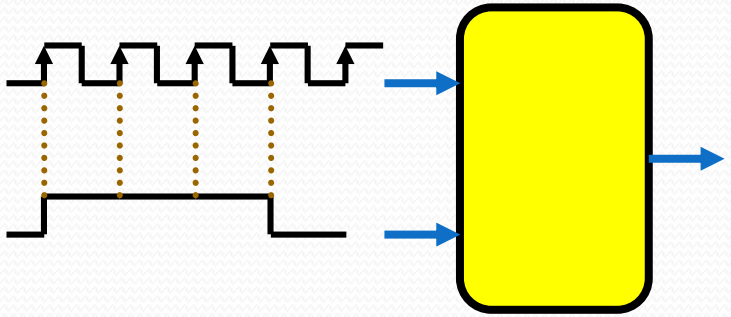


Fig: Logic diagram of sequential circuit

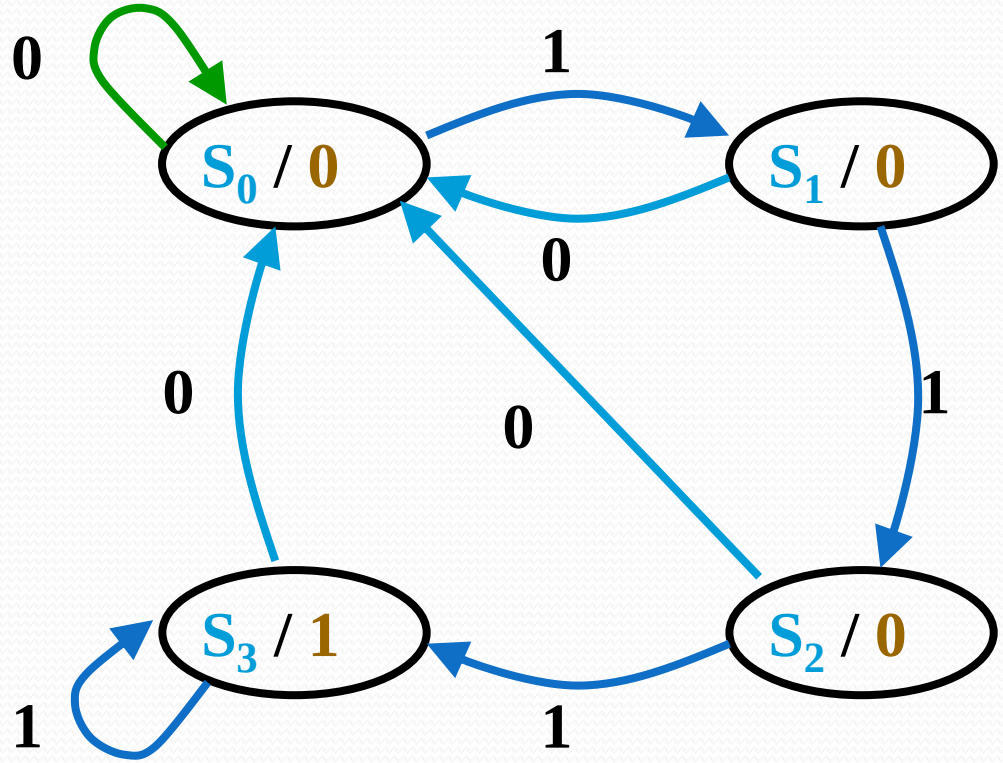
Design of Clocked Sequential Circuits

- *Example*:

Detect 3 or more consecutive 1's



Moore Method:



State	*A* *B*
S_0	0 0
S_1	0 1
S_2	1 0
S_3	1 1

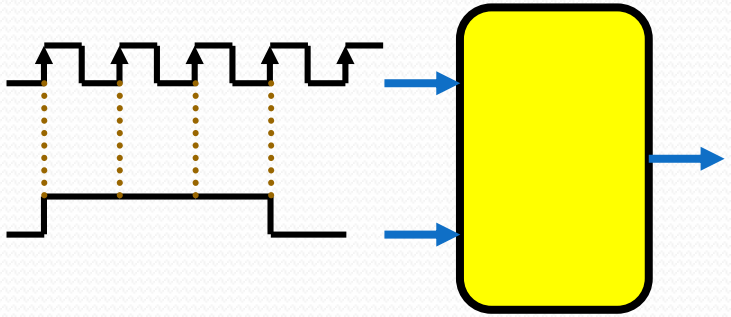
Design of Clocked Sequential Circuits

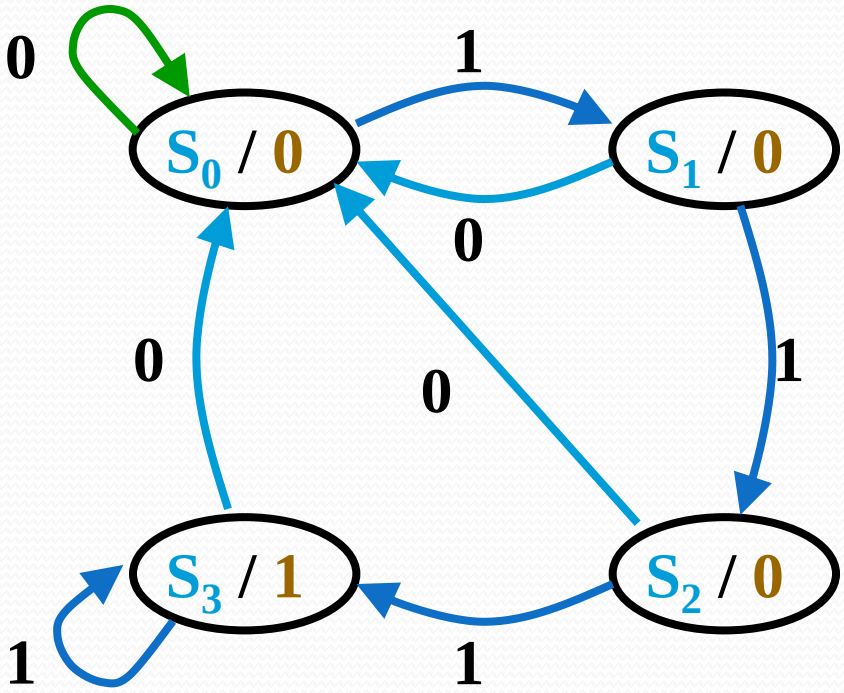
- *Example*:

Detect 3 or more consecutive 1's

Moore Method:

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





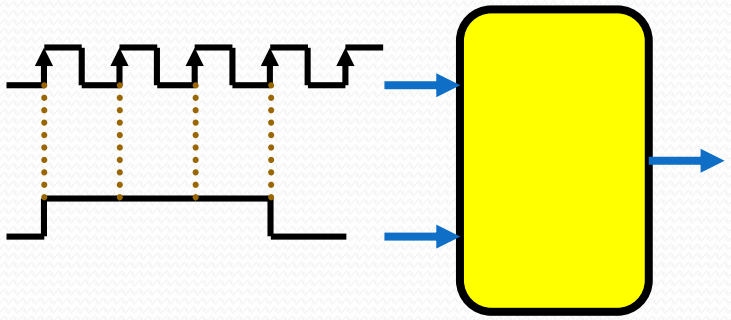
Design of Clocked Sequential Circuits

- *Example*:

Detect 3 or more consecutive 1's

Moore Method:

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



Synthesis using *D* Flip-Flops

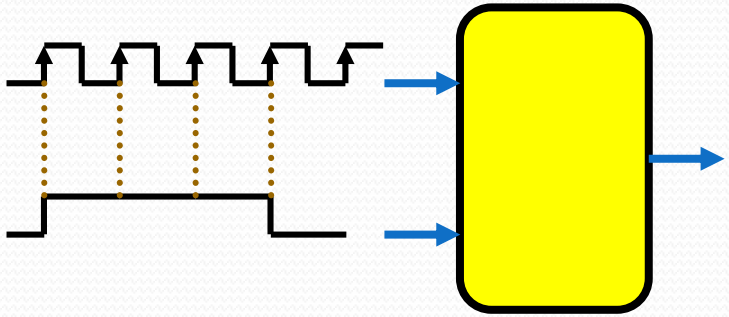
$$A(t+1) = D_A(A, B, x)$$
$$= \sum(3, 5, 7)$$
$$B(t+1) = D_B(A, B, x)$$
$$= \sum(1, 5, 7)$$
$$y(A, B, x) = \sum(6, 7)$$

Design of Clocked Sequential Circuits

- *Example*:

Detect 3 or more consecutive 1's

Moore Method:

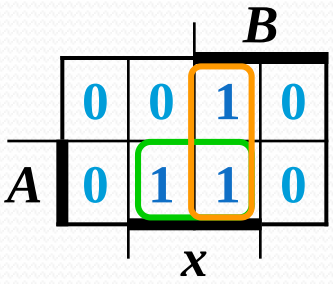


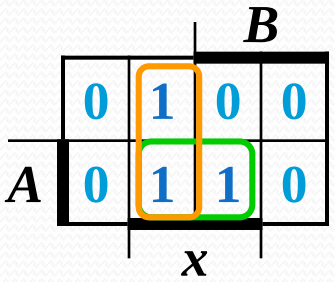
Synthesis using *D* Flip-Flops

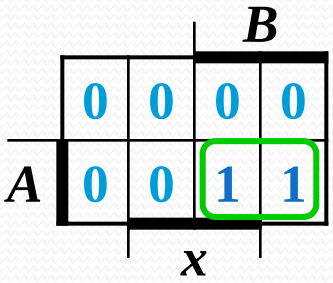
$$D_A(A, B, x) = \sum(3, 5, 7)$$
$$= A\,x + B\,x$$

$$D_B(A, B, x) = \sum(1, 5, 7)$$
$$= A\,x + B'\,x$$

$$y(A, B, x) = \sum(6, 7)$$
$$= A\,B$$





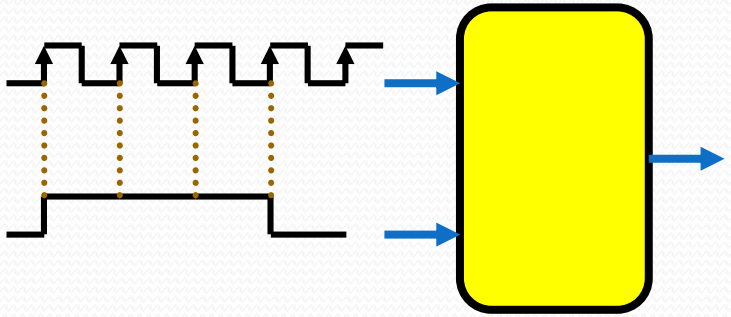


Design of Clocked Sequential Circuits

- *Example*:

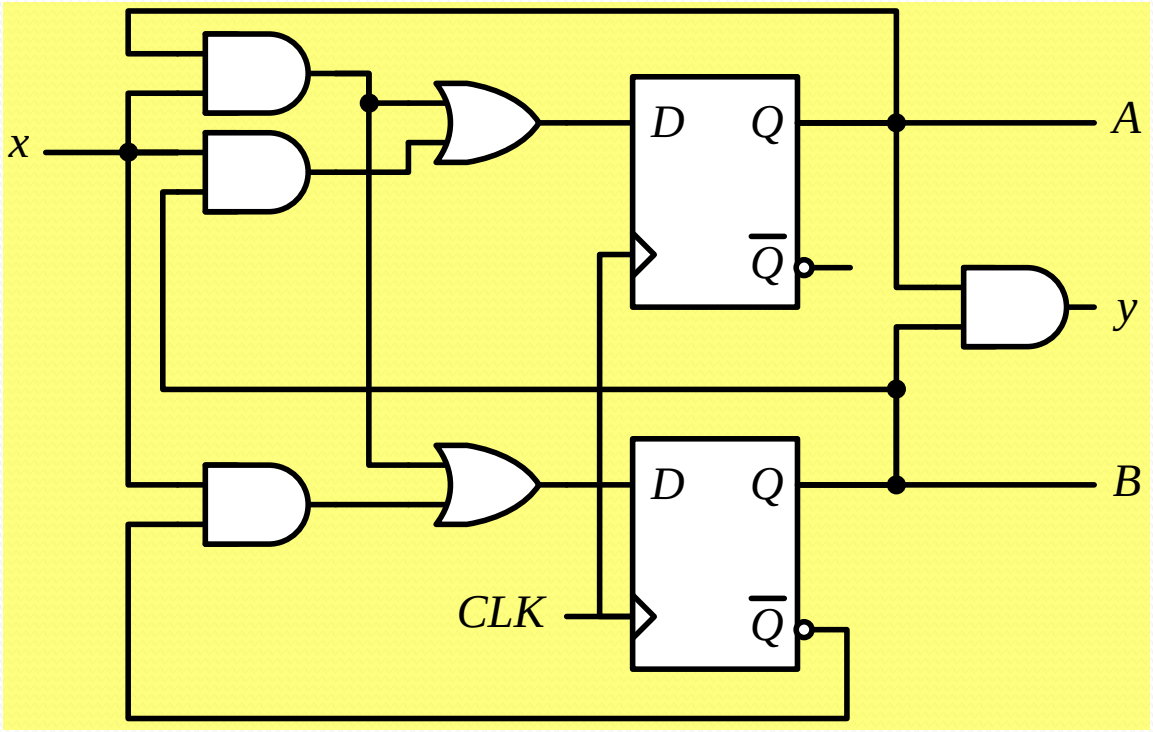
Detect 3 or more consecutive 1's

Moore Method:



Synthesis using *D* Flip-Flops

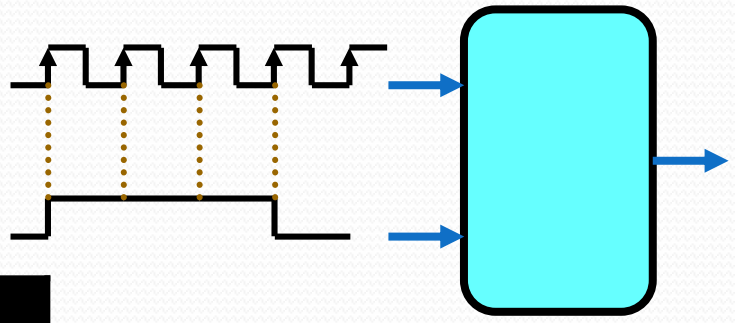
$$D_A = A\,x + B\,x$$
$$D_B = A\,x + B'\,x$$
$$y = A\,B$$



Design of Clocked Sequential Circuits with *T* F.F.

- *Example*:

Detect 3 or more consecutive 1's



Present State		Input	Next State		Flip-Flop Inputs			
A	*B*	*x*	*A*	*B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	0	0	x	x	1
0	1	1	1	0	1	x	x	1
1	0	0	0	0	x	1	0	x
1	0	1	1	1	x	0	1	x
1	1	0	0	0	x	1	x	1
1	1	1	1	1	x	0	x	0

Synthesis using *JK* F.F.

$J_A(A, B, x) = \sum(3)$

$d_{JA}(A, B, x) = \sum(4,5,6,7)$

$K_A(A, B, x) = \sum(4, 6)$

$d_{KA}(A, B, x) = \sum(0,1,2,3)$

$J_B(A, B, x) = \sum(1, 5)$

$d_{JB}(A, B, x) = \sum(2,3,6,7)$

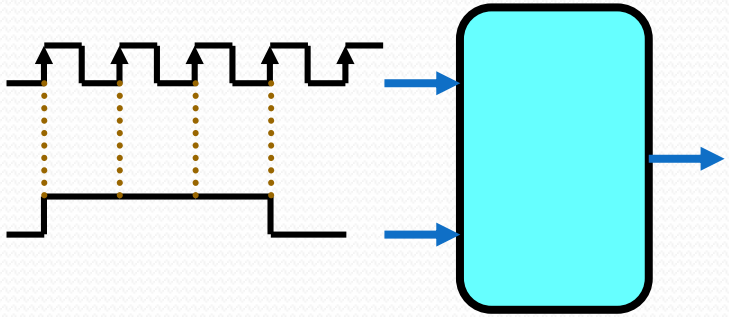
$K_B(A, B, x) = \sum(2, 3, 6)$

$d_{KB}(A, B, x) = \sum(0,1,4,5)$

Design of Clocked Sequential Circuits with *T* F.F.

- *Example*:

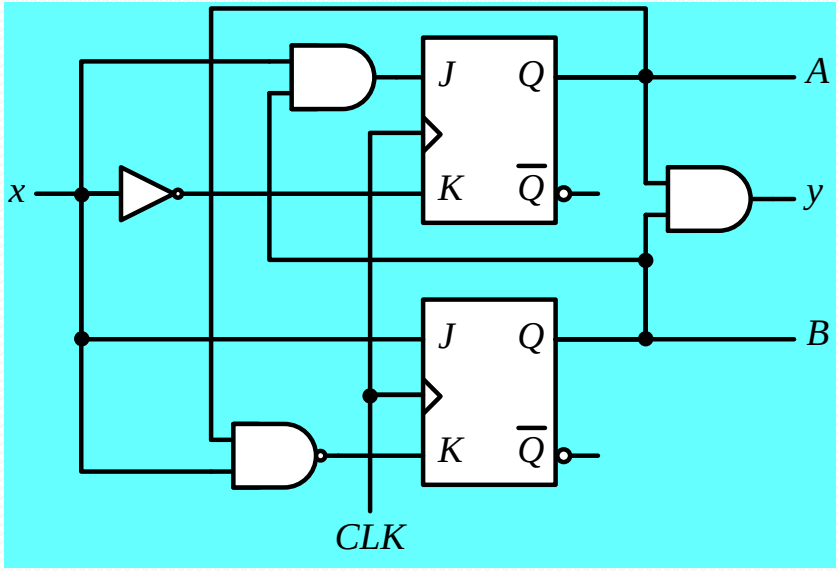
Detect 3 or more consecutive 1's

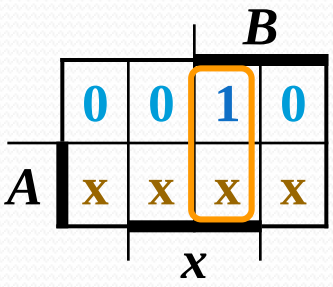


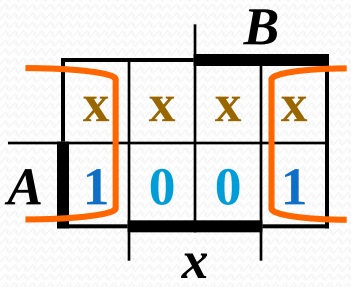
Synthesis using *JK* Flip-Flops

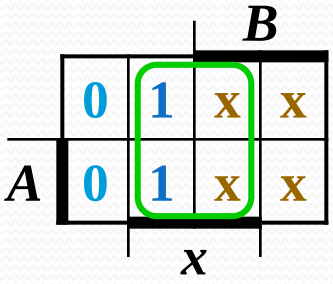
$J_A = B\,x \qquad K_A = x'$

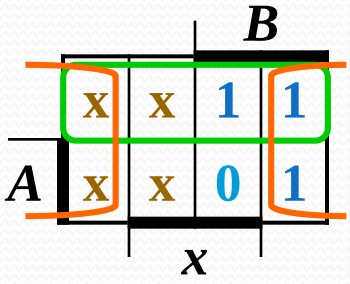
$J_B = x \qquad K_B = A' + x'$







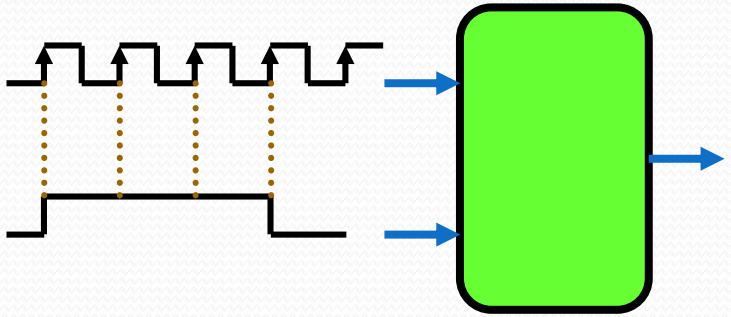




Design of Clocked Sequential Circuits with *T* F.F.

- *Example*:

Detect 3 or more consecutive 1's



Present State		Input	Next State		F.F. Input	
A	*B*	*x*	*A*	*B*	T_A	T_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	0	0	1
0	1	1	1	0	1	1
1	0	0	0	0	1	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	1	1	1	0	0

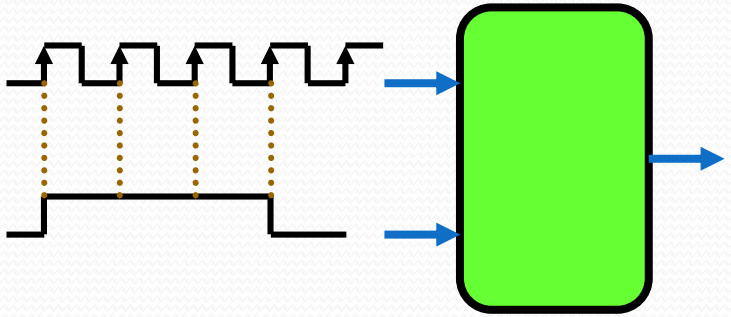
Synthesis using *T* Flip-Flops

$$T_A(A, B, x) = \sum(3, 4, 6)$$
$$T_B(A, B, x) = \sum(1, 2, 3, 5, 6)$$

Design of Clocked Sequential Circuits with *T* F.F.

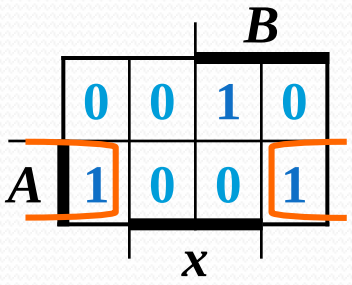
- *Example*:

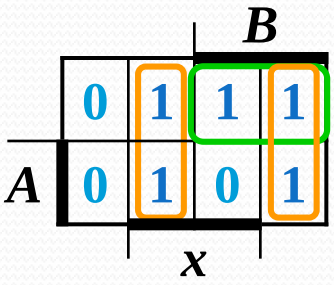
Detect 3 or more consecutive 1's

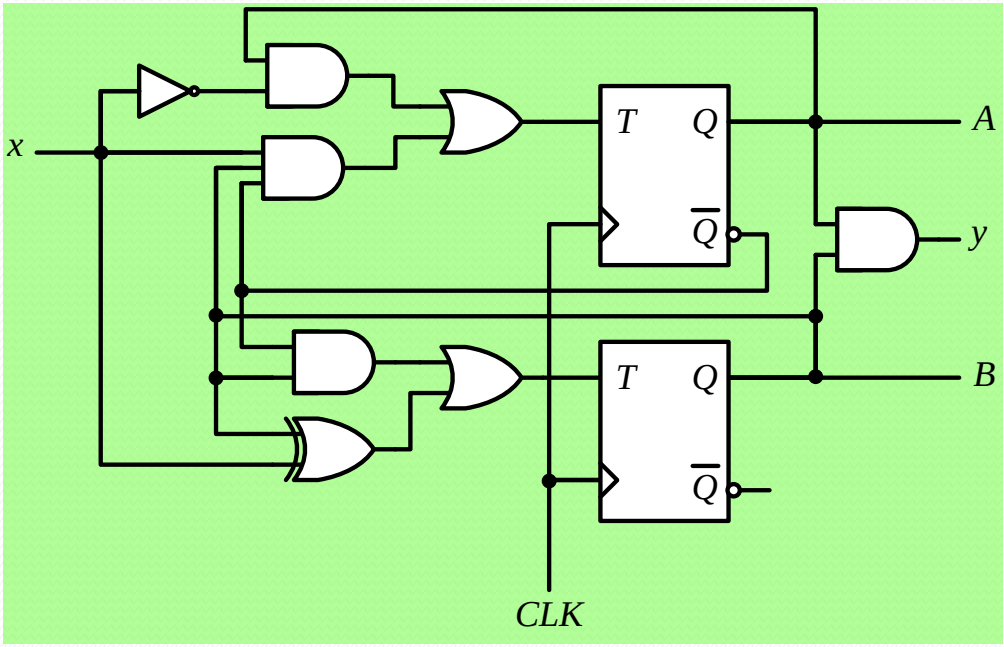


Synthesis using *T* Flip-Flops

$$T_A = A\,x' + A'\,B\,x$$
$$T_B = A'B + B \oplus x$$







Conversion Between Flip-Flop Types

Procedure uses excitation tables

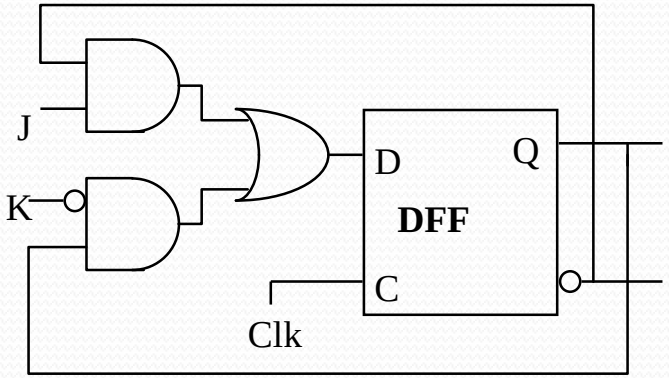
Method: to realize a type **A** flip-flop using a type **B** flip-flop:

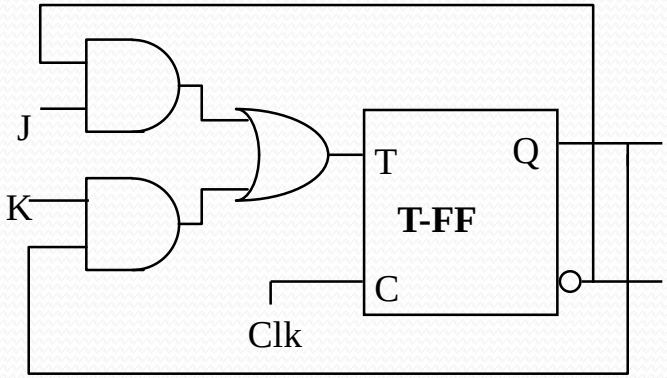
1. Start with the state-table for the A flip-flop
2. Create K-maps to express **B** Flip-flop as functions of inputs of **A** Flip-Flop.
3. Fill in K-maps with appropriate values for **B** Flip-flop to cause the same state transition as in the **A** Flip-Flop transition table.

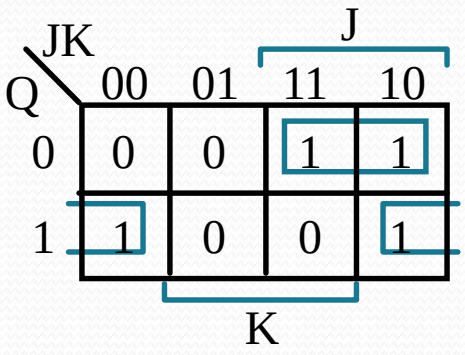
Q	Q^+	R	S	J	K	T	D
0	0	X	0	0	X	0	0
0	1	0	1	1	X	1	1
1	0	1	0	X	1	1	0
1	1	0	X	X	0	0	1

Example: Implement JK-FF using a D-FF and T-FF

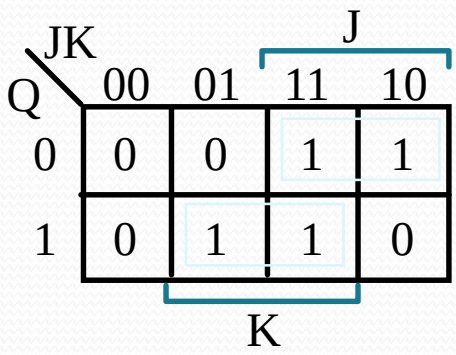
J	K	Q	Q+	D	T
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1







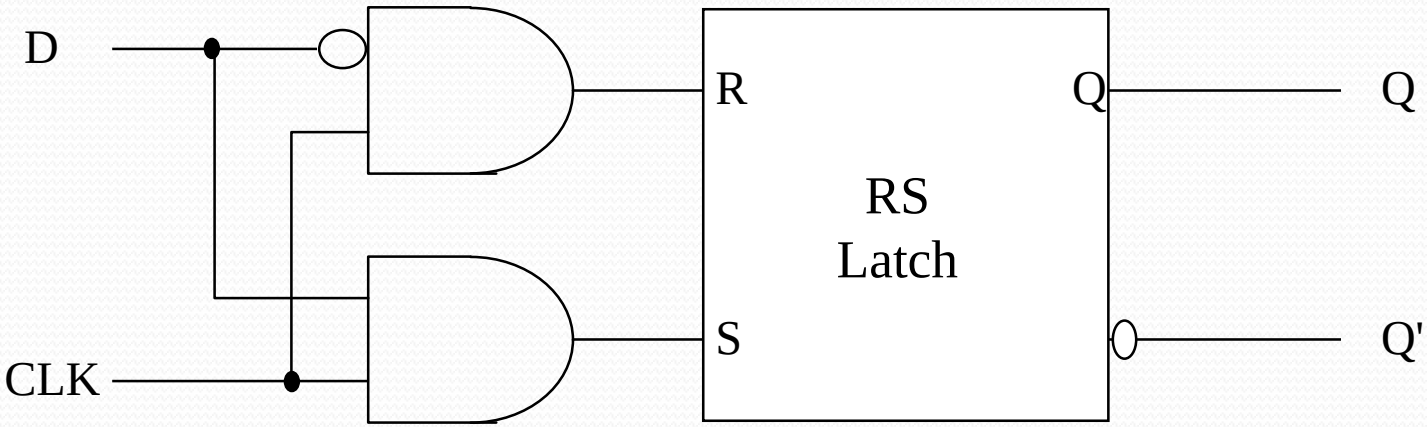
D= JQ' + K'Q



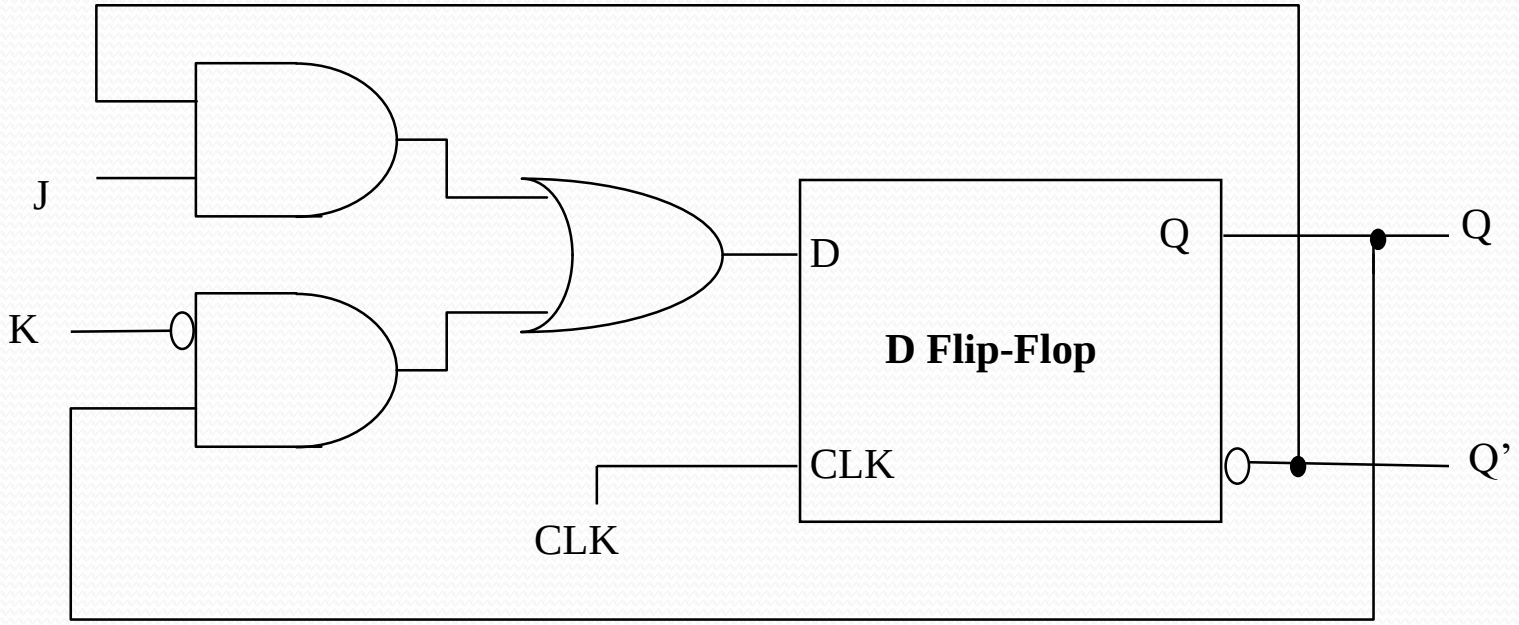
T= JQ' + KQ

- **Realization of One Flip-Flop using Another Flip-Flop**

1. D flip-flop using SR flip-flop



2. JK flip-flop using D flip-flop



3. T-FF using a D-FF