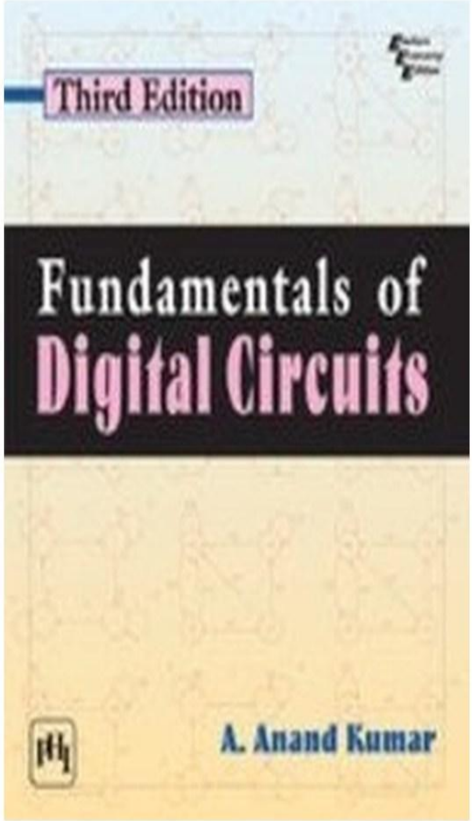
Digital Electronics

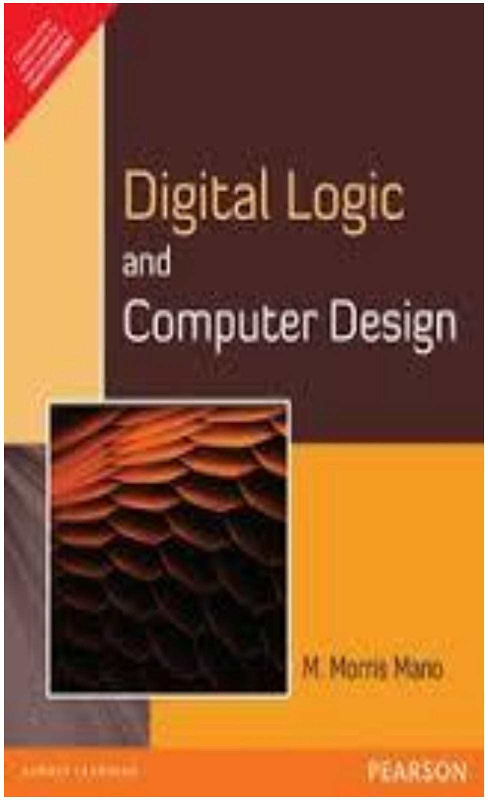
(19-204-0302)

Text Books

- 1. M. Morris Mano, *Digital Logic & Computer Design*, 4/e, Pearson Education, 2013.

- 2. FUNDAMENTALS OF DIGITAL CIRCUITS THIRD EDITION A. Anand Kumar





Module - 3

Syllabus - Module 3

- Sequential Logic Circuits: Latches, Flip Flops, SR, JK, T, D. Triggering of FF, Conversion
- Master/Slave FF, Analysis of clocked sequential circuits state reduction and assignment
- Design of clocked sequential circuits, Shift registers, Design of Counters, Asynchronous and Synchronous ripple counters, Ring counters, Johnson counter

Sequential Logic Circuits

- Sequential switching circuits are those whose output levels at any instant of time are dependent not only on the levels present at the inputs at that time, but also on the state of the circuit, i.e. on the prior input level conditions (i.e. on its past inputs).

- The past history is provided by feedback from the output back to the input.

- It means that sequential switching circuits have memory.

- Sequential circuits are thus made of combinational circuits and memory elements.

Examples for Switching Circuits

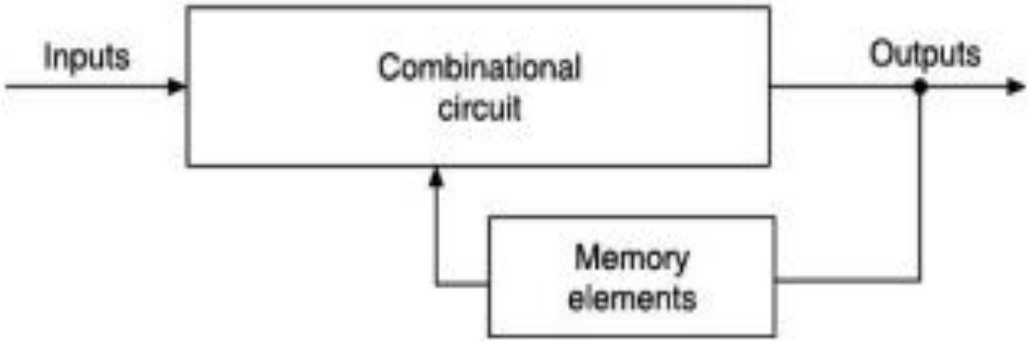
- Combinational

- Parallel adders, subtractors, encoders, decoders, code converters, parity bit generators, etc.

- Sequential

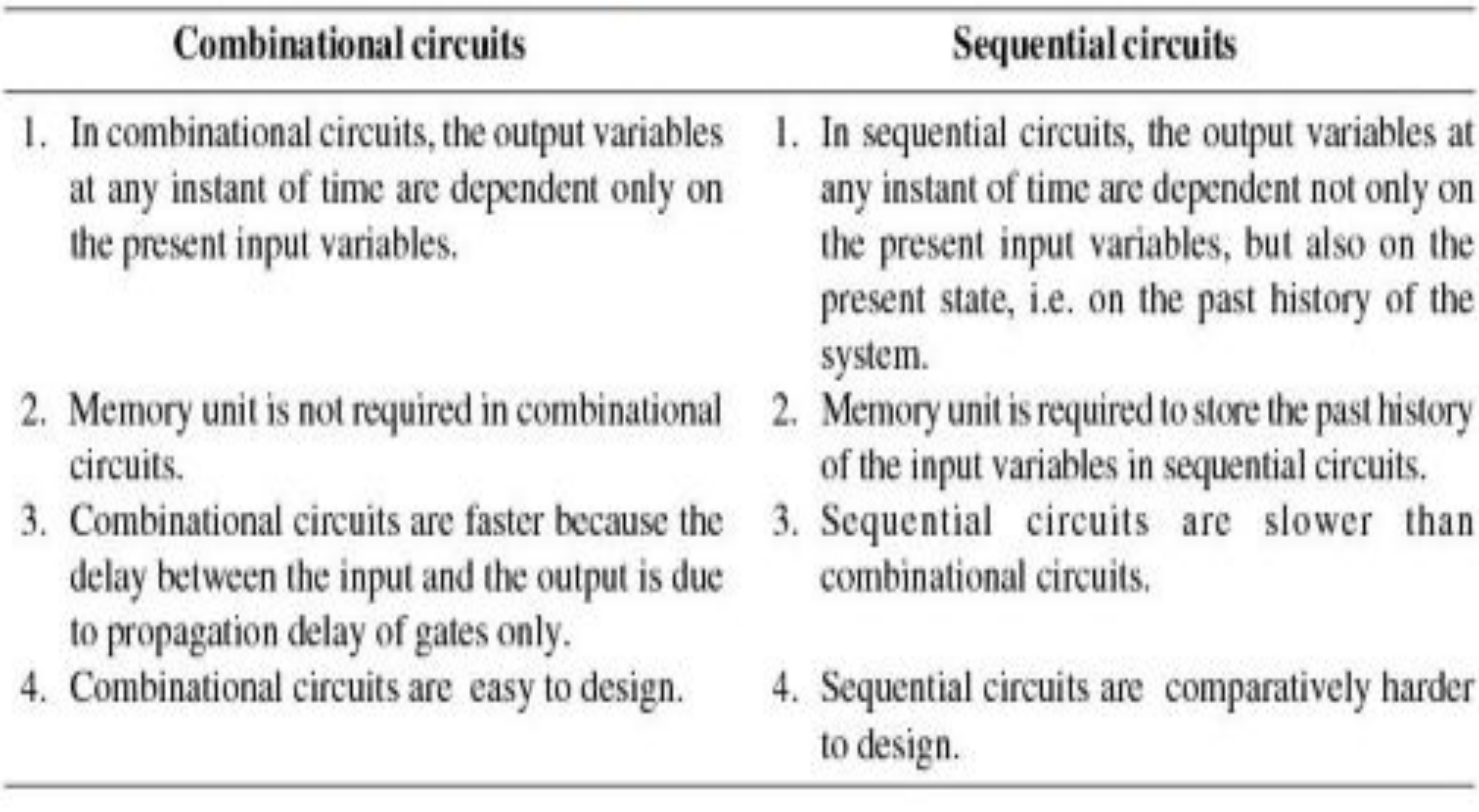
- Counters, shift registers, serial adders, sequence generators, logic function generators, etc.

Block diagram of a sequential circuit.



- Figure shows a block diagram of a sequential circuit.
- The memory elements are connected to the combinational circuit as a feedback path.
- The information stored in the memory element at any given time defines the present state of the sequential circuit.
- The present state and the external inputs determine the outputs and the next state of the sequential circuit.

Comparison between combinational and sequential circuits



Combinational circuits	Sequential circuits
1. In combinational circuits, the output variables at any instant of time are dependent only on the present input variables.	1. In sequential circuits, the output variables at any instant of time are dependent not only on the present input variables, but also on the present state, i.e. on the past history of the system.
2. Memory unit is not required in combinational circuits.	2. Memory unit is required to store the past history of the input variables in sequential circuits.
3. Combinational circuits are faster because the delay between the input and the output is due to propagation delay of gates only.	3. Sequential circuits are slower than combinational circuits.
4. Combinational circuits are easy to design.	4. Sequential circuits are comparatively harder to design.

Classification of Sequential Circuits

- The sequential circuits may be classified as synchronous sequential circuits and asynchronous sequential circuits depending on the timing of signals.

- Synchronous sequential circuits.

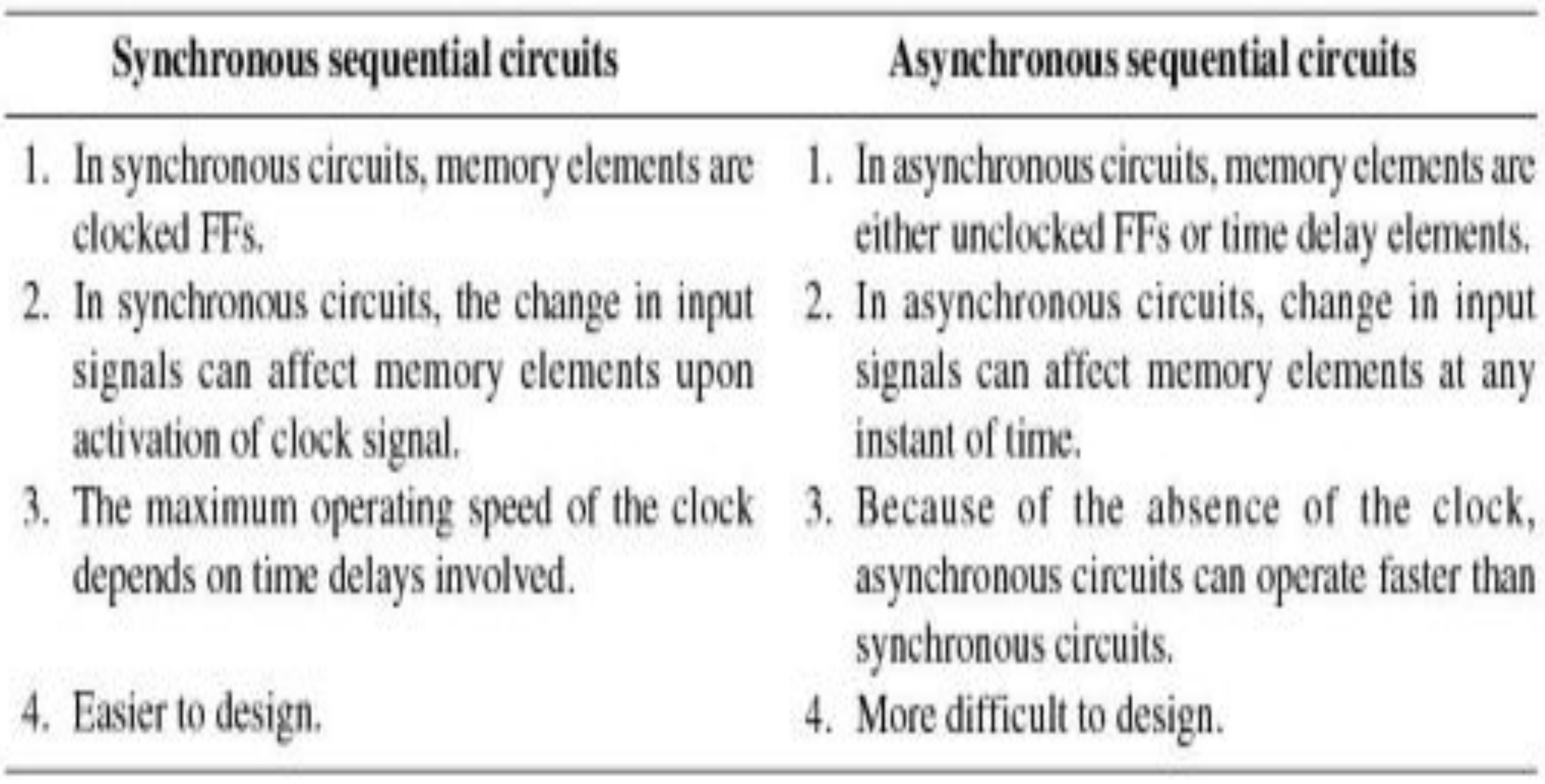
- The sequential circuits which are controlled by a clock are called synchronous sequential circuits.

- These circuits will be active only when clock signal is present.

Asynchronous Sequential Circuits.

- The sequential circuits which are not controlled by a clock are called asynchronous sequential circuits, i.e. the sequential circuits in which events can take place any time the inputs are applied are called asynchronous sequential circuits.

Comparison between synchronous and asynchronous sequential circuits



Synchronous sequential circuits	Asynchronous sequential circuits
1. In synchronous circuits, memory elements are clocked FFs.	1. In asynchronous circuits, memory elements are either unclocked FFs or time delay elements.
2. In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.	2. In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
3. The maximum operating speed of the clock depends on time delays involved.	3. Because of the absence of the clock, asynchronous circuits can operate faster than synchronous circuits.
4. Easier to design.	4. More difficult to design.

Flip Flops

- The most important memory element in a sequential circuit is the flip-flop, which is made up of an assembly of logic gates.

- A flip-flop (FF), known more formally as a bistable multivibrator, has two stable states.

- It can remain in either of the states indefinitely. Its state can be changed by applying the proper triggering signal.

- It is also called a binary or one-bit memory.

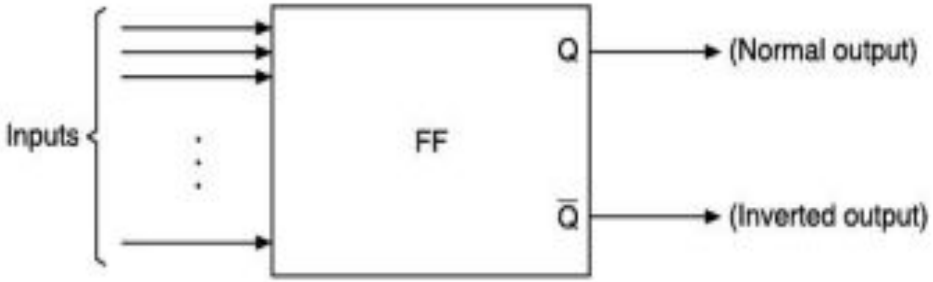


Figure 10.3 General flip-flop symbol.

- The flip-flop has two outputs, labelled Q & $\overline{Q}$

- The Q output is the normal output of the flip-flop and $\overline{Q}$ is the inverted output.

- The state of the flip-flop always refers to the state of the normal output Q.

- A flip-flop is said to be in HIGH state or logic 1 state or SET state when Q = 1, and in LOW state or logic 0 state or RESET state or CLEAR state when Q = 0.

- The input signals which command the flip-flop to change state are called excitations.

- Flip-flop serves as a storage device.

- It stores a 1 when its Q output is a 1, and stores a 0 when its Q output is a 0.

- Flip-flops are the fundamental components of shift registers and counters.

Latched Flipflops

- It refers to non-clocked flip-flops, because these flip-flops ‘latch on’ to a 1 or a 0 immediately upon receiving the input pulse called SET or RESET.

- They are not dependent on the clock signal for their operation, i.e. a latch is a sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

Active High Input Latch & Active Low Input Latch.

- A latch may be an active-high input latch or an active-low input latch.

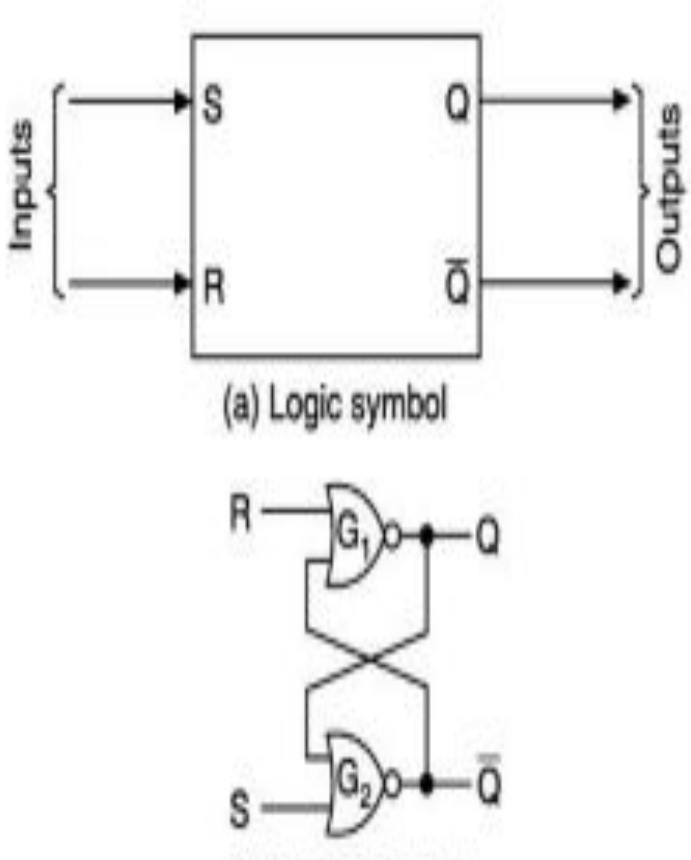
- Active-high means that the SET and RESET inputs are normally resting in the LOW state and one of them will be pulsed HIGH whenever we want to change the latch outputs.

- Active-low means that the SET and RESET inputs are normally resting in the HIGH state and one of them will be pulsed LOW whenever we want to change the latch outputs.

The S-R Latch (SR Flip Flop)

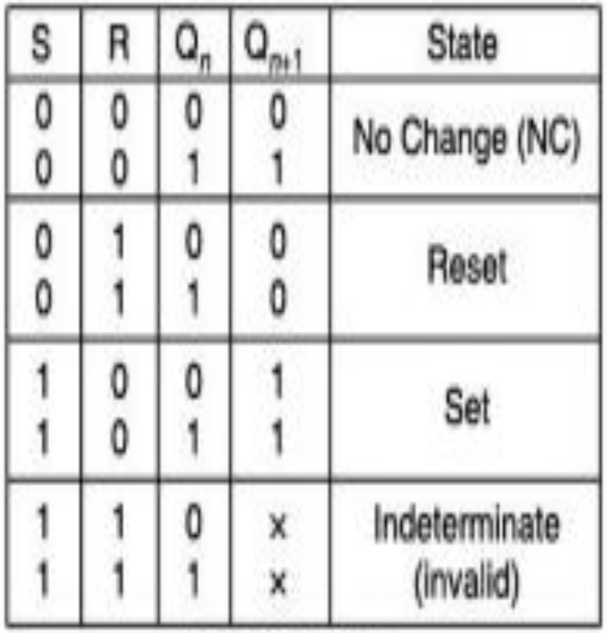
- The simplest type of flip-flop is called an S-R latch.
- It has two outputs labeled Q and $\overline{Q}$ and two inputs labeled S and R.
- It can be constructed using either two cross-coupled NAND gates or two-cross coupled NOR gates.
- Using two NOR gates, an active-high S-R latch can be constructed and using two NAND gates an active-low S-R latch can be constructed.

S R Latch / S R Flip Flop (Active High)



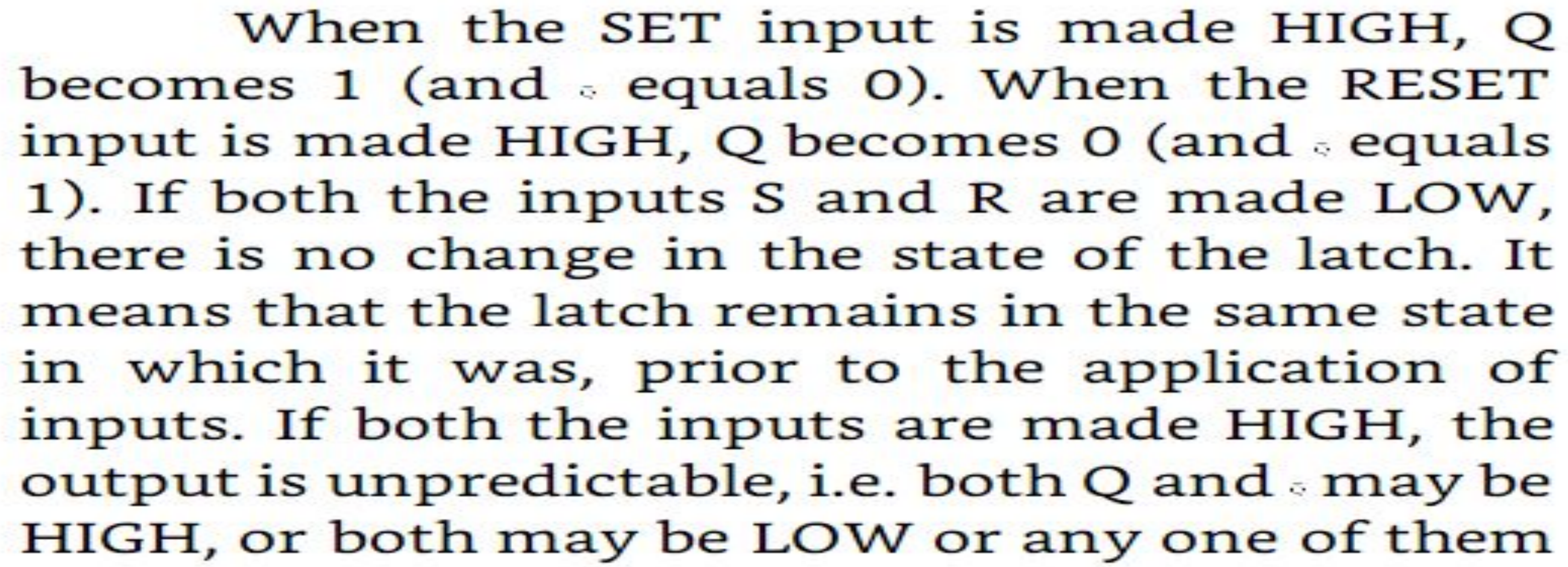
(a) Logic symbol

(b) Logic diagram

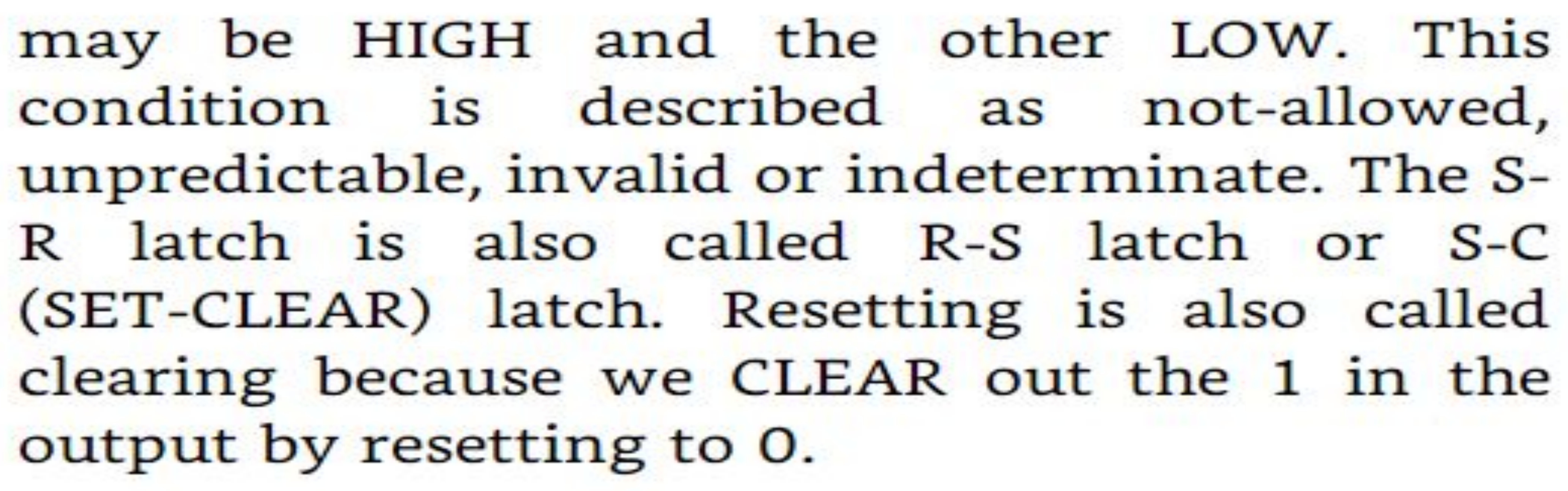


S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Indeterminate (invalid)
1	1	1	x	

(c) Truth table



When the SET input is made HIGH, Q becomes 1 (and equals 0). When the RESET input is made HIGH, Q becomes 0 (and equals 1). If both the inputs S and R are made LOW, there is no change in the state of the latch. It means that the latch remains in the same state in which it was, prior to the application of inputs. If both the inputs are made HIGH, the output is unpredictable, i.e. both Q and may be HIGH, or both may be LOW or any one of them



may be HIGH and the other LOW. This condition is described as not-allowed, unpredictable, invalid or indeterminate. The *S*-R latch is also called R-*S* latch or *S*-C (SET-CLEAR) latch. Resetting is also called clearing because we CLEAR out the 1 in the output by resetting to 0.

Gated Latches (Clocked Flip-Flops)

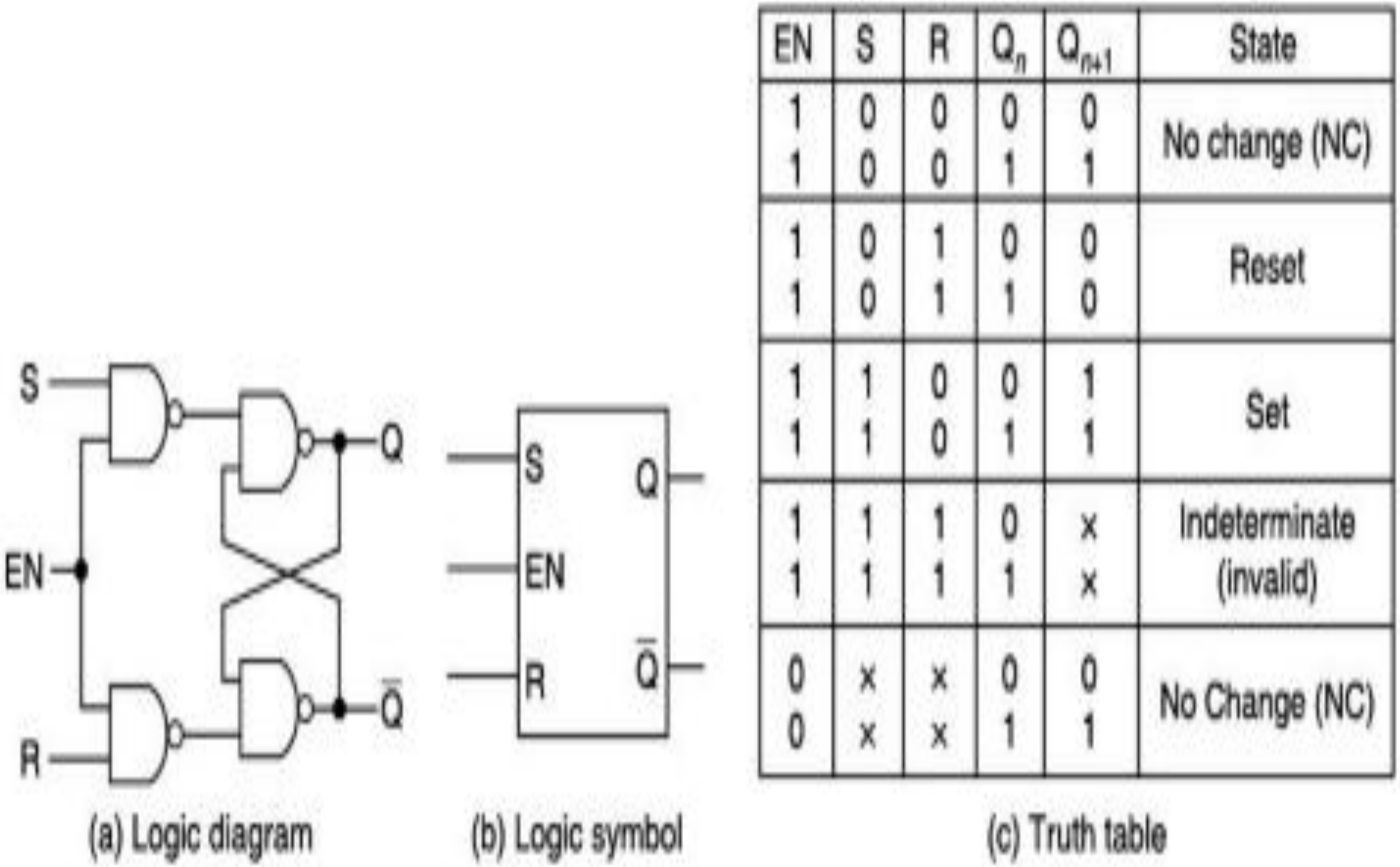
- In the latches described earlier, the output can change state any time the input conditions are changed. So, they are called asynchronous latches.

- Synchronous SR Latch

- A gated S-R latch requires an enable (EN) input. Its S and R inputs will control the state of the flip-flop only when the enable is HIGH.

- When the enable is LOW, the inputs become ineffective and no change of state can take place.

- The ENABLE input may be a clock. So, a gated S-R latch is also called a clocked S-R latch or synchronous S-R latch.



(a) Logic diagram

(b) Logic symbol

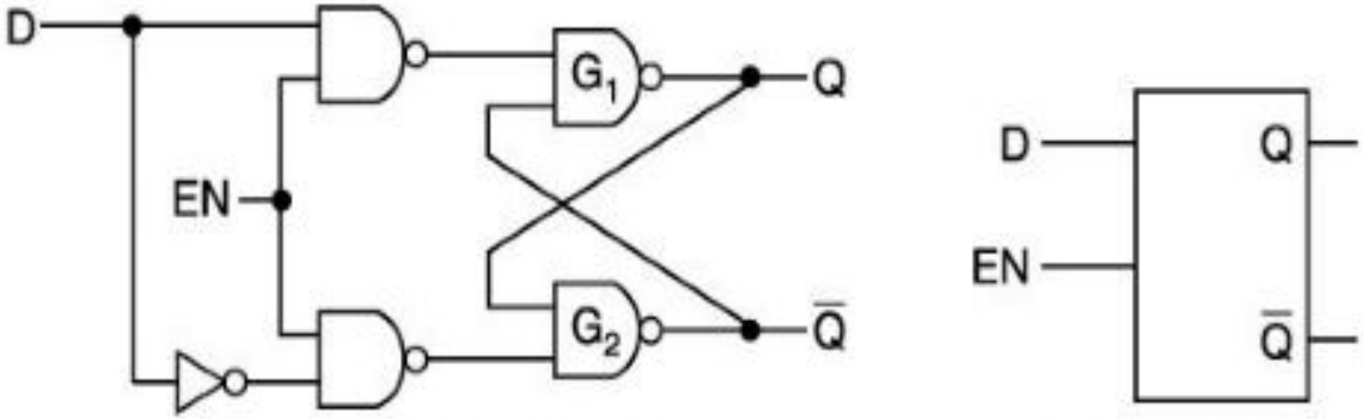
EN	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	x	Indeterminate (invalid)
1	1	1	1	x	
0	x	x	0	0	No Change (NC)
0	x	x	1	1	

(c) Truth table

Figure 10.7 A gated S-R latch.

The gated D-latch (D Flip Flop)

- A single input (D Input) latch is called a D Latch.
- The output Q follows the D input when EN is High.
- D = 1, cause the latch to SET when ENABLED.
- D = 0, cause the latch to RESET when ENABLED.



(a) Logic diagram

(b) Logic symbol

EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	×	0	0	No Change (NC)
0	×	1	1	

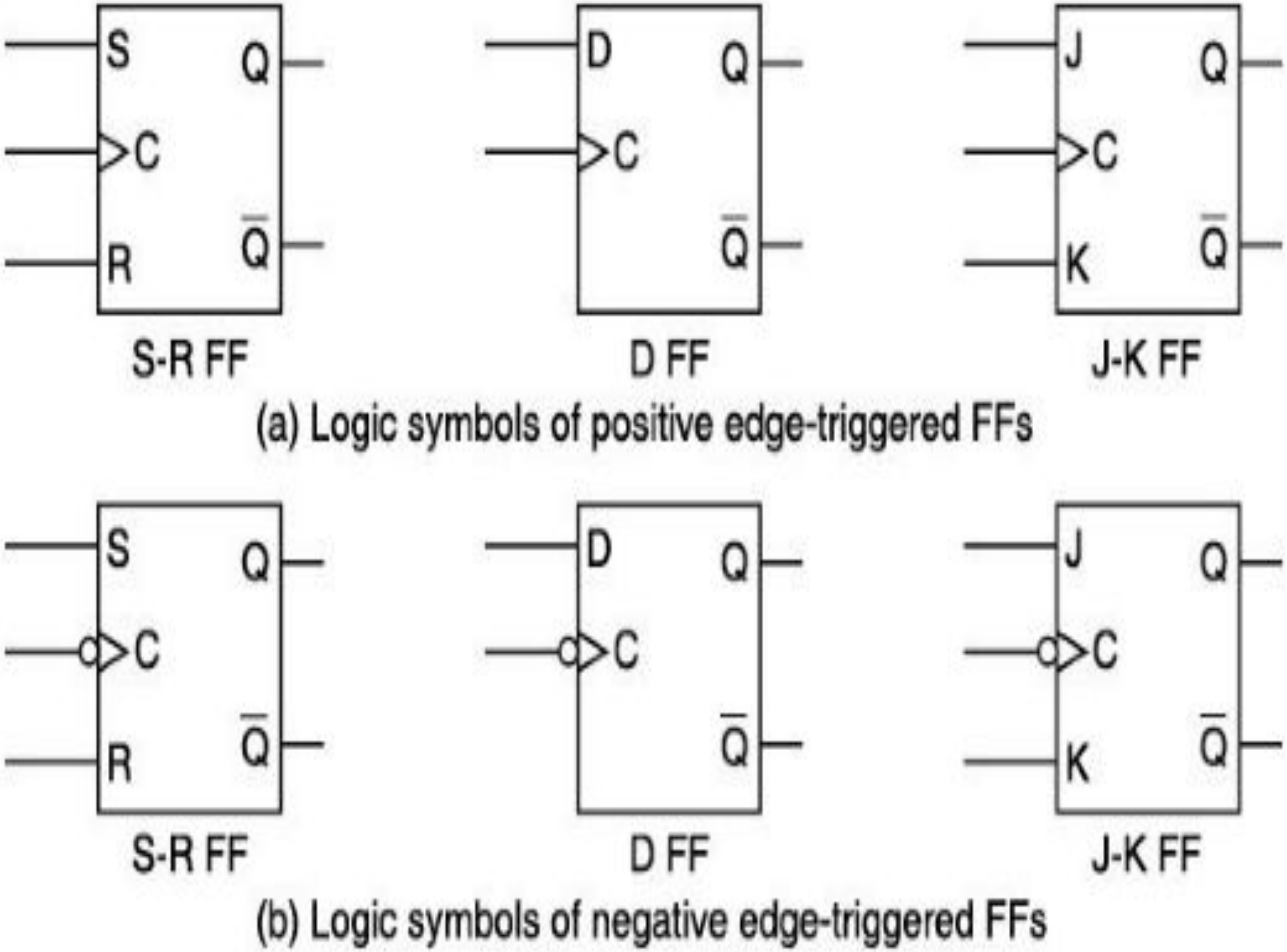
(c) Truth table

Figure 10.9 A gated D-latch.

Edge-Triggered Flip-Flops

- The flip-flops using the clock signal are called the clocked flip-flops.
- The clock signal is distributed to all parts of the system and most of the system outputs can change state only when the clock makes a transition.
- Clocked flip-flops may be positive edge-triggered or negative edge-triggered.
- Positive edge-triggered flip-flops are those in which ‘state transitions’ take place only at the positive-going (0 to 1, or LOW to HIGH) edge of the clock pulse, and negative edge-triggered flip-flops are those in which ‘state transitions’ take place only at the negative-going (1 to 0, or HIGH to Low) edge of the clock signal.
- Positive-edge triggering is indicated by a ‘triangle’ at the clock terminal of the flip-flop. Negative-edge triggering is indicated by a ‘triangle’ with a bubble at the clock terminal of the flip-flop.

The edge-triggering is also called *dynamic triggering.*

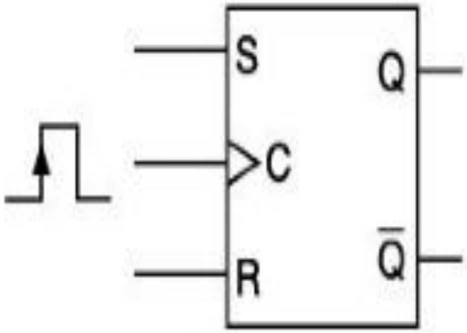


(a) Logic symbols of positive edge-triggered FFs

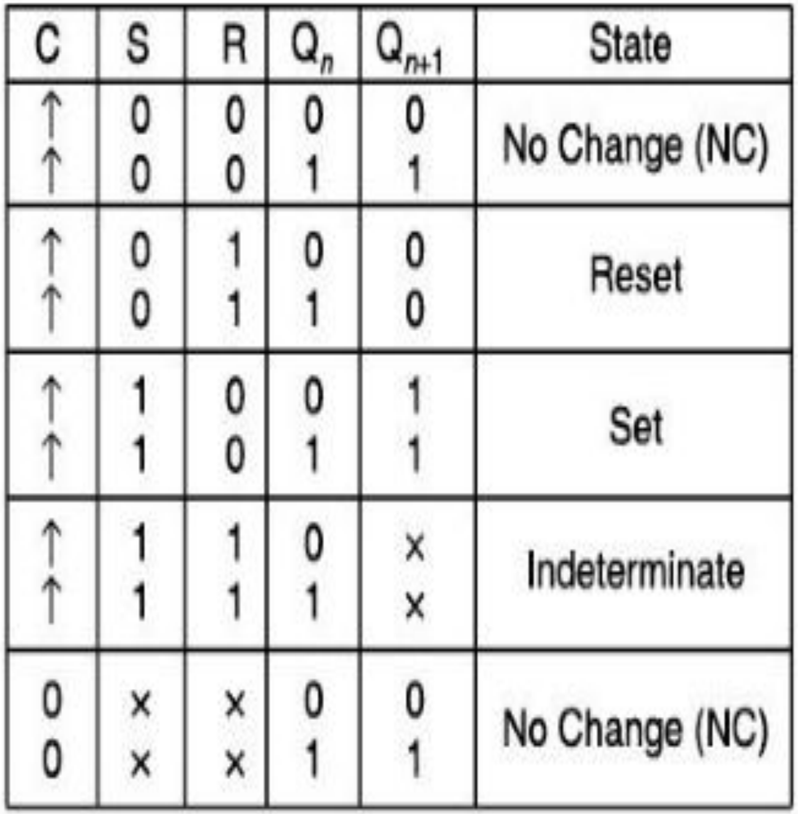
(b) Logic symbols of negative edge-triggered FFs

Figure 10.10 Edge-triggered flip-flops.

The edge-triggered S-R flip-flop



(a) Logic symbol



C	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change (NC)
↑	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	×	Indeterminate
↑	1	1	1	×	
0	×	×	0	0	No Change (NC)
0	×	×	1	1	

(b) Truth table

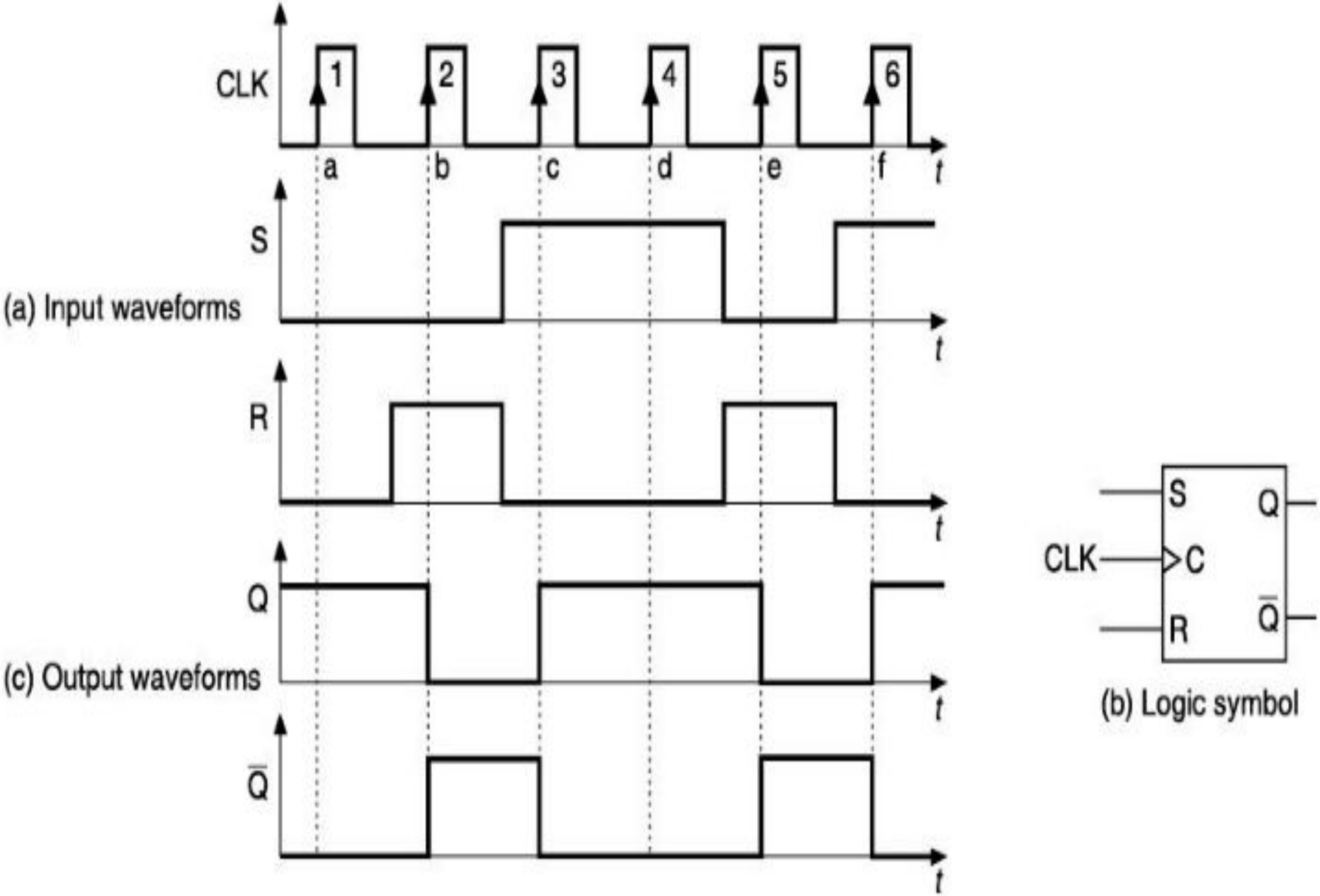
Figure 10.12 Positive edge-triggered S-R flip-flop.

- The S and R inputs of the S-R flip-flop are called the synchronous control inputs because data on these inputs affect the flip-flop’s output only on the triggering (positive going) edge of the clock pulse.

- Without a clock pulse, the S and R inputs cannot affect the output.

- When S is HIGH and R is LOW, the Q output goes HIGH on the positive-going edge of the clock pulse and the flip-flop is SET. (If it is already in SET state, it remains SET).

- When S is LOW and R is HIGH, the Q output goes LOW on the positive-going edge of the clock pulse and the flip-flop is RESET, i.e. cleared. (If it is already in RESET state, it remains RESET).



(a) Input waveforms

(c) Output waveforms

(b) Logic symbol

Figure 10.13 Example 10.2: Waveforms—positive edge-triggered S-R flip-flop.

1. Initially, S = 0 and R = 0 and Q is assumed to be HIGH.
2. At the positive-going transition of the first clock pulse (i.e. at a), both S and R are LOW. So, no change of state takes place. Q remains HIGH and $\bar{Q}$ remains LOW.
3. At the leading edge of the second clock pulse (i.e. at b), S = 0 and R = 1. So, the flip-flop resets. Hence, Q goes LOW and $\bar{Q}$ goes HIGH.
4. At the positive-going edge of the third clock pulse (i.e. at c), S = 1 and R = 0. So, the flip-flop sets. Hence, Q goes HIGH and $\bar{Q}$ goes LOW.
5. At the rising edge of the fourth clock pulse, S = 1 and R = 0. Since the flip-flop is already in a SET state, it remains SET. That is, Q remains HIGH and $\bar{Q}$ remains LOW.
6. The fifth pulse resets the flip-flop at its positive-going edge because S = 0 and R = 1 is the input condition and Q = 1 at that time.
7. The sixth pulse sets the flip-flop at its rising edge because S = 1 and R = 0 is the input condition and Q = 0 at that time.

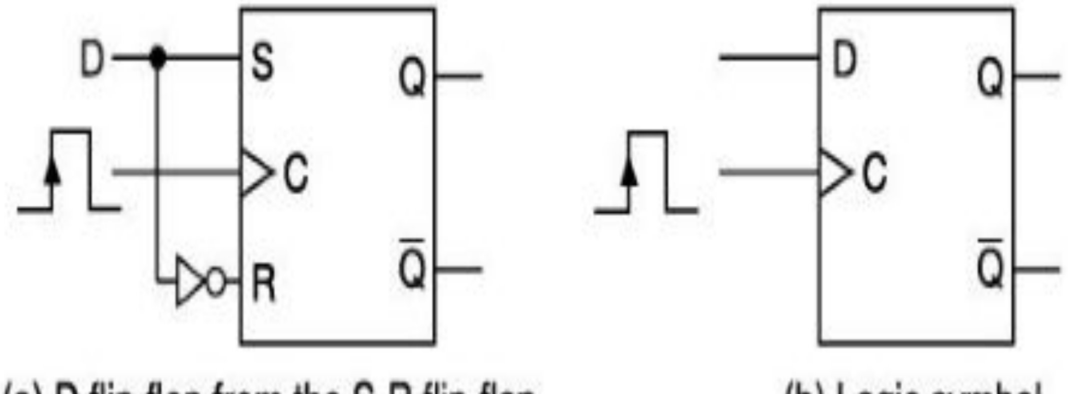
- The truth table of a negative edge-triggered S-R flip-flop is the same as that of a positive edge triggered S-R flip-flop except that the arrows point downwards.

- This flip-flop will trigger only when the clock input goes from 1 to 0.

The edge-triggered D flip-flop:

- The edge-triggered D flip-flop has only one input terminal.

- The D flip-flop may be obtained from an S-R flip-flop by just putting one inverter between the S and R terminals.



(a) D flip-flop from the S-R flip-flop

(b) Logic symbol

C	D	Q_n	Q_{n+1}	State
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
0	×	0	0	No Change (NC)
0	×	1	1	

(c) Truth table

Figure 10.15 The positive edge-triggered D flip-flop.

- The output Q will go to the same state that is present on the D input at the positive-going transition of the clock pulse.

- In other words, the level present at D will be stored in the flip-flop at the instant the positive-going transition occurs.

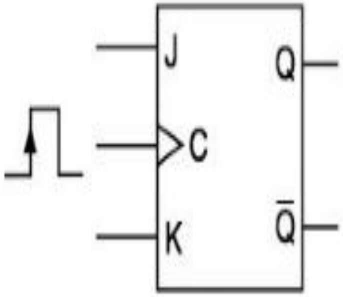
- That is, if D is a 1 and the clock is applied, Q goes to a 1 and $\overline{Q}$ to a 0 at the rising edge of the pulse and thereafter remain so.

- If D is a 0 and the clock is applied, Q goes to a 0 and $\overline{Q}$ to a 1 at the rising edge of the clock pulse and thereafter remain so.

- The <u>negative edge-triggered D flip-flop</u> operates in the same way as the positive edge-triggered D flip-flop except that the change of state takes place at the negative-going edge of the clock pulse.
- In the truth table of the negative edge triggered flip-flop the arrows point downwards.

The edge-triggered J-K flip-flop

- The functioning of the J-K flip-flop is identical to that of the S-R flip-flop, except that it has no invalid state like that of the S-R flip-flop.



(a) Logic symbol

C	J	K	Q_n	Q_{n+1}	State
↑	0	0	0	0	No Change (NC)
↑	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	
0	x	x	0	0	No Change (NC)
0	x	x	1	1	

(b) Truth table

Figure 10.17 Positive edge-triggered J-K flip-flop.

- When J = 0 and K = 0, no change of state takes place even if a clock pulse is applied.

- When J = 0 and K = 1, the flip-flop resets at the positive-going edge of the clock pulse.

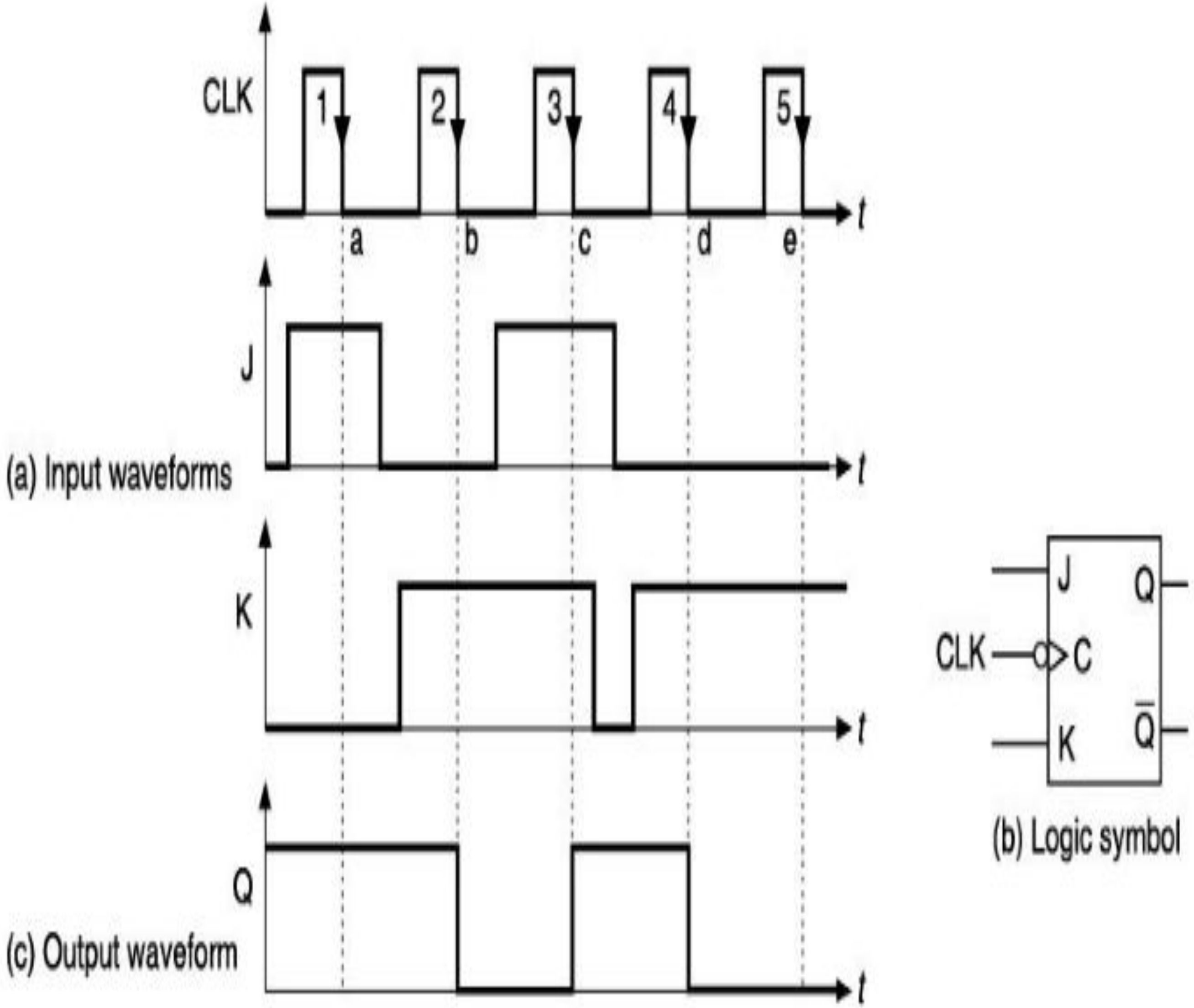
- When J = 1 and K = 0, the flip-flop sets at the positive-going edge of the clock pulse.

- When J = 1 and K = 1, the flip-flop toggles, i.e. goes to the opposite state at the positive-going edge of the clock pulse.

- In this mode, the flip-flop toggles or changes state for each occurrence of the positive-going edge of the clock pulse.

- A negative edge-triggered J-K flip-flop operates in the same way as a positive edge-triggered J-K flip-flop except that the change of state takes place at the negative-going edge of the clock pulse.

- In the truth-table of a negative edge-triggered J-K flip-flop the arrows point downwards.



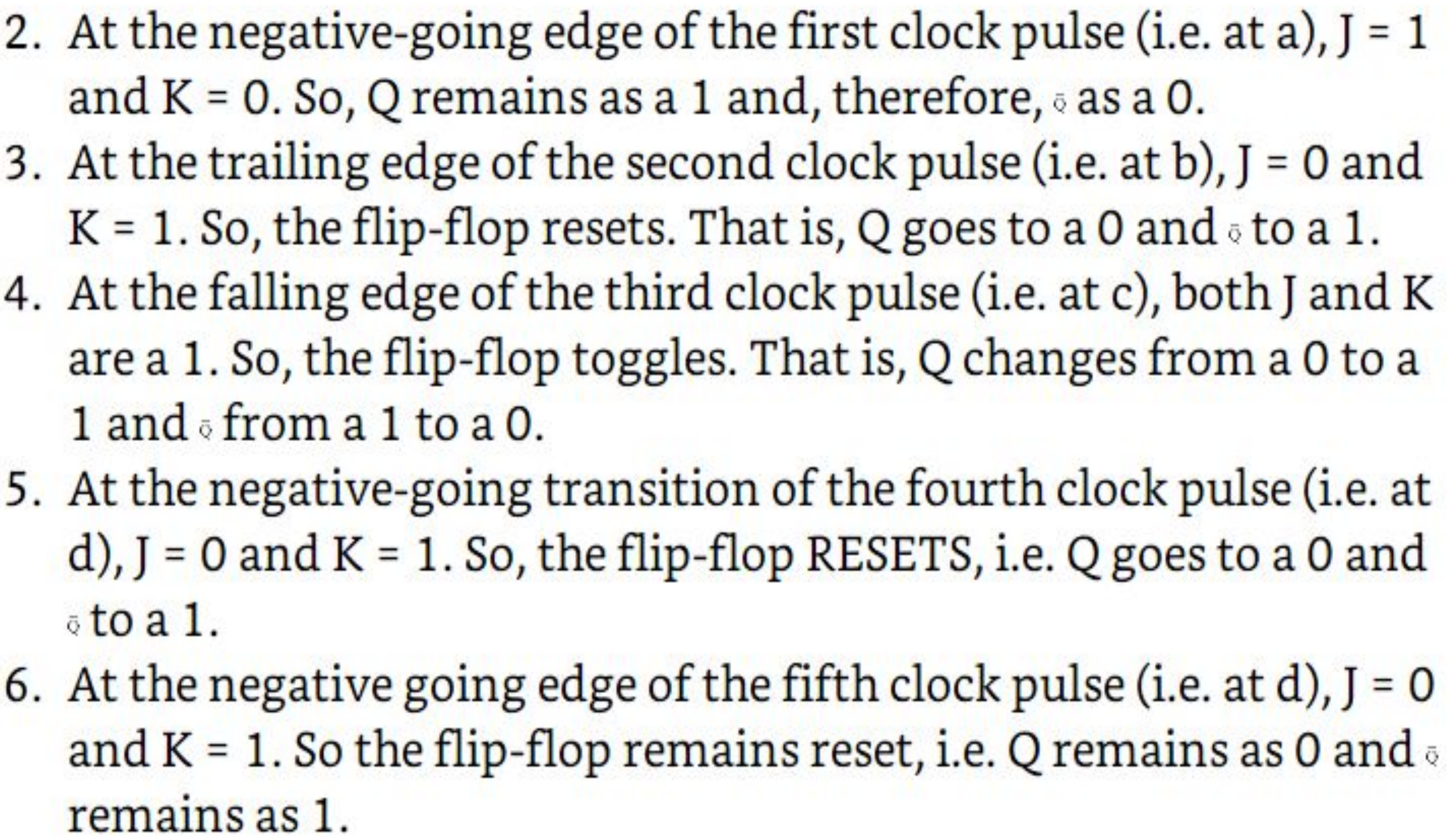
(a) Input waveforms

(b) Logic symbol

(c) Output waveform

Figure 10.18 Example 10.3: Waveforms—edge-triggered J-K flip-flop.

- 1. Initially J = 0, K = 0 and CLK = 0. Assume that the initial state of the flip-flop is a 1, i.e. Q = 1 initially.



2. At the negative-going edge of the first clock pulse (i.e. at a), J = 1 and K = 0. So, Q remains as a 1 and, therefore, $\bar{Q}$ as a 0.
3. At the trailing edge of the second clock pulse (i.e. at b), J = 0 and K = 1. So, the flip-flop resets. That is, Q goes to a 0 and $\bar{Q}$ to a 1.
4. At the falling edge of the third clock pulse (i.e. at c), both J and K are a 1. So, the flip-flop toggles. That is, Q changes from a 0 to a 1 and $\bar{Q}$ from a 1 to a 0.
5. At the negative-going transition of the fourth clock pulse (i.e. at d), J = 0 and K = 1. So, the flip-flop RESETS, i.e. Q goes to a 0 and $\bar{Q}$ to a 1.
6. At the negative going edge of the fifth clock pulse (i.e. at d), J = 0 and K = 1. So the flip-flop remains reset, i.e. Q remains as 0 and $\bar{Q}$ remains as 1.

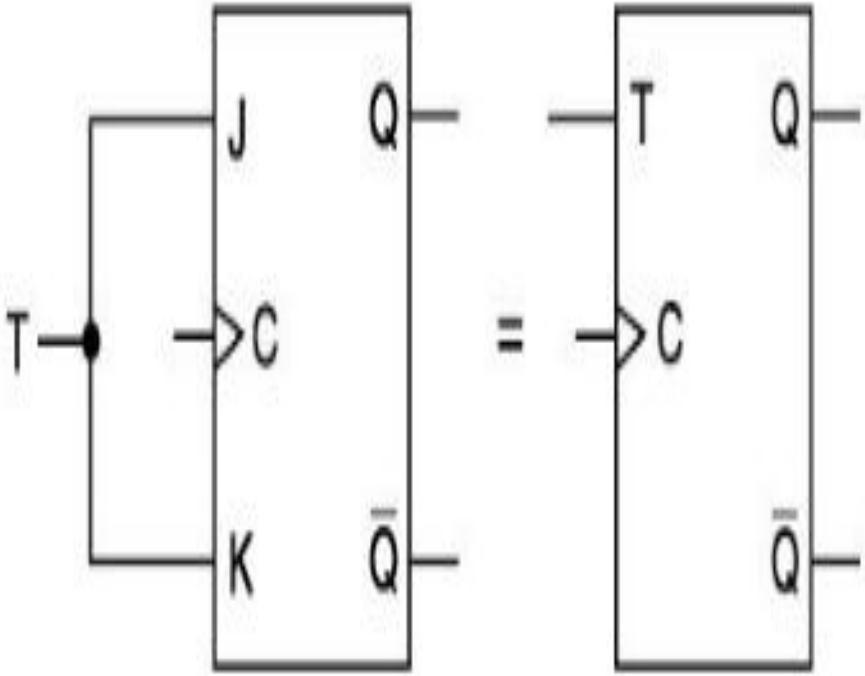
The edge-triggered T flip-flop

- A T flip-flop has a single control input, labelled T for toggle.

- When T is HIGH, the flip-flop toggles on every new clock pulse.

- When T is LOW, the flip-flop remains in whatever state it was before.

- It is easy to convert a J-K flip-flop to the functional equivalent of a T flip-flop by just connecting J and K together and labelling the common connection as T.



(a) T flip-flop from JK flip-flop

(b) Logic symbol

C	T	Q_n	Q_{n+1}	State
↑	0	0	0	No Change (NC)
↑	0	1	1	
↑	1	0	1	Toggle
↑	1	1	0	
0	x	0	0	No Change (NC)
0	x	1	1	

(c) Truth table

Figure 10.20 Edge-triggered T flip-flop.

Excitation table (State Table) of a flip-flop

- A table which lists the present state, the next state and the excitations of a flip-flop is called the excitation table of a flip-flop,

  i.e., the excitation table is a table which indicates the excitations required to take the flip-flop from the present state to the next state.

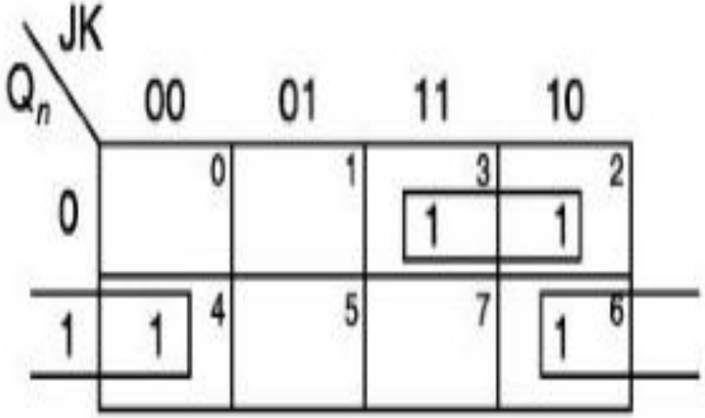
- The characteristic equation of a flip-flop is the equation expressing the next state of a flip-flop in terms of its present state and present excitations.

- To obtain the characteristic equation of a flip-flop,
- 1. Write the excitation requirements of the flip-flop.
- 2. Draw a K-map for the next state of the flip- flop in terms of its present state and inputs and simplify it

Excitation Table – J K Flip Flop

Present state	Inputs		Next state
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a) Excitation requirements of JK flip-flop



Q_n \ JK	00	01	11	10
0	0	1	3 1	2 1
1	1 4	5	7	1 6

K-map for Q_{n+1}

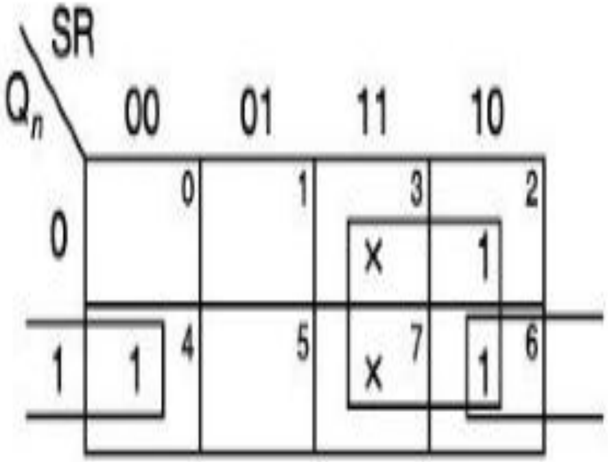
The characteristic equation of a JK flip-flop is

$$Q_{n+1} = \overline{Q}_n J + Q_n \overline{K}$$

Excitation Table – S R Flip Flop

Present state	Inputs		Next state
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

(b) Excitation requirements of SR flip-flop



Q_n \ SR	00	01	11	10
0			x	1
1	1		x	1

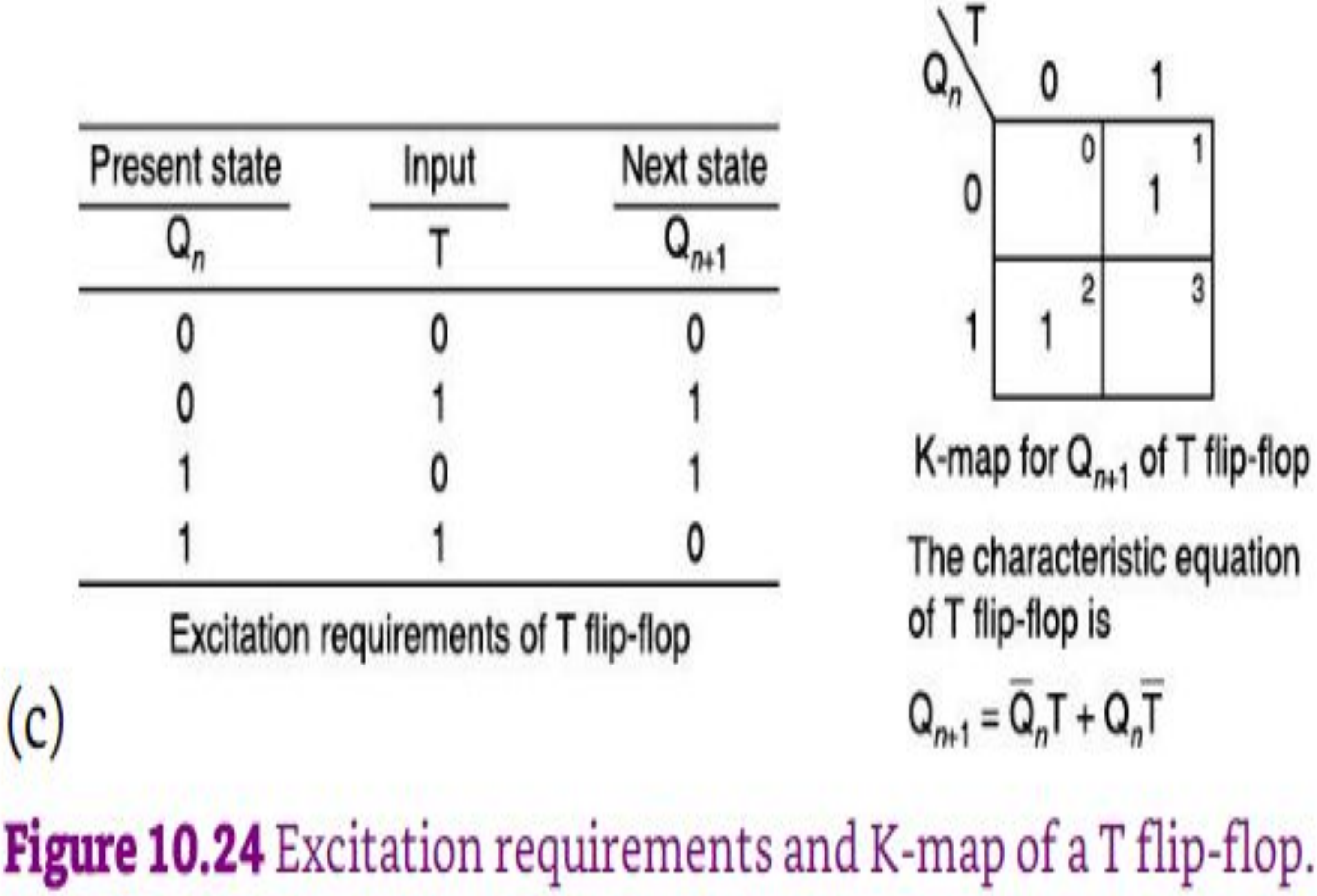
K-map for Q_{n+1} of SR flip-flop

The characteristic equation of SR flip-flop

$$Q_{n+1} = S + Q_n\bar{R}$$

Figure 10.23 Excitation requirements and K-map of an S-R flip-flop.

Excitation Table – T Flip Flop



Present state	Input	Next state
Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation requirements of T flip-flop

(c)

$Q_n \backslash T$	0	1
0	^{0}	1 ^{1}
1	1 ^{2}	^{3}

K-map for Q_{n+1} of T flip-flop

The characteristic equation of T flip-flop is

$$Q_{n+1} = \bar{Q}_n T + Q_n \bar{T}$$

Figure 10.24 Excitation requirements and K-map of a T flip-flop.

Race Around Condition

- For a J-K flip-flop, consider the assignment of excitations J = K = 1. If the width of the clock pulse is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0, 0 to 1 and so on, and at the end of the clock pulse, its state will be uncertain. This phenomenon is called the race around condition.

- The outputs Q and $\bar{Q}$ will change on their own if the clock pulse width is too long compared with the propagation delay of each NAND gate.

- This problem is eliminated using master-slave flip-flop

Master-Slave (Pulse-Triggered) Flip-Flops

- The master-slave flip-flop was developed to to avoid the problems of logic race in clocked flip-flops.

- In MS FF, a known time delay (equal to the width of one clock pulse) between the time that the flip-flop responds to a clock pulse and the time the response appears at its output.

- A master-slave flip-flop is also called a pulse-triggered flip-flop because the length of the time required for its output to change state equals the width of one clock pulse.

- The master-slave or pulse-triggered flip-flop actually contains two flip-flops— a master flip-flop and a slave flip-flop.

- The control inputs are applied to the master flip-flop .

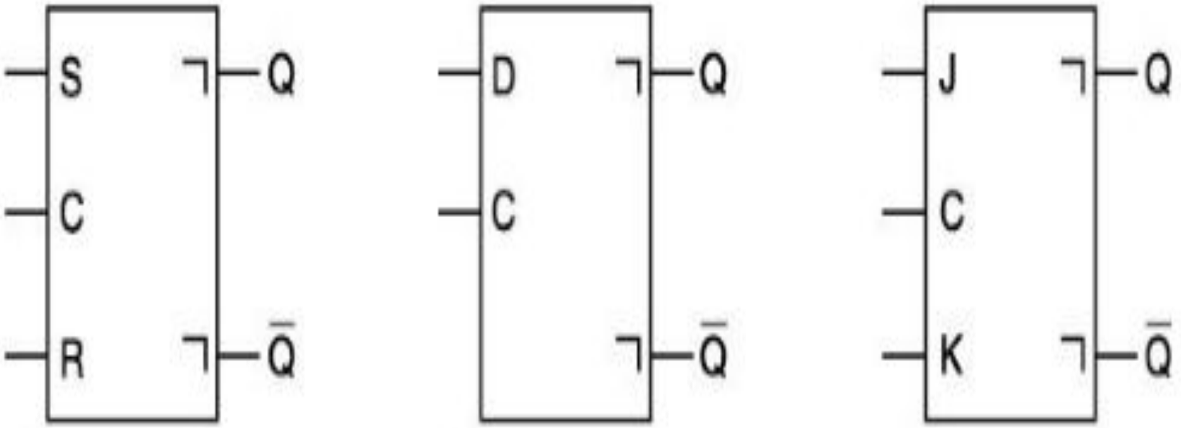
- On the rising edge of the clock pulse, the levels on the control inputs are used to determine the output of the master.

- On the falling edge of the clock pulse, the state of the master is transferred to the slave, whose outputs are Q and $\overline{Q}$

- Thus, the actual outputs of the flip-flop, i.e. Q and $\overline{Q}$ change just after the negative-going transition of the clock.

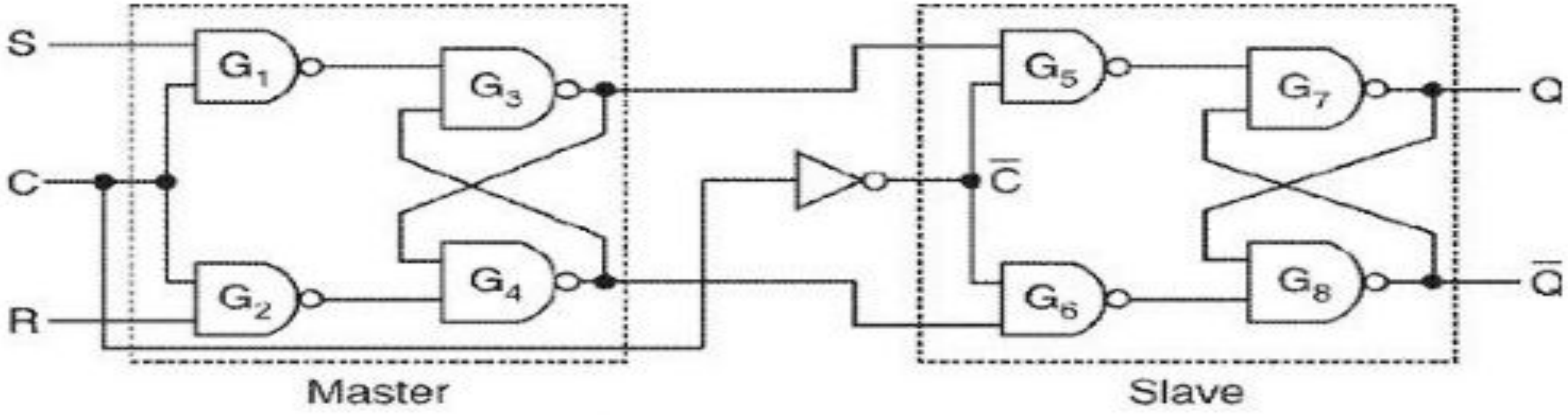
- There are three basic types of master-slave flip-flops— S-R, D, and J-K.

- The key to identifying a master-slave flip-flop by its logic symbol is the postponed output symbol ¬ at the outputs.

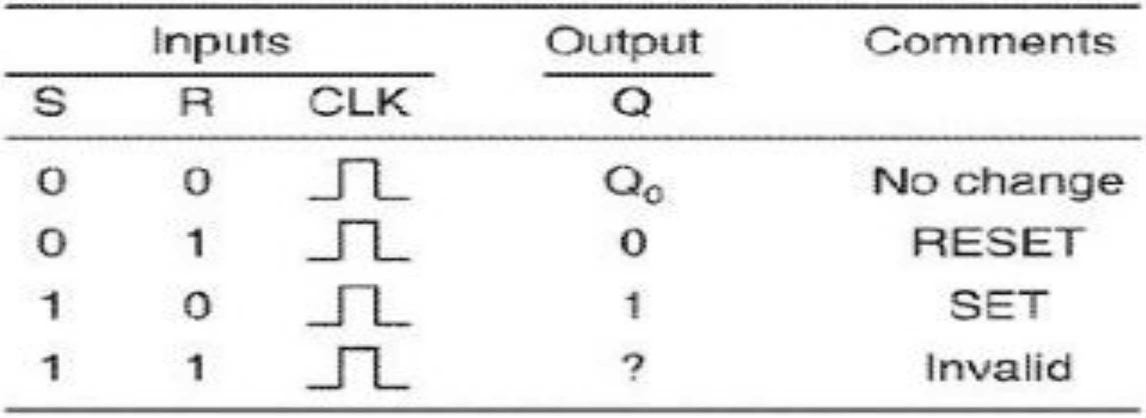


(a) S-R flip-flop (b) D flip-flop (c) J-K flip-flop

The Master-Slave (Pulse-Triggered) S-R Flip-Flop



(a) Logic diagram



Inputs			Output	Comments
S	R	CLK	Q	
0	0	⎍	Q_0	No change
0	1	⎍	0	RESET
1	0	⎍	1	SET
1	1	⎍	?	Invalid

(b) Truth table

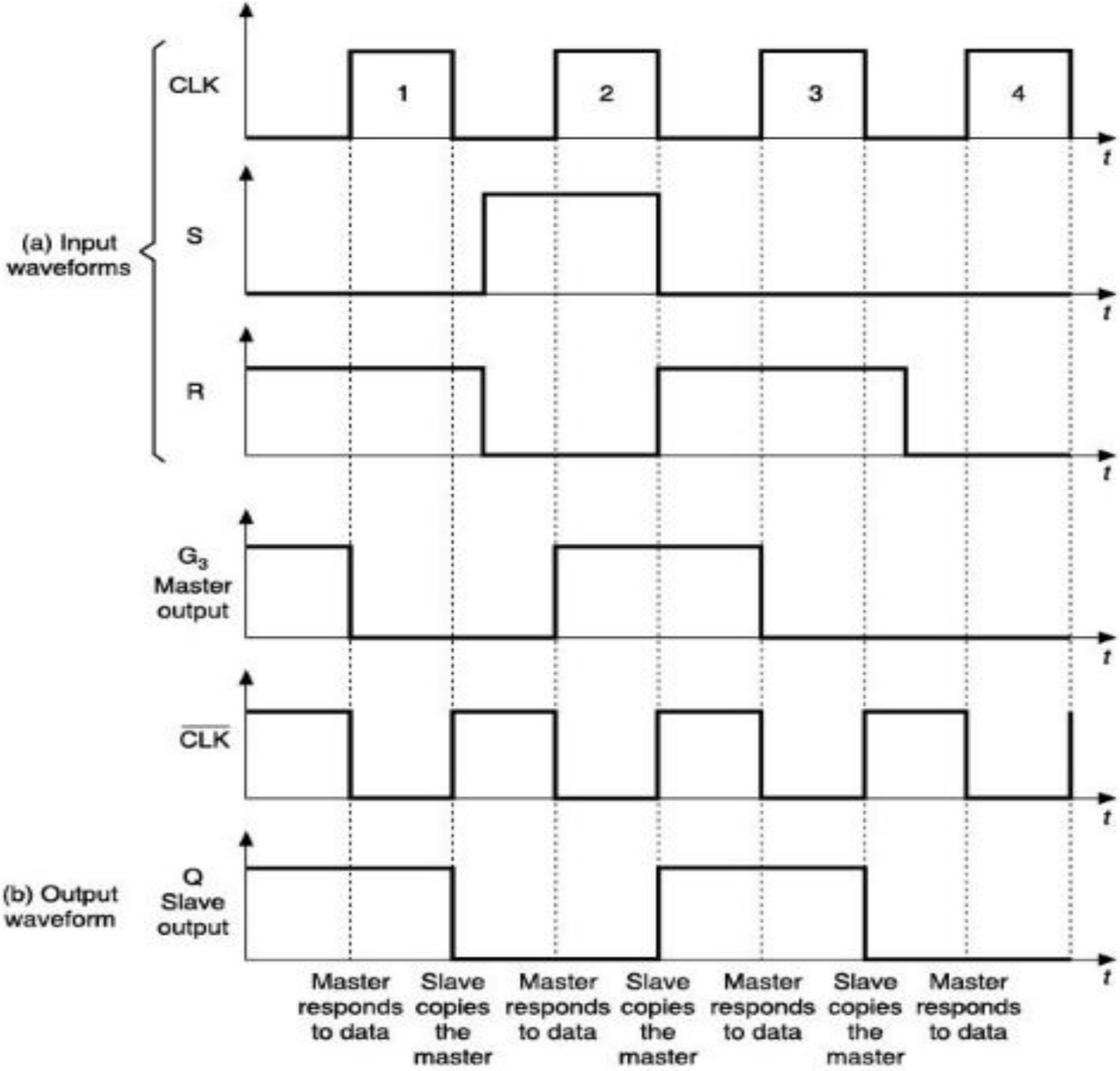
- The external control inputs S and R are applied to the master section.

- The master section is basically a gated S-R latch, and responds to the external S-R inputs applied to it at the positive-going edge of the clock signal.

- The slave section is the same as the master section except that it is clocked on the inverted clock pulse and thus responds to its control inputs (which are nothing but the outputs of the master flip-flop) at the negative-going edge of the clock pulse.

- The master section assumes the state determined by the S and R inputs at the positive-going edge of the clock pulse and the slave section copies the state of the master section at the negative-going edge of the clock pulse.

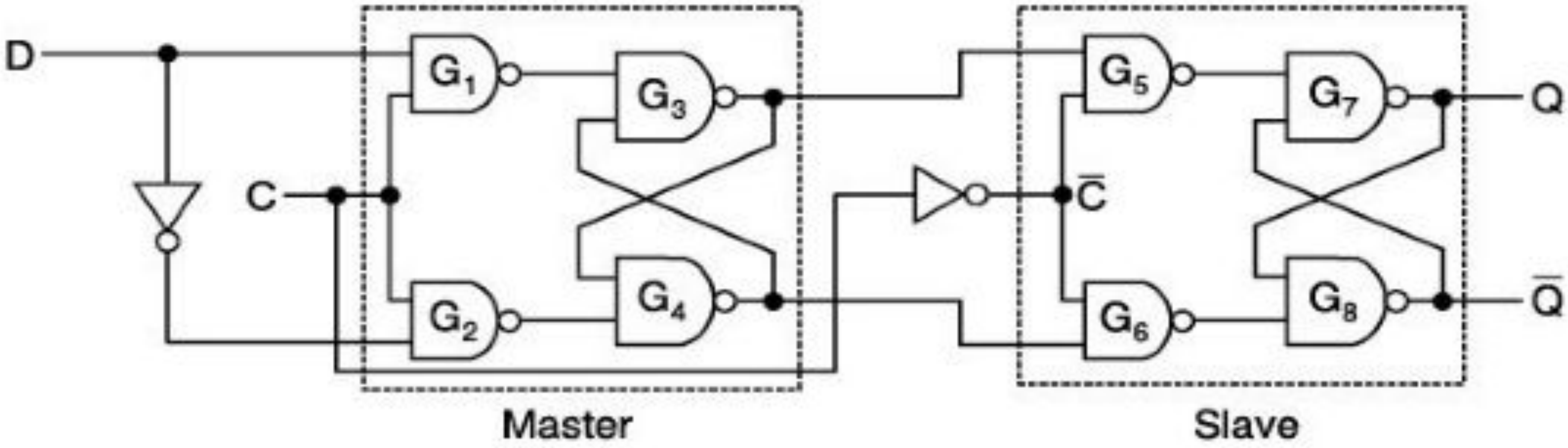
- The state of the slave then immediately appears on its Q and outputs.


(a) Input waveforms
CLK
1
2
3
4
t
S
t
R
t
G3 Master output
t
CLK
t
(b) Output waveform
Q Slave output
t
Master responds to data
Slave copies the master
Master responds to data
Slave copies the master
Master responds to data
Slave copies the master
Master responds to data

Let us assume that, initially the flip-flop is SET, i.e. Q = 1 and the control inputs are S = 0 and R = 1 and the output of master is a 1.
At the positive-going edge of the first clock pulse, the master resets, i.e. the output of G3 goes LOW. At the negative-going edge of the first clock pulse, the slave copies it. So, Q goes LOW. The inputs S and R now change when the clock is LOW; so it does not affect the operation. At the positive-going edge of the second clock pulse, S = 1 and R = 0 (and G3 = 0, Q = 0). So, the master sets, i.e. the output of G3 goes HIGH. At the negative-going edge of the second clock pulse, the slave copies this action of the master and, therefore, Q goes HIGH.

At the positive-going edge of the third clock pulse, S = 0 and R = 1, so, the master resets, i.e. the output of G3 goes LOW. At the negative-going edge of the third clock pulse, the slave copies this action of the master and, therefore, Q goes LOW.
At the positive-going edge of the fourth clock pulse, S = 0 and R = 0 (the output of G3 = 0, Q = 0). So, there is no change in the state of the master. Hence, there will not be any change in the state of the slave at the negative-going edge of that clock pulse and Q, therefore, remains LOW.

The Master-Slave (Pulse-Triggered) D Flip-Flop



(a) Logic diagram

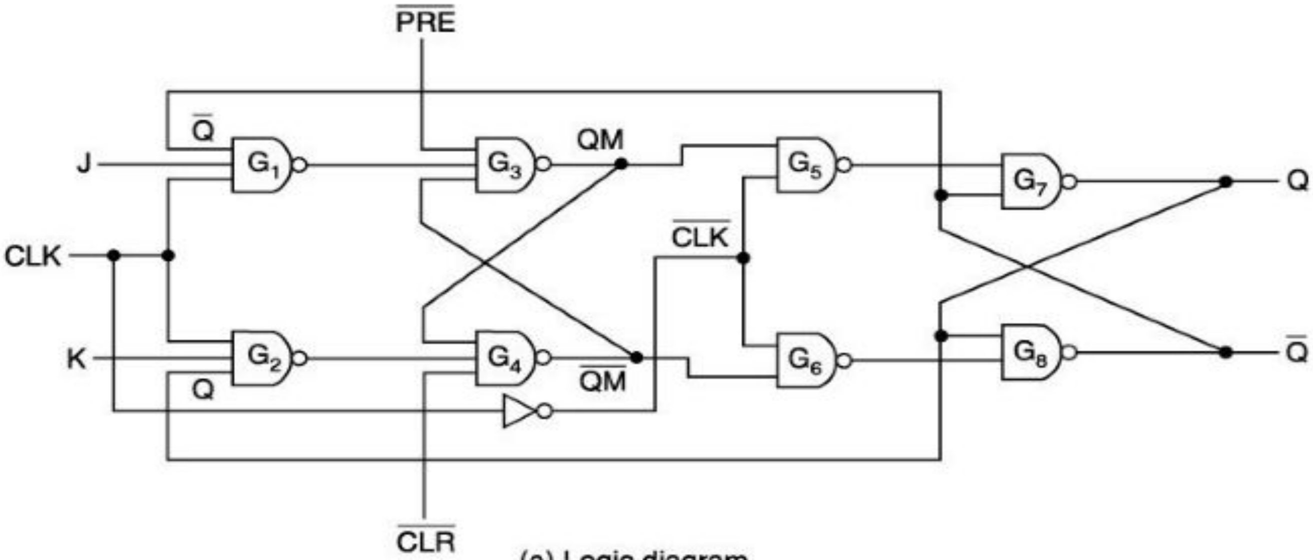
Inputs		Output	Comments
D	CLK	Q	
0	⎍	0	RESET
1	⎍	1	SET

(b) Truth table

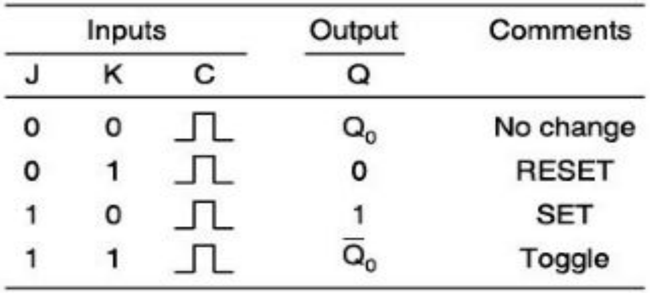
Figure 10.39 The master-slave D flip-flop.

- The D input is transferred to the master at the positive-going edge of the clock pulse and the same is copied by the slave and, therefore, appears at the Q output of the slave at the negative-going edge of the clock pulse.

The Master-Slave (Pulse-Triggered) J-K Flip-Flop



(a) Logic diagram



Inputs			Output	Comments
J	K	C	Q	
0	0	⎍	Q_0	No change
0	1	⎍	0	RESET
1	0	⎍	1	SET
1	1	⎍	$\overline{Q}_0$	Toggle

(b) Truth table

- The truth table operation is the same as that of a negative edge-triggered J-K flip-flop except for the way in which it is triggered.

- The logic diagram of the master-slave J-K flip-flop is similar to that of the master-slave S-R flip-flop.

- The difference is that the Q output is connected back to the input of G2 and the $\overline{Q}$ tput is connected back to the input of G1 and the external inputs are designated as J and K.

- The M-S J-K flip-flop is a J-K flip-flop followed by an S-R-flip-flop, i.e. the master flip-flop is a J-K flip-flop and slave flip-flop is an S-R flip-flop.

- The problem of logic race is eliminated because while the clock drives the master, the inverted clock drives the slave.

Flip-Flop Excitation Tables

- The excitation table of a flip-flop can be obtained from its truth table.

- It indicates the inputs required to be applied to the flip-flop to take it from the present state to the next state.

The truth table and excitation table of an S-R flip flop

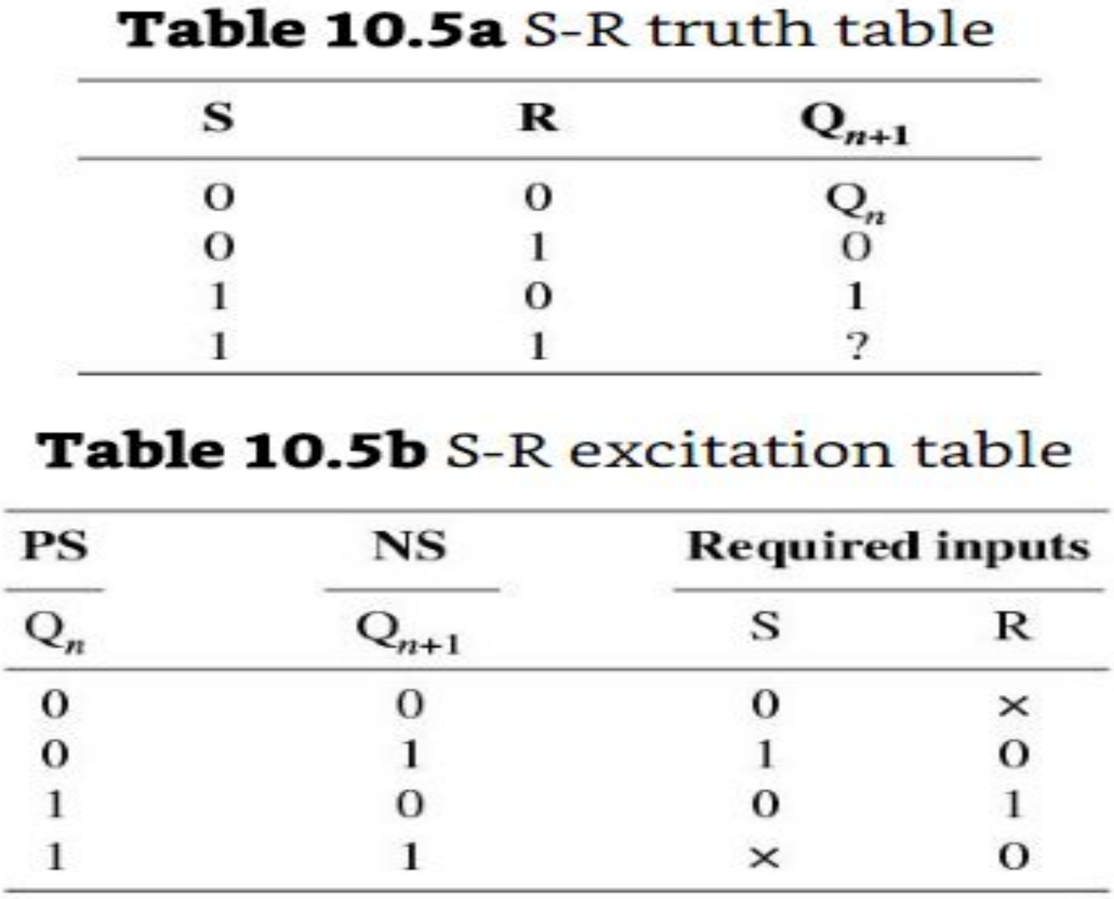


Table 10.5a S-R truth table

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

Table 10.5b S-R excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

The truth table and excitation table of a J-K flip-flop

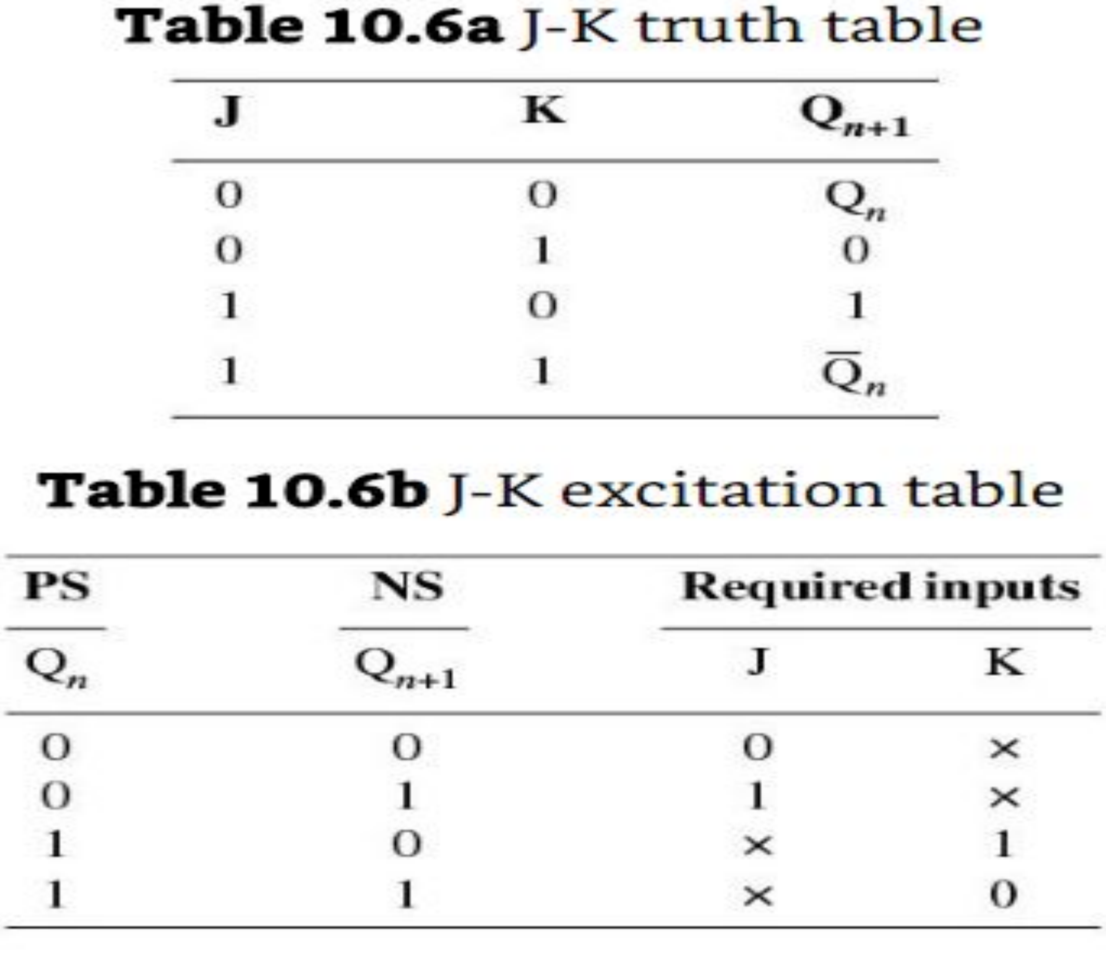


Table 10.6a J-K truth table

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

Table 10.6b J-K excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

The truth table and the excitation table of the D flip-flop

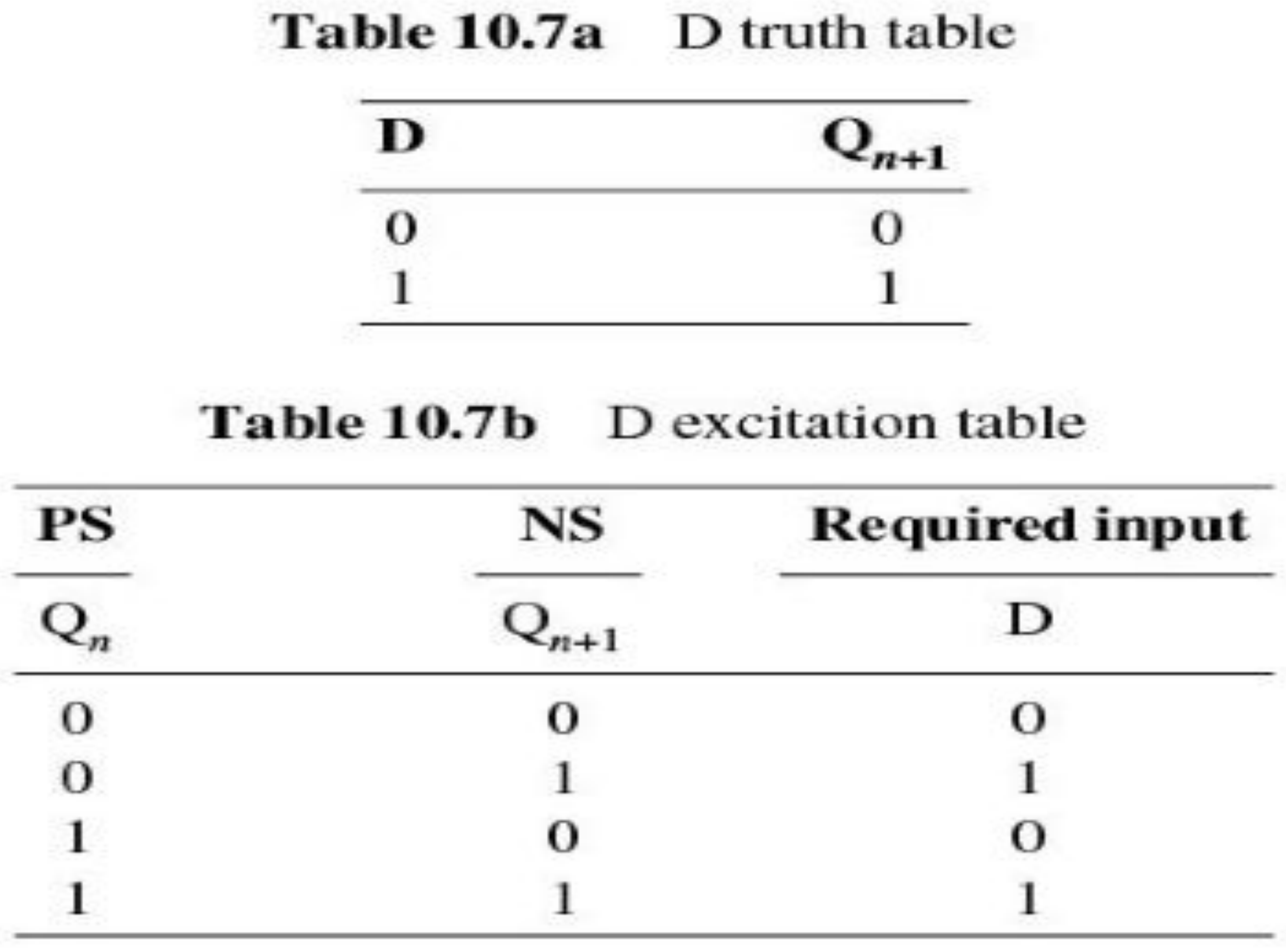


Table 10.7a D truth table

D	Q_{n+1}
0	0
1	1

Table 10.7b D excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

The truth table and the excitation table of the T flip-flop

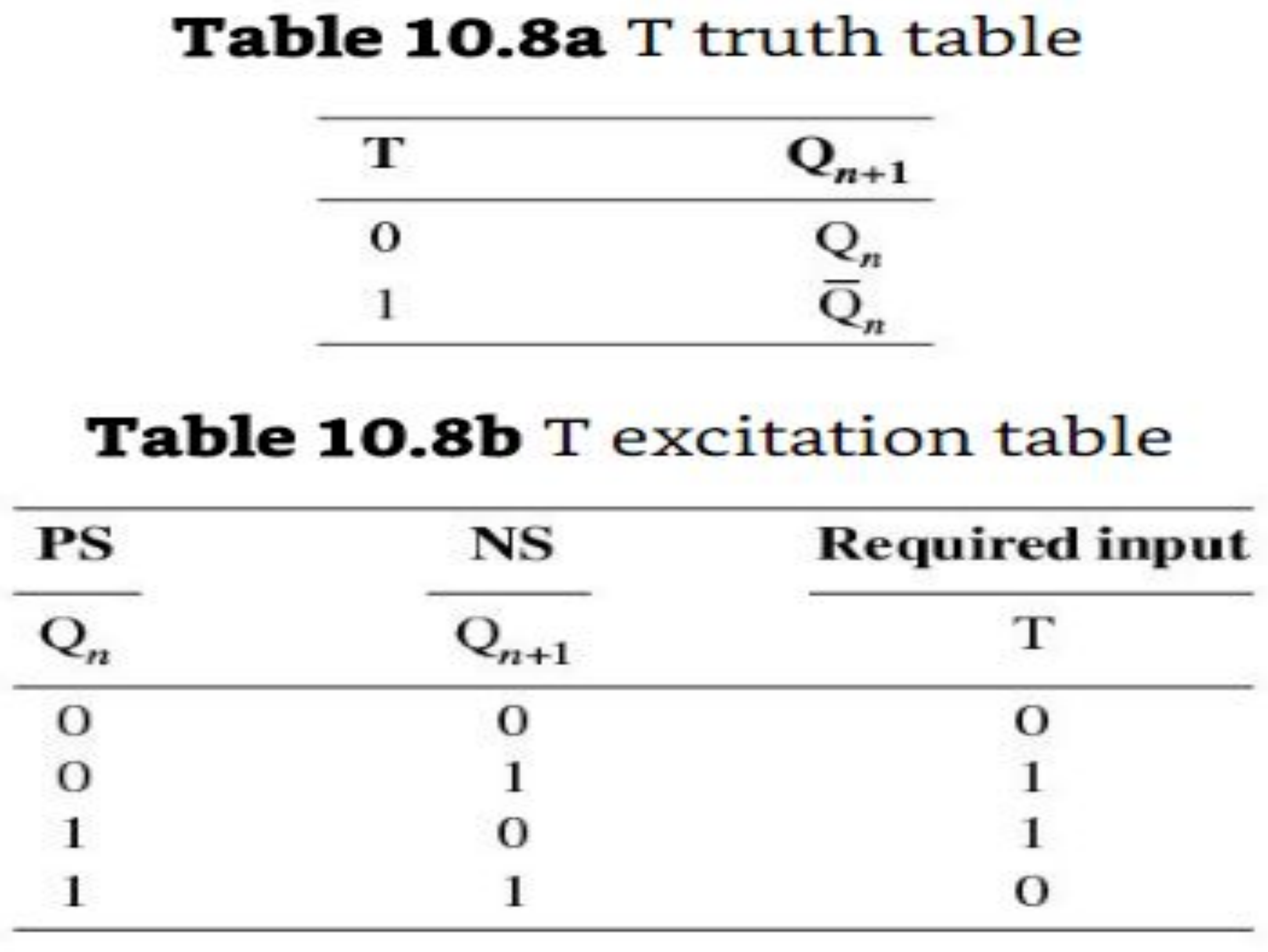


Table 10.8a T truth table

T	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

Table 10.8b T excitation table

PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Conversion of Flip-Flops

- To convert one type of flip-flop into another type, we have to obtain the expressions for the inputs of the existing flip-flop in terms of the inputs of the required flip-flop and the present state variables of the existing flip-flop and implement them.

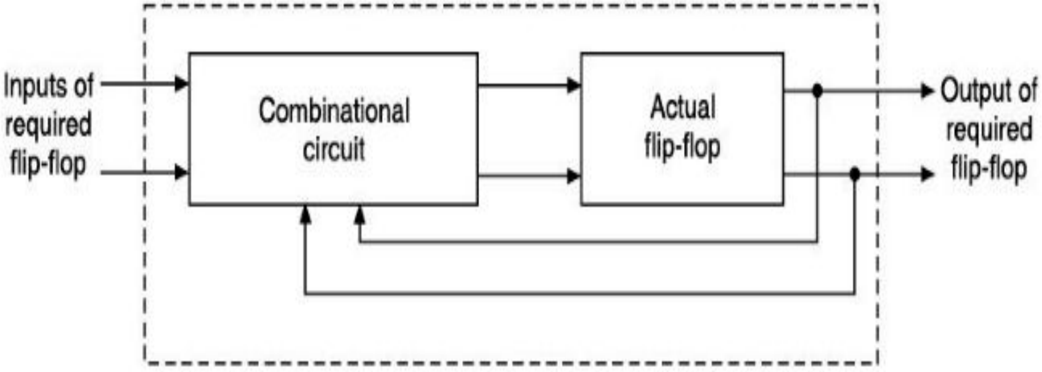


Figure 10.43 Block diagram for conversion of flip-flop.

S-R flip-flop to J-K flip-flop

- Here the external inputs to the already available S-R flip-flop will be J and K.

- S and R are the outputs of the combinational circuit, which are also the actual inputs to the S-R flip-flop.

- We write a truth table with J, K, Qn, Qn + 1, S, and R, where Qn is the present state of the flip-flop and Qn + 1 is the next state obtained when the particular J and K inputs are applied,

- i.e. Qn denotes the state of the flip-flop before the application of the inputs and Qn + 1 refers to the state obtained by the flip-flop after the application of inputs.

- J, K and Qn can have eight combinations.

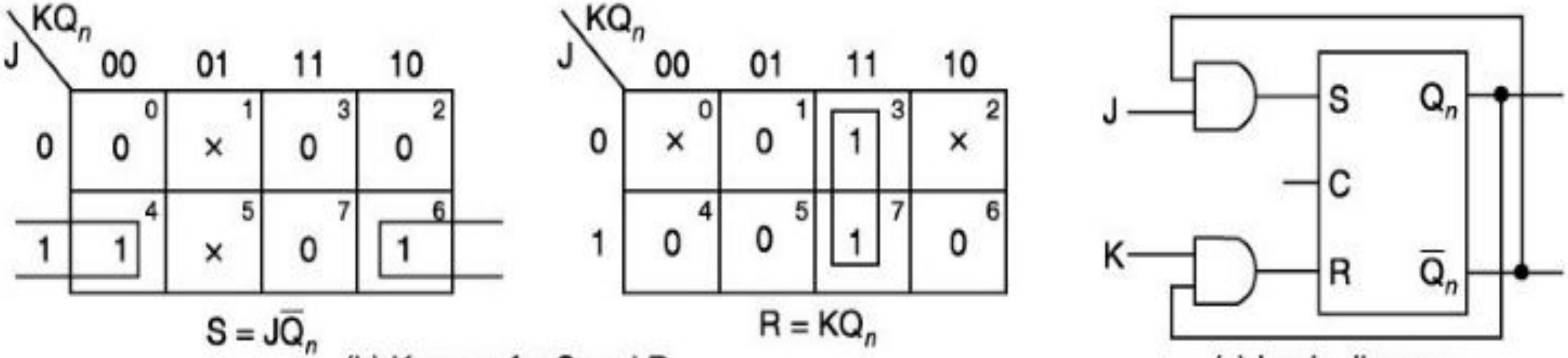
- For each combination of J, K and Qn, find the corresponding Qn + 1, i.e. determine to which next state (Qn + 1) the J-K flip-flop will go from the present state Qn if the present inputs J and K are applied.

- Complete the table by writing the values of S and R required to get each Qn + 1 from the corresponding Qn,

- i.e. write what values of S and R are required to change the state of the flip-flop from Qn to Qn + 1.

External Inputs		Present State	Next State	Flip-flop Inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	×	0
1	1	0	1	1	0
1	1	1	0	0	1

(a) Conversion table



(b) K-maps for S and R

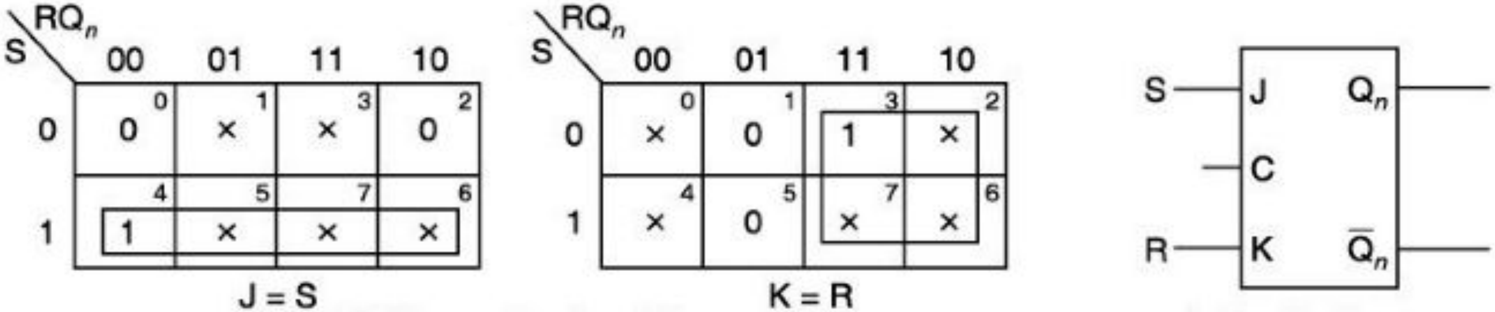
(c) Logic diagram

Figure 10.44 Conversion of S-R flip-flop to J-K flip-flop.

J-K flip-flop to S-R flip-flop

External Inputs		Present State	Next State	Flip-flop Inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	×
0	0	1	1	×	0
0	1	0	0	0	×
0	1	1	0	×	1
1	0	0	1	1	×
1	0	1	1	×	0

(a) Conversion table



S \ RQ_n	00	01	11	10
0	0	×	×	0
1	1	×	×	×

J = S

S \ RQ_n	00	01	11	10
0	×	0	1	×
1	×	0	×	×

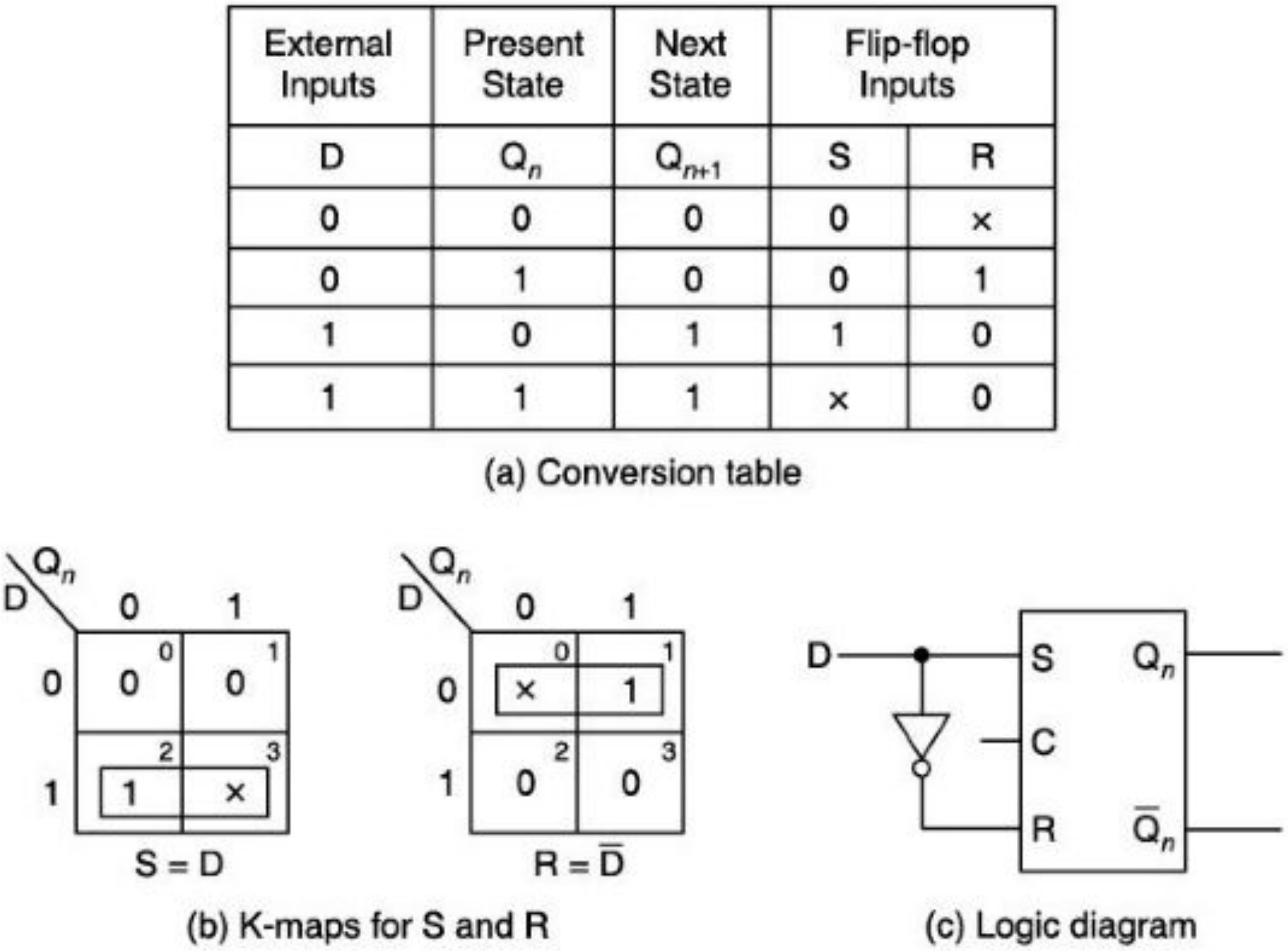
K = R

(b) K-maps for J and K

(c) Logic diagram

Figure 10.45 Conversion of J-K flip-flop to S-R flip-flop.

S-R flip-flop to D flip-flop:



External Inputs	Present State	Next State	Flip-flop Inputs	
D	Q_n	Q_{n+1}	S	R
0	0	0	0	×
0	1	0	0	1
1	0	1	1	0
1	1	1	×	0

(a) Conversion table

D \ Q_n	0	1
0	0	0
1	1	×

S = D

D \ Q_n	0	1
0	×	1
1	0	0

$R = \overline{D}$

(b) K-maps for S and R

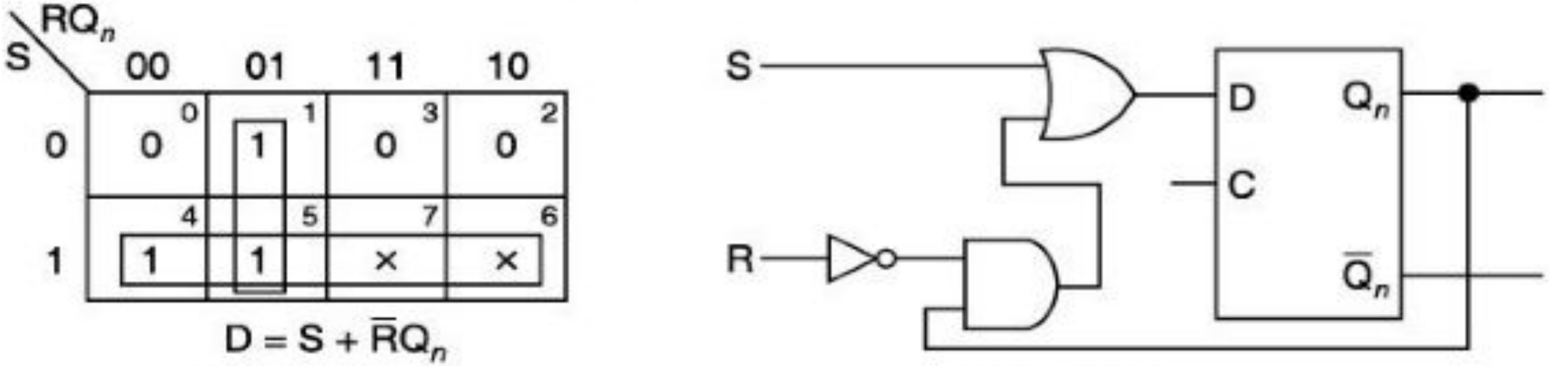
(c) Logic diagram

Figure 10.46 Conversion of S-R flip-flop to D flip-flop.

D flip-flop to S-R flip-flop

External Inputs		Present State	Next State	Flip-flop Input
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1

(a) Conversion table



S \ RQ_n	00	01	11	10
0	0 (0)	1 (1)	0 (3)	0 (2)
1	1 (4)	1 (5)	× (7)	× (6)

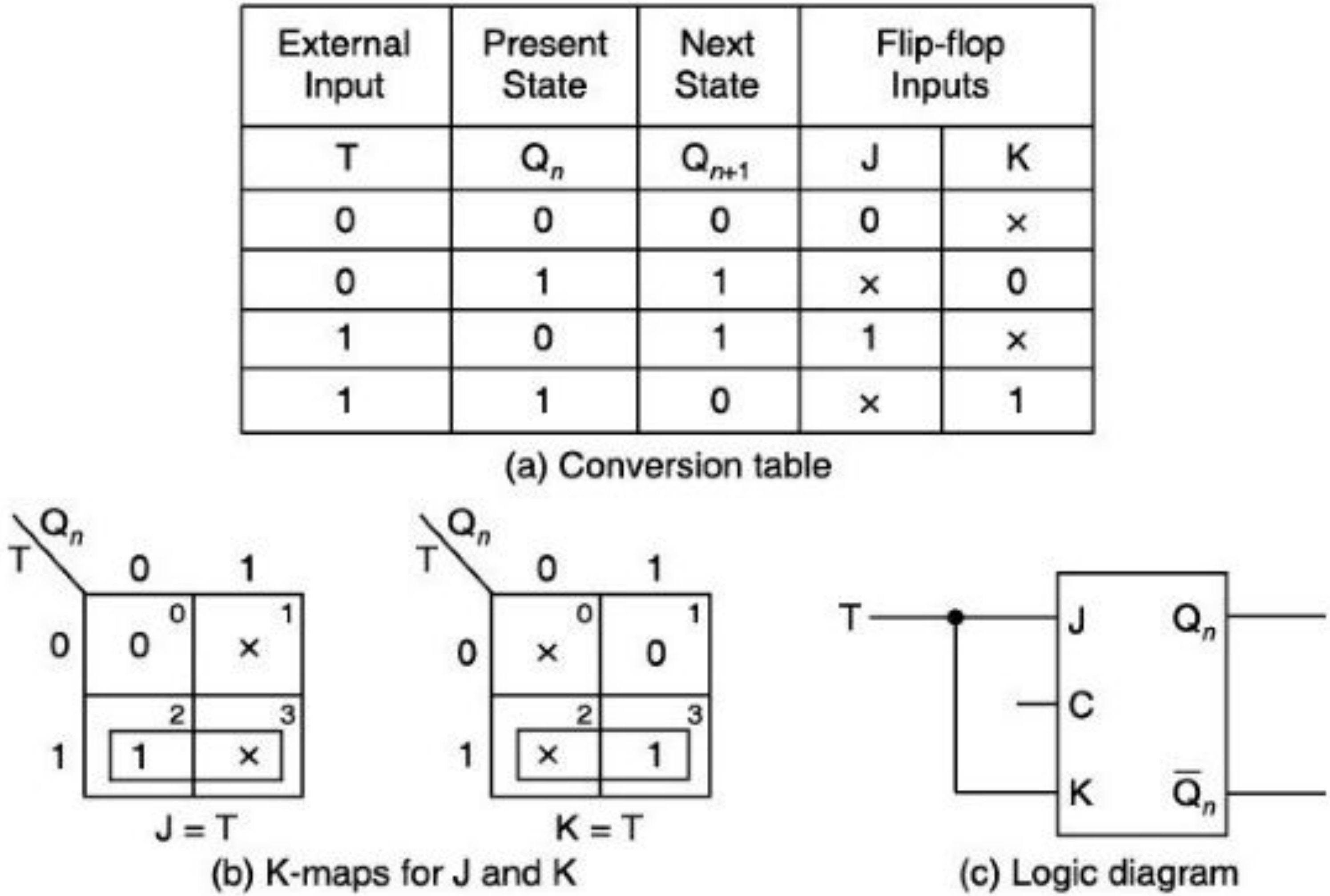
$D = S + \overline{R}Q_n$

(b) K-map for D

(c) Logic diagram

Figure 10.47 Conversion of D flip-flop to S-R flip-flop.

J-K flip-flop to T flip-flop



External Input	Present State	Next State	Flip-flop Inputs	
T	Q_n	Q_{n+1}	J	K
0	0	0	0	×
0	1	1	×	0
1	0	1	1	×
1	1	0	×	1

(a) Conversion table

T \ Q_n	0	1
0	0	×
1	1	×

J = T

T \ Q_n	0	1
0	×	0
1	×	1

K = T

(b) K-maps for J and K

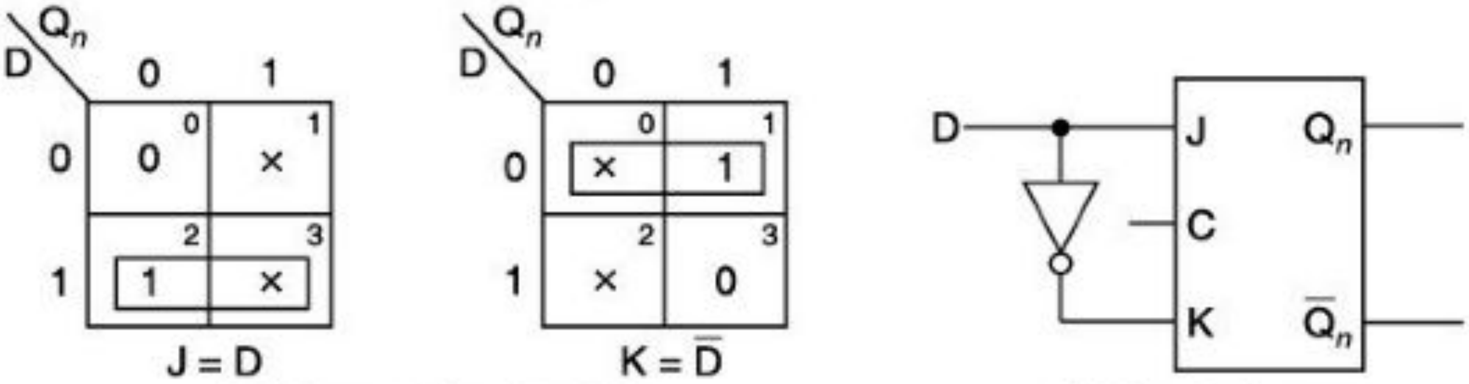
(c) Logic diagram

Figure 10.48 Conversion of J-K flip-flop to T flip-flop.

J-K flip-flop to D flip-flop

External Input	Present State	Next State	Flip-flop Inputs	
D	Q_n	Q_{n+1}	J	K
0	0	0	0	×
0	1	0	×	1
1	0	1	1	×
1	1	1	×	0

(a) Conversion table

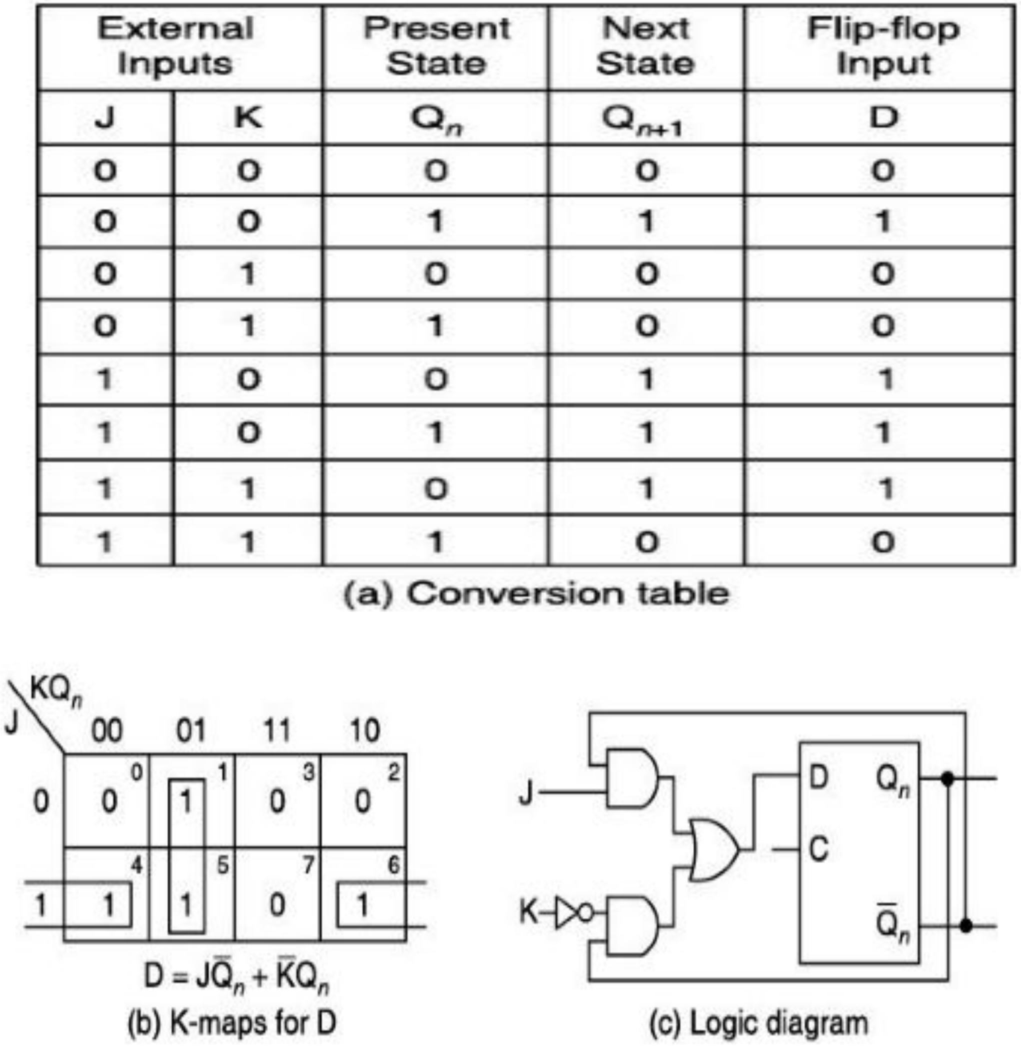


(b) K-maps for J and K

(c) Logic diagram

Figure 10.49 Conversion of J-K flip-flop to D flip-flop.

D flip-flop to J-K flip-flop



External Inputs		Present State	Next State	Flip-flop Input
J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

(a) Conversion table

J \ KQ_n	00	01	11	10
0	0 [0]	1 [1]	0 [3]	0 [2]
1	1 [4]	1 [5]	0 [7]	1 [6]

$D = J\overline{Q}_n + \overline{K}Q_n$

(b) K-maps for D

(c) Logic diagram

Figure 10.50 Conversion of D flip-flop to J-K flip-flop.

Applications of Flip-Flops

- Some of the basic applications are parallel data storage, serial data storage, transfer of data, frequency division, counting, parallel to serial conversion, serial to parallel conversion, synchronizing the effect of asynchronous data, detection of an input sequence, etc.

Types of Data Transfer

- 1. Parallel Data Transfer
- 2. Serial Data Transfer

- Parallel

- Parallel data transfer is the simultaneous transmission of all bits of data from one device to another.

- Serial

- Serial data transfer is the transmission of one bit of data at a time from one device to another. Serial data must be transmitted under the synchronization of a clock, since the clock provides the means to specify the time at which each new bit is sampled.

Shift Register

- A register is a set of FFs used to store binary data.

- The storage capacity of a register is the number of bits (1s and 0s) of digital data it can retain.

- Loading a register means setting or resetting the individual FFs, i.e. inputting data into the register so that their states correspond to the bits of data to be stored.

- Loading may be serial or parallel.

- In serial loading, data is transferred into the register in serial form, i.e. one bit at a time, whereas in parallel loading, the data is transferred into the register in parallel form meaning that all the FFs are triggered into their new states at the same time.

- A register may output data either in serial form or in parallel form.

- Serial output means that the data is transferred out of the register, one bit at a time serially.

- Parallel output means that the entire data stored in the register is available in parallel form, and can be transferred out at the same time.

Buffer Register

- Some registers do nothing more than storing a binary word.

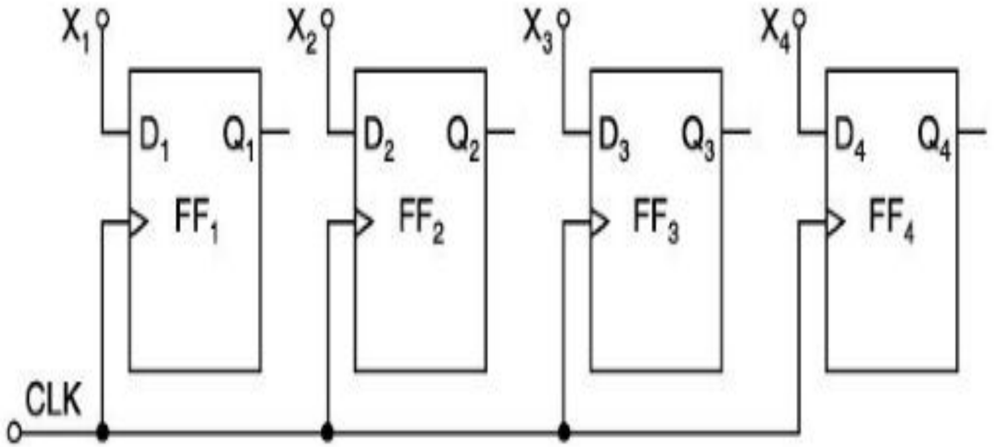
- The buffer register is the simplest of registers.

- It simply stores the binary word.

- The buffer may be a controlled buffer.

- Most of the buffer registers use D flip-flops.

4-bit buffer register



- The binary word to be stored is applied to the data terminals.

- On the application of clock pulse, the output word becomes the same as the word applied at the input terminals,

- i.e. the input word is loaded into the register by the application of clock pulse.

- When the positive clock edge arrives, the stored word becomes:

$$Q_4Q_3Q_2Q_1 = X_4X_3X_2X_1$$

or

$$Q = X$$

Data Transmission in Shift Registers

- A number of FFs connected together such that data may be shifted into and shifted out of them is called a **shift register**.

- Data may be shifted into or out of the register either in serial form or in parallel form.

- There are four basic types of shift registers:

- 1. serial-in, serial-out
- 2. serial-in, parallel-out
- 3. parallel-in, serial-out
- 4. parallel-in, parallel-out

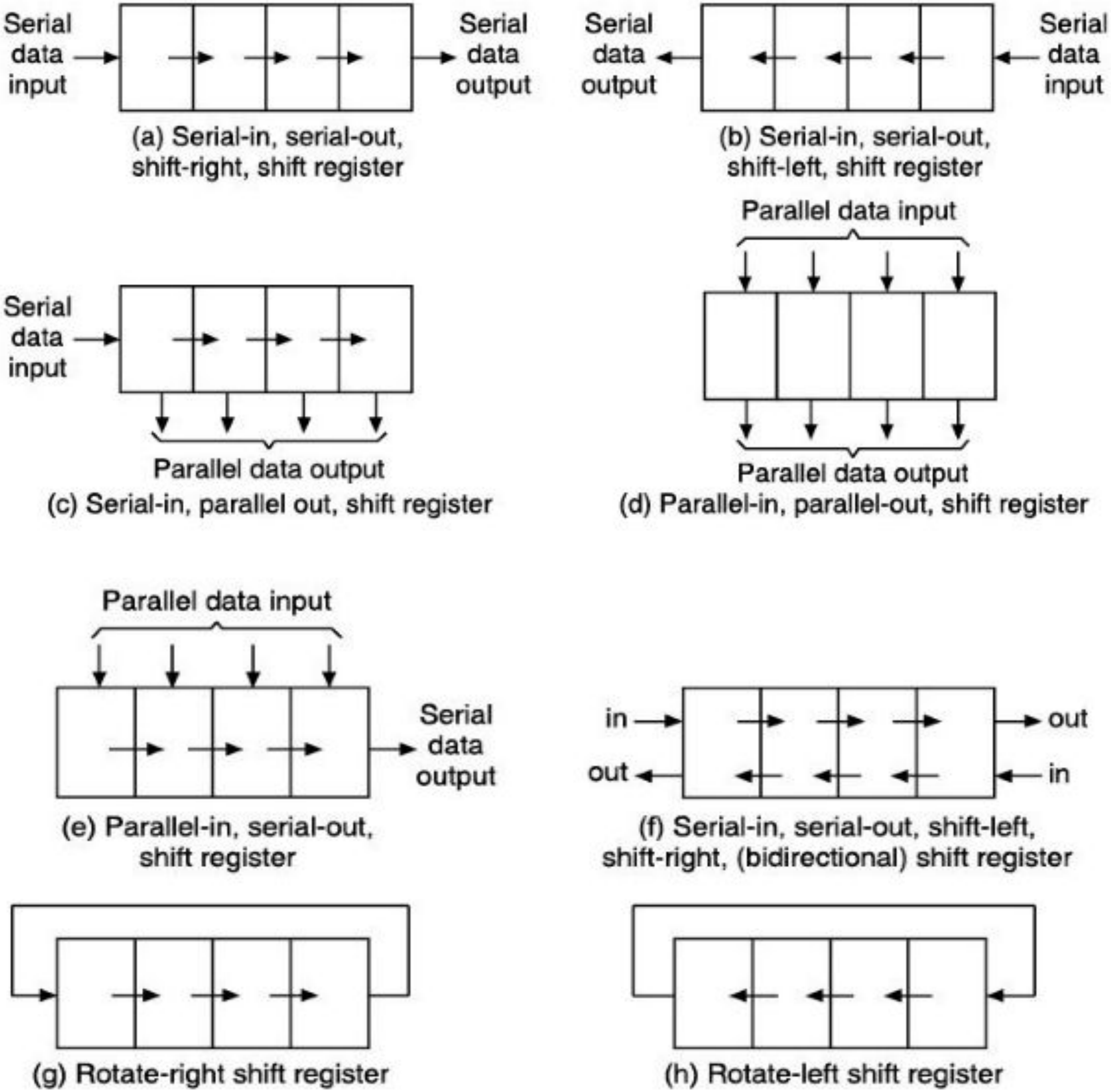

Serial data input
Serial data output
(a) Serial-in, serial-out, shift-right, shift register
Serial data output
Serial data input
(b) Serial-in, serial-out, shift-left, shift register
Serial data input
Parallel data output
(c) Serial-in, parallel out, shift register
Parallel data input
Parallel data output
(d) Parallel-in, parallel-out, shift register
Parallel data input
Serial data output
(e) Parallel-in, serial-out, shift register
in
out
out
in
(f) Serial-in, serial-out, shift-left, shift-right, (bidirectional) shift register
(g) Rotate-right shift register
(h) Rotate-left shift register

Figure 11.3 Data transfer in registers.

Serial-in, Serial-out, Shift Register

- This type of shift register accepts data serially, i.e. one bit at a time, and also outputs data serially.

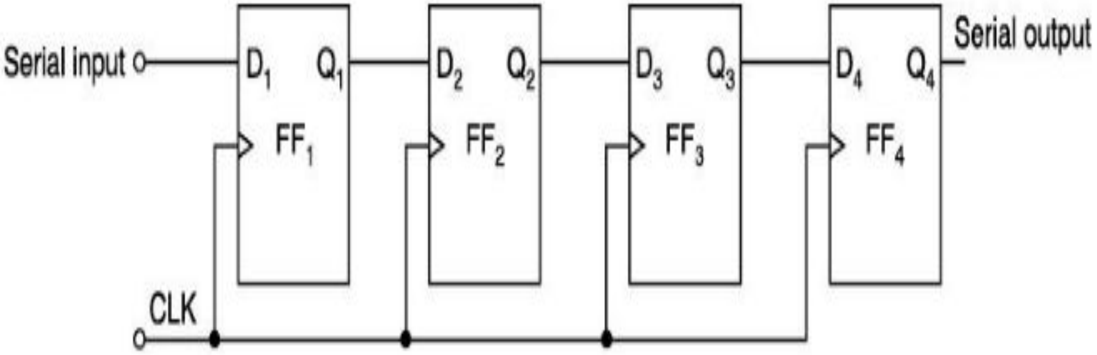


Figure 11.4 4-bit serial-in, serial-out, shift-right, shift register.

- With four stages, i.e. four FFs, the register can store upto four bits of data.

- Serial data is applied at the D input of the first FF.

- The Q output of the first FF is connected to the D input of the second FF, the Q output of the second FF is connected to the D input of the third FF and the Q output of the third FF is connected to the D input of the fourth FF.

- The data is outputted from the Q terminal of the last FF.

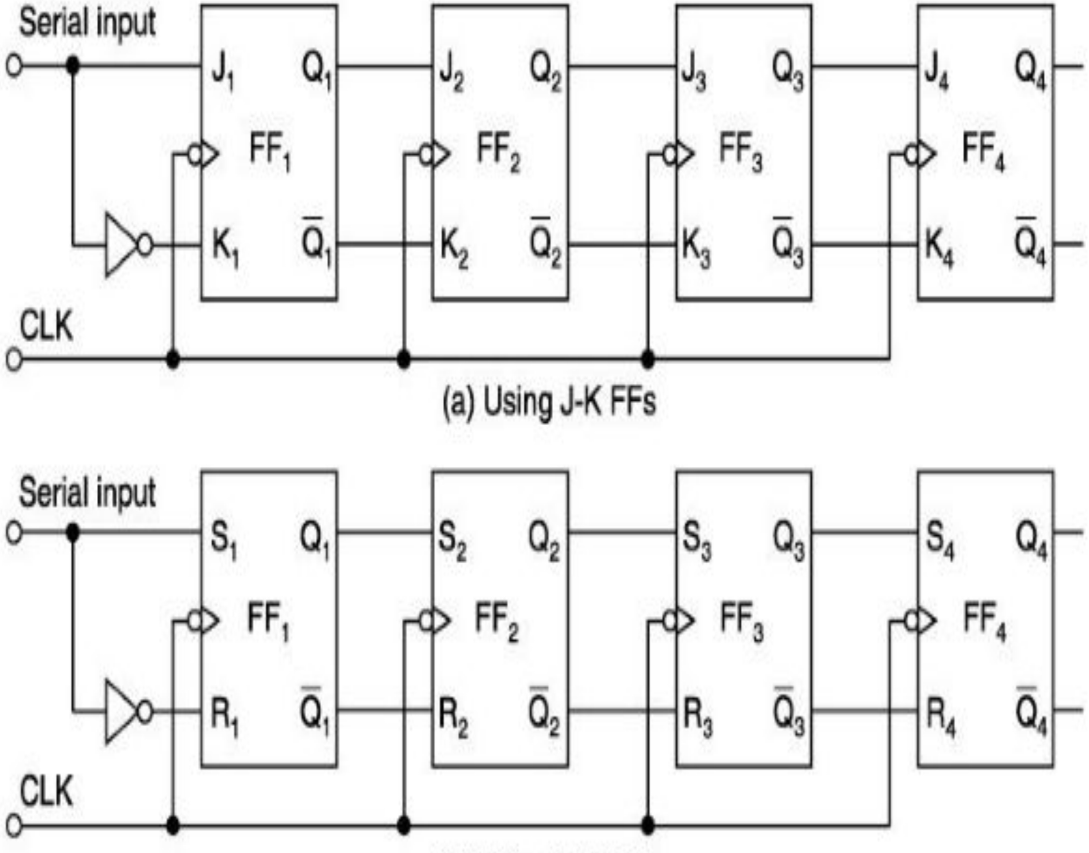
- When serial data is transferred into a register, each new bit is clocked into the first FF at the positive-going edge of each clock pulse.

- The bit that was previously stored by the first FF is transferred to the second FF.

- The bit that was stored by the second FF is transferred to the third FF, and so on.

- The bit that was stored by the last FF is shifted out.

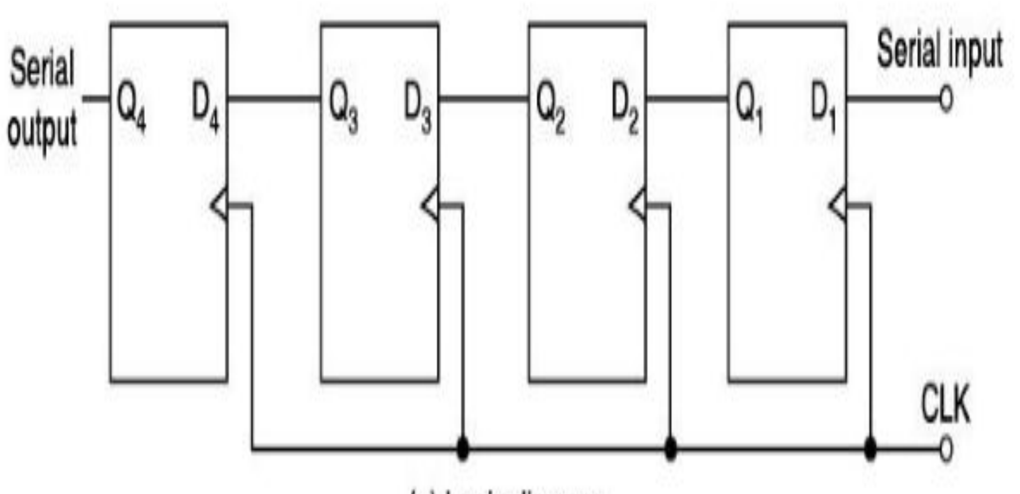
4 Bit Serial-in, Serial-out, Shift Register using JK/SR FF



(a) Using J-K FFs

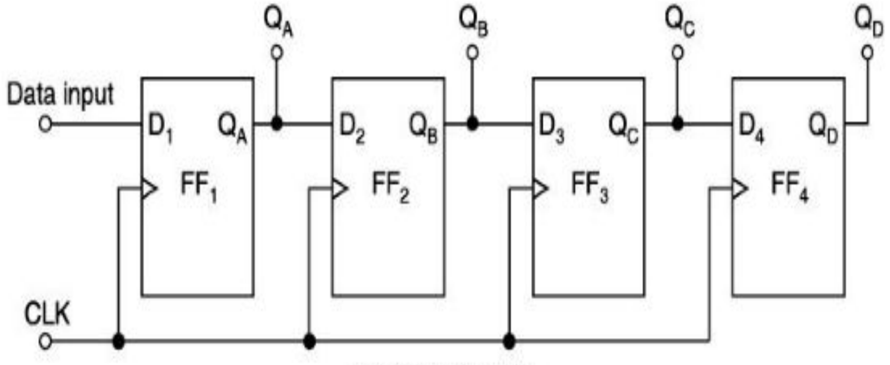
(b) Using S-R FFs

4-bit serial-in, serial-out, shift-left, shift register.

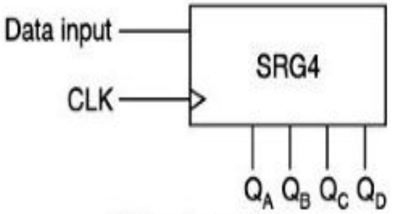


(a) Logic diagram

Serial-in, Parallel-out, Shift Register



(a) Logic diagram



(b) Logic symbol

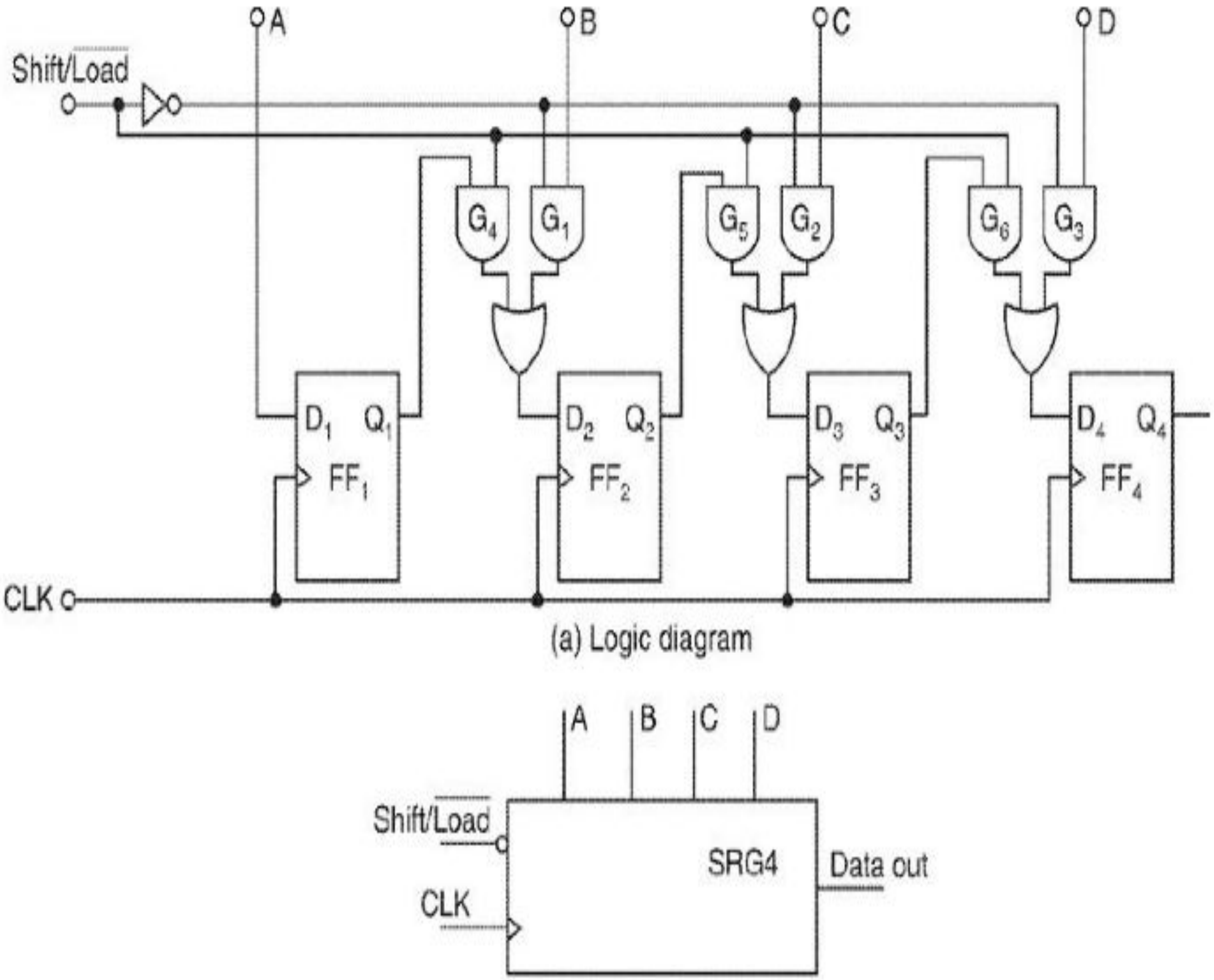
- In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form.

- Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously.

- The serial-in, parallel-out, shift register can be used as a serial-in, serial-out, shift register if the output is taken from the Q terminal of the last FF.

Parallel-in, Serial-out, Shift Register

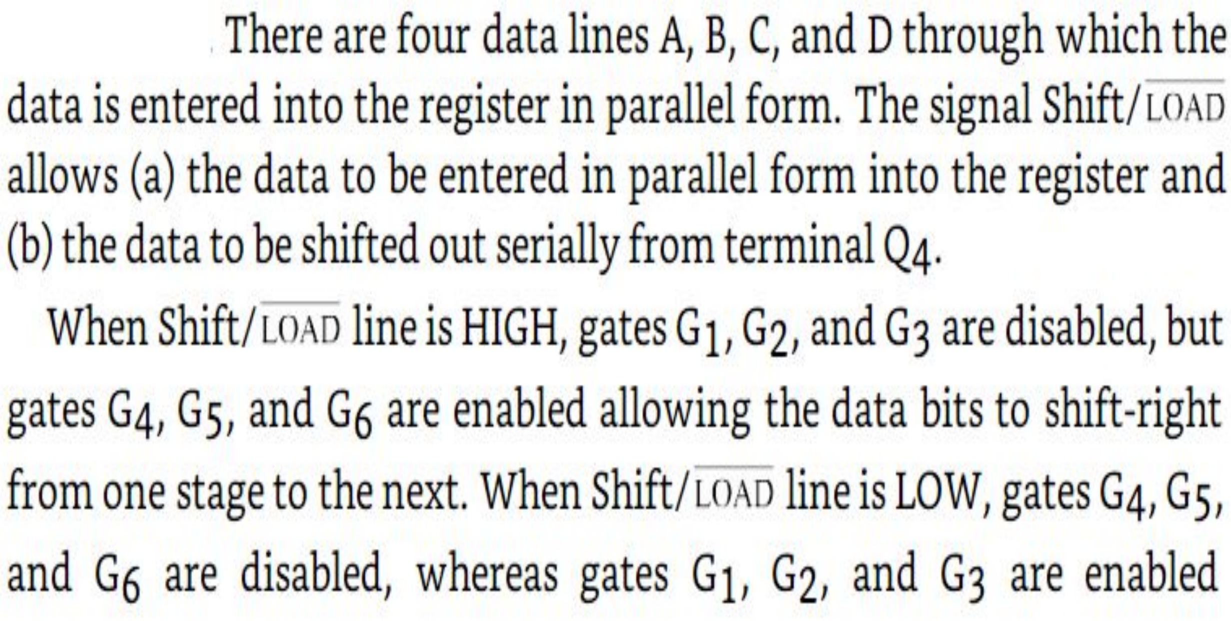
- For a parallel-in, serial-out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, but the data bits are transferred out of the register serially, i.e. on a bit-by-bit basis over a single line.



(a) Logic diagram

(b) Logic symbol

Figure 11.8 A 4-bit parallel-in, serial-out, shift register.



There are four data lines A, B, C, and D through which the data is entered into the register in parallel form. The signal Shift/$\overline{\text{LOAD}}$ allows (a) the data to be entered in parallel form into the register and (b) the data to be shifted out serially from terminal Q_4.

When Shift/$\overline{\text{LOAD}}$ line is HIGH, gates G_1, G_2, and G_3 are disabled, but gates G_4, G_5, and G_6 are enabled allowing the data bits to shift-right from one stage to the next. When Shift/$\overline{\text{LOAD}}$ line is LOW, gates G_4, G_5, and G_6 are disabled, whereas gates G_1, G_2, and G_3 are enabled

- There are four data lines A, B, C, and D through which the data is entered into the register in parallel form.

- The signal Shift/Load allows (a) the data to be entered in parallel form into the register and (b) the data to be shifted out serially from terminal Q4.

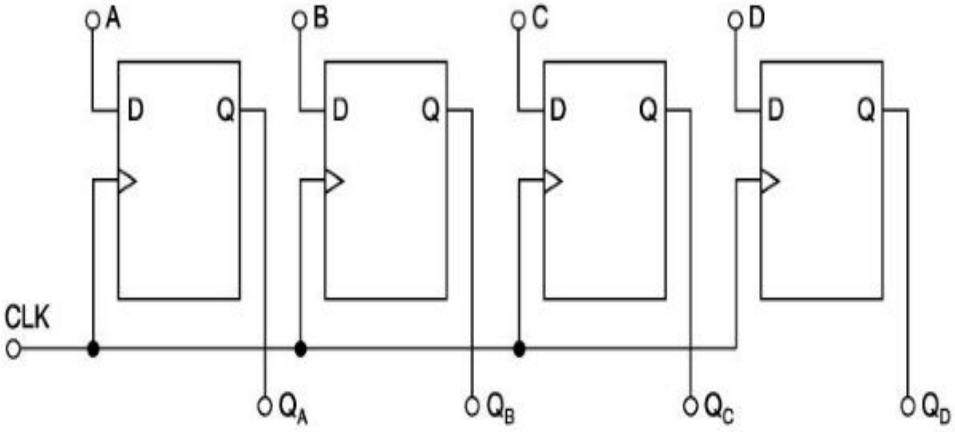
- When Shift/Load line is high, gates G1, G2, and G3 are disabled, but gates G4, G5, and G6 are enabled allowing the data bits to shift-right from one stage to the next.

- When Shift/Load line is low, gates G4, G5, and G6 are disabled, whereas gates G1, G2, and G3 are enabled allowing the data input to appear at the D inputs of the respective FFs.

- When a clock pulse is applied, these data bits are shifted to the Q output terminals of the FFs and, therefore, data is inputted in one step.

- The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the Shift/Load input.

Parallel-in, Parallel-out, Shift Register



- In a parallel-in, parallel-out, shift register, the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form.

- Data is applied to the D input terminals of the FFs.

- When a clock pulse is applied, at the positive-going edge of that pulse, the D inputs are shifted into the Q outputs of the FFs.

- The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

Bidirectional Shift Register

- A bidirectional shift register is one in which the data bits can be shifted from left to right or from right to left.

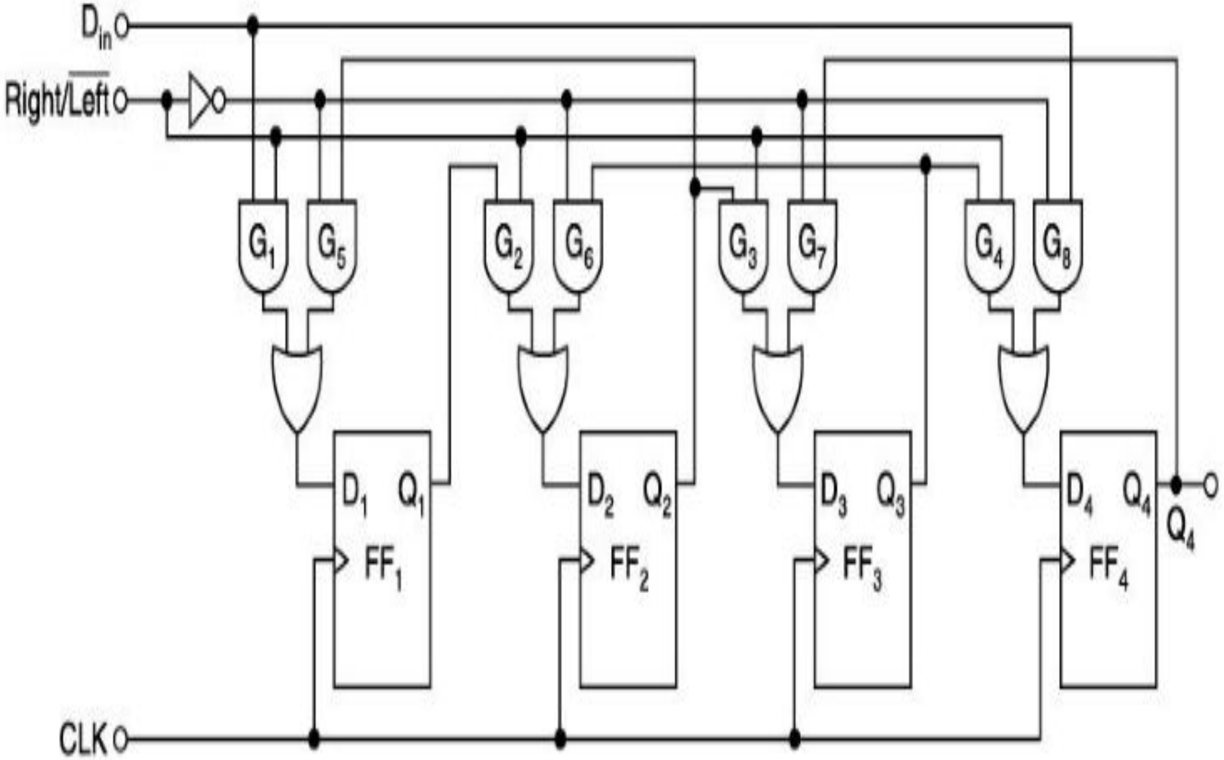


Figure 11.10 Logic diagram of a 4-bit bidirectional shift register.

- Figure shows the logic diagram of a 4-bit serial-in, serial-out, bidirectional (shift-left, shift-right) shift register.

- Right/Left is the mode signal.

- When Right/Left is a 1, the logic circuit works as a shift-right shift register.

- When Right/Left is a 0, it works as a shift-left shift register.

- A high on the Right/Left control input enables the AND gates G1, G2, G3, and G4 and disables the AND gates G5, G6, G7, and G8, and the state of Q output of each FF is passed through the gate to the D input of the following FF.

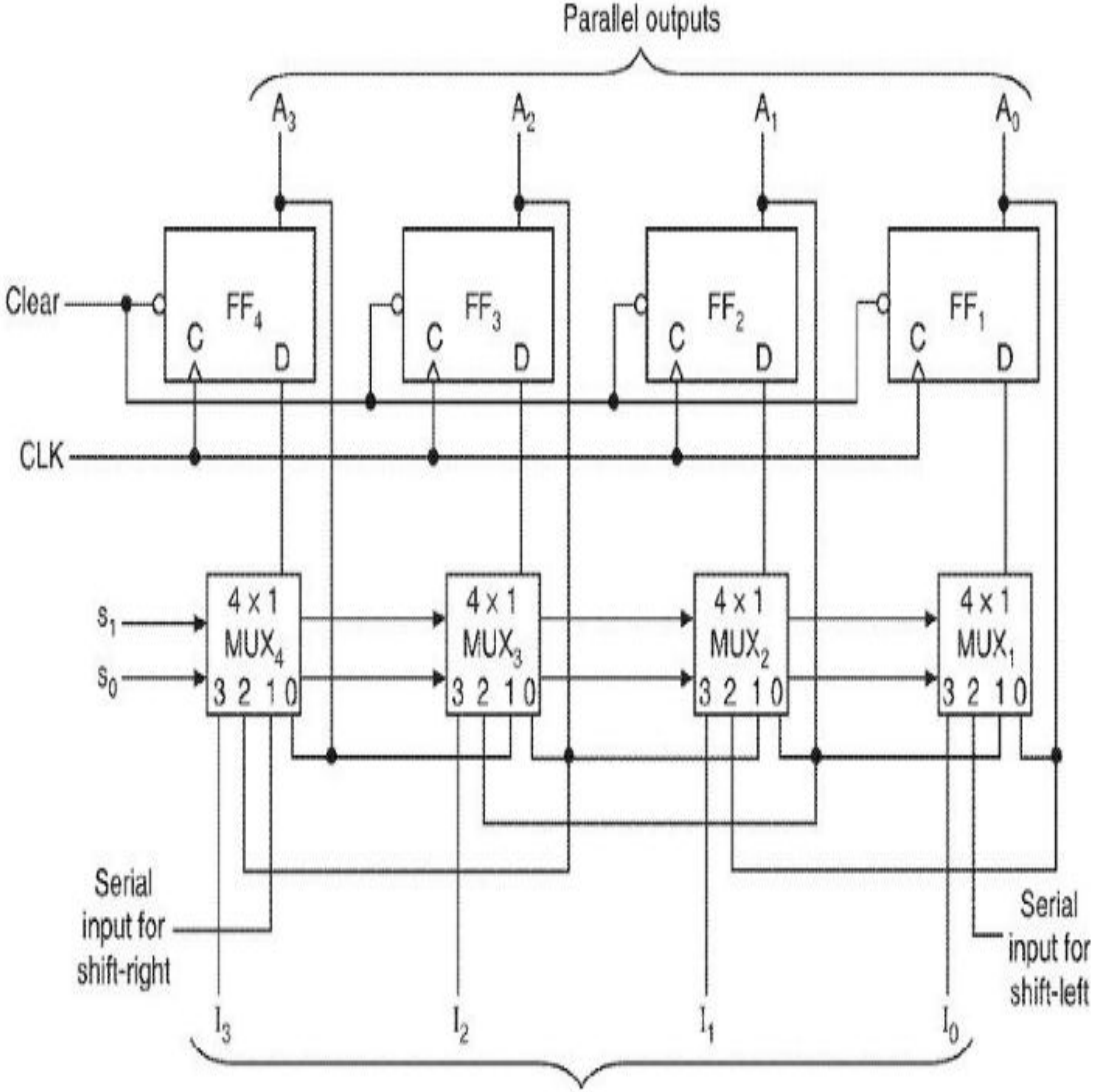
- When a clock pulse occurs, the data bits are then effectively shifted one place to the right.

- A low on the Right/Left control input enables the AND gates G5, G6, G7, and G8 and disables the AND gates G1, G2, G3, and G4, and the Q output of each FF is passed to the D input of the preceding FF.

- When a clock pulse occurs, the data bits are then effectively shifted one place to the left. Hence, the circuit works as a bidirectional shift register.

Universal Shift Registers

- A Universal shift register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be either in serial form or in parallel form.


Parallel outputs
A3
A2
A1
A0
Clear
FF4
C
D
FF3
C
D
FF2
C
D
FF1
C
D
CLK
s1
s0
4 × 1
MUX4
3 2 1 0
4 × 1
MUX3
3 2 1 0
4 × 1
MUX2
3 2 1 0
4 × 1
MUX1
3 2 1 0
Serial input for shift-right
Serial input for shift-left
I3
I2
I1
I0

- A universal shift register can be realized using multiplexers.

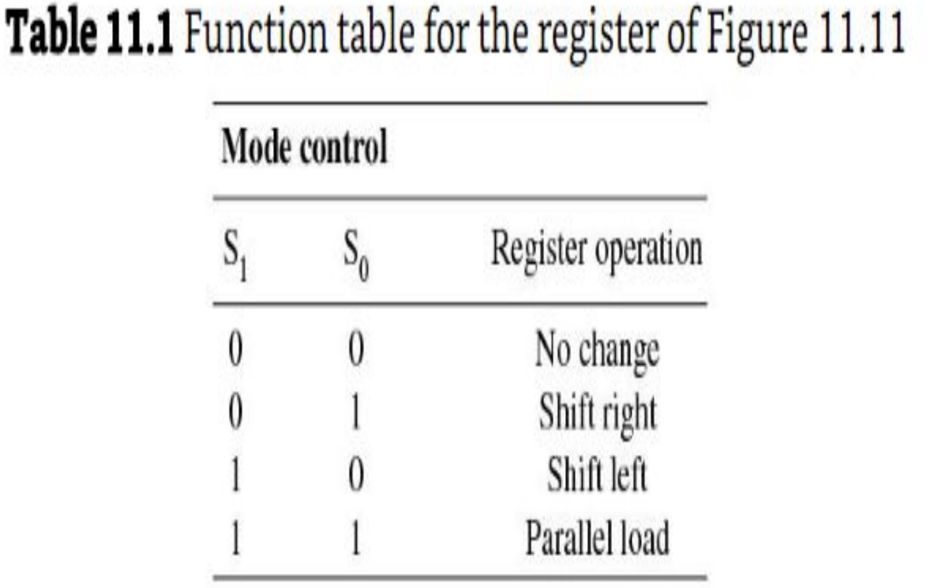


Table 11.1 Function table for the register of Figure 11.11

Mode control		
S_1	S_0	Register operation
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

- It consists of four D flip-flops and four multiplexers.

- The four multiplexers have two common selection inputs S1 and S0.

- Input 0 in each multiplexer is selected when S1S0 = 00, input 1 is selected when S1S0 = 01, and input 2 is selected when S1S0 = 10 and input 3 is selected when S1S0 = 11.

- The selection inputs control the mode of operation of the register according to the function entries in Table 11.1.

- When S1S0 = 0, the present value of the register is applied to the D inputs of flip-flops.

- This condition forms a path from the output of each flip-flop into the input of the same flip-flop.

- The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs.

- When S1S0 = 01, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop FF4.

- When S1S0 = 10, a shift-left operation results with the other serial input going into flip-flop FF1.

- Finally when S1S0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge.

Applications of Shift Registers

- 1. Time delays:

- 2. Serial/ Parallel data conversion:

- Ring counters:

COUNTERS

- A digital counter is a set of flip-flops (FFs) whose states change in response to pulses applied at the input to the counter.

- The FFs are interconnected such that their combined state at any time is the binary equivalent of the total number of pulses that have occurred up to that time.

- Ie, A counter is used to count pulses.

Types of Counters

- 1. Asynchronous Counters (Ripple Counters)
- 2. Synchronous Counters

- Asynchronous Counters

- Asynchronous counters are also called ripple counters.

- In ripple counters, the FFs within the counter are not made to change the states at exactly the same time. This is because the FFs are not triggered simultaneously.

- The clock does not directly control the time at which every stage changes state.

- Asynchronous counter uses T FFs to perform a counting function.

- Ripple counters are also called serial or series counters.

Synchronous Counters

- Synchronous counters are clocked such that each FF in the counter is triggered at the same time.

- This is accomplished by connecting the clock line to each stage of the counter.

- Synchronous counters are faster than asynchronous counters, because the propagation delay involved is less.

Comparison of synchronous and asynchronous counters

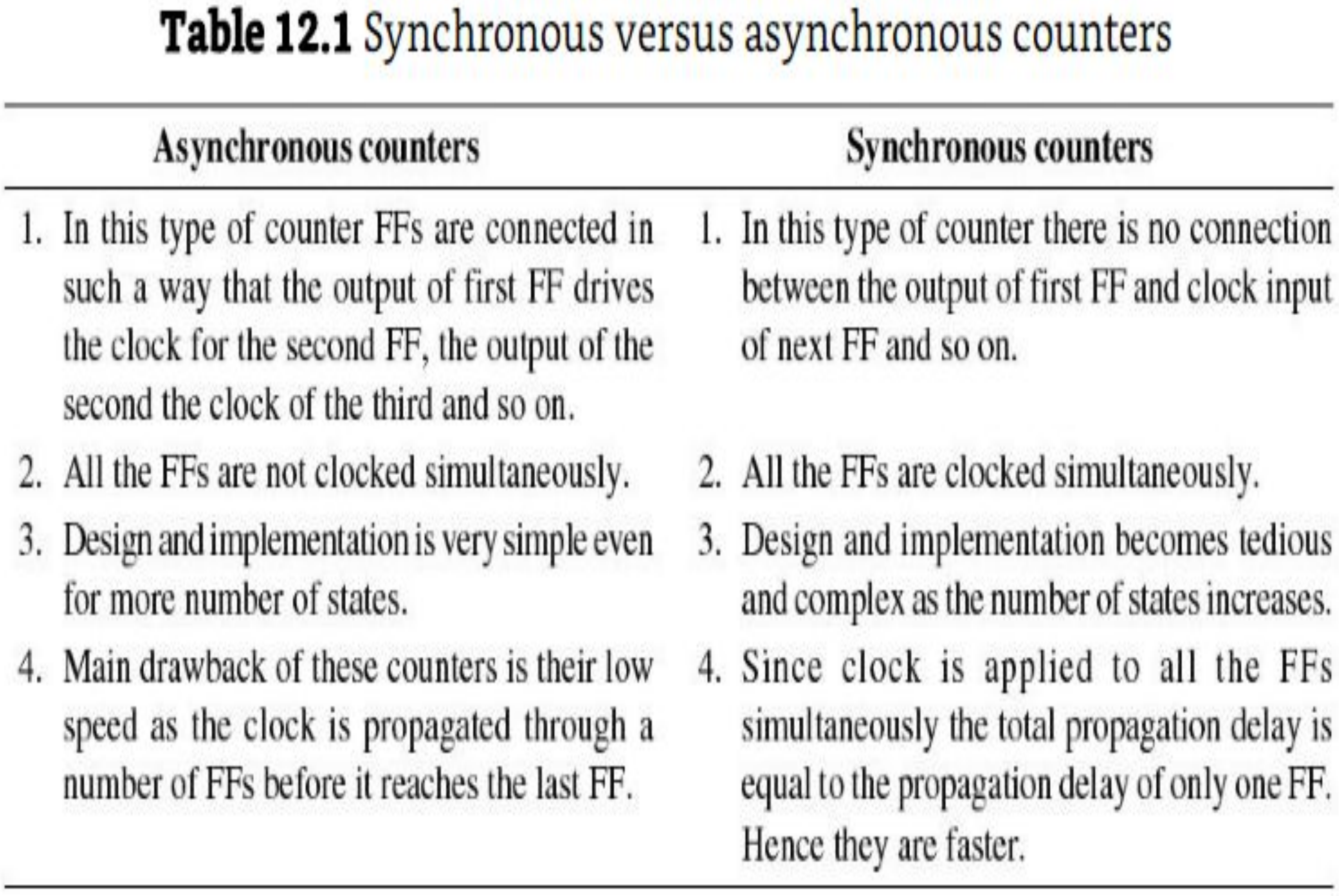


Table 12.1 Synchronous versus asynchronous counters

Asynchronous counters	Synchronous counters
1. In this type of counter FFs are connected in such a way that the output of first FF drives the clock for the second FF, the output of the second the clock of the third and so on.	1. In this type of counter there is no connection between the output of first FF and clock input of next FF and so on.
2. All the FFs are not clocked simultaneously.	2. All the FFs are clocked simultaneously.
3. Design and implementation is very simple even for more number of states.	3. Design and implementation becomes tedious and complex as the number of states increases.
4. Main drawback of these counters is their low speed as the clock is propagated through a number of FFs before it reaches the last FF.	4. Since clock is applied to all the FFs simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster.

- Up Counter

- A counter may be an up-counter or a down-counter. An up-counter is a counter which counts in the upward direction, i.e. 0, 1, 2, 3,…, N.

- Down Counter

- A down-counter is a counter which counts in the downward direction, i.e. N, N – 1, N – 2, N – 3, …, 1, 0.

- State

- Each of the counts of the counter is called the state of the counter.

- Modulus of the counter.

- In other words, the number of input pulses that causes the counter to reset to its initial count is called the modulus of the counter.

- Since a 2-bit counter has 4 states, it is called a mod-4 counter.

- It divides the input clock signal frequency by 4, therefore, it is also called a divide-by-4 counter. It requires two FFs.

- Similarly, a 3-bit counter uses 3 FFs and has $2^3 = 8$ states. It divides the input clock frequency by 2^3, i.e. 8.

- In general, an n-bit counter will have n FFs and 2^n states, and divides the input frequency by 2^n. Hence, it is a divide-by-2^n counter.

- The number of FFs required to construct a mod-N counter equals the smallest n for which N = 2^n.

- A mod-N counter divides the input frequency by N, hence, it is called a divide-by-N counter.

- Full modulus counter.

- A counter which goes through all the possible states before restarting is called the full modulus counter.

- Variable modulus counter.

- A counter in which the maximum number of states can be changed is called the variable modulus counter.

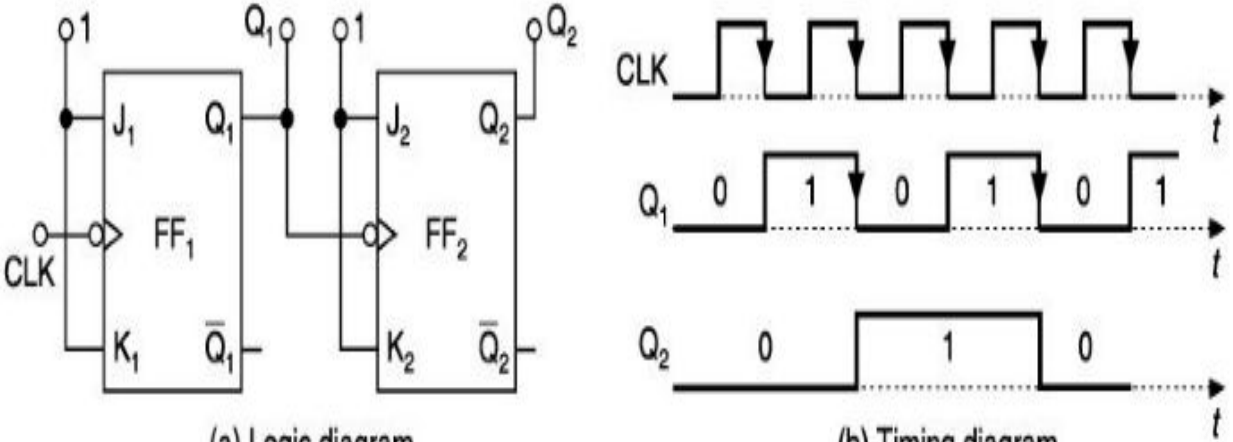
- Terminal count.

- The final state of the counter sequence is called the terminal count.

Asynchronous Counters

- Two-bit Ripple Up-counter Using Negative Edge-triggered Flip-Flops

- The 2-bit up-counter counts in the order 0, 1, 2, 3, 0, 1, …, i.e. 00, 01, 10, 11, 00, 01,…, etc.



(a) Logic diagram

(b) Timing diagram

- Figure shows a 2-bit ripple up-counter, using negative edge-triggered J-K FFs, and its timing diagram.

- The counter is initially reset to 00.

- When the first clock pulse is applied, FF1 toggles at the negative-going edge of this pulse, therefore, Q1 goes from low to high.

- This becomes a positive-going signal at the clock input of FF2. so, FF2 is not affected, and hence, the state of the counter after one clock pulse is Q1 = 1 and Q2 = 0, i.e. 01.

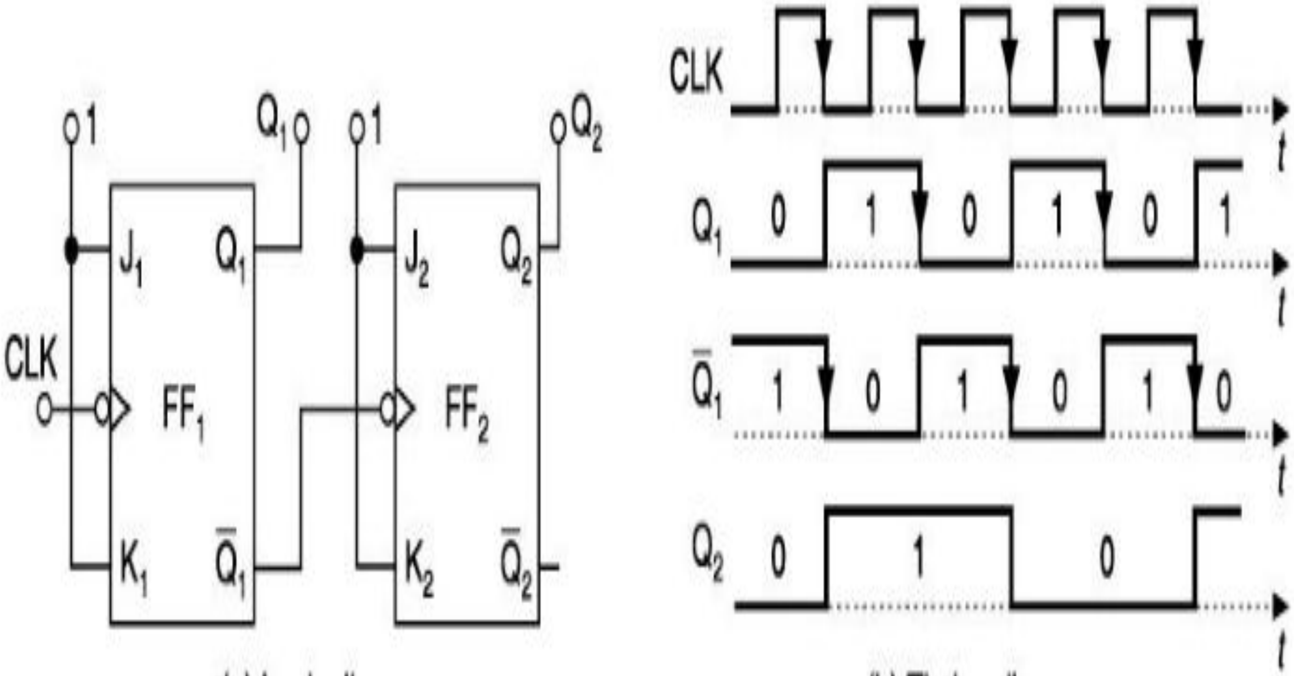
- At the negative-going edge of the second clock pulse, FF1 toggles. So, Q1 changes from high to low and this negative-going signal applied to CLK of FF2 activates FF2, and hence, Q2 goes from low to high. Therefore, Q1 = 0 and Q2 = 1, i.e. 10 is the state of the counter after the second clock pulse.

- At the negative-going edge of the third clock pulse, FF1 toggles. So Q1 changes from a 0 to a 1. This becomes a positive-going signal to FF2, hence, FF2 is not affected. Therefore, Q2 = 1 and Q1 = 1, i.e. 11 is the state of the counter after the third clock pulse.

- At the negative-going edge of the fourth clock pulse, FF1 toggles. So, Q1 goes from a 1 to a 0. This negative-going signal at Q1 toggles FF2, hence, Q2 also changes from a 1 to a 0. Therefore, Q2 = 0 and Q1 = 0, i.e. 00 is the state of the counter after the fourth clock pulse.

- For subsequent clock pulses, the counter goes through the same sequence of states. So, it acts as a mod-4 counter with Q1 as the LSB and Q2 as the MSB. The counting sequence is thus 00, 01, 10, 11, 00, 01,…, etc.

Two-bit Ripple Down-counter Using Negative Edge-triggered Flip-Flops

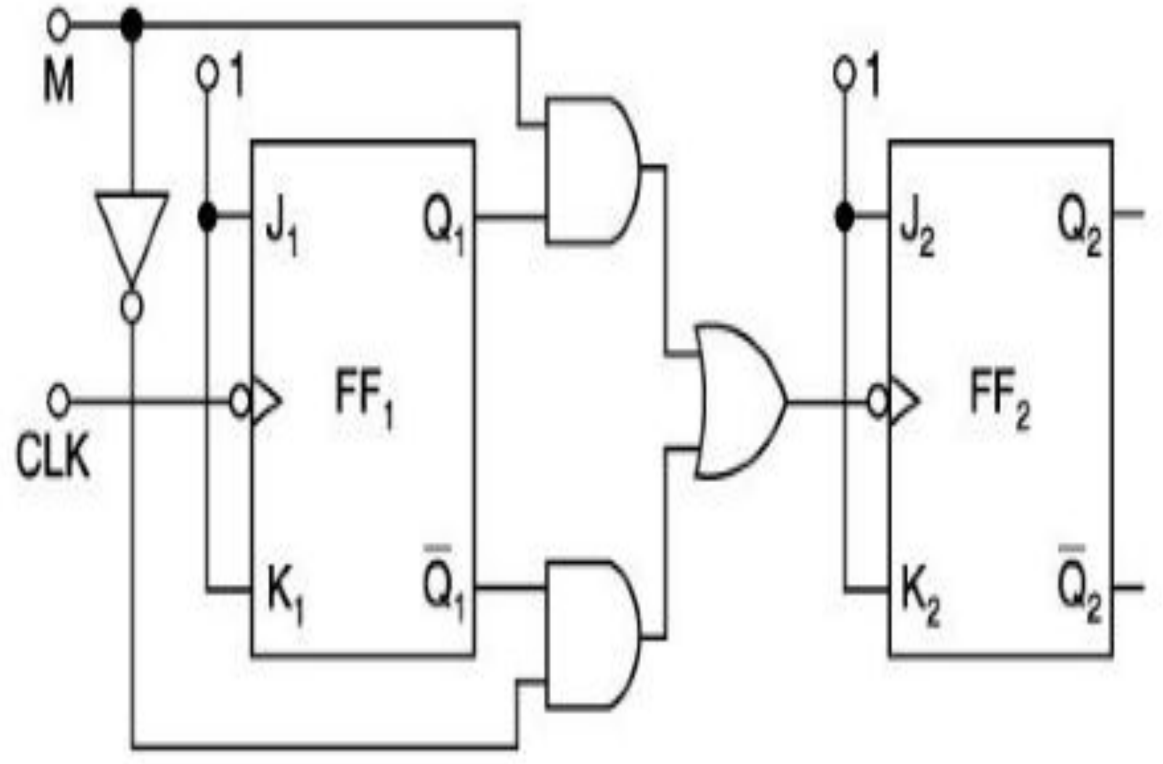


(a) Logic diagram

(b) Timing diagram

A 2-bit down-counter counts in the order 0, 3, 2, 1, 0, 3,…,
i.e. 00, 11, 10, 01, 00, 11, …, etc.

Two-bit Ripple Up-down Counter Using Negative Edge-triggered Flip-Flops



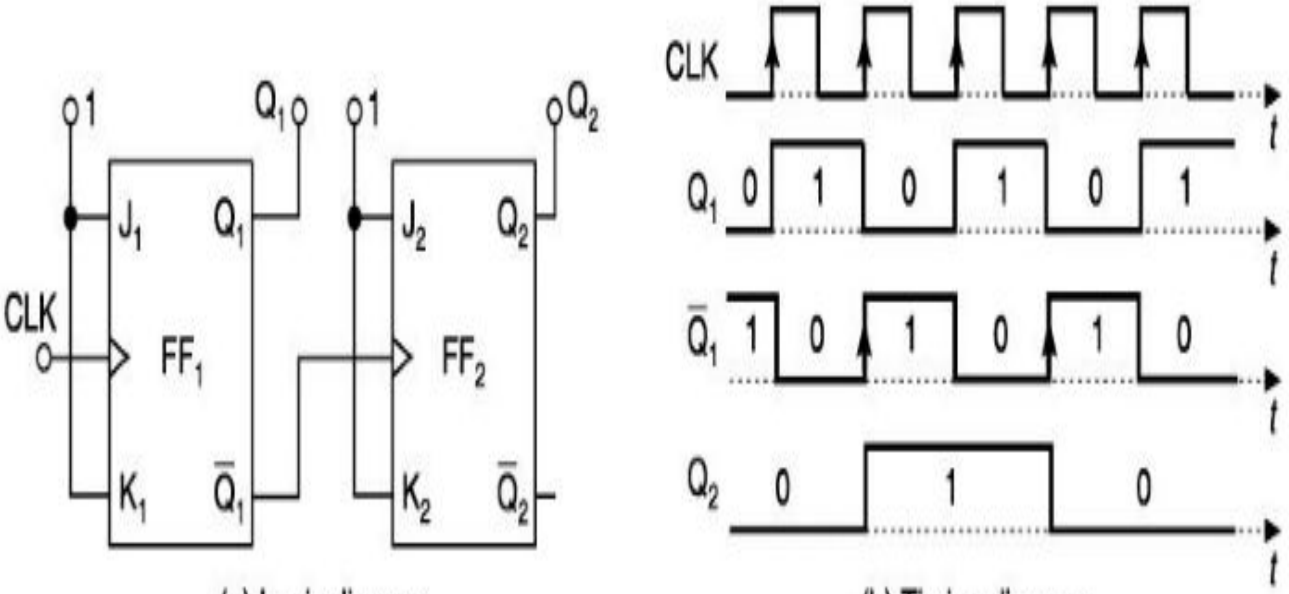
- An up-down counter is a counter which can count both in upward and downward directions.

- An up-down counter is also called a forward/ backward counter or a bidirectional counter.

- A control signal or a mode signal M is used to choose the direction of count.

- When M = 1 for up counting, Q1 is transmitted to clock of FF2 and when M = 0 for down counting, 1 is transmitted to clock of FF2

Two-bit Ripple Up-counter Using Positive Edge-triggered Flip-Fops



(a) Logic diagram

(b) Timing diagram

- The $\overline{Q1}$ output of the first FF is connected to the clock of FF2.

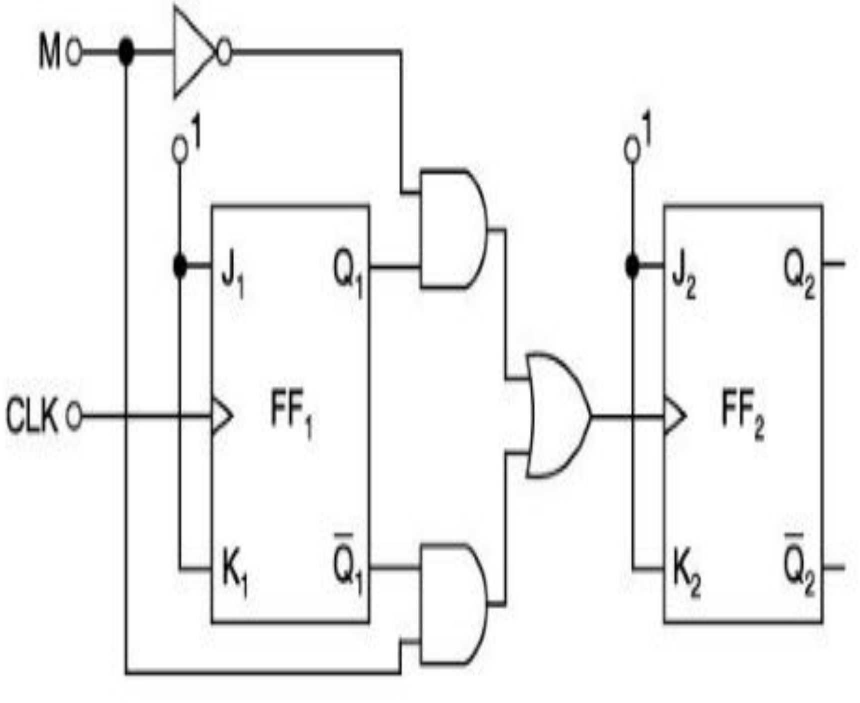
- The external clock signal is applied to the first flip-flop FF1.

- The FF1 toggles at the positive-going edge of each clock pulse and FF2 toggles whenever $\overline{Q1}$ changes from a 0 to a 1.

- State transitions occur at the positive-going edges of the clock pulses.

- The counting sequence is 00, 01, 10, 11, 00, 01, …, etc.

Two-bit Ripple Up/ Down Counter Using Positive Edge-triggered Flip-Flops



When M = 1 for up co $\overline{Q1}$ ng, is transmitted to the clock of FF2 and when M = 0 for down counting, Q1 is transmitted to the clock of FF2.

Design of Asynchronous Counters

- To design an asynchronous counter, first write the counting sequence, then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R or using K-map

- Provide a feedback such $\overline{R}$ that R or resets all the FFs after the desired count.

Design of a Mod-6 Asynchronous Counter Using T FFs

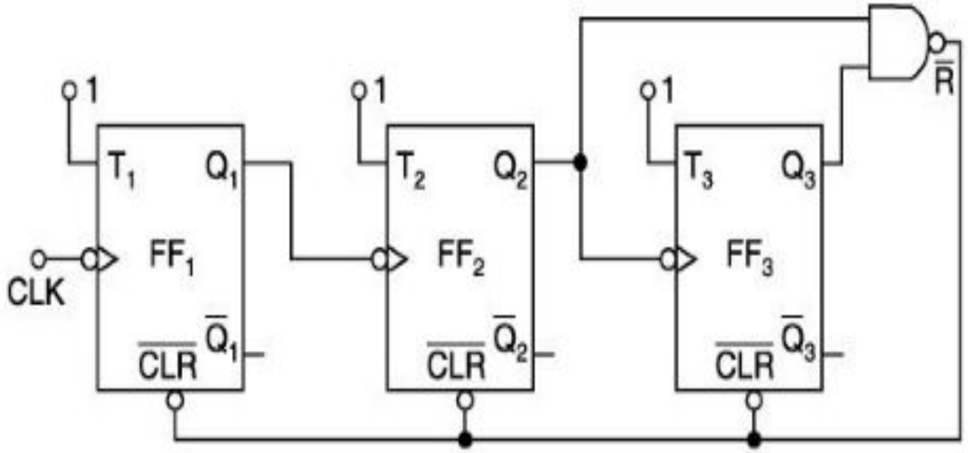
- A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101.

- When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided.

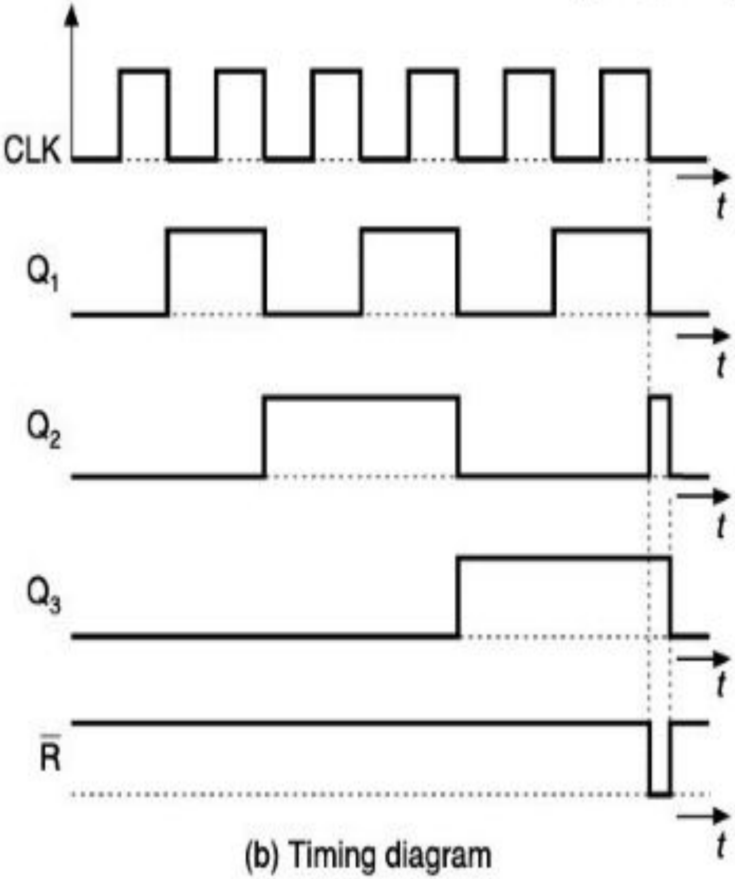
- It is a ‘divide-by-6 counter’, in the sense that it divides the input clock frequency by 6.

- It requires three FFs, because the smallest value of n satisfying the condition $N <= 2^n$ is n = 3; three FFs can have eight possible states, out of which only six are utilized and the remaining two states 110 and 111, are invalid.

- If initially the counter is in 000 state, then after the first clock pulse it goes to 001, after the second clock pulse, it goes to 010, and so on. After the sixth clock pulse, it goes to 000.



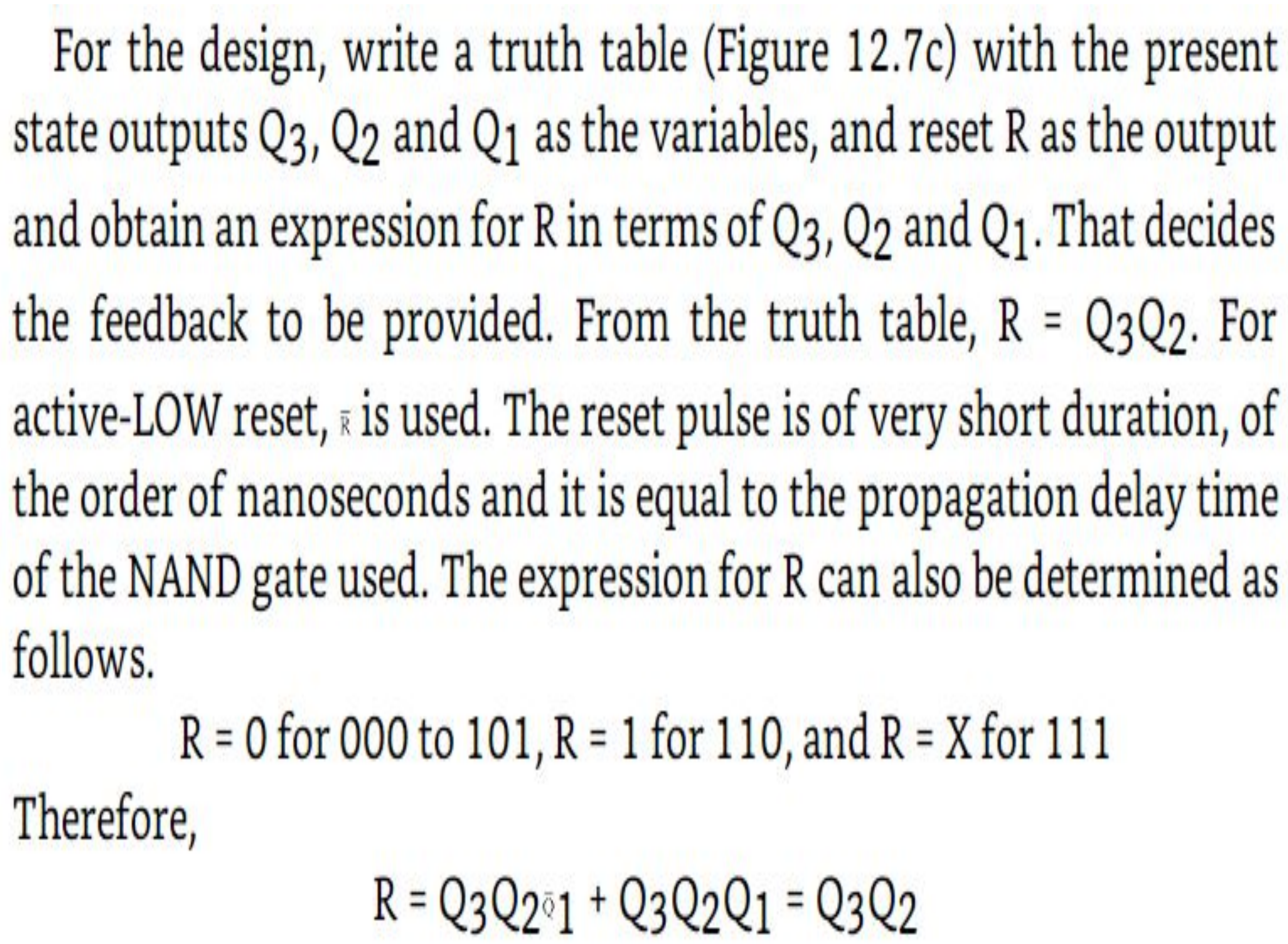
(a) Logic diagram



(b) Timing diagram

After pulses	State			R
	Q_3	Q_2	Q_1	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
	0	0	0	0
7	0	0	1	0

(c) Table for R



For the design, write a truth table (Figure 12.7c) with the present state outputs Q_3, Q_2 and Q_1 as the variables, and reset R as the output and obtain an expression for R in terms of Q_3, Q_2 and Q_1. That decides the feedback to be provided. From the truth table, $R = Q_3Q_2$. For active-LOW reset, $\bar{R}$ is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.

R = 0 for 000 to 101, R = 1 for 110, and R = X for 111

Therefore,

$$R = Q_3Q_2\bar{Q}_1 + Q_3Q_2Q_1 = Q_3Q_2$$

Design of a Mod-10 Asynchronous Counter Using T FFs (BCD/Decade)

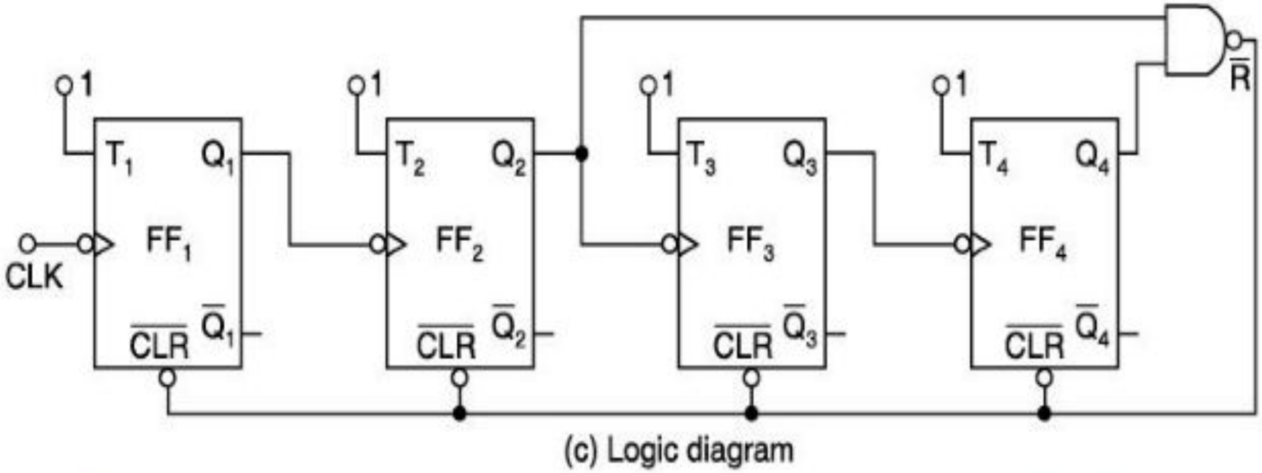
- A mod-10 counter is a decade counter.
- It is also called a BCD counter or a divide-by-10 counter.
- It requires four FFs (the smallest value of n satisfying the condition 10 <=2^n, is n = 4). So, there are 16 possible states, out of which ten are valid and the remaining six are invalid.
- The counter has ten stable states, 0000 through 1001, i.e. it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001.

After pulses	Count			
	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	0	0	0	0

(a) Count table

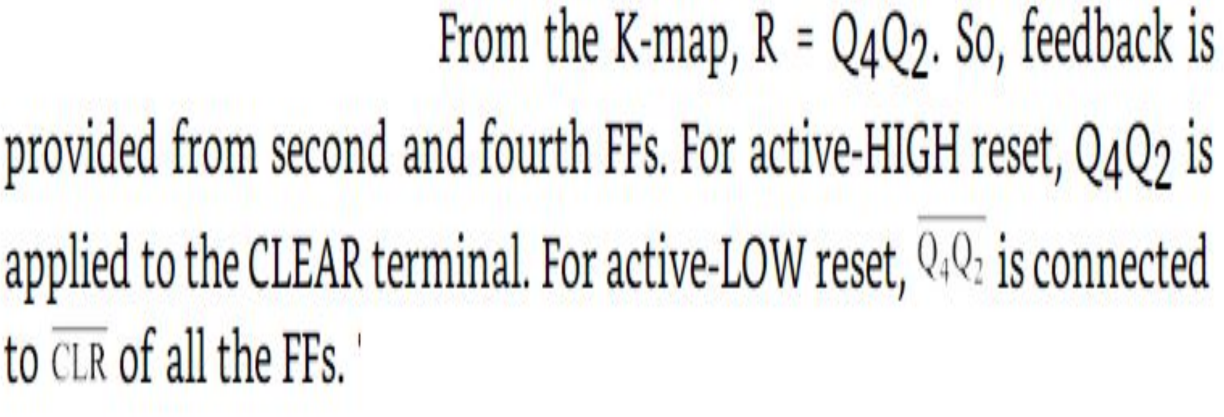
$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12 ×	13 ×	15 ×	14 ×
10	8	9	11 ×	10 1

(b) K-Map



(c) Logic diagram

Figure 12.8 Asynchronous mod-10 counter using T flip-flops.



From the K-map, $R = Q_4Q_2$. So, feedback is provided from second and fourth FFs. For active-HIGH reset, Q_4Q_2 is applied to the CLEAR terminal. For active-LOW reset, $\overline{Q_4Q_2}$ is connected to $\overline{CLR}$ of all the FFs.

Synchronous Counters

- Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. The propagation delay thus gets accumulated,

- Synchronous counters are counters in which all the FFs are triggered simultaneously (in parallel) by the clock-input pulses.

- Since all the FFs change state simultaneously in synchronization with the clock pulse, the propagation delays of FFs do not add together (as in ripple counters) to produce the overall delay.

Design of Synchronous Counters

- For a systematic design of synchronous counters, the following procedure is used.

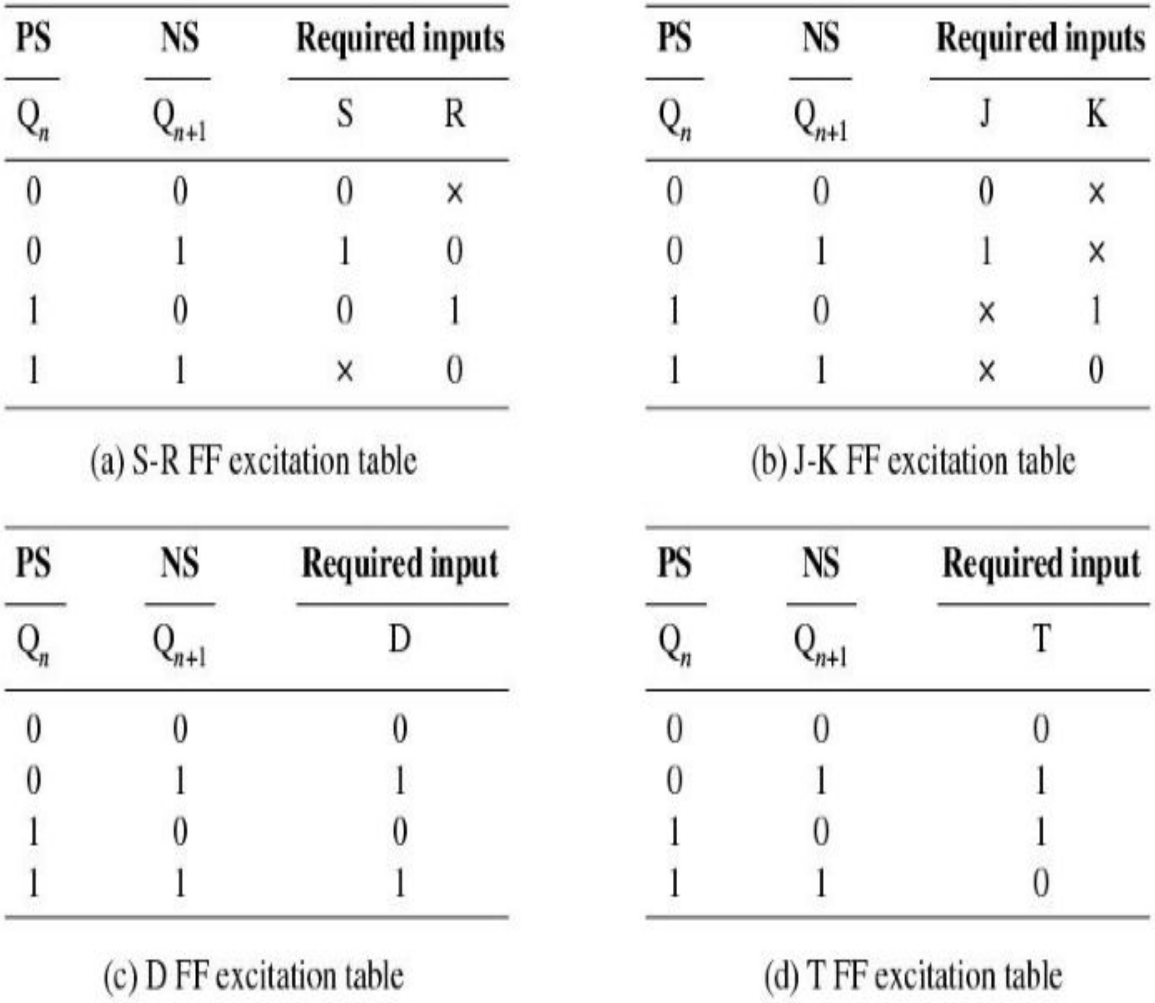
Step 1. Number of flip-flops:

Step 2. State diagram: (A state diagram, which can also be called the transition diagram, is a graphical means of depicting the sequence of states through which the counter progresses.)

Step 3. Choice of flip-flops and excitation table:

Step 4. Minimal expressions for excitations:

The excitation tables of various flip-flops used in the counters



PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

(a) S-R FF excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

(b) J-K FF excitation table

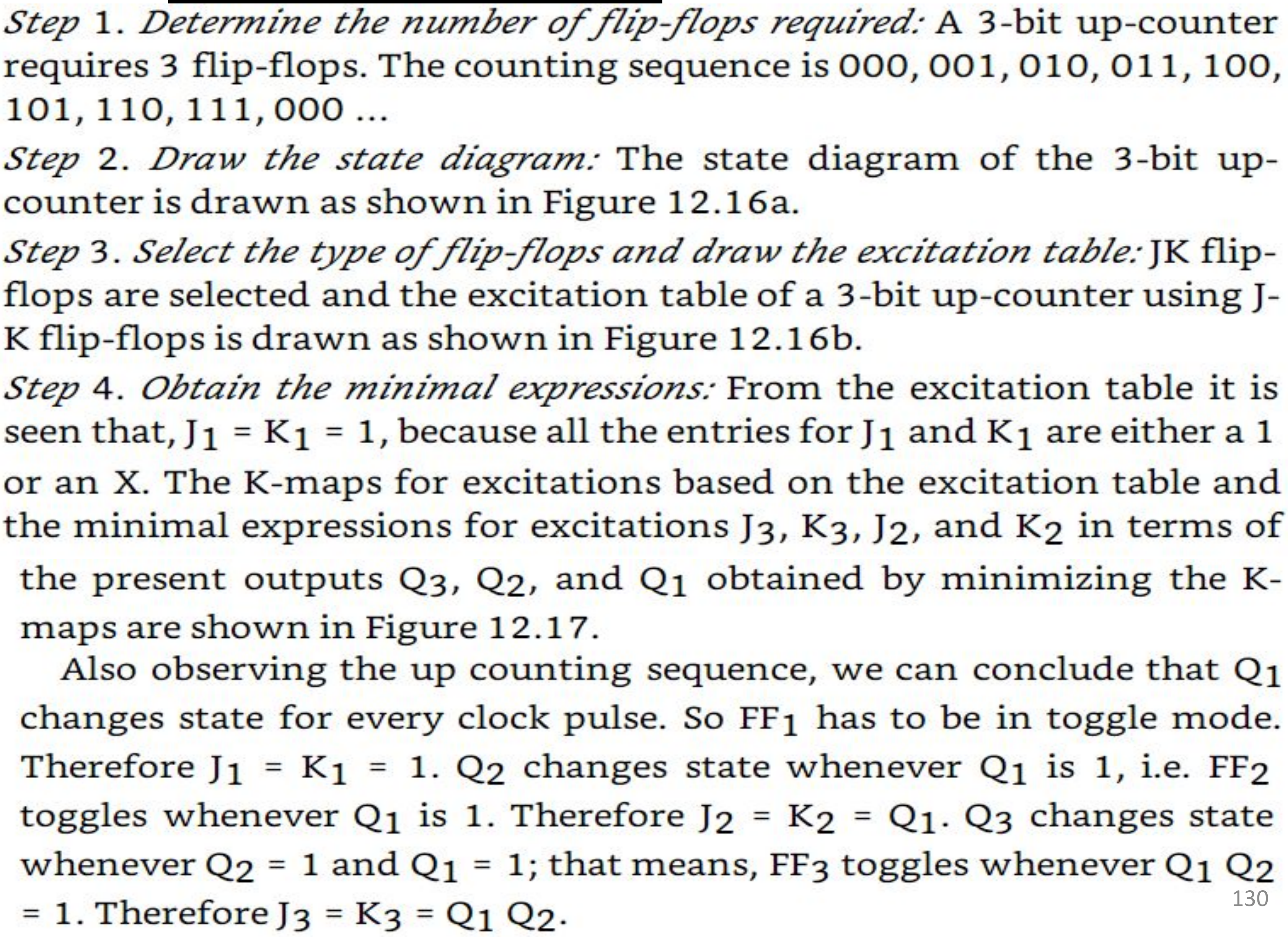
PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

(c) D FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(d) T FF excitation table

Design of a Synchronous 3-bit Up Counter Using J-K FFs



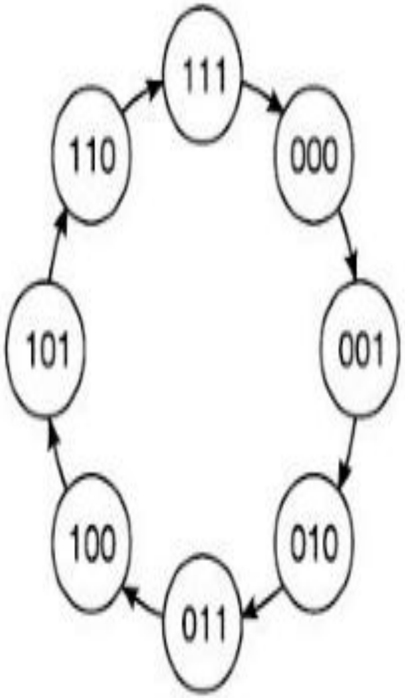
Step 1. *Determine the number of flip-flops required:* A 3-bit up-counter requires 3 flip-flops. The counting sequence is 000, 001, 010, 011, 100, 101, 110, 111, 000 ...

Step 2. *Draw the state diagram:* The state diagram of the 3-bit up-counter is drawn as shown in Figure 12.16a.

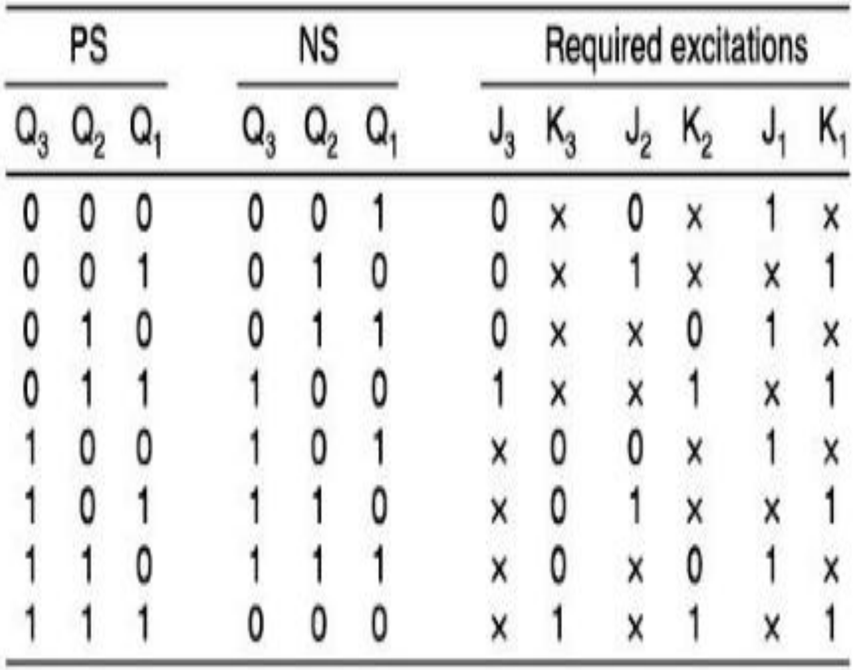
Step 3. *Select the type of flip-flops and draw the excitation table:* JK flip-flops are selected and the excitation table of a 3-bit up-counter using J-K flip-flops is drawn as shown in Figure 12.16b.

Step 4. *Obtain the minimal expressions:* From the excitation table it is seen that, $J_1 = K_1 = 1$, because all the entries for J_1 and K_1 are either a 1 or an X. The K-maps for excitations based on the excitation table and the minimal expressions for excitations J_3, K_3, J_2, and K_2 in terms of the present outputs Q_3, Q_2, and Q_1 obtained by minimizing the K-maps are shown in Figure 12.17.

Also observing the up counting sequence, we can conclude that Q_1 changes state for every clock pulse. So FF_1 has to be in toggle mode. Therefore $J_1 = K_1 = 1$. Q_2 changes state whenever Q_1 is 1, i.e. FF_2 toggles whenever Q_1 is 1. Therefore $J_2 = K_2 = Q_1$. Q_3 changes state whenever $Q_2 = 1$ and $Q_1 = 1$; that means, FF_3 toggles whenever $Q_1 Q_2 = 1$. Therefore $J_3 = K_3 = Q_1 Q_2$.



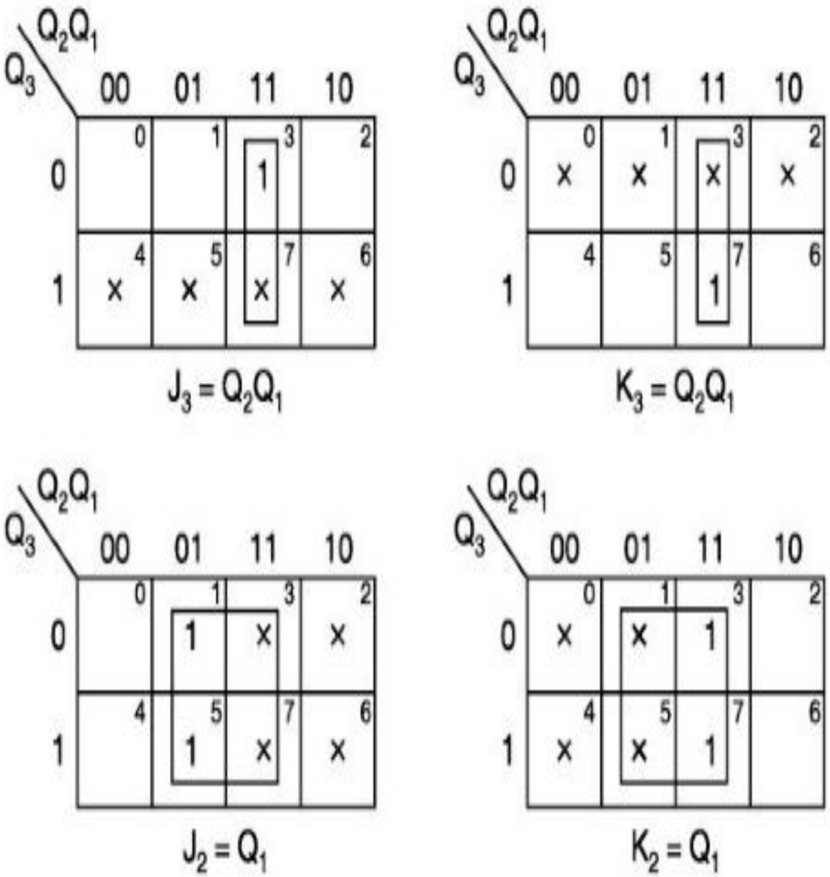
(a) State diagram



PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
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	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
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	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1

(b) Excitation table

Figure 12.16 A 3-bit up-counter.



$J_3 = Q_2Q_1$

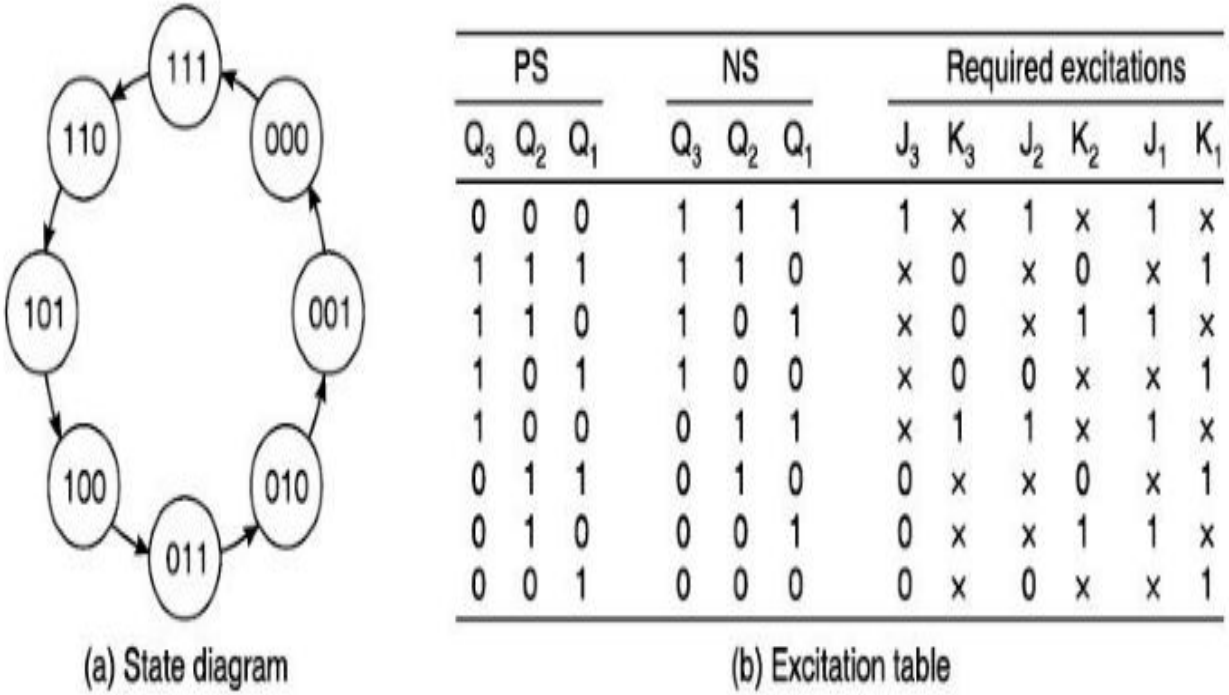
$K_3 = Q_2Q_1$

$J_2 = Q_1$

$K_2 = Q_1$

Figure 12.17 Karnaugh maps for a 3-bit up-counter.

Design of Synchronous 3-bit Down-counter

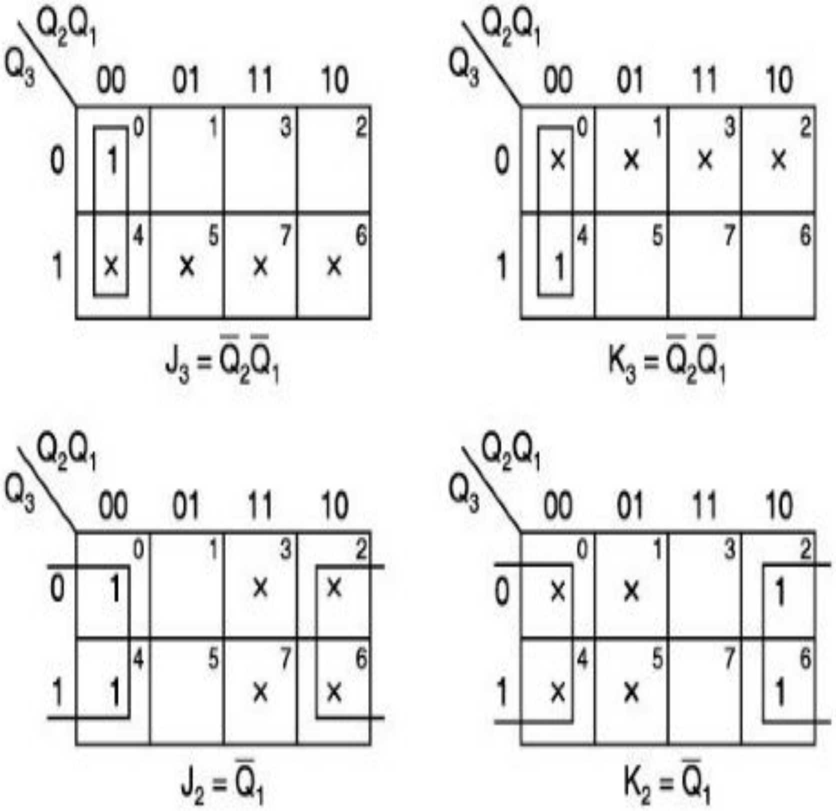


(a) State diagram

PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	1	1	1	1	x	1	x	1	x
1	1	1	1	1	0	x	0	x	0	x	1
1	1	0	1	0	1	x	0	x	1	1	x
1	0	1	1	0	0	x	0	0	x	x	1
1	0	0	0	1	1	x	1	1	x	1	x
0	1	1	0	1	0	0	x	x	0	x	1
0	1	0	0	0	1	0	x	x	1	1	x
0	0	1	0	0	0	0	x	0	x	x	1

(b) Excitation table

Figure 12.18 A 3-bit down-counter.



$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	1			
1	x	x	x	x

$J_3 = \overline{Q}_2\overline{Q}_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	x	x	x	x
1	1			

$K_3 = \overline{Q}_2\overline{Q}_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	1		x	x
1	1		x	x

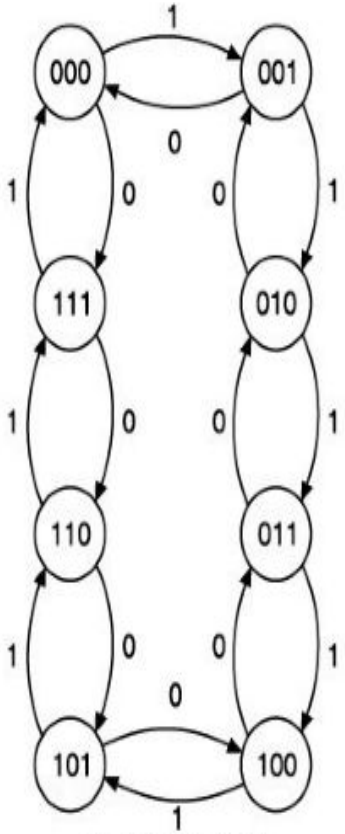
$J_2 = \overline{Q}_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	x	x		1
1	x	x		1

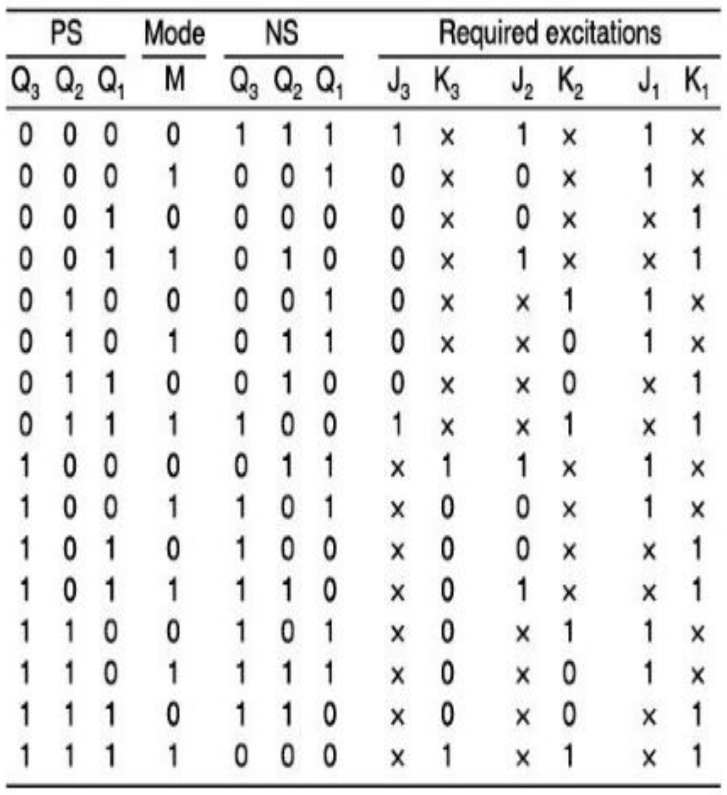
$K_2 = \overline{Q}_1$

Figure 12.19 K-maps for a three-bit down counter using J-K FFs.

Design of a Synchronous 3-bit Up-down Counter Using J-K FFs



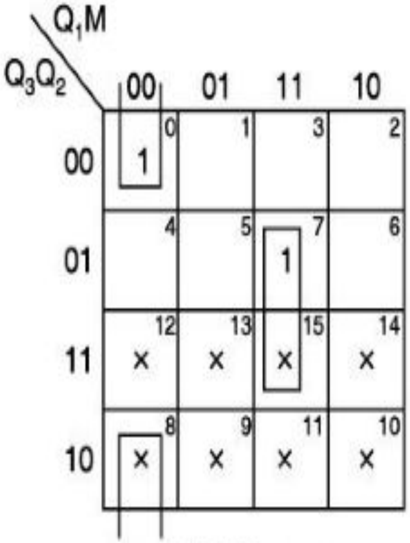
(a) State diagram



PS			Mode	NS			Required excitations					
Q_3	Q_2	Q_1	M	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	1	1	1	1	×	1	×	1	×
0	0	0	1	0	0	1	0	×	0	×	1	×
0	0	1	0	0	0	0	0	×	0	×	×	1
0	0	1	1	0	1	0	0	×	1	×	×	1
0	1	0	0	0	0	1	0	×	×	1	1	×
0	1	0	1	0	1	1	0	×	×	0	1	×
0	1	1	0	0	1	0	0	×	×	0	×	1
0	1	1	1	1	0	0	1	×	×	1	×	1
1	0	0	0	0	1	1	×	1	1	×	1	×
1	0	0	1	1	0	1	×	0	0	×	1	×
1	0	1	0	1	0	0	×	0	0	×	×	1
1	0	1	1	1	1	0	×	0	1	×	×	1
1	1	0	0	1	0	1	×	0	×	1	1	×
1	1	0	1	1	1	1	×	0	×	0	1	×
1	1	1	0	1	1	0	×	0	×	0	×	1
1	1	1	1	0	0	0	×	1	×	1	×	1

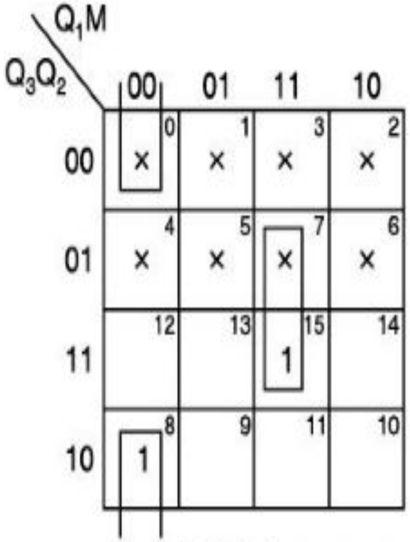
(b) Excitation table

Figure 12.13 Synchronous 3-bit up-down counter.



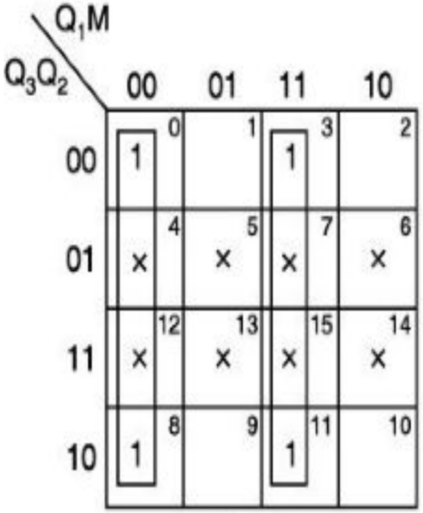
$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	0 1	1	3	2
01	4	5	7 1	6
11	12 ×	13 ×	15 ×	14 ×
10	8 ×	9 ×	11 ×	10 ×

$J_3 = \overline{Q}_2\overline{Q}_1\overline{M} + Q_2Q_1M$



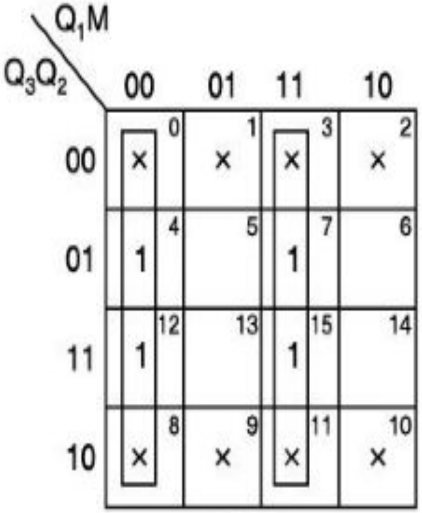
$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	0 ×	1 ×	3 ×	2 ×
01	4 ×	5 ×	7 ×	6 ×
11	12	13	15 1	14
10	8 1	9	11	10

$K_3 = \overline{Q}_2\overline{Q}_1\overline{M} + Q_2Q_1M$



$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	0 1	1	3 1	2
01	4 ×	5 ×	7 ×	6 ×
11	12 ×	13 ×	15 ×	14 ×
10	8 1	9	11 1	10

$J_2 = \overline{Q}_1\overline{M} + Q_1M$



$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	0 ×	1 ×	3 ×	2 ×
01	4 1	5	7 1	6
11	12 1	13	15 1	14
10	8 ×	9 ×	11 ×	10 ×

$K_2 = \overline{Q}_1\overline{M} + Q_1M$

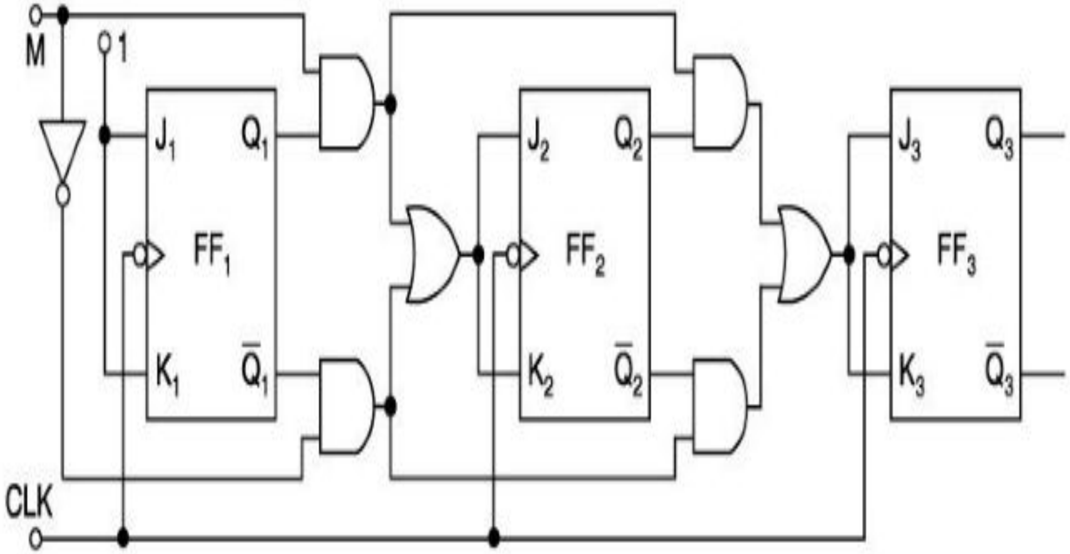


Figure 12.15 Logic diagram of the synchronous 3-bit up-down counter using J-K

Design of a Synchronous Modulo-10 up/ down Counter Using T FFs

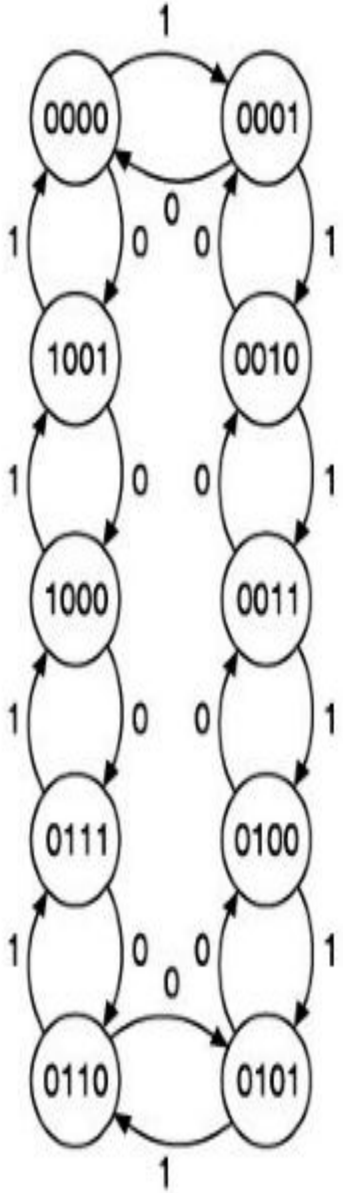
- Step 1. The number of flip-flops:

- A modulo-10 counter has 10 states and so it requires 4 FFs. (10 <= 2^4). 4-FFs can have16 states. So out of 16, six states (1010 through 1111) are invalid.

- The entries for excitations corresponding to invalid states are don’t cares.

- For selecting up and down modes a control or mode signal is required. Let us say it counts up when the mode signal M = 1 and counts down when M = 0.

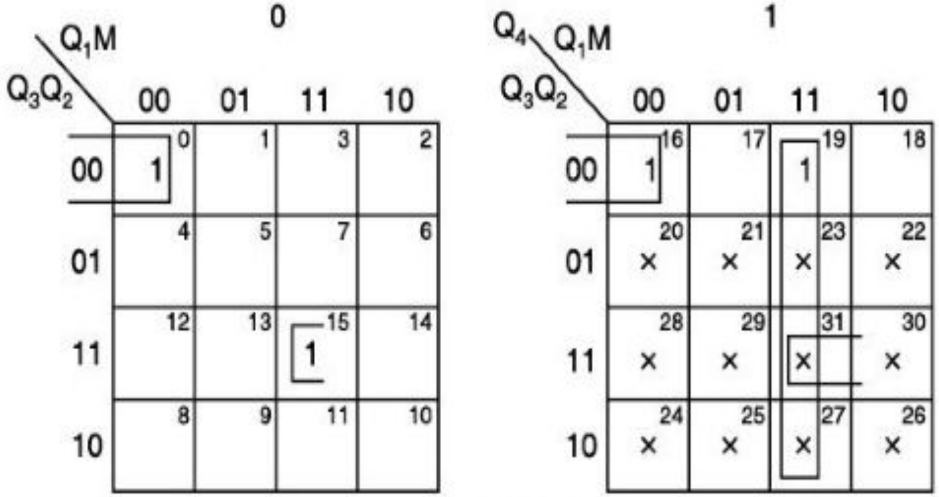
- The clock signal is applied to all the FFs simultaneously.



(a) State diagram

PS				Mode	NS				Required excitations			
Q_4	Q_3	Q_2	Q_1	M	Q_4	Q_3	Q_2	Q_1	T_4	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	0	1	0	0	1	1
0	0	1	0	1	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1	1	0	0	0	1
0	1	1	1	0	0	1	1	0	0	0	0	1
0	1	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	1	0	0	1

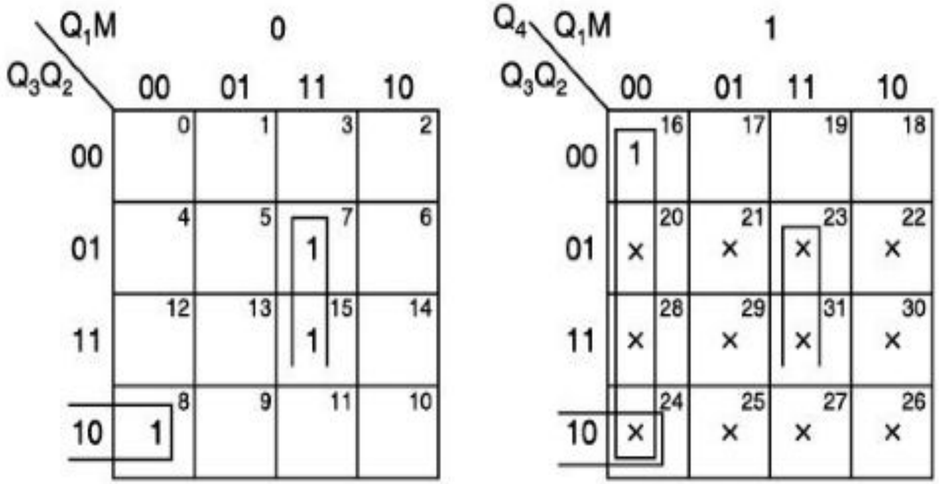
(b) Excitation table



Q_3Q_2 \ Q_1M (Q_4 = 0)	00	01	11	10
00	1			
01				
11			1	
10				

Q_3Q_2 \ Q_1M (Q_4 = 1)	00	01	11	10
00	1		1	
01	×	×	×	×
11	×	×	×	×
10	×	×	×	×

$$T_4 = Q_4Q_1M + \bar{Q}_3\bar{Q}_2\bar{Q}_1\bar{M} + Q_3Q_2Q_1M$$

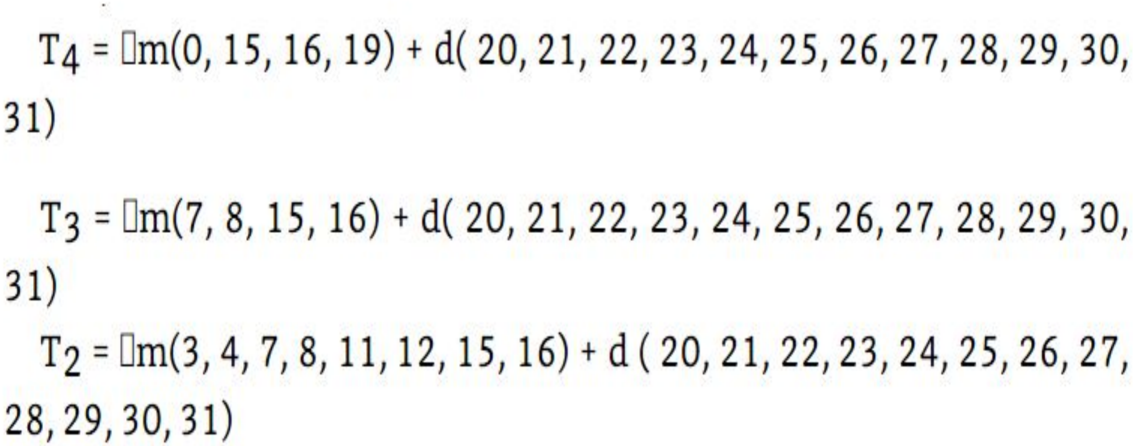


Q_3Q_2 \ Q_1M (Q_4 = 0)	00	01	11	10
00				
01			1	
11			1	
10	1			

Q_3Q_2 \ Q_1M (Q_4 = 1)	00	01	11	10
00	1			
01	×	×	×	×
11	×	×	×	×
10	×	×	×	×

$$T_3 = Q_4\bar{Q}_1\bar{M} + Q_2Q_1M + Q_3\bar{Q}_2\bar{Q}_1\bar{M}$$

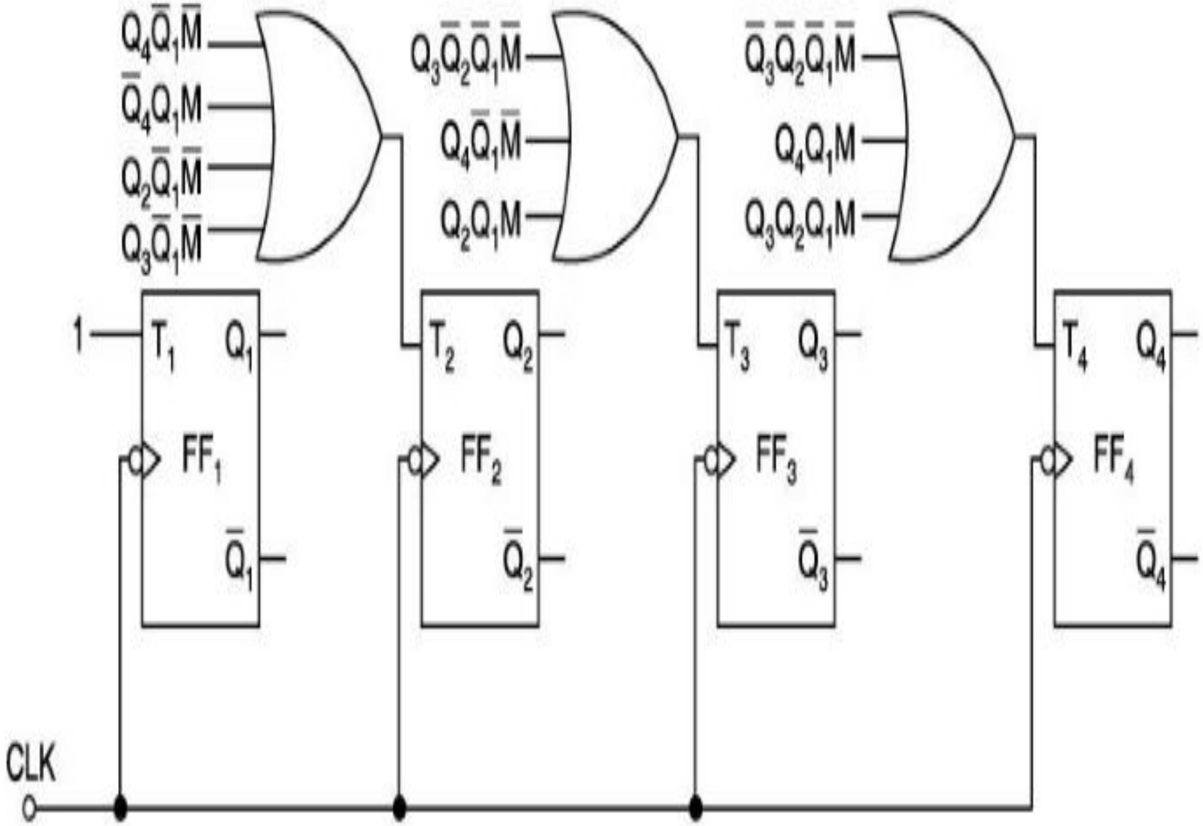
$$T_2 = Q_4\bar{Q}_1\bar{M} + \bar{Q}_4Q_1M + Q_2\bar{Q}_1\bar{M} + Q_3\bar{Q}_1\bar{M}$$



T_4 = □m(0, 15, 16, 19) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)

T_3 = □m(7, 8, 15, 16) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)

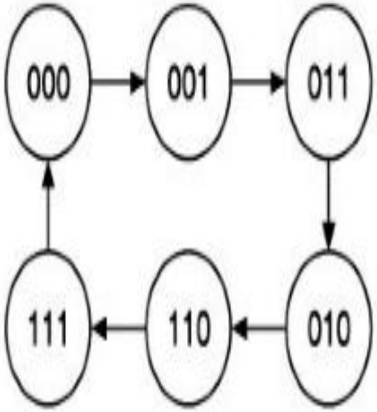
T_2 = □m(3, 4, 7, 8, 11, 12, 15, 16) + d (20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)


$Q_4\overline{Q}_1\overline{M}$
$\overline{Q}_4Q_1M$
$Q_2\overline{Q}_1\overline{M}$
$Q_3\overline{Q}_1\overline{M}$
$Q_3\overline{Q}_2\overline{Q}_1\overline{M}$
$Q_4\overline{Q}_1\overline{M}$
Q_2Q_1M
$\overline{Q}_3\overline{Q}_2\overline{Q}_1\overline{M}$
Q_4Q_1M
$Q_3Q_2Q_1M$
1
T_1
Q_1
FF_1
$\overline{Q}_1$
T_2
Q_2
FF_2
$\overline{Q}_2$
T_3
Q_3
FF_3
$\overline{Q}_3$
T_4
Q_4
FF_4
$\overline{Q}_4$
CLK

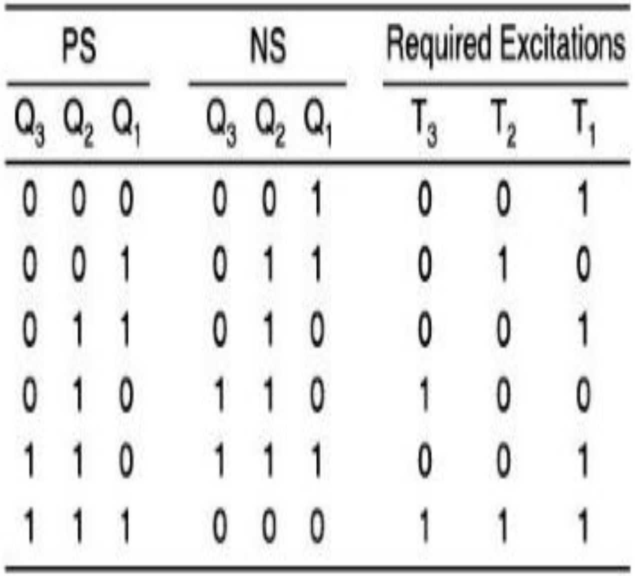
Home Work

- 1. Design a Modulo-9 Synchronous Counter Using T FFs.
- 2. Design of a Synchronous Mod-6 Counter Using J-K FFs

Design of a Synchronous Modulo-6 Gray Code Counter

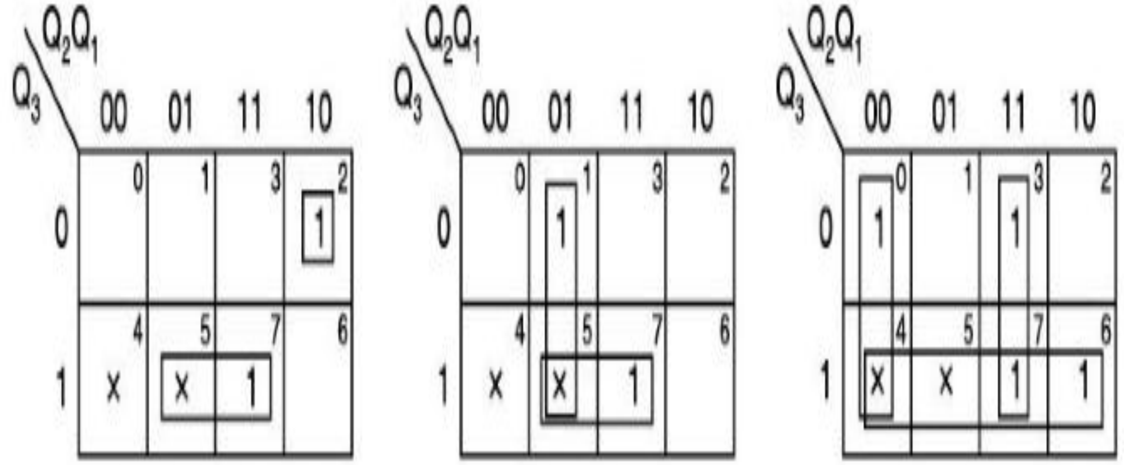


(a) State diagram



PS			NS			Required Excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

(b) Excitation table



$Q_3 \backslash Q_2Q_1$	00	01	11	10
0				1
1	X	X	1	

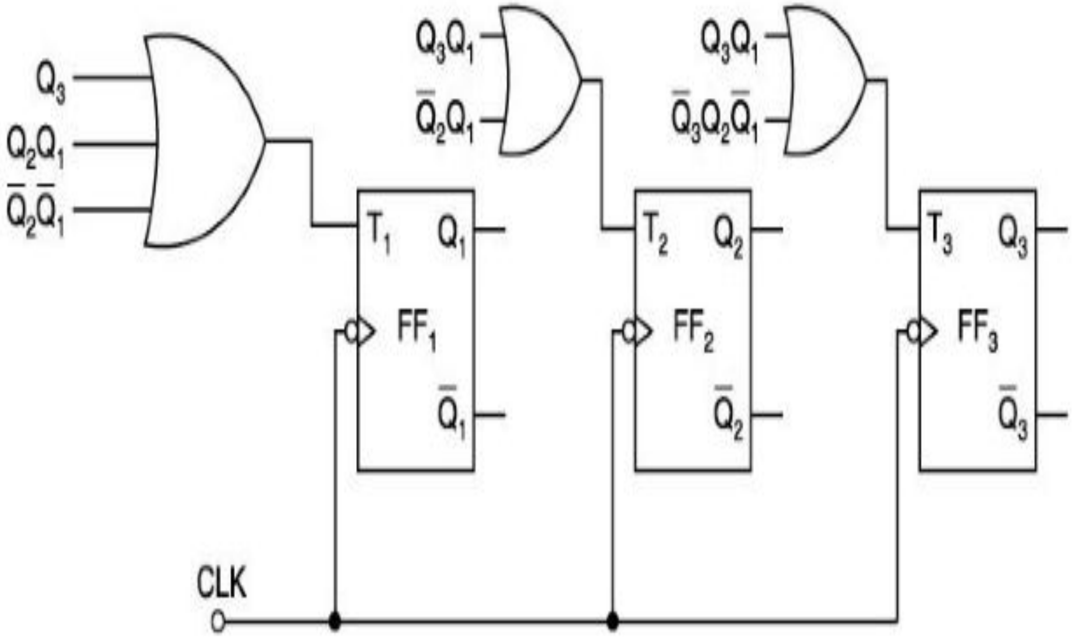
$T_3 = Q_3Q_1 + \bar{Q}_3Q_2\bar{Q}_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0		1		
1	X	X	1	

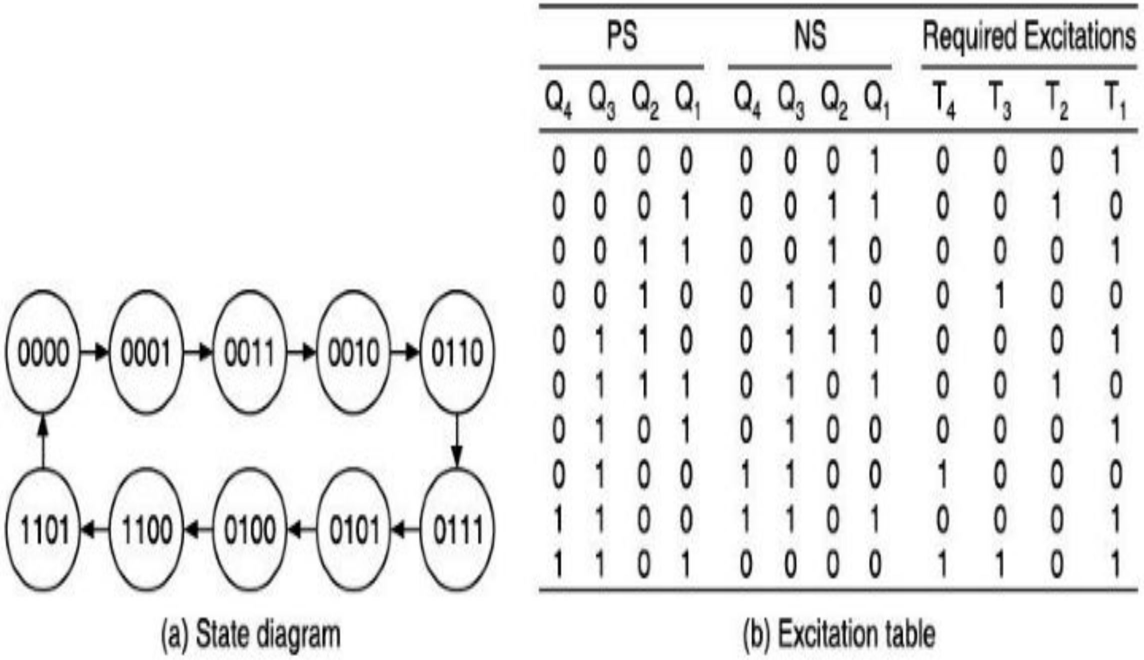
$T_2 = Q_3Q_1 + \bar{Q}_2Q_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	1		1	
1	X	X	1	1

$T_1 = Q_3 + Q_2Q_1 + \bar{Q}_2\bar{Q}_1$


Q_3
Q_2Q_1
$\overline{Q}_2\overline{Q}_1$
Q_3Q_1
$\overline{Q}_2Q_1$
Q_3Q_1
$\overline{Q}_3Q_2\overline{Q}_1$
T_1
Q_1
FF_1
$\overline{Q}_1$
T_2
Q_2
FF_2
$\overline{Q}_2$
T_3
Q_3
FF_3
$\overline{Q}_3$
CLK

Design of a Synchronous Modulo-10 Gray Code Counter

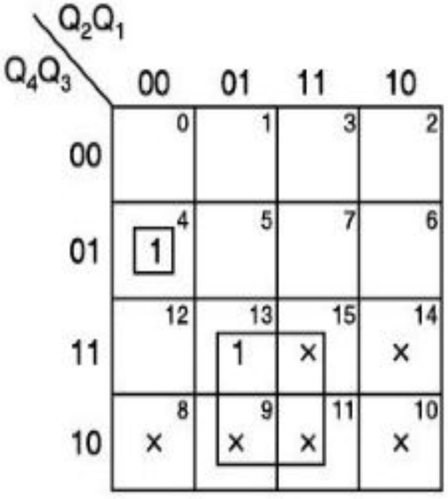


(a) State diagram

PS				NS				Required Excitations			
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	T_4	T_3	T_2	T_1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	1	0	0	1	0
0	0	1	1	0	0	1	0	0	0	0	1
0	0	1	0	0	1	1	0	0	1	0	0
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	0	1	0	1	0	0	1	0
0	1	0	1	0	1	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	0
1	1	0	0	1	1	0	1	0	0	0	1
1	1	0	1	0	0	0	0	1	1	0	1

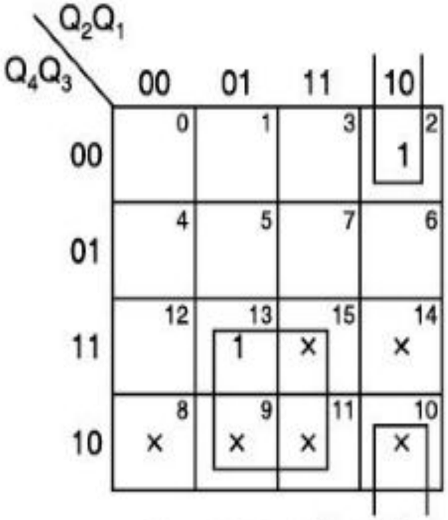
(b) Excitation table

Figure 12.29 Synchronous mod-10 Gray code counter.



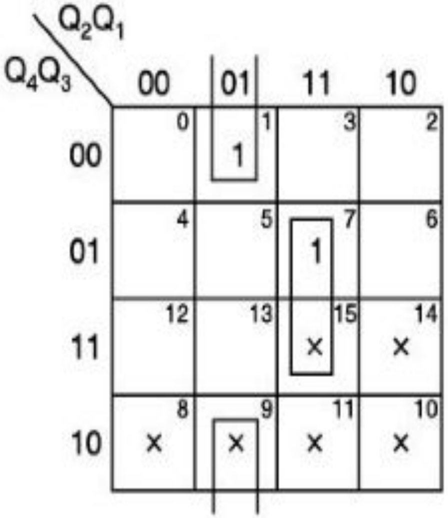
$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0	1	3	2
01	4 1	5	7	6
11	12	13 1	15 ×	14 ×
10	8 ×	9 ×	11 ×	10 ×

$T_4 = Q_4Q_1 + \bar{Q}_4Q_3\bar{Q}_2\bar{Q}_1$



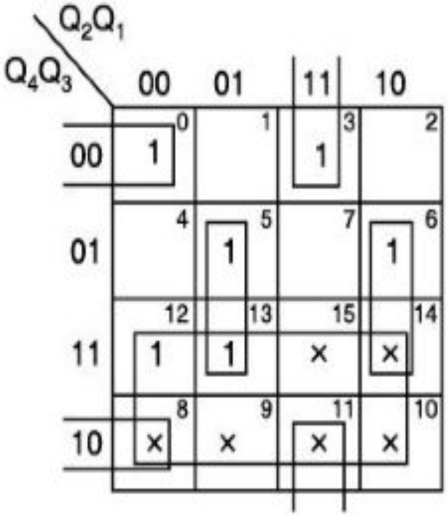
$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0	1	3	2 1
01	4	5	7	6
11	12	13 1	15 ×	14 ×
10	8 ×	9 ×	11 ×	10 ×

$T_3 = Q_4Q_1 + \bar{Q}_3Q_2\bar{Q}_1$



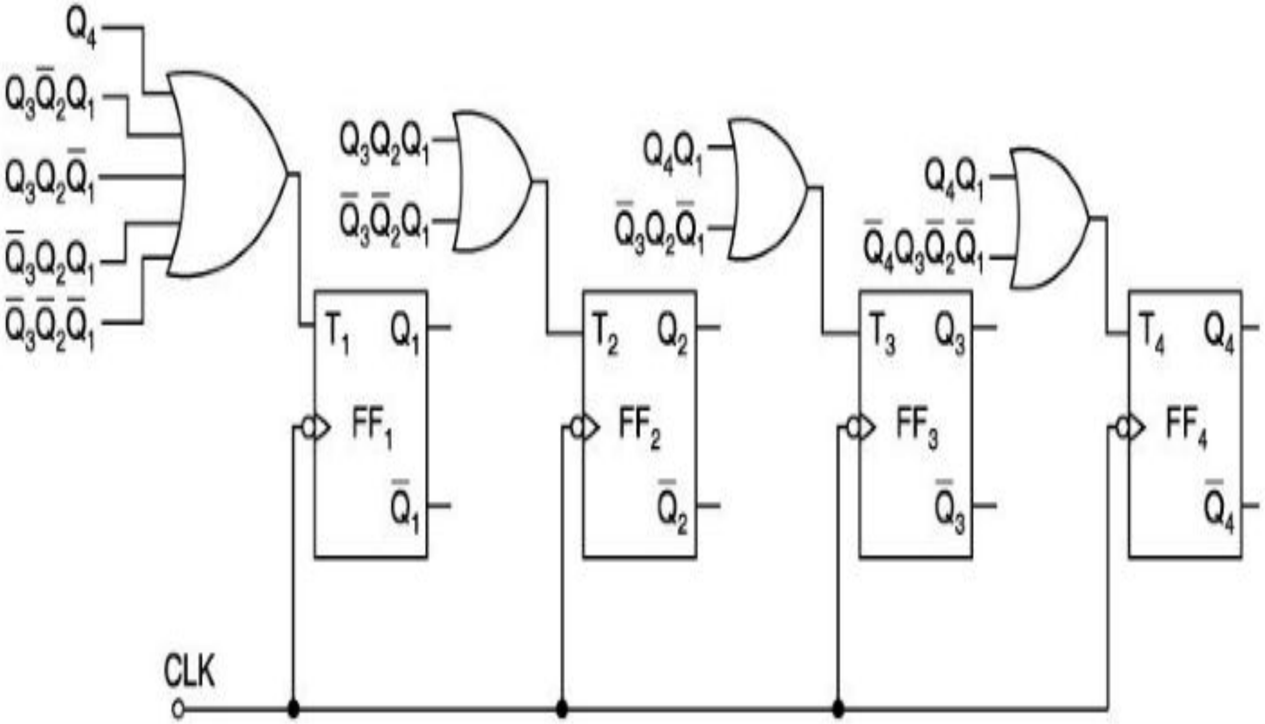
$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0	1 1	3	2
01	4	5	7 1	6
11	12	13	15 ×	14 ×
10	8 ×	9 ×	11 ×	10 ×

$T_2 = Q_3Q_2Q_1 + \bar{Q}_3\bar{Q}_2Q_1$

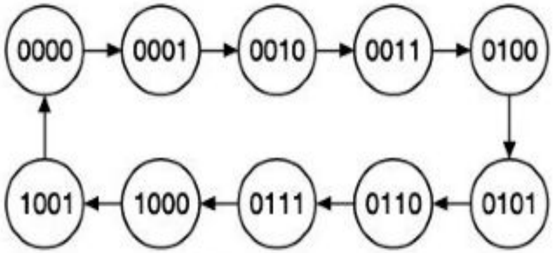


$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0 1	1	3 1	2
01	4	5 1	7	6 1
11	12 1	13 1	15 ×	14 ×
10	8 ×	9 ×	11 ×	10 ×

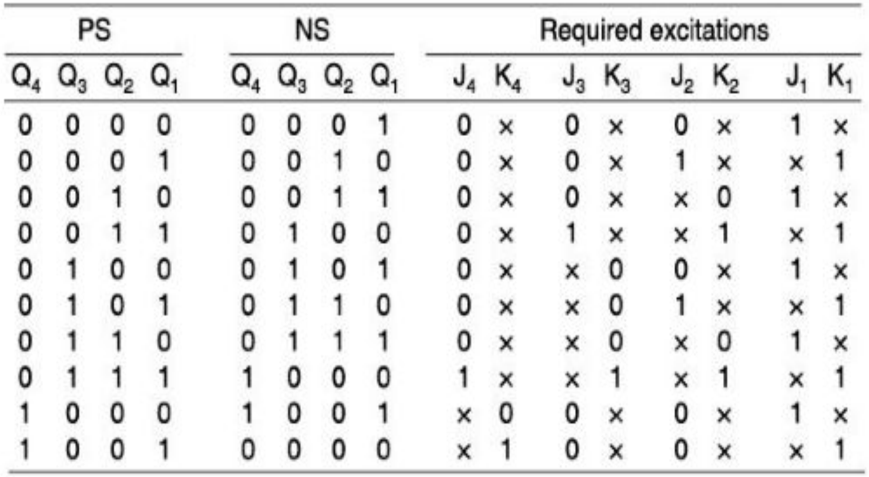
$T_1 = Q_4 + Q_3\bar{Q}_2Q_1 + Q_3Q_2\bar{Q}_1$
$+ \bar{Q}_3Q_2Q_1 + \bar{Q}_3\bar{Q}_2\bar{Q}_1$


Q_4
$Q_3\overline{Q}_2Q_1$
$Q_3Q_2\overline{Q}_1$
$\overline{Q}_3Q_2Q_1$
$\overline{Q}_3\overline{Q}_2\overline{Q}_1$
$Q_3Q_2Q_1$
$\overline{Q}_3\overline{Q}_2Q_1$
Q_4Q_1
$\overline{Q}_3Q_2\overline{Q}_1$
Q_4Q_1
$\overline{Q}_4Q_3\overline{Q}_2\overline{Q}_1$
T_1
Q_1
FF_1
$\overline{Q}_1$
T_2
Q_2
FF_2
$\overline{Q}_2$
T_3
Q_3
FF_3
$\overline{Q}_3$
T_4
Q_4
FF_4
$\overline{Q}_4$
CLK

Design of a Synchronous BCD Counter Using J-K FFs



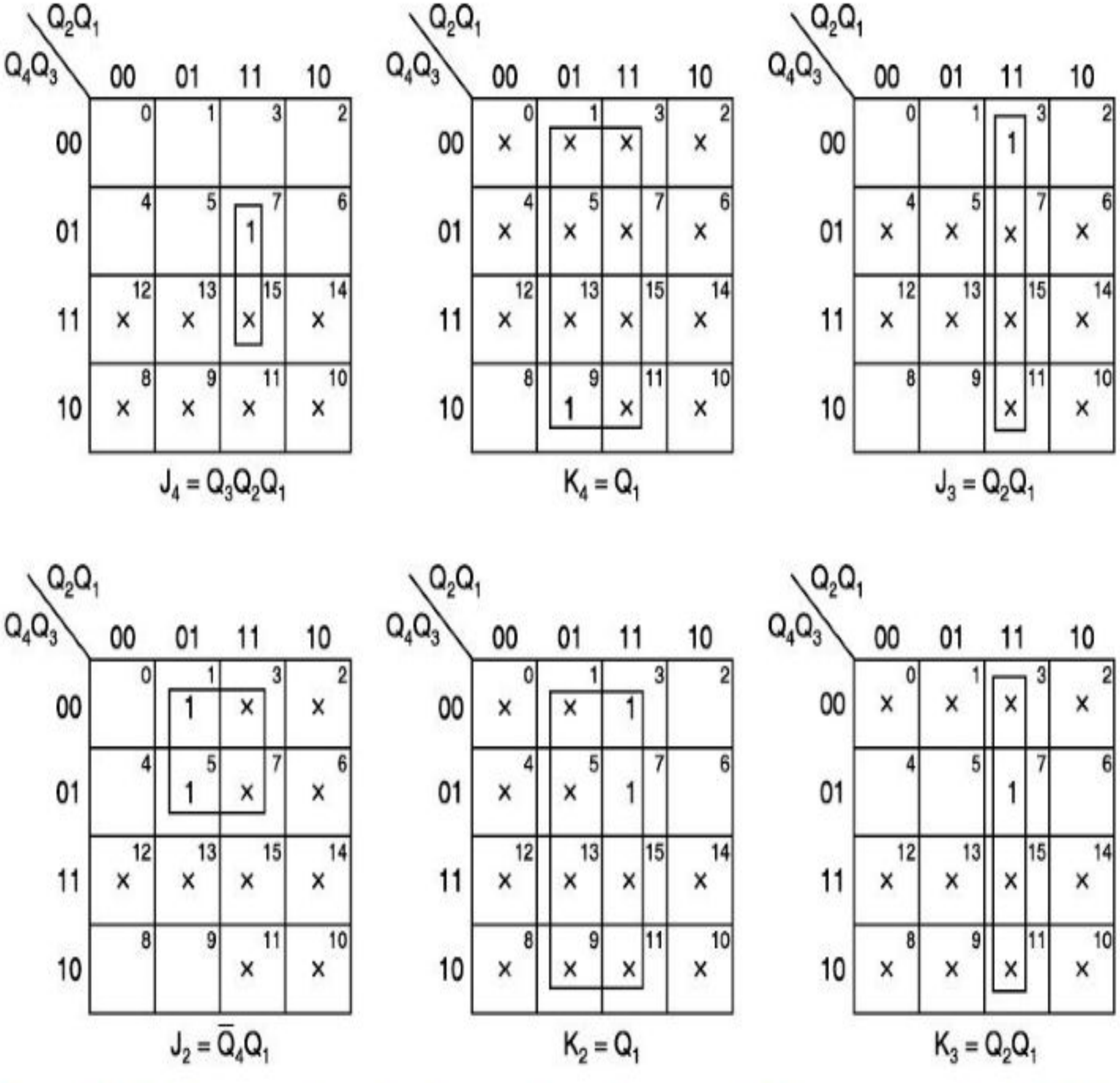
(a) State diagram



PS				NS				Required excitations							
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	×	0	×	0	×	1	×
0	0	0	1	0	0	1	0	0	×	0	×	1	×	×	1
0	0	1	0	0	0	1	1	0	×	0	×	×	0	1	×
0	0	1	1	0	1	0	0	0	×	1	×	×	1	×	1
0	1	0	0	0	1	0	1	0	×	×	0	0	×	1	×
0	1	0	1	0	1	1	0	0	×	×	0	1	×	×	1
0	1	1	0	0	1	1	1	0	×	×	0	×	0	1	×
0	1	1	1	1	0	0	0	1	×	×	1	×	1	×	1
1	0	0	0	1	0	0	1	×	0	0	×	0	×	1	×
1	0	0	1	0	0	0	0	×	1	0	×	0	×	×	1

(b) Excitation table

Figure 12.32 Synchronous BCD (mod-10) counter.



Q_4Q_3 \ Q_2Q_1	00	01	11	10
00				
01			1	
11	×	×	×	×
10	×	×	×	×

$J_4 = Q_3Q_2Q_1$

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	×	×	×	×
01	×	×	×	×
11	×	×	×	×
10		1	×	×

$K_4 = Q_1$

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00			1	
01	×	×	×	×
11	×	×	×	×
10			×	×

$J_3 = Q_2Q_1$

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00		1	×	×
01		1	×	×
11	×	×	×	×
10			×	×

$J_2 = \overline{Q}_4Q_1$

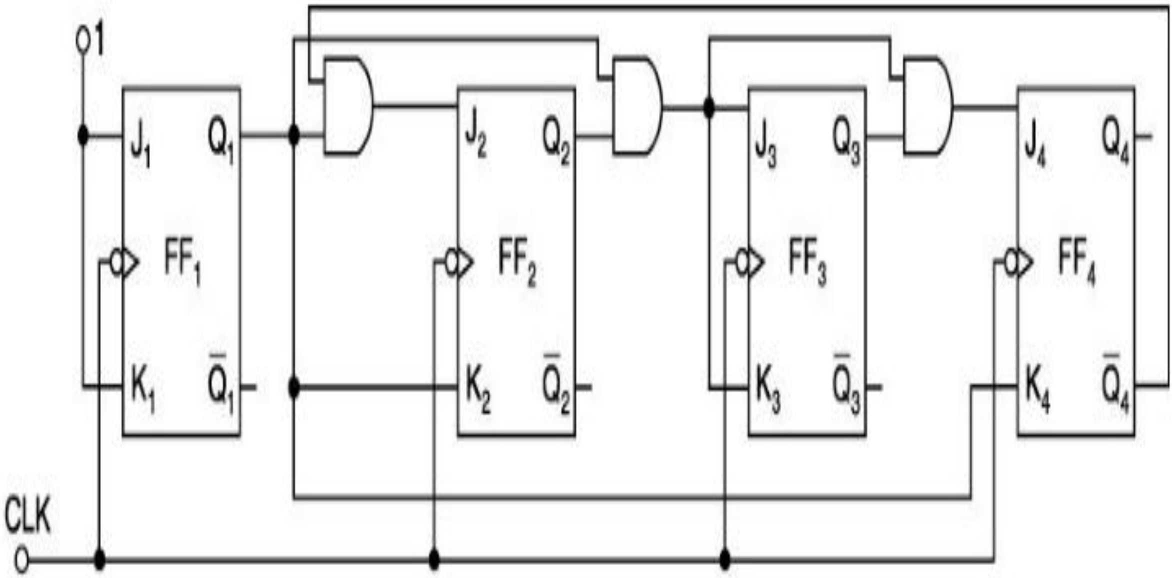
Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	×	×	1	
01	×	×	1	
11	×	×	×	×
10	×	×	×	×

$K_2 = Q_1$

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	×	×	×	×
01			1	
11	×	×	×	×
10	×	×	×	×

$K_3 = Q_2Q_1$

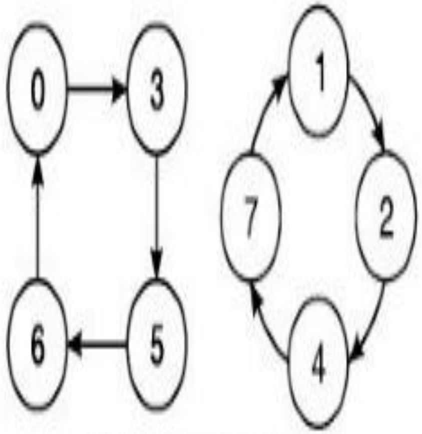
Figure 12.33 K-maps for excitations of synchronous BCD counter using J-K flip-


1
J1
Q1
FF1
K1
Q̄1
J2
Q2
FF2
K2
Q̄2
J3
Q3
FF3
K3
Q̄3
J4
Q4
FF4
K4
Q̄4
CLK

Design a type T counter that goes through states 0, 3, 5, 6, 0, 3 , 5 . ...

- Step 1. The number of flip-flops:

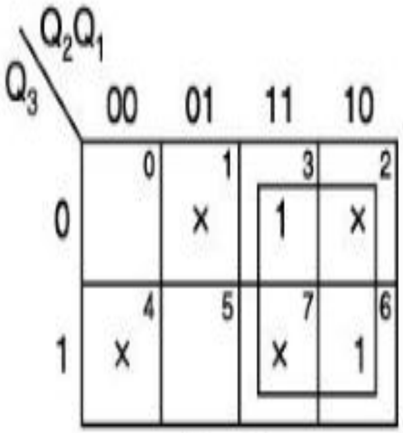
- Three FFs can have 8 states, out of which states 000, 011, 101, 110 are valid and states 001, 010, 100, 111 are invalid. The entries for excitations corresponding to invalid states are don’t cares.



(a) State diagram

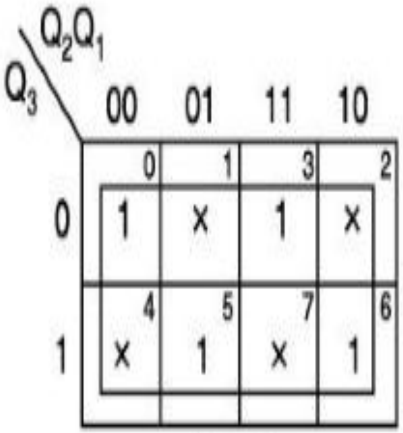
PS			NS			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	0
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	1	0

(b) Excitation table



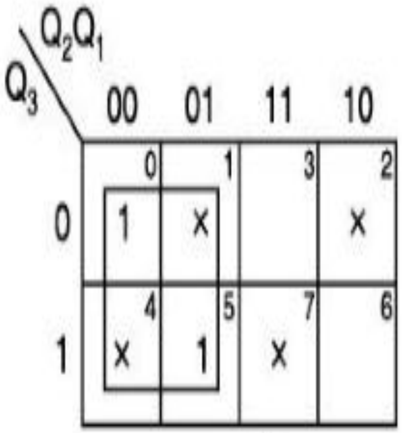
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	^{0}	^{1} ×	^{3} 1	^{2} ×
1	^{4} ×	^{5}	^{7} ×	^{6} 1

$T_3 = Q_2$



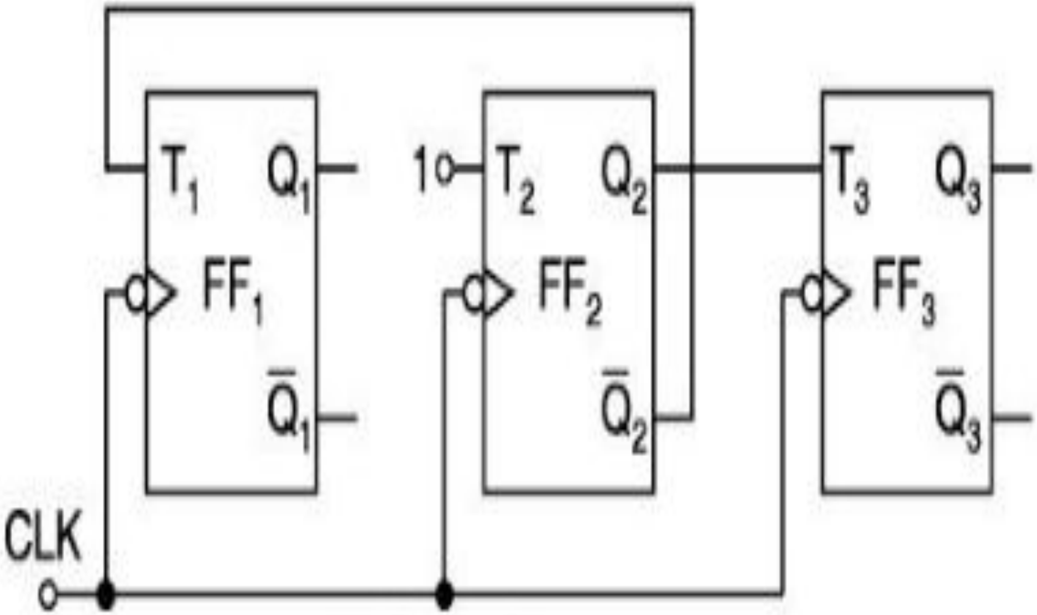
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	^{0} 1	^{1} ×	^{3} 1	^{2} ×
1	^{4} ×	^{5} 1	^{7} ×	^{6} 1

$T_2 = 1$



$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	^{0} 1	^{1} ×	^{3}	^{2} ×
1	^{4} ×	^{5} 1	^{7} ×	^{6}

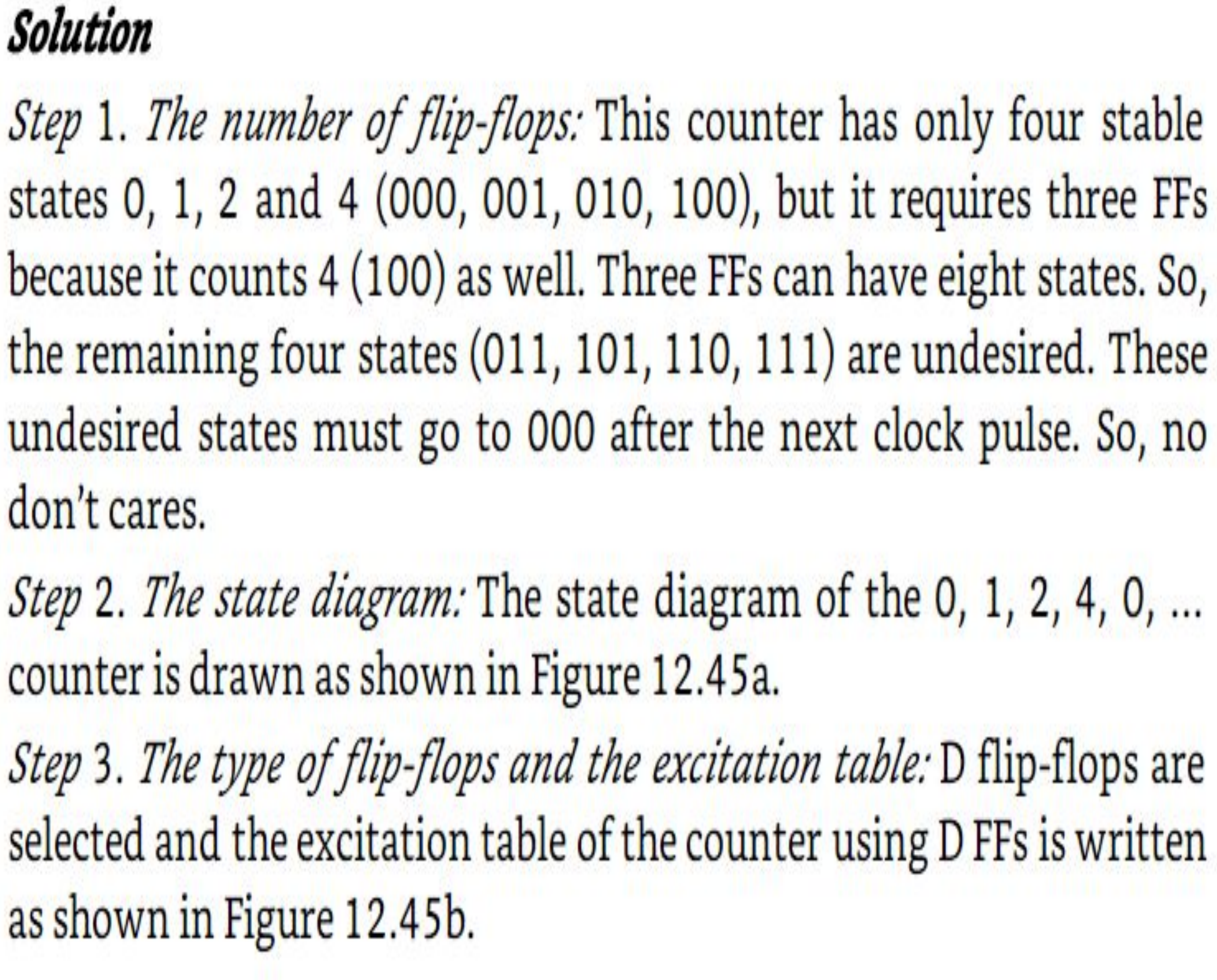
$T_1 = \bar{Q}_2$



(a) Logic diagram

- Question

- Design a type D counter that goes through states 0, 1, 2, 4, 0, … . The undesired (unused) states must always go to zero (000) on the next clock pulse.

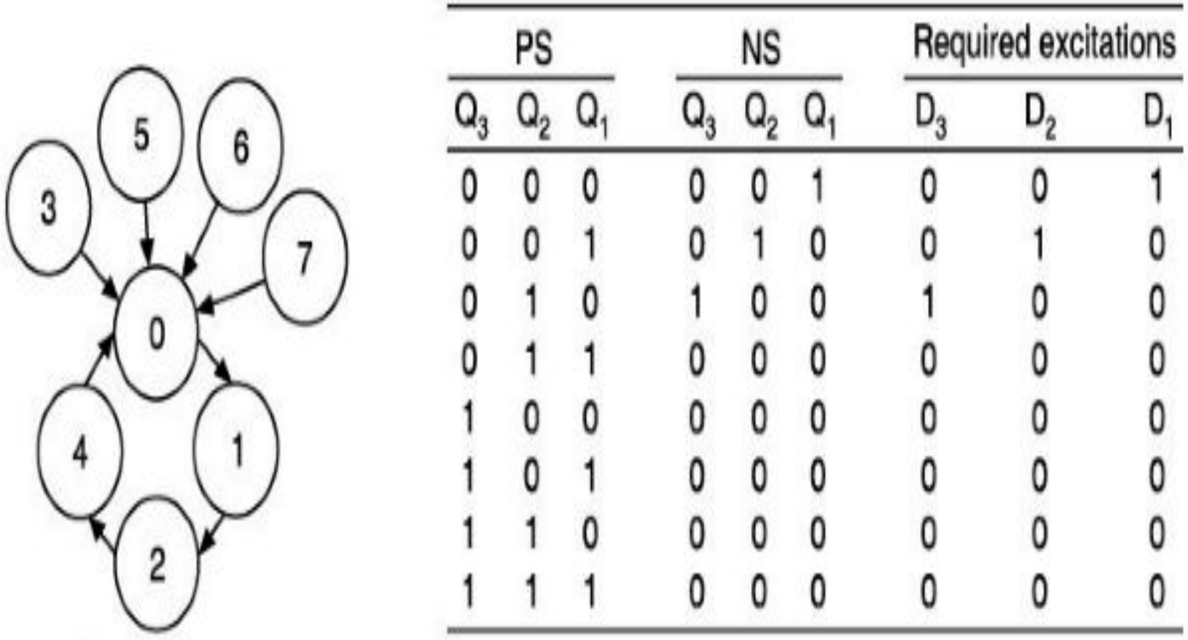


Solution

Step 1. *The number of flip-flops:* This counter has only four stable states 0, 1, 2 and 4 (000, 001, 010, 100), but it requires three FFs because it counts 4 (100) as well. Three FFs can have eight states. So, the remaining four states (011, 101, 110, 111) are undesired. These undesired states must go to 000 after the next clock pulse. So, no don't cares.

Step 2. *The state diagram:* The state diagram of the 0, 1, 2, 4, 0, ... counter is drawn as shown in Figure 12.45a.

Step 3. *The type of flip-flops and the excitation table:* D flip-flops are selected and the excitation table of the counter using D FFs is written as shown in Figure 12.45b.



(a) State diagram

PS			NS			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	D_3	D_2	D_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	1	0	0	1	0	0
0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0

(b) Excitation table

Figure 12.45 Example 12.5: 0, 1, 2, 4, 0, ... counter.

Step 4. *The minimal expressions:* From the excitation table we can see that no minimization is possible. So the expressions for excitations read from the excitation table are:

D3 = ◌3Q2◌1; D2 = ◌3◌2Q1; D1 = ◌3◌2◌1

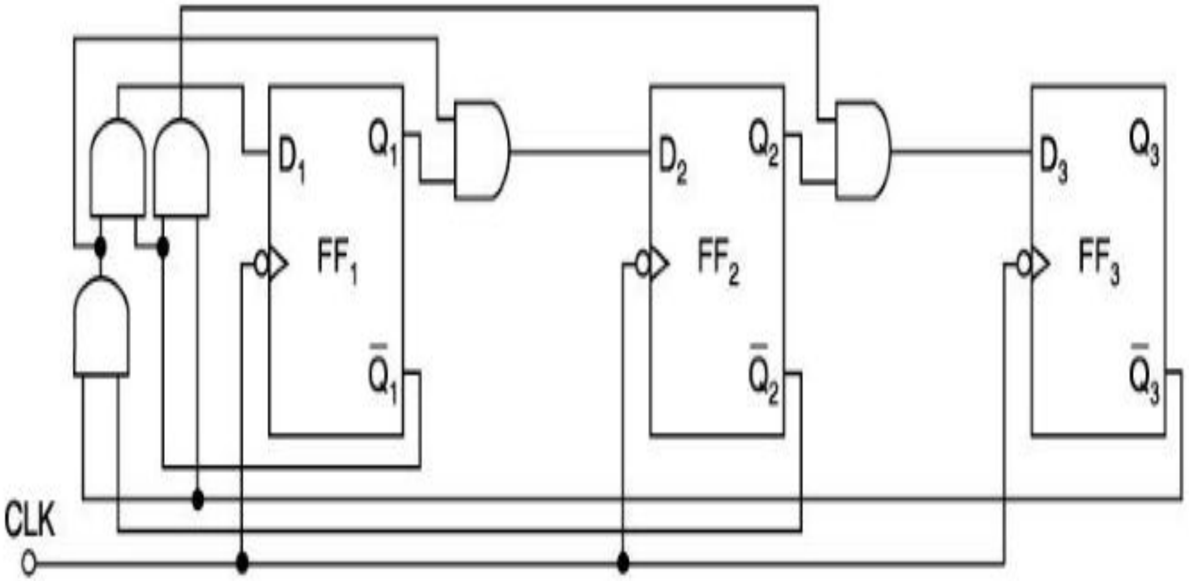
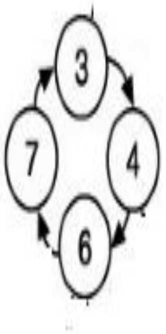
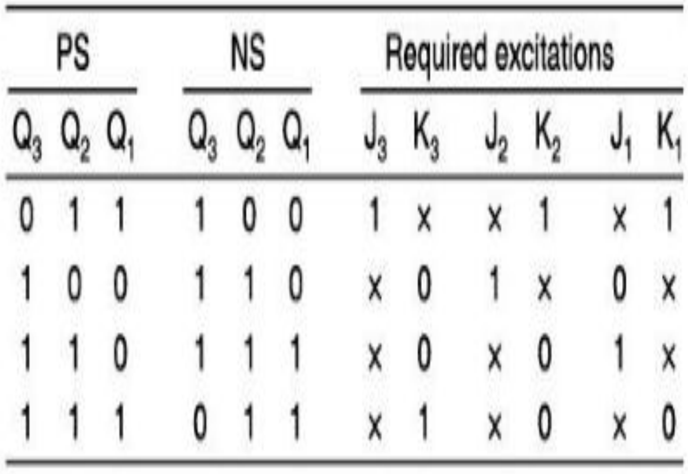


Figure 12.46 Example 12.5: Logic diagram of type D counter that goes through states 0, 1, 2, 4, 0, … .

Design a J-K counter that goes through states 3, 4, 6, 7 and 3…..

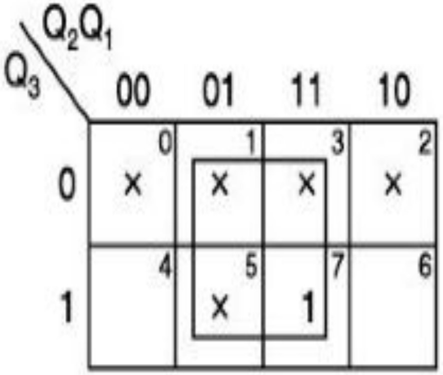


(a) State diagram



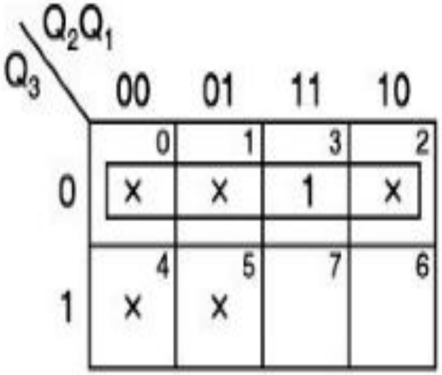
PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	1	0	x	0	1	x	0	x
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	1	1	x	1	x	0	x	0

(b) Excitation table



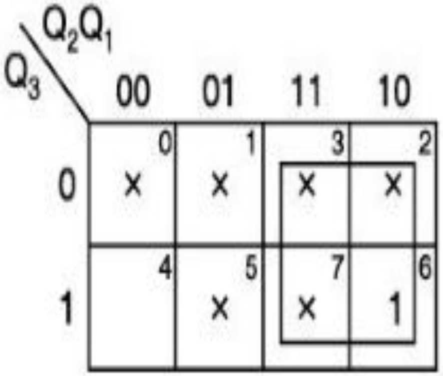
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0 ×	1 ×	3 ×	2 ×
1	4	5 ×	7 1	6

$K_3 = Q_1$



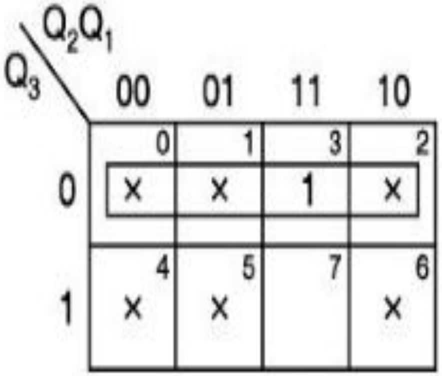
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0 ×	1 ×	3 1	2 ×
1	4 ×	5 ×	7	6

$K_2 = \overline{Q}_3$



$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0 ×	1 ×	3 ×	2 ×
1	4	5 ×	7 ×	6 1

$J_1 = Q_2$



$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	0 ×	1 ×	3 1	2 ×
1	4 ×	5 ×	7	6 ×

$K_1 = \overline{Q}_3$

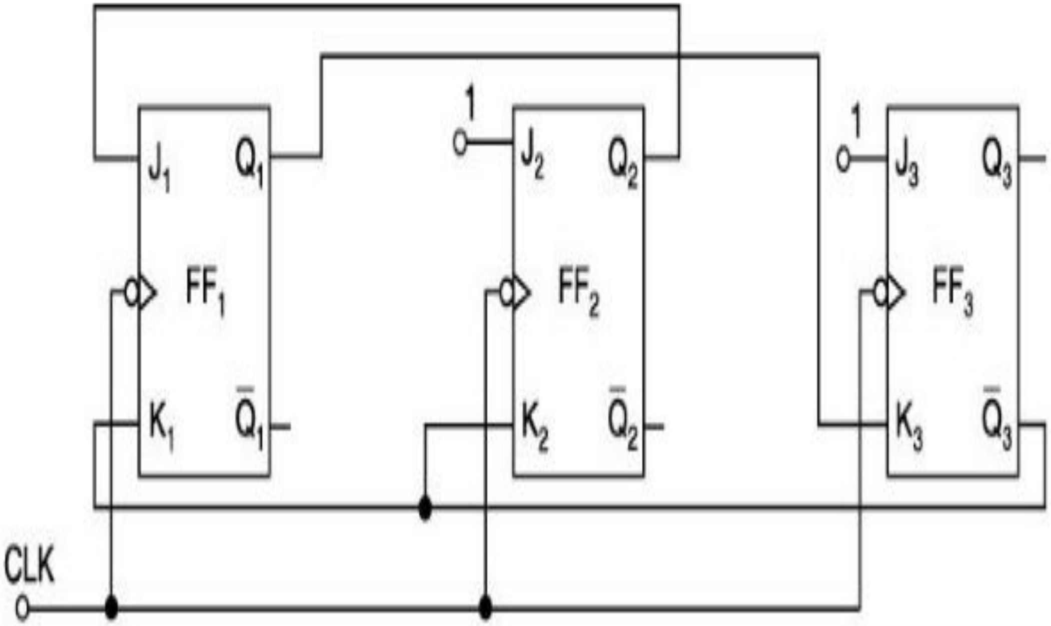


Figure 12.49 Example 12.3: Logic diagram of the J-K counter that goes through states 3, 4, 6, 7, 3, ...

Shift Register Counters

- One of the applications of shift registers is that they can be arranged to form several types of counters.

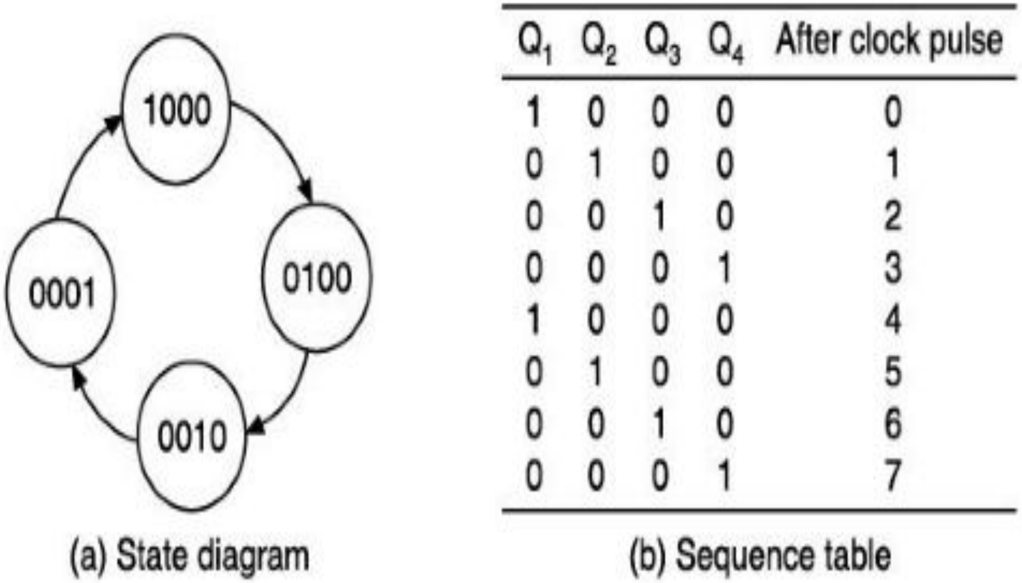
- Shift register counters are obtained from serial-in, serial-out shift registers by providing feedback from the output of the last FF to the input of the first FF.

- The most widely used shift register counters are

- 1. Ring Counter

- 2. Twisted Ring Counter (Johnson Counter)

Ring Counter



(a) State diagram

Q_1	Q_2	Q_3	Q_4	After clock pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

(b) Sequence table

Figure 12.59 State diagram and sequence table of a 4-bit ring counter.

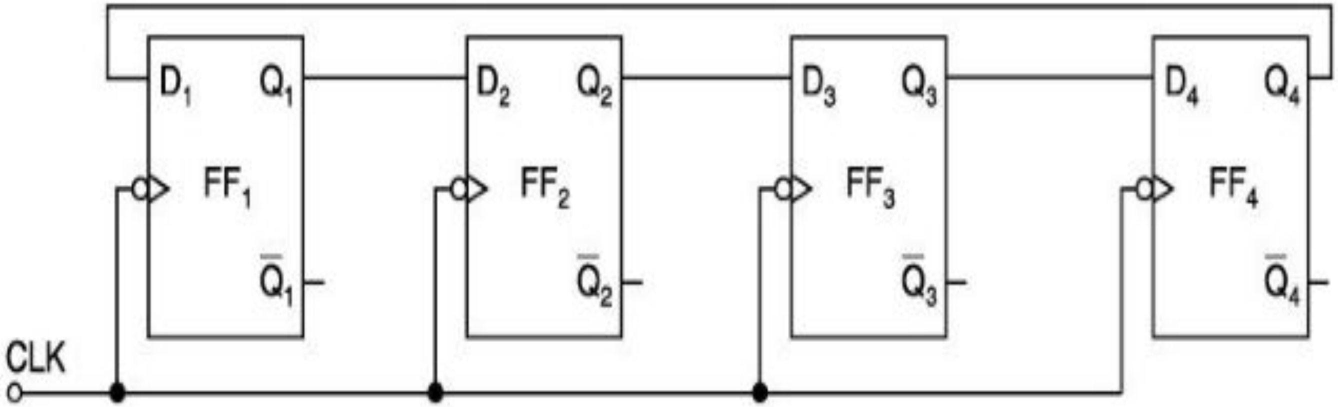


Figure 12.57 Logic diagram of a 4-bit ring counter using D flip-flops.

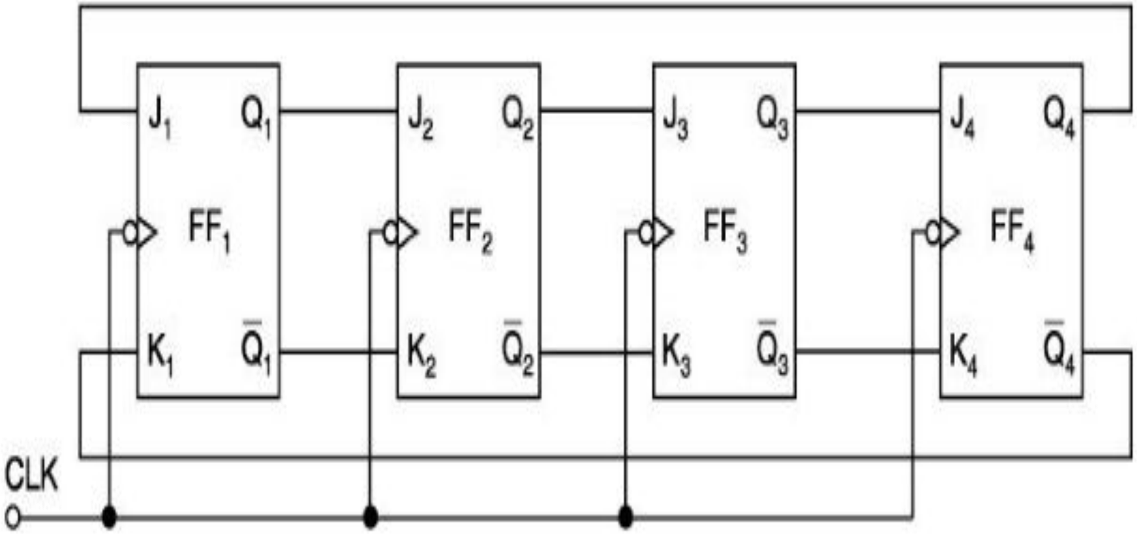


Figure 12.58 Logic diagram of a 4-bit ring counter using J-K flip-flops.

- Initially, the first FF is preset to a 1.

- So, the initial state is 1000, i.e. Q1 = 1, Q2 = 0, Q3 = 0 and Q4 = 0.

- After each clock pulse, the contents of the register are shifted to the right by one bit and Q4 is shifted back to Q1.

- The sequence repeats after four clock pulses.

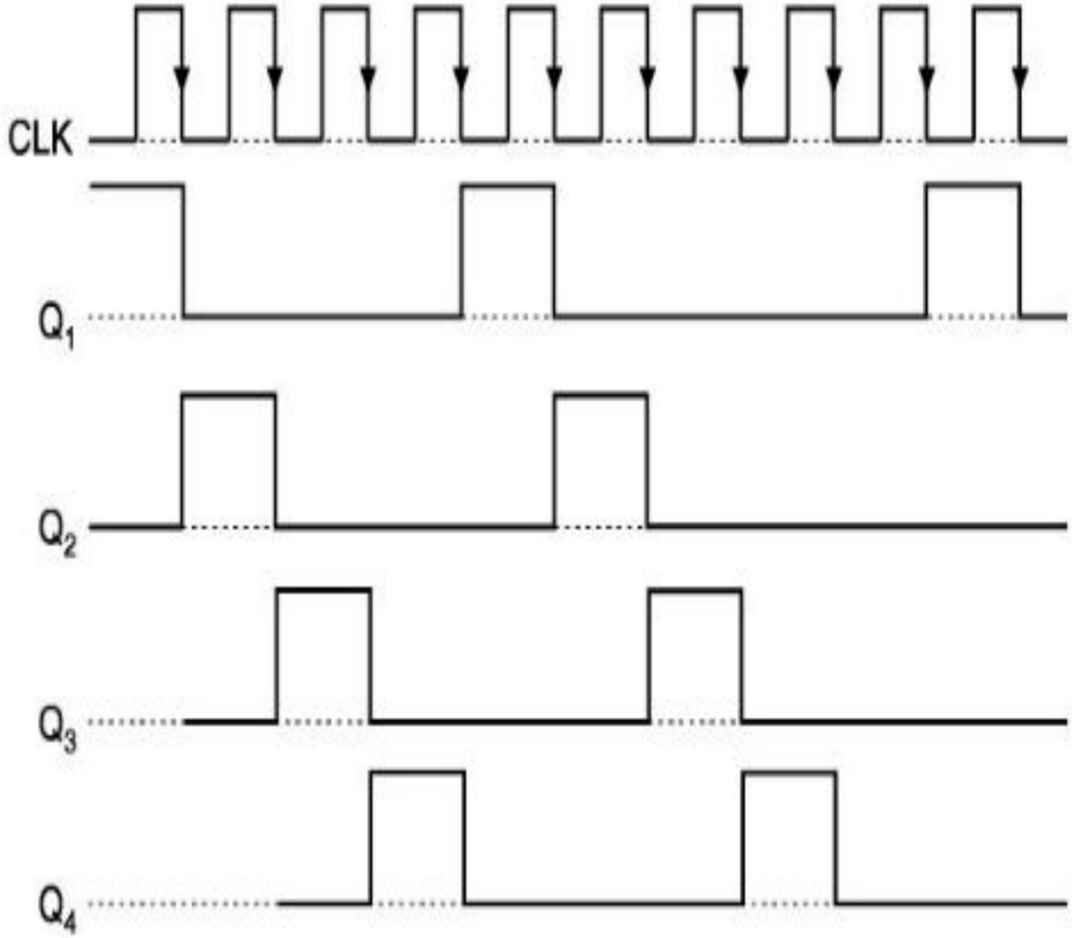
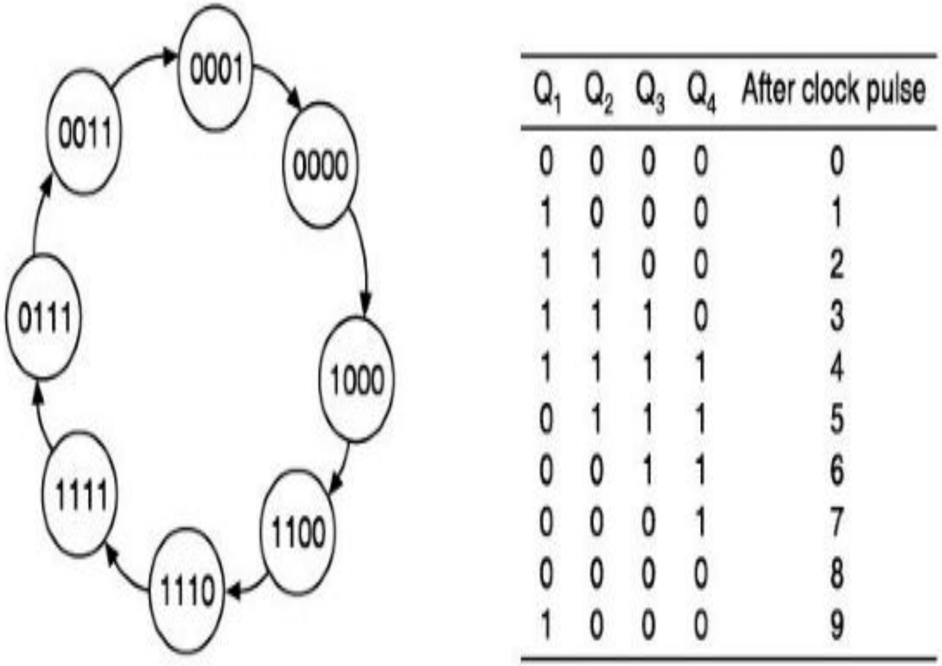


Figure 12.60 Timing diagram of a 4-bit ring counter.

Twisted Ring Counter (Johnson Counter)

- This counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF.

- The Q output of each stage is connected to the D input of the next stage, but the output of the la $\overline{Q}$ stage is connected to the D input of first stage, therefore, the name twisted ring counter.



(a) State diagram

Q_1	Q_2	Q_3	Q_4	After clock pulse
0	0	0	0	0
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9

(b) Sequence table

Figure 12.63 State diagram and sequence table of a twisted ring counter.

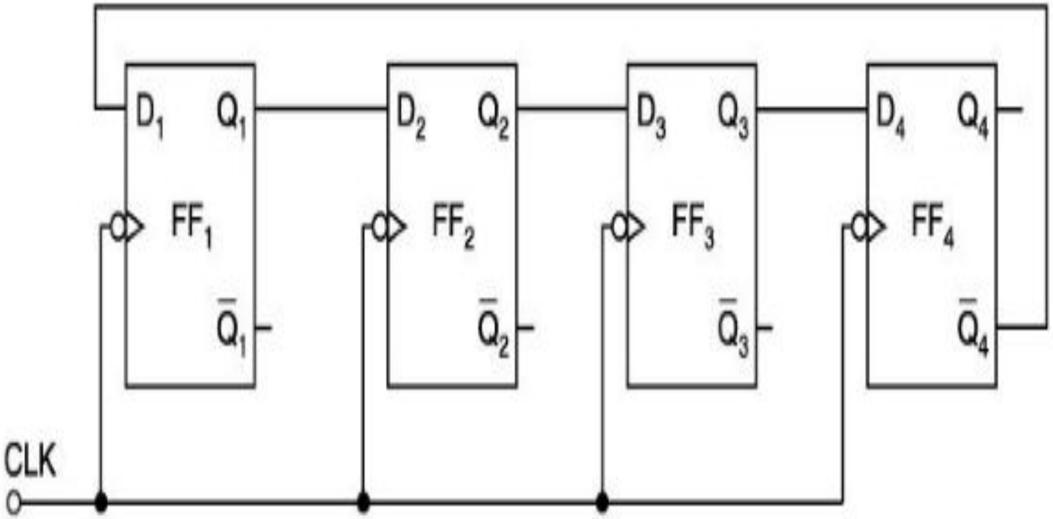


Figure 12.61 Logic diagram of a 4-bit twisted ring counter using D flip-flops.

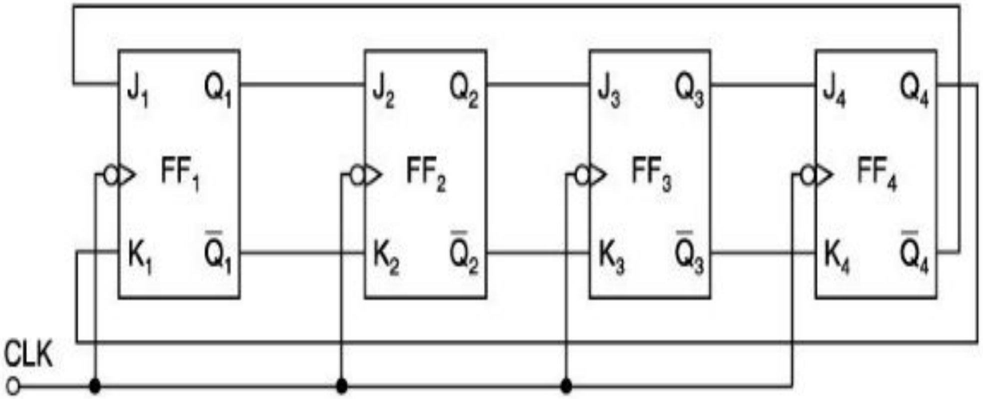


Figure 12.62 Logic diagram of a 4-bit twisted ring counter using J-K flip-flops.

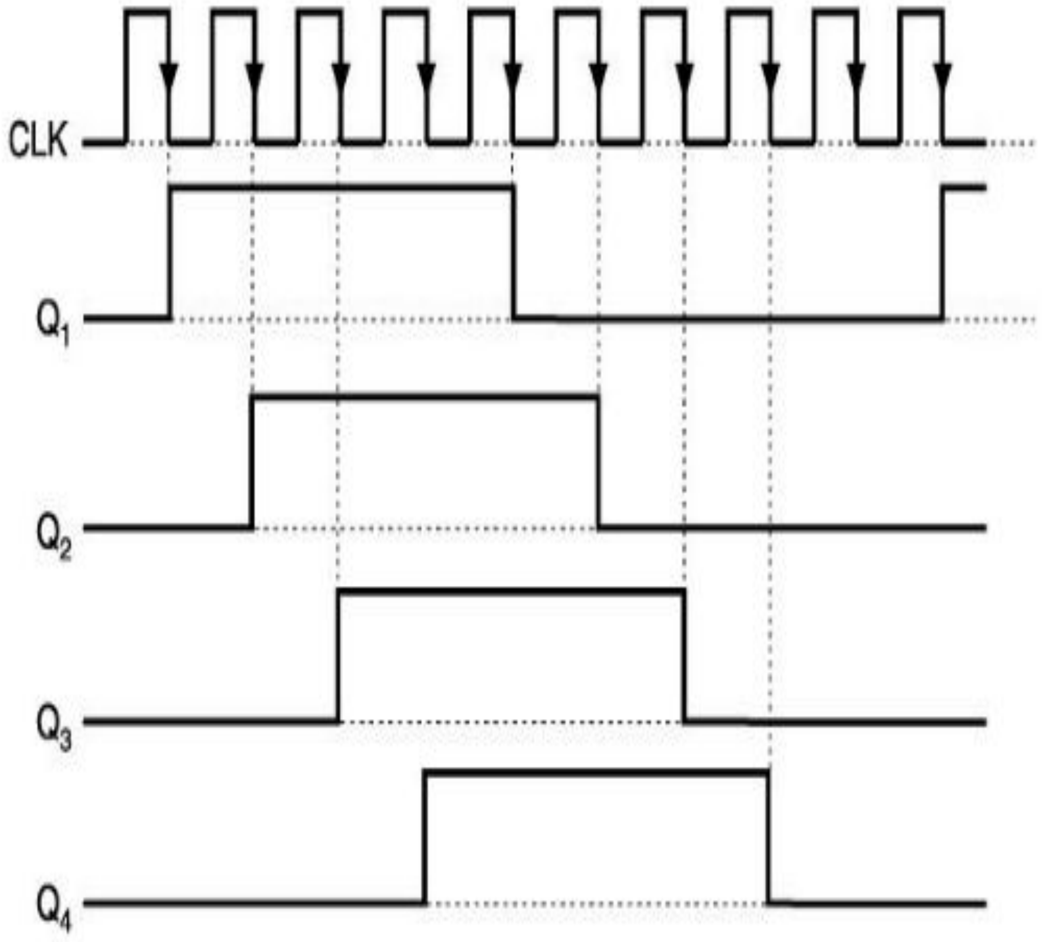


Figure 12.64 Timing diagram of a 4-bit twisted ring counter.