Chapter 5 Sequential Circuit

Mr. Kalpesh Jadav

Assistant Prof

EC department , PIET

Difference between Combinational and sequential circuit

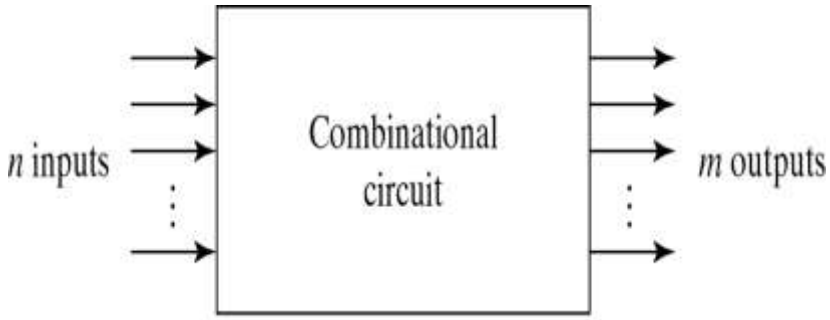
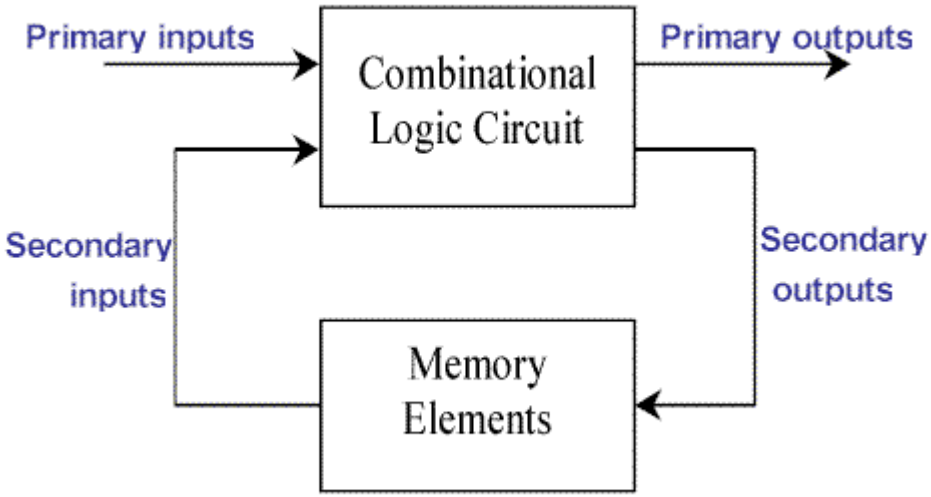


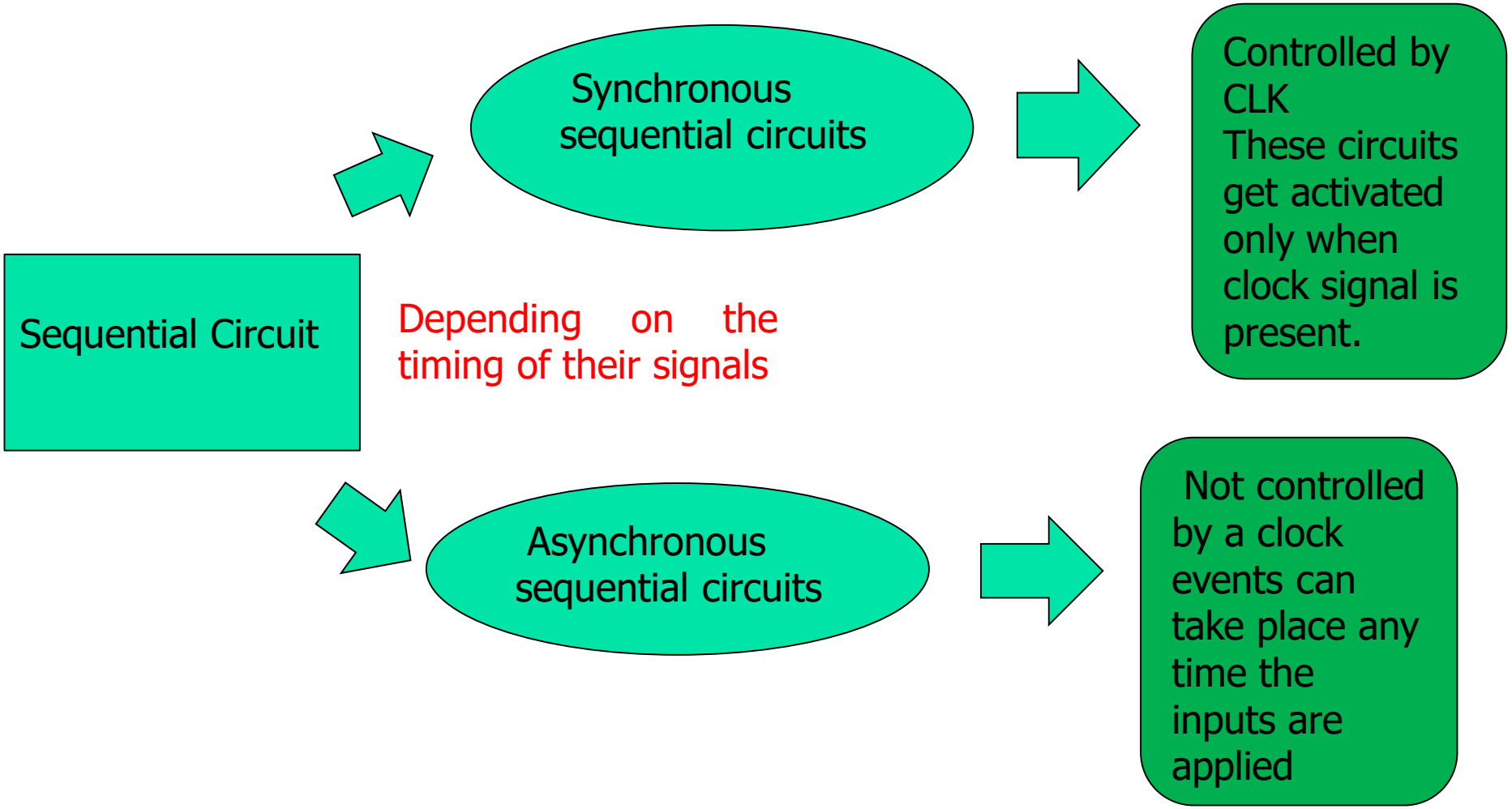
Fig. 4-1 Block Diagram of Combinational Circuit

Output is only dependant
on present inputs
Ex. Adder



Output is only dependant
on present inputs as well
as previous output
Ex. Counter
3+1=4
4+1=5 etc.

Classification of Sequential Circuit

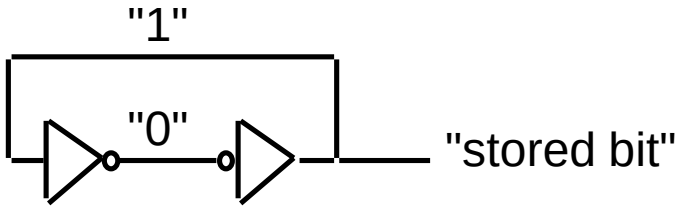


Comparison Between Synchronous and Asynchronous Sequential Circuit

Synchronous	Asynchronous
In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.	In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
In synchronous circuits, memory elements are clocked FFs	In asynchronous circuits, memory elements are either un-clocked FFs or time delay elements.
The maximum operating speed of the clock depends on time delays involved.	Since the clock is not present, asynchronous circuits can operate faster than synchronous circuits
They are easier to design.	More difficult to design.

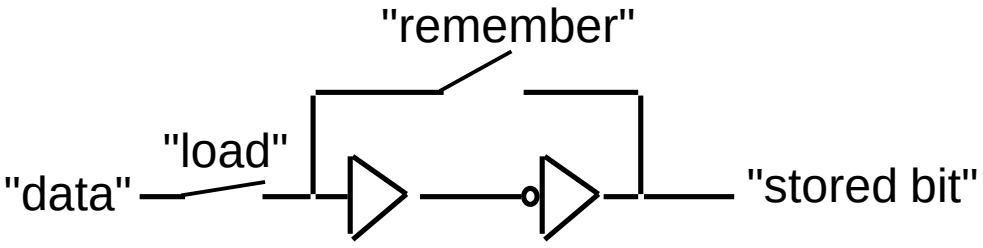
How do we store info? Feedback

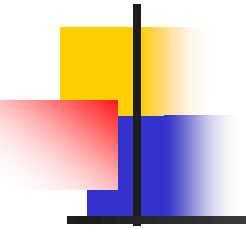
- Two inverters can hold a bit
  - As long as power is applied



Storing a new memory

- Temporarily break the feedback path





Latches and Flip-Flops

- The two most popular varieties of storage cells used to build sequential circuits are: latches and flip-flops.
  - **Latch:** **level sensitive** storage element
  - **Flip-Flop:** **edge triggered** storage element

- Common examples of latches:
  S-R latch, \S-\R latch, D latch (= gated D latch)

- Common examples of flip-flops:
  D-FF, D-FF with enable, Scan-FF, JK-FF, T-FF

LATCHES AND FLIP-FLOPS

Flip-flop :

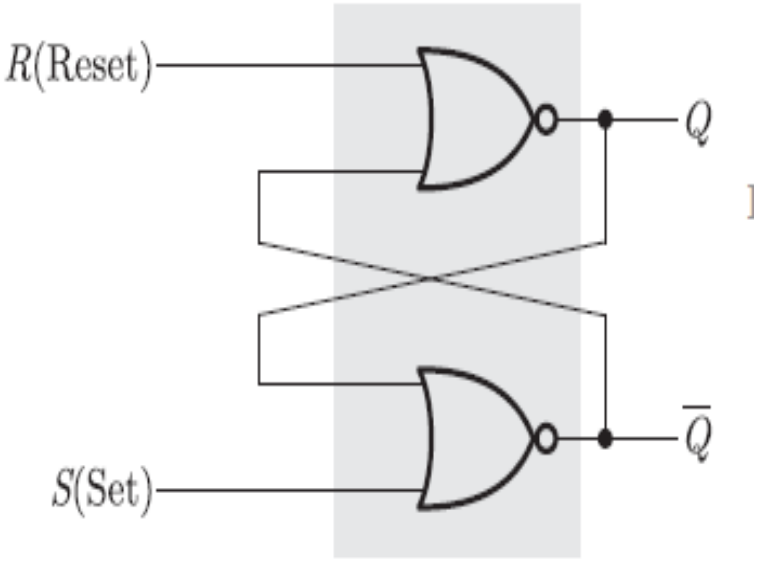
- It is an electronic circuit or device which is used to store a data in binary form.
- Flip-slop is an one-bit memory device and it can store either 1 or 0.
- Flip-flops is a sequential device that changes its output only when a clocking signal is changing.

Latch:

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

S - *R* Latch using NOR Gates

X Y	NOR
0 0	1
0 1	0
1 0	0
1 1	0

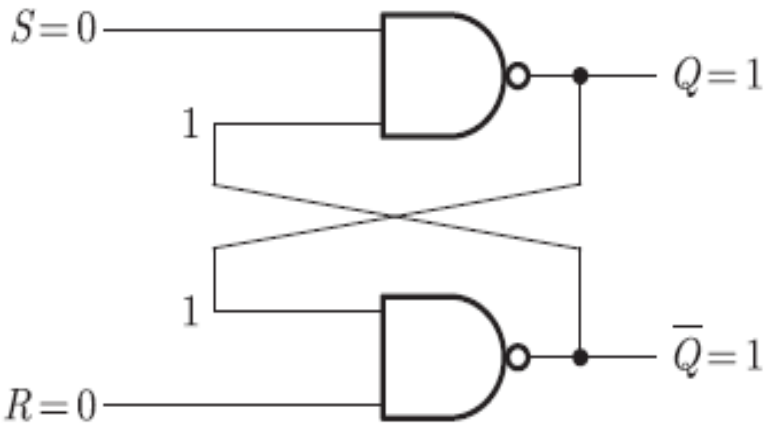


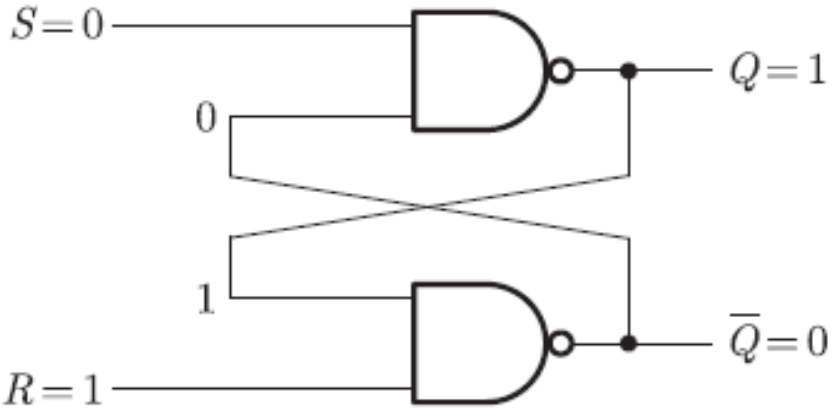
S stands for set, it means that when *S* is 1, it stores 1. Similarly, *R* stands for reset and if *R* = 1, flip-flop reset and it's output will be 0. This circuit is called as NOR gate latch or *S R* latch.

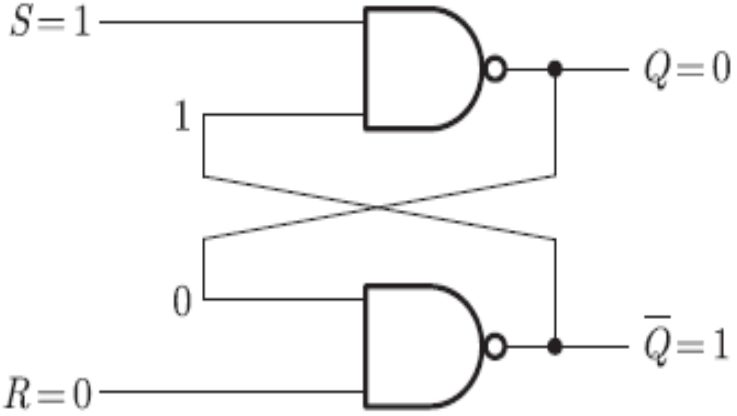
Inputs		Output	Operation Mode
S	*R*	Q_{n+1}	
0	0	Q_n	No Change
0	1	0	Reset
1	0	1	Set
1	1	?	Forbidden

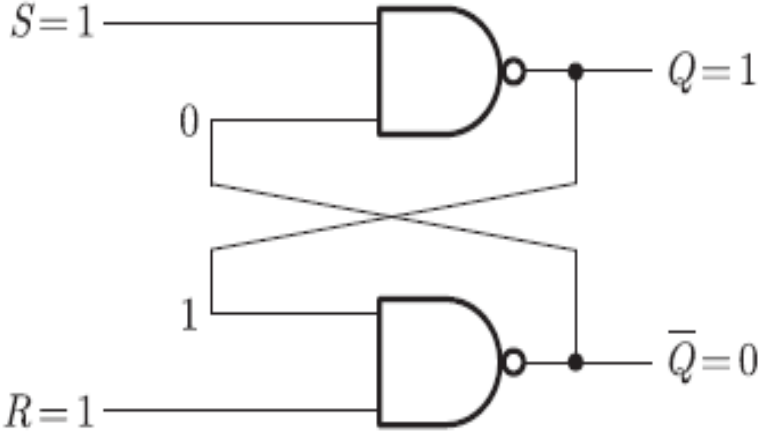
Q_n represents the state of the flip-flop before applying the inputs (i.e. the present state of the flip-flop). Q_{n+1} represents the state of the flip-flop after the application of the inputs (i.e. the next state of the flip-flop).

S-*R* Latch using NAND Gates

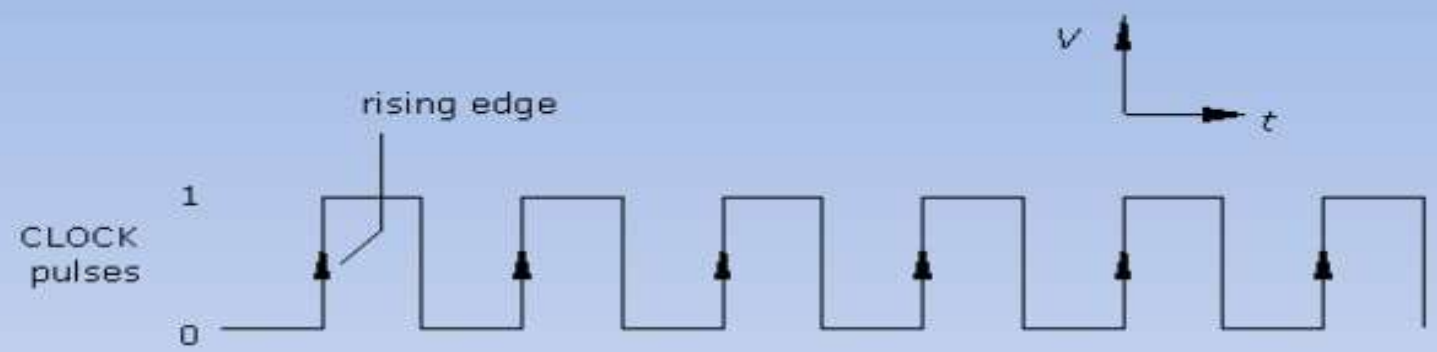


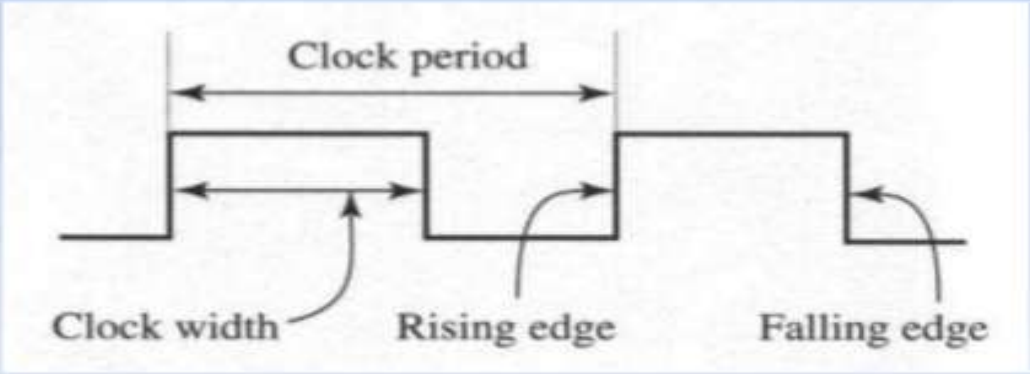






What is Clock pulse?





Triggering of Flip flop

The process of applying the clock signal to change the state of a flip-flop is called triggering.

Level Triggering

any changes at the input during the time the clock is active (HIGH) are reflected at the output as per its truth table. Since the flip-flop changes its state only when clock pulse is HIGH, this is also referred to as positive level triggered flip-flop.

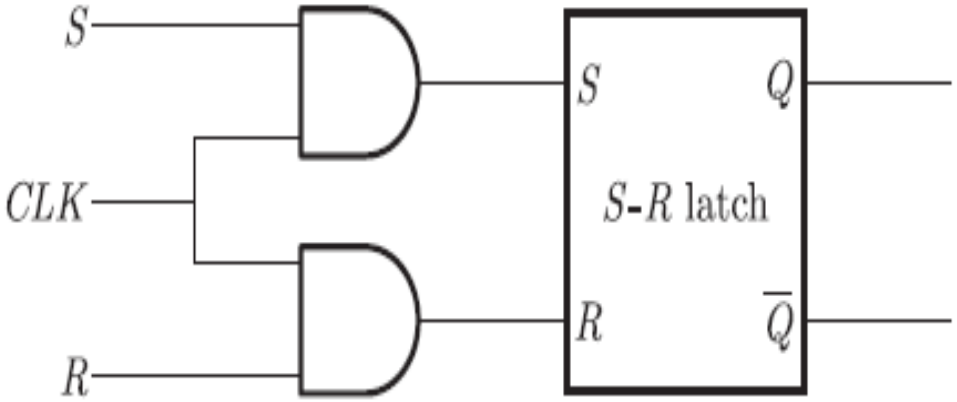
(a) Positive level (b) Negative level

Edge Triggering

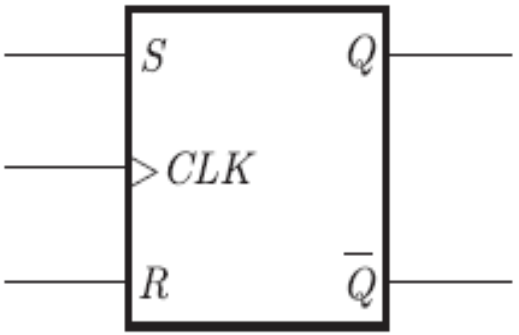
the input signals affect the flip-flop only if they are present at the positive going or negative going edge of the clock pulse.

(a) Positive edge (b) Negative edge

S-R Flip-Flop

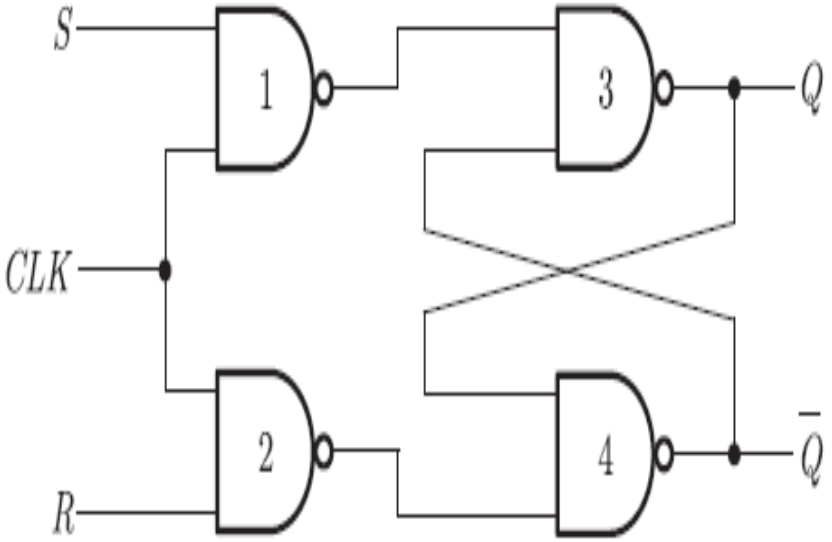


Logic diagram of Clocked S-R flip-flop

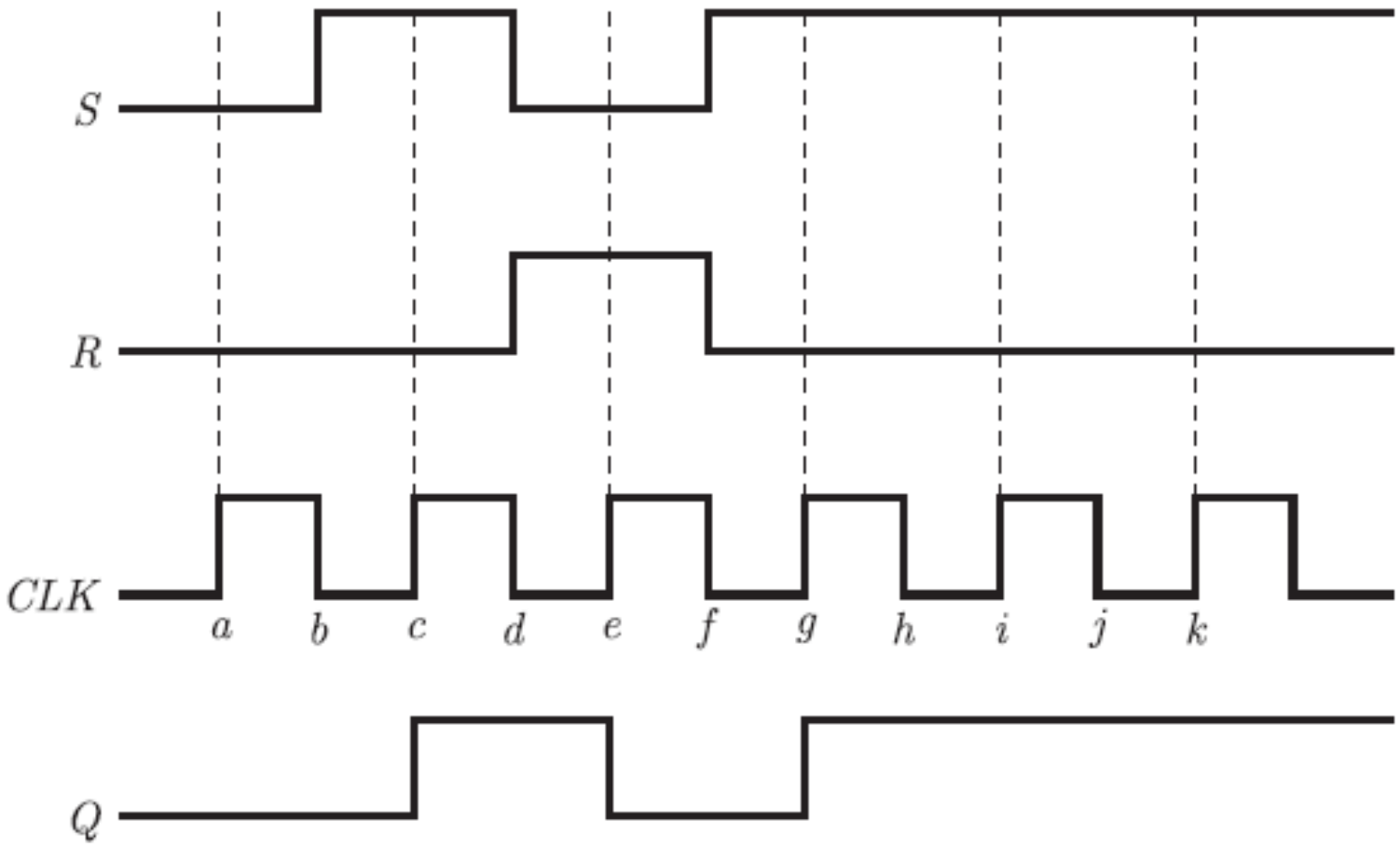


Block diagram of a clocked S-R flip-flop

CLK	Inputs		Output	Operation Mode
	S	*R*	Q_{n+1}	
1	0	0	Q_n	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	?	Invalid (Forbidden)
0	X	X	Q_n	



Timing diagram

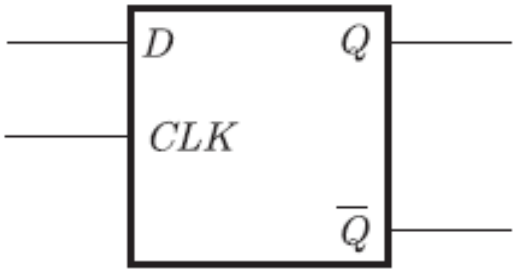


Conti.

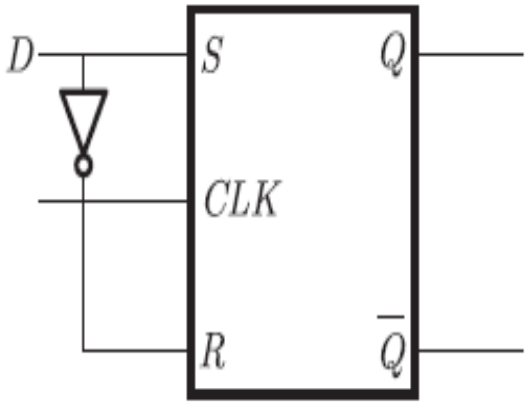
1. Initially all the inputs are zero and the output Q is 0.
2. At the time when first clock pulse goes HIGH (at point a), inputs are $S=0$, $R=0$, hence the flip-flop does not change state on this transition and output will remain in same state i.e., $Q=0$.
3.. Again, when the clock pulse goes HIGH at point c, inputs are $S=1$ and $R=0$. This cause flip-flop to go into SET state i.e., $Q=1$.
4. At point e clock pulse goes HIGH and inputs are $S=0$ and $R=1$, hus, the output of flip-flop will be reset to $Q=0$ state.

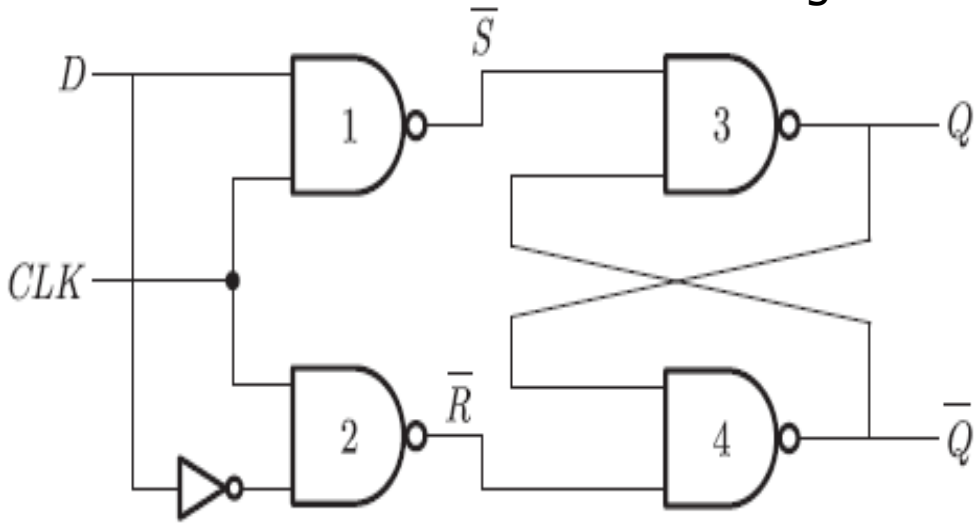
Similarly we can obtain output of flip-flop at all clock transitions. The condition $R=1$ and $S=1$ will not be used here.

D-Flip Flop



Block diagram of D-flip flop



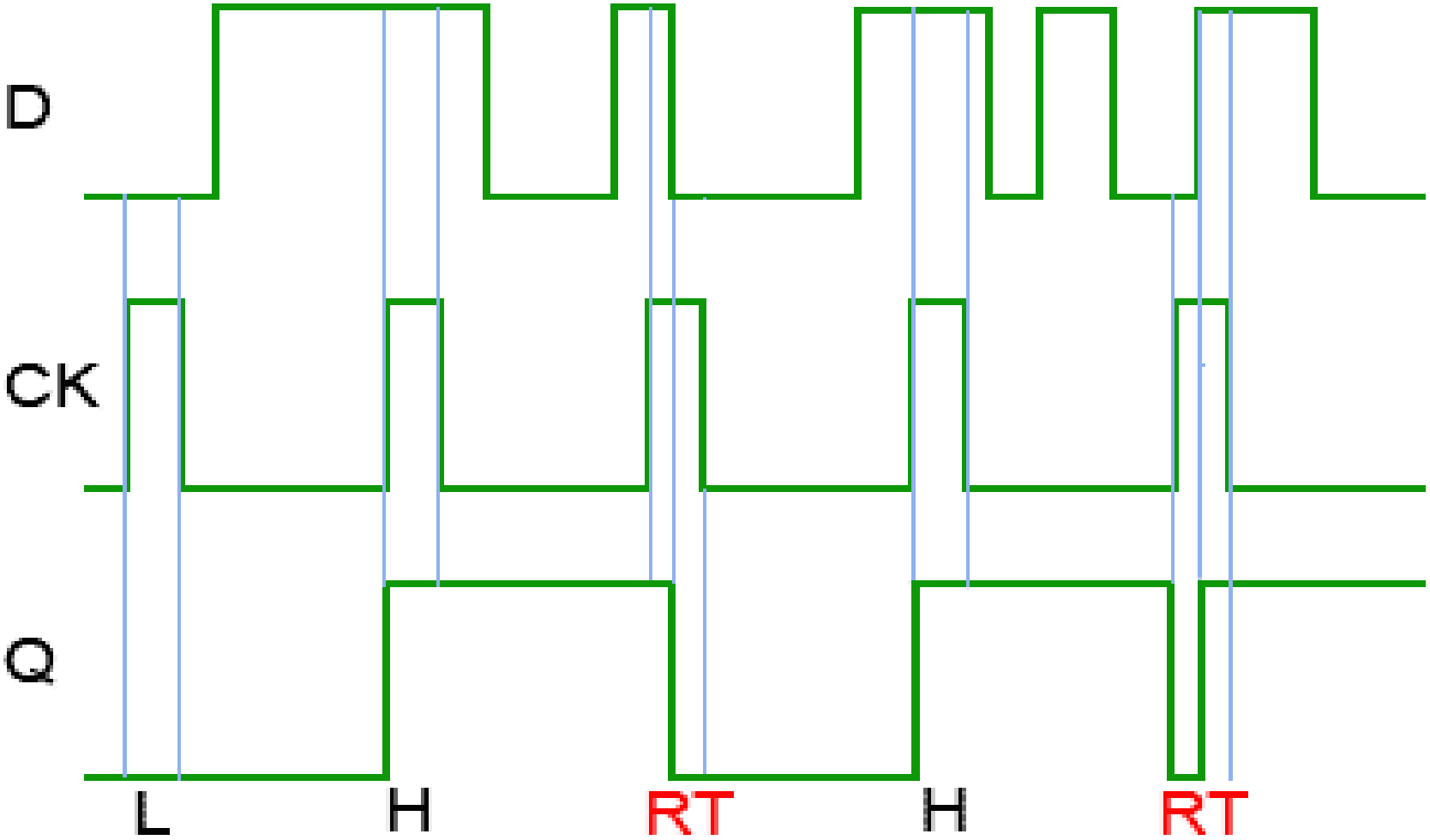


(a) Clocked D-flip flop (b) using NAND gate

CLK	Input D	Output Q_{n+1}
1	0	0
1	1	1
0	X	No change

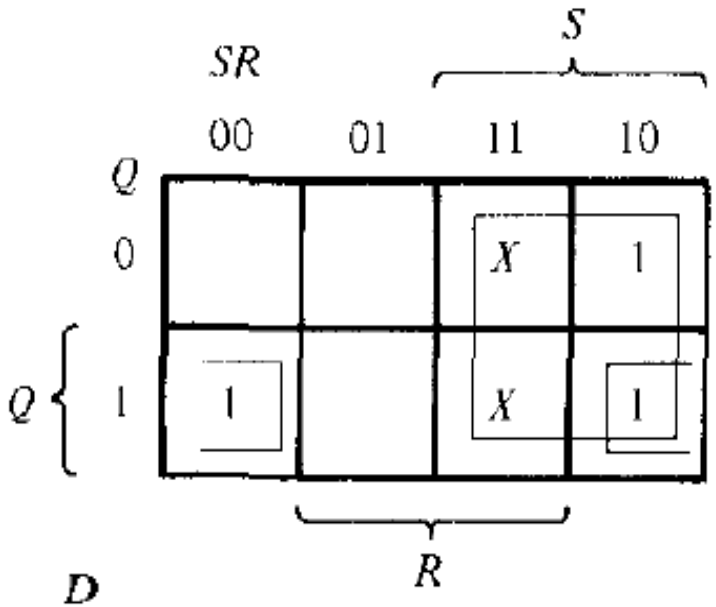
Truth Table of D-flip flop

Timing diagram of D-flip flop



Characteristic equation of SR FF and D FF

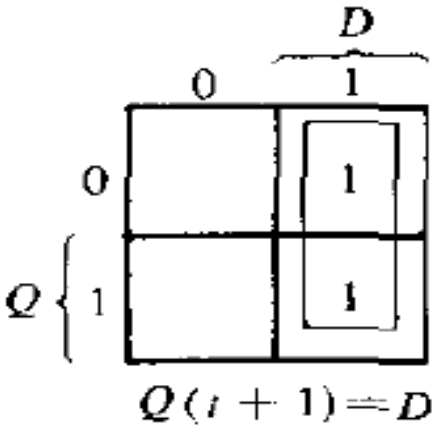
CLK	Inputs		Output	Operation Mode
	S	R	Q_{n+1}	
1	0	0	Q_n	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	?	Invalid (Forbidden)
0	X	X	Q_n	



$Q(t+1) = S + R'Q$

$SR = 0$

CLK	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

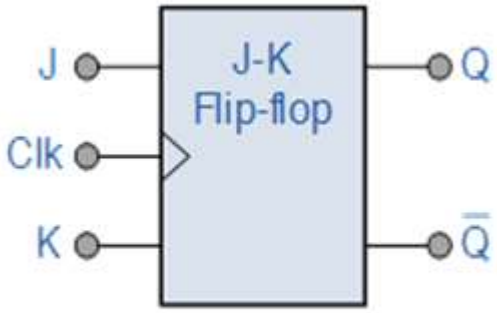


$Q(t+1) = D$

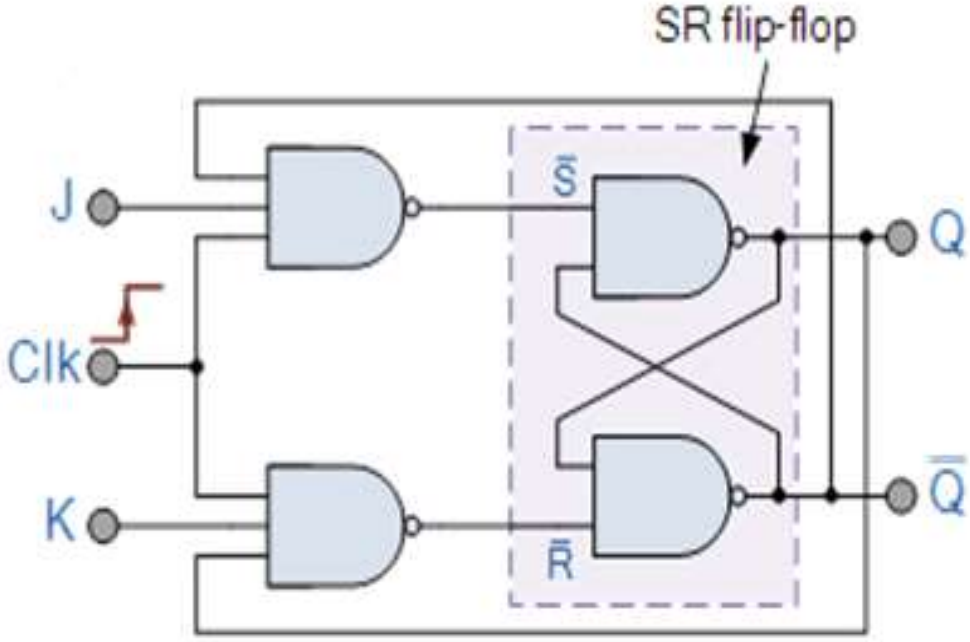
JK flip flop

CLK	J	K	Q_{n+1}
0	x	x	Q_n
1	0	0	Qn
1	1	0	1
1	0	1	0
1	1	1	Bar Qn

Truth Table

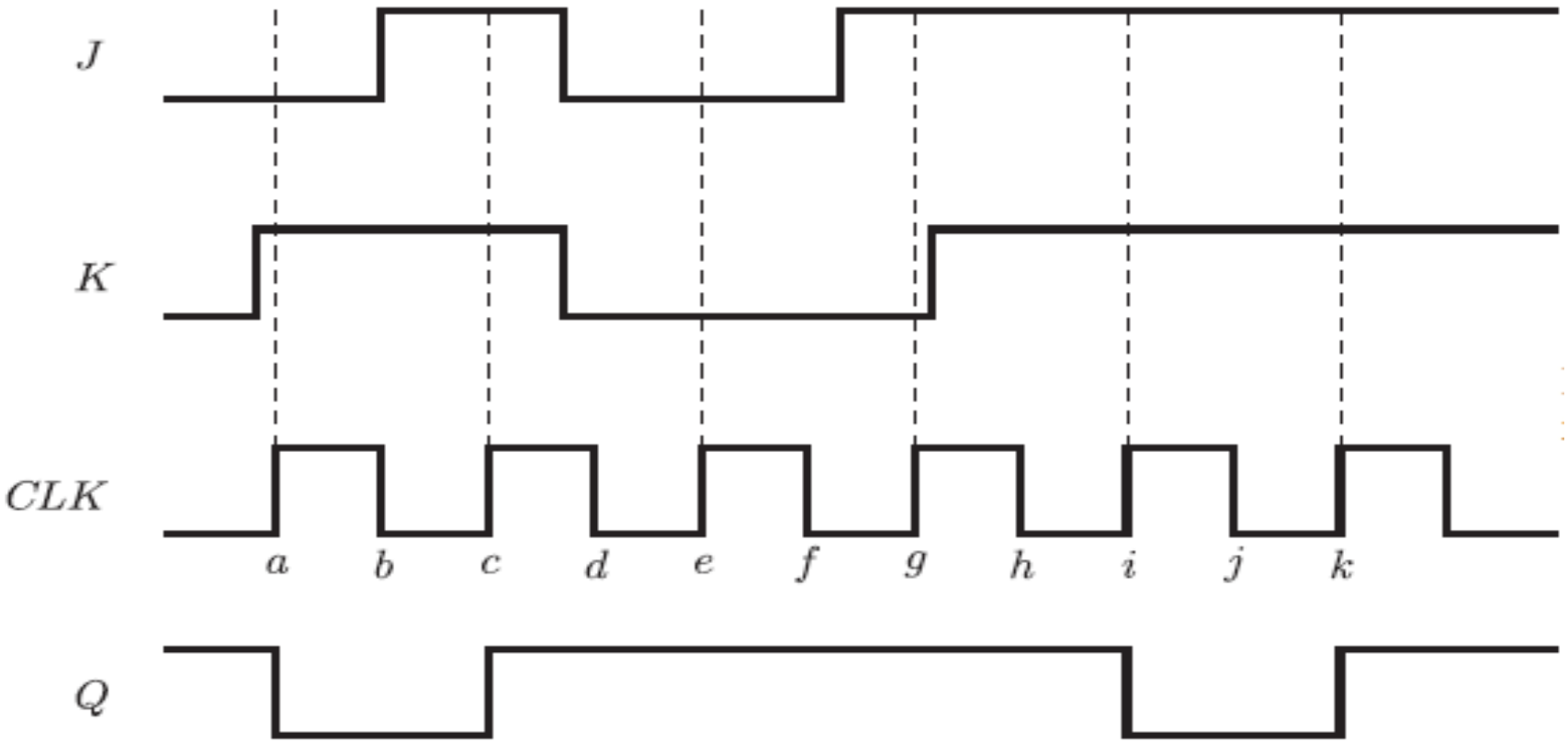


Symbol



Circuit

Timing diagram of JK flip flop

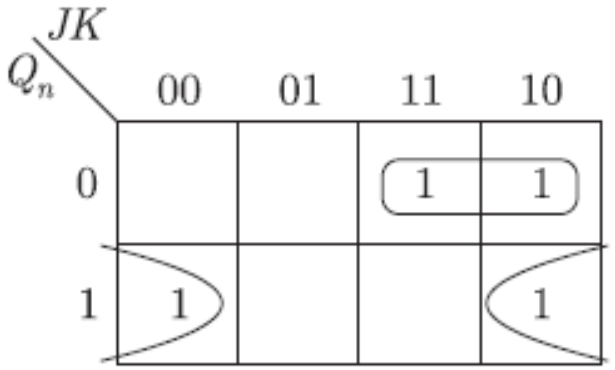


Timing diagram of JK flip flop

1. Initially all the inputs are zero and the output Q is 1.
2. At the time when first clock pulse goes HIGH (at point a), inputs are $J = 0$, $K = 1$. Thus, the output of flip-flop will be reset to $Q = 0$ state.
3.. Again, when the clock pulse goes HIGH at point c, inputs are $J = K = 1$. Therefore, the flip-flop output toggle to its opposite state i.e., $Q = 1$.
4. At point e clock pulse goes HIGH and inputs are $J = K = 0$, hence the flip-flop does not change state on this transition and output will be $Q = 1$.
5. At point g, clock pulse goes HIGH and input are $J = 1$, $K = 0$ This cause flip-flop to go into SET state i.e., $Q = 1$. Since it is already 1, and it will remain in the same state.
6. At point i, clock pulse goes high and inputs are $J = 1 = K$. Therefore, the flip-flop toggles to its opposite state and output will be 0.
7. We can determine output for remaining clock transitions in the same way.

Characteristic equation of JK FF and T FF

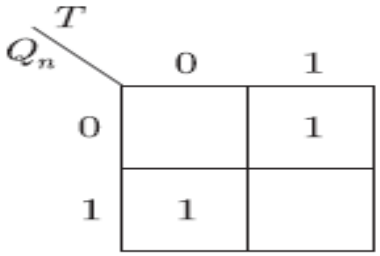
Inputs		Present state	Next state
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Characteristic Equation of J-K FF

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

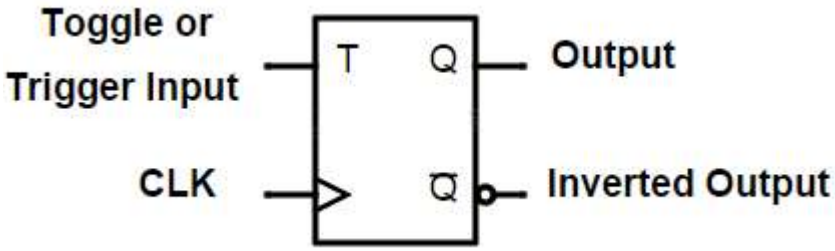
Input	Present state	Next state
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

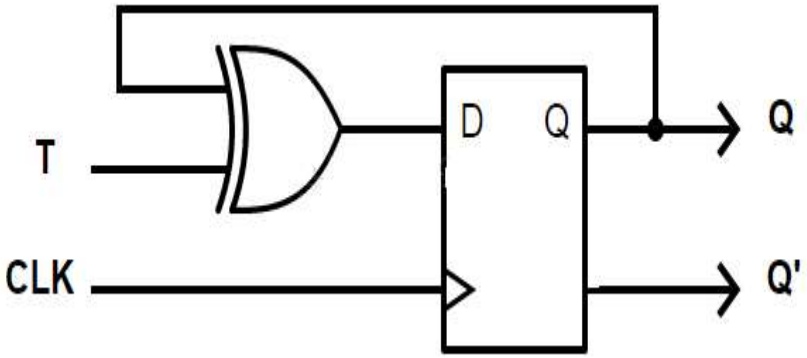


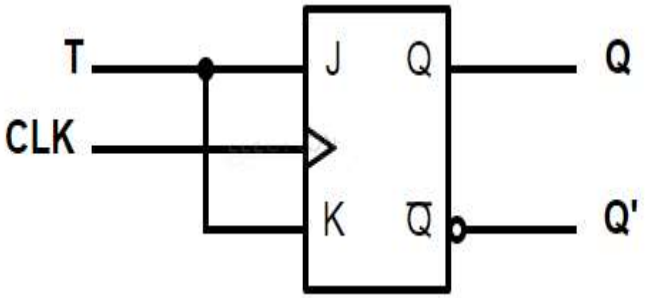
Characteristic Equation of T FF:

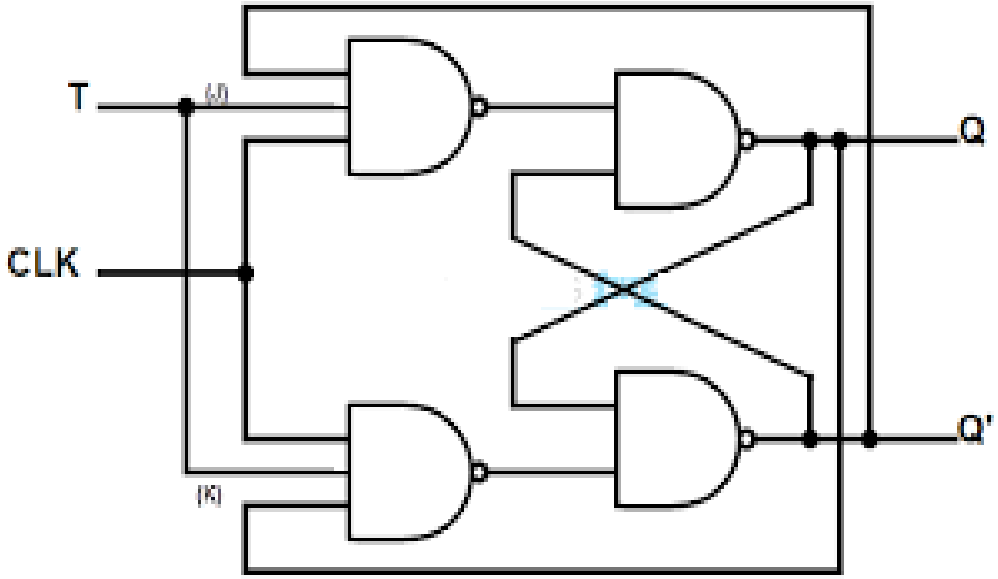
$$Q_{n+1} = T\overline{Q}_n + \overline{T}Q_n$$

T(toggle)-flip flop









CLK	Input T	Output Q_{n+1}
↑	0	Q_n
↑	1	$\overline{Q}_n$
0	X	No change

Excitation Table flip-flop

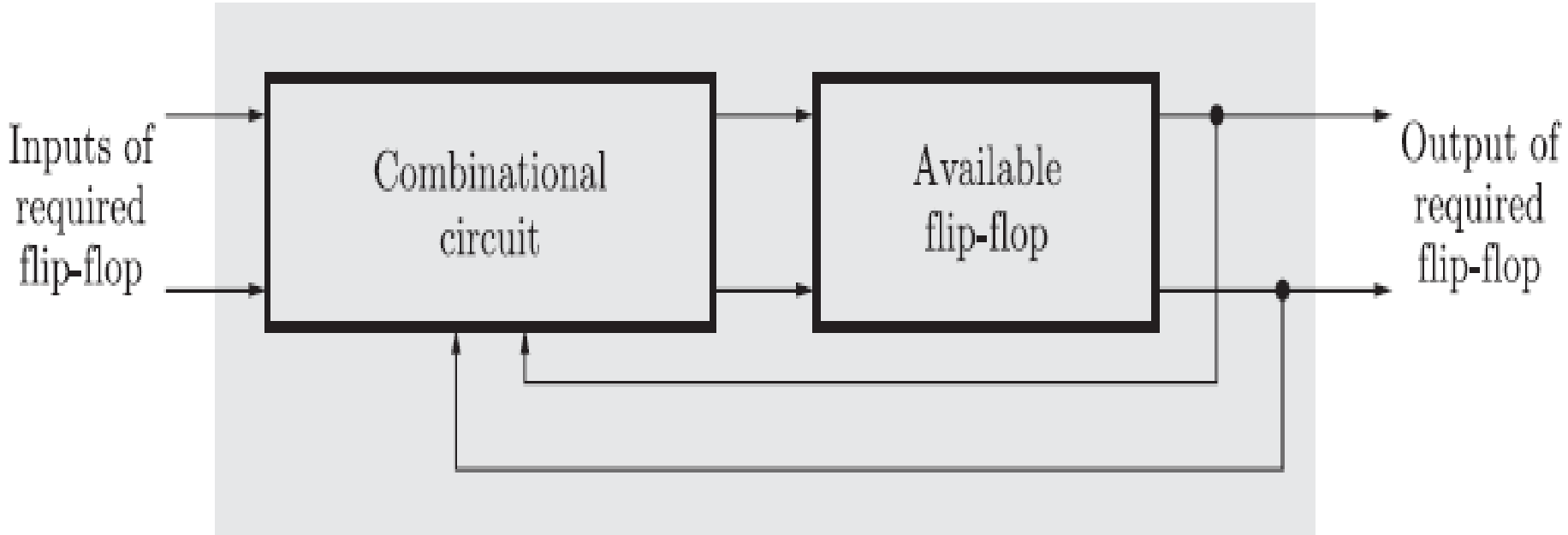
Table 7.11.5b: Excitation Table of S-R Flip-flop

Present State	Next State	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Table 7.11.6b: Excitation table of J-K Flip-flop

Present State	Next State	Required Inputs	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

CONVERSION OF FF



CONTI.

1. The first step of conversion is that write the present and next state table (i.e., characteristic table) of the required flip-flop.
2. In the table, for each transition $Q_n \rightarrow Q_{n+1}$, write the values of required inputs of given flip-flop. For this, excitation table of given flip-flop should be referred. The complete table is called conversion table as shown in Table 7.12.2.
3. Draw K-maps for each of the input variable of given flip-flop in terms of present state and inputs of required flip-flop. From the K-map, obtain the simplified expression of each of the input of given flip-flop.
4. Now, design a combination circuit for the above obtained expression and connect this with given flip-flop in correct manner.

Conversion from SR ff to D ff

Step 1: Construct the present-state-and-next-state table of a *D* flip flop as shown in Table

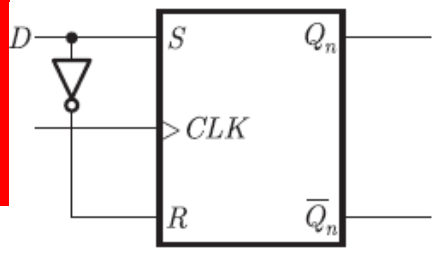
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Step 2: We write the values of inputs *S* and *R* that are required to change the state of the flip-flop from Q_n to Q_{n+1}. The complete conversion table as shown in Table

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

Step 3: Draw K-map for *S* and *R* with inputs *D* and Q_n as shown below.

Step 4: From the K-map of we get the simplified expressions, $S = D$ and $R = \overline{D}$. The logic diagram showing the conversion *S-R* flip-flop to *D* flip-flop is shown in



K-map for S

Q_n \ D	0	1
0	0	1
1	0	X

$S = D$

K-map for R

Q_n \ D	0	1
0	X	0
1	1	0

$R = \overline{D}$

Conversion of *S* - *R* Flip-Flop to *T* Flip-Flop

Step 1: Construct the present-state-and-next-state table of a *T* flipflop as shown in

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

Step 2: Write the values of inputs *S* and *R* that are required to change the state of the flip-flop from Q_n to Q_{n+1}. The complete conversion table as shown in Table

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

Step 3: Draw K-map for *S* and *R* with inputs *D* and Q_n as shown below

K-map for S

Q_n \ T	0	1
0	0	1
1	X	0

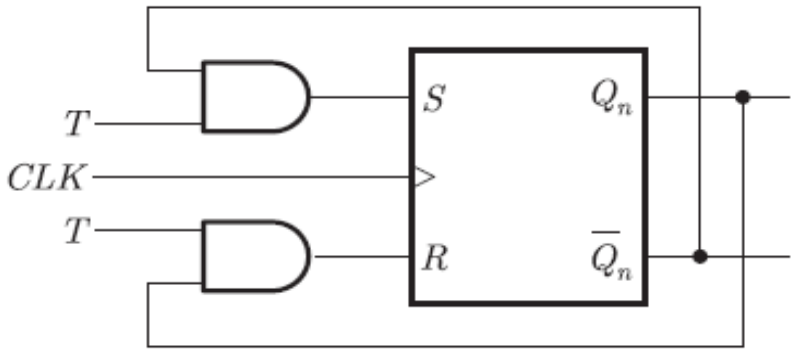
$S = T\overline{Q}_n$

K-map for R

Q_n \ T	0	1
0	X	0
1	0	1

$R = TQ_n$

Step 4: From the K-map of we get the simplified expression as $S\ TQ_n =$ and $R = TQ_n$. The logic diagram showing the conversion *S R* - flip-flop to *T* flip-flop is shown in Figure



Conversion of *D* Flip-Flop to *J* -*K* Flip-Flop

Step 1: Construct the present-state-and-next-state table of a *J* –*K* flip-flop as shown in Table 7.12.1.

Step 2: We write the values of input *D* that is required to change the state of the flip-flop from Q_n to Q_{n+1}.

Table 7.12.16: Conversion table

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

Table 7.12.1: Present-state-and-next-state table of a *J*–*K* flip-flop

J	*K*	Q_n	Q_{n+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

Step 3: Draw K-map for *D* with inputs *J* , *K* and Q_n as shown below.

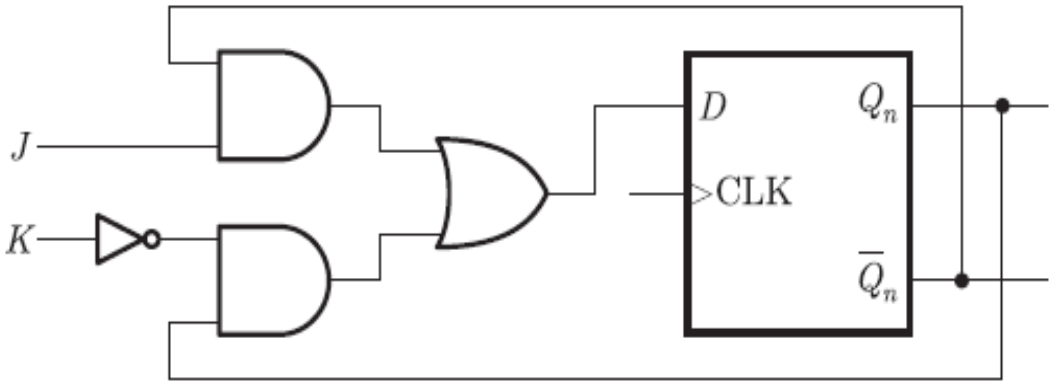
K-map for *D*

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

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

Step 4: From the K-map, we get simplified expression for *D* as

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

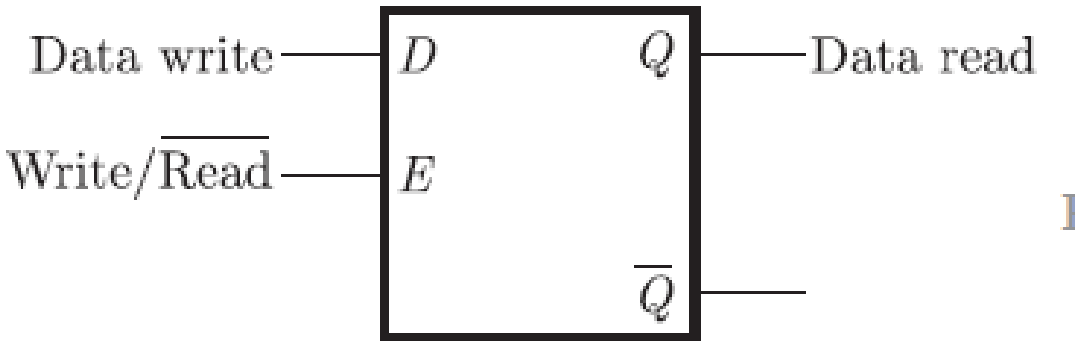


APPLICATION OF FLIF FLOP

Some of the basic applications are frequency division and counting circuits and data storage and transfer circuits. These are briefly described in the next subsections.

Registers

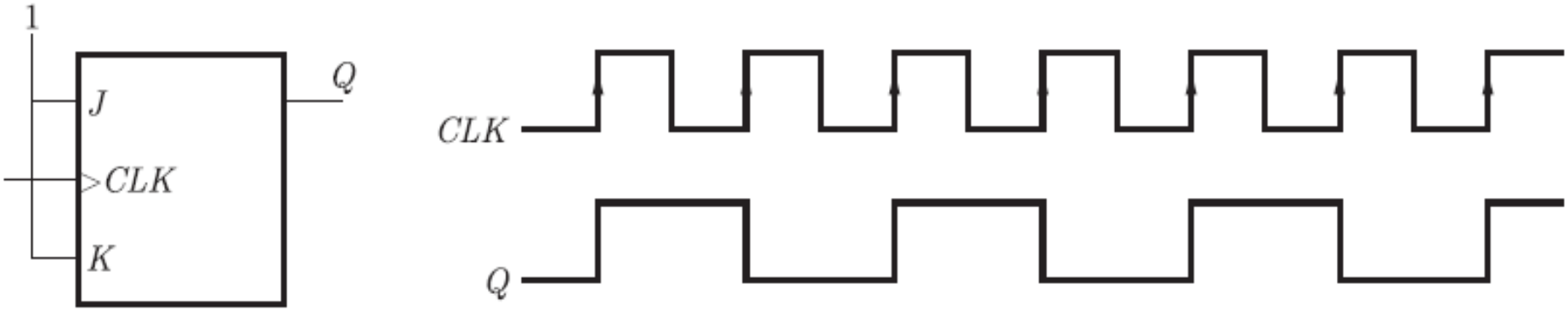
Shift registers are most commonly used to store digital data. Shift register can be constructed by a group of flip-flops. The output of one flip-flop is taken to the next flip-flop as input. A common clock is be written into memory and data can also be read from memory. A 1-bit read/write memory is shown in



APPLICATION OF FLIF FLOP

Frequency Division

For example, if a waveform is applied to the clock input of *J K* -flip-flop, which operates in toggle mode, the frequency of output waveform is half of the input frequency as shown in Figure Therefore, the frequency has been divided by 2.



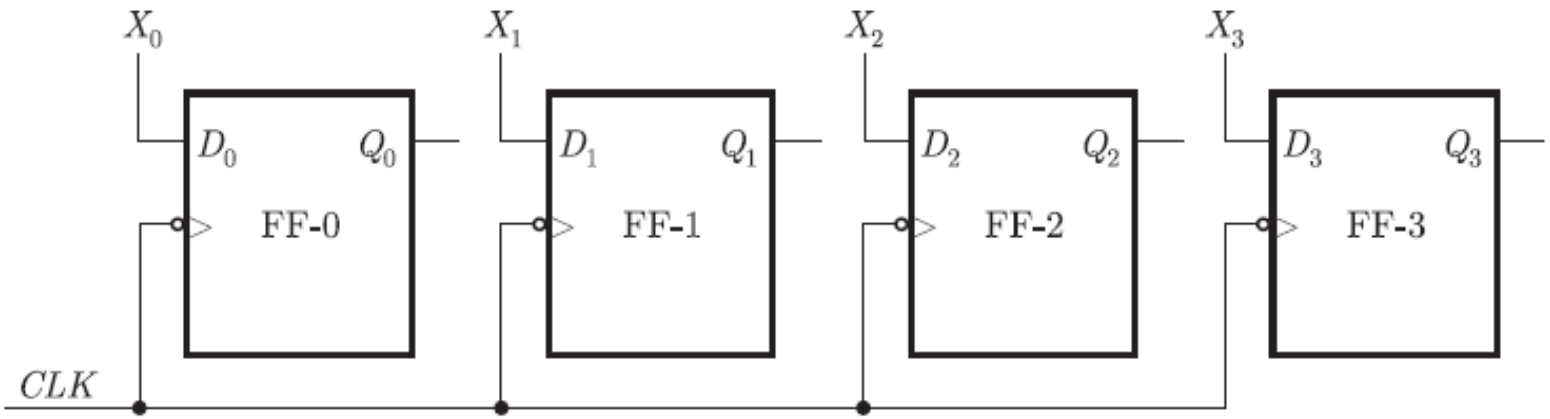
BUFFER REGISTER

shows the simplest register constructed with four *D* flip flops. The register is also called buffer register. Each *D* flip-flop is triggered with a common negative edge clock pulse. 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.

Therefore, when the first negative clock edge arrives, the stored binary information becomes,

$$Q_3Q_2Q_1Q_0 = X_3X_2X_1X_0$$

In this register, four *D* flip-flops are used. So it can store 4-bit binary information. Thus the number of flip-flop stages in register determines its total storage capacity.



SHIFT REGISTER

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

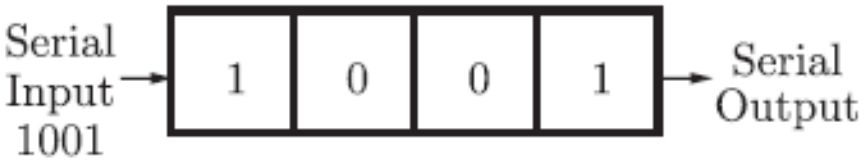
Classification of Shift Registers

Data may be shifted into or out of the register either in serial form or in parallel form. So, there are four basic types of shift registers:

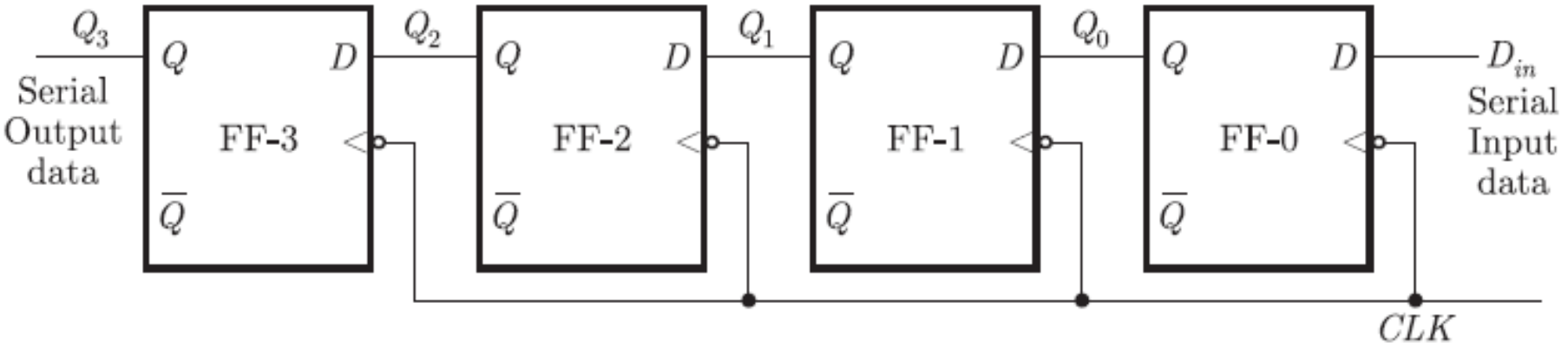
1. Serial-in Serial-out (SISO) Shift Registers
2. Serial-in, Parallel-out (SIPO) Shift Registers
3. Parallel-in Serial-out (PISO) Shift Registers
4. Parallel-in Parallel-out (PIPO) Shift Registers

Serial-in Serial-out (SISO) Shift Register

In Serial-in, Serial-out shift register, data input is in serial form and common clock pulse is applied to each of the flip-flop. After each clock pulse, data moves by one position. The output can be obtained in serial form. In this type of shift register, data moves either in left or right direction.



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



SISO

First of all assume that all the flip-flops initially are in the rest condition i.e., $Q3 = Q2 = Q1 = Q0 = 0$. Let us consider a four bit binary number 1111 which has to be entered into the register. When this is to be done, this number must be applied to *Din* bit-by-bit with MSB bit applied first.

1. We apply MSB bit of the number to be entered to *Din* . Therefore, *Din* = *D*0 = 1. Now, we apply the clock. On the first falling edge of clock, the FF-0 is set and the stored word in the register will be

$$Q3Q2Q1Q0 = 0\ 0\ 0\ 1$$

2. Then, we apply the next bit to *Din* . Hence, *Din* = 1. As soon as the next negative edge of the clock hits, FF-1 will set and the stored word will change to,

$$Q3Q2Q1Q0 = 0\ 0\ 11$$

3. Next, we apply the next bit to be stored, i.e., 1 to *Din* . Then, we apply the clock pulse. As soon as the third negative clock edge hits, FF-2 sets and the output will get modified to

$$Q3Q2Q1Q0 = 0\ 111$$

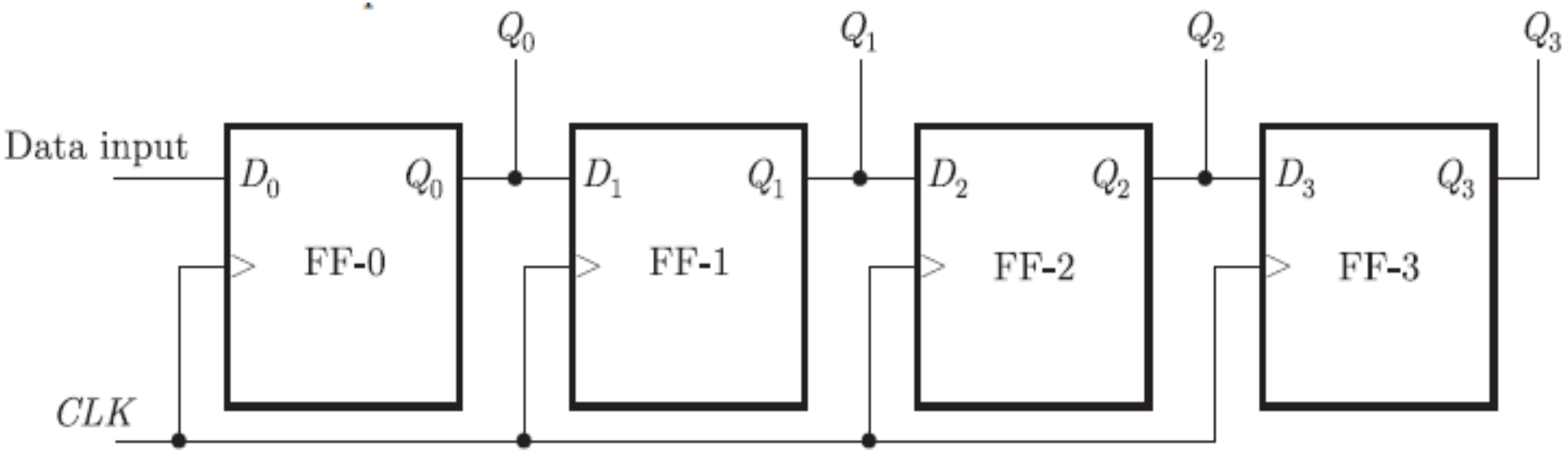
4. Similarly, with *Din* = 1, and with the fourth negative clock edge arriving, the second word in the register will be given by

$$Q3Q2Q1Q0 = 1111$$

The data in each stage after each of the four shift pulses is shown in Table below.

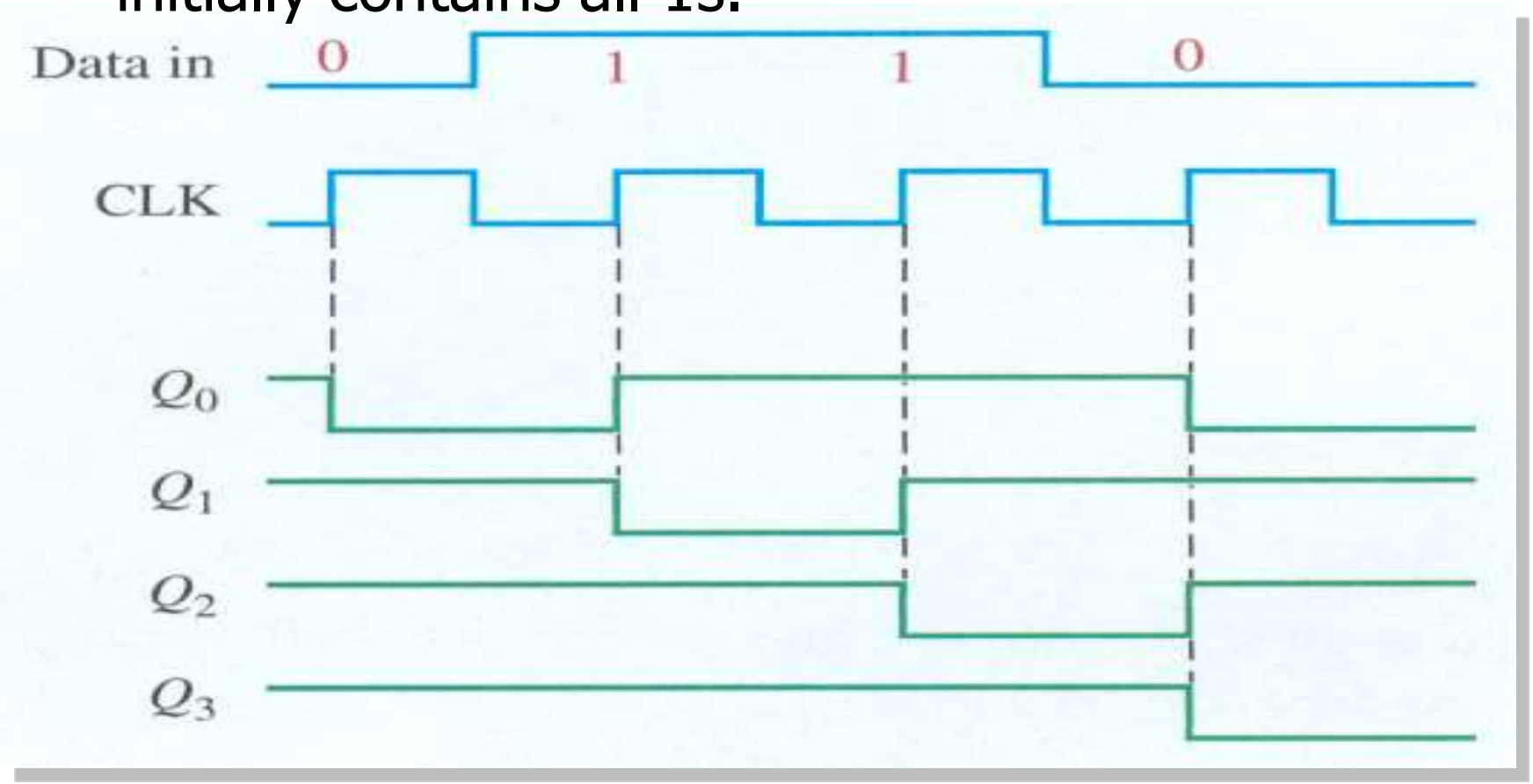
serial-in, parallel-out shift register

data bits are entered in shift register serially and the data bits are taken out of the register in parallel. Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously



SIPO

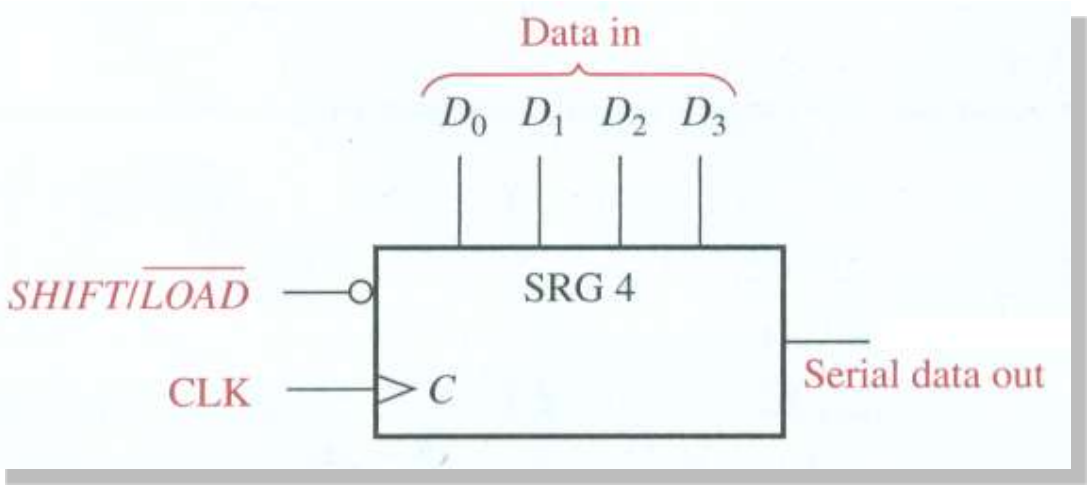
- **Example:** Show the state of the 4-bit register foe the data input and clock waveforms. The register initially contains all 1s.



Parallel-in, Serial-out shift register

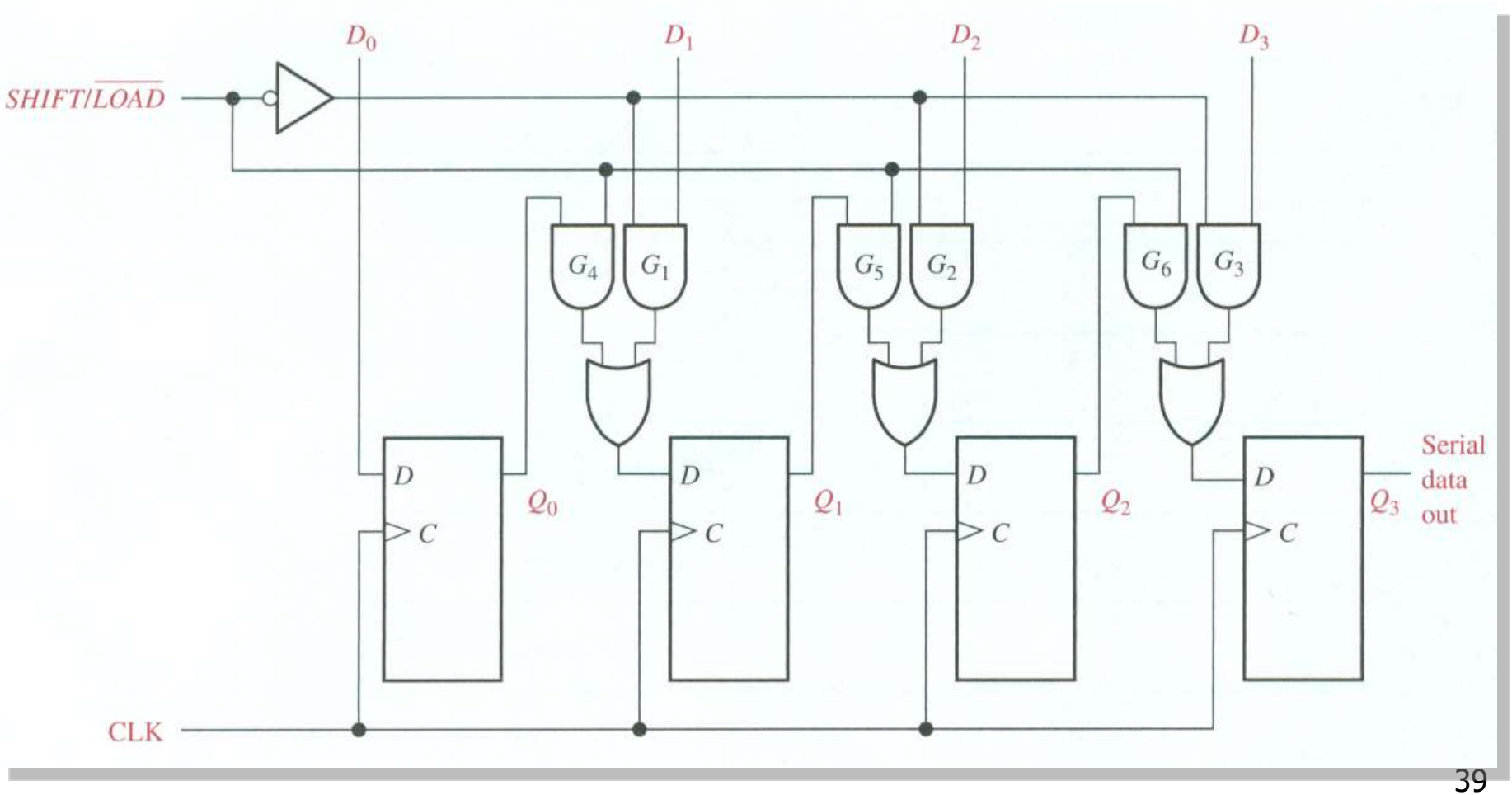
- The bits are entered simultaneously into their respective stages.
- The serial output appears bit by bit per clock pulse.
- To store 4 bits, we need 1 clock pulse
- To shift them out them, we need another 3 clock pulses.

- 4-bit parallel in/serial out



PISO

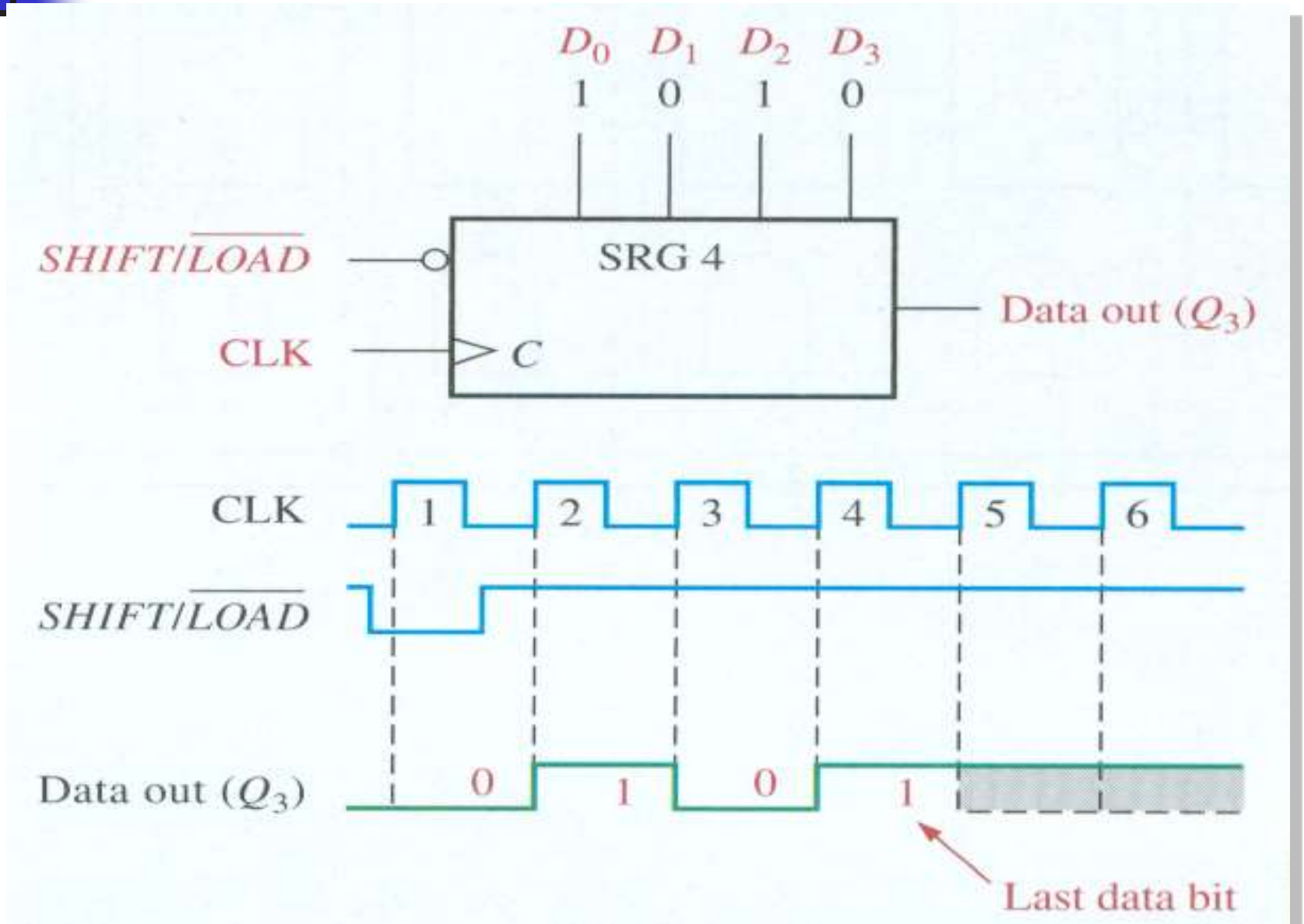
The signal Shift/LOAD allows the data to be entered in parallel form into the register and the data to be shifted out serially from terminal Q_3.



PISO

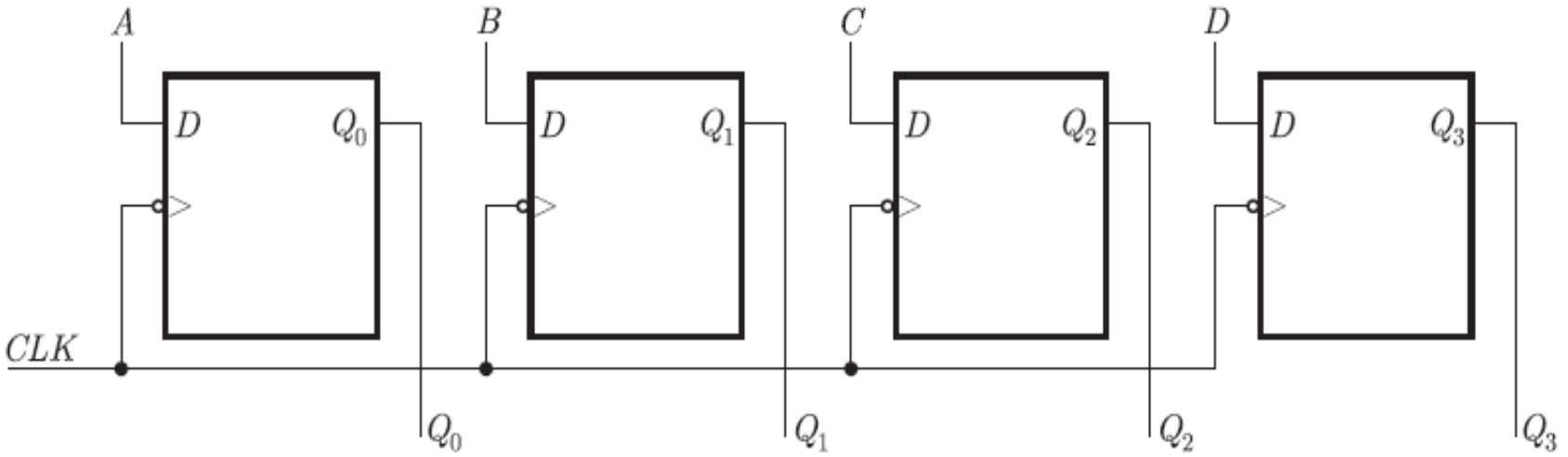
When Shift/LOAD line is LOW, gates 4, 5, and 6 are disabled,
whereas gates 1, 2, and 3 enabled allowing the data input to appear at the *D* inputs of the respective flip-flops. After application of a clock pulse, the flip-flop will be set if its *D* input is 1 and the flip-flop will be reset if its *D* input is 0. Therefore, all four bits will be stored simultaneously in the register.
When Shift/LOAD line is HIGH, gates 1, 2, and 3 are disable,
but gates 4, 5, and 6 are enabled allowing the data bits to shift-right from one stage to the next.

PISO


D_0 D_1 D_2 D_3
1 0 1 0
SRG 4
SHIFT/$\overline{LOAD}$
CLK
C
Data out (Q_3)
CLK
1 2 3 4 5 6
SHIFT/$\overline{LOAD}$
Data out (Q_3)
0 1 0 1
Last data bit

PARALLEL-IN PARALLEL-OUT SHIFT REGISTER

In Parallel-in, Parallel-out shift registers, all data bits are entered simultaneously on the parallel input lines rather than on a bit by bit basis on serial data inputs. After simultaneously entry of all data bits, the data bits are available on the output lines immediately



a 4-bit parallel-in, parallel-out, shift register using *D* flip-flops. The D_0, D_1, D_2, D_3 are the parallel inputs and the Q_0, Q_1, Q_2, Q_3 are the parallel outputs. When a clock pulse is applied, at the negative-going edge of that pulse, the *D* inputs are shifted into the *Q* outputs of the flip-flops. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

APPLICATIONS OF SHIFT REGISTERS

Shift registers can be found in many applications. The common
applications of shift registers are listed as below

1. Timing circuits to produce time delay
2. Shift register counters: Ring counter and Johnson counter,
3. Serial to parallel converters,
4. Parallel to serial converters,
5. Sequence generators

COUNTERS

A counter is one of the most useful subsystems in a digital system.

Counter is a sequential circuit that is used to counting the number of clock pulses arriving at its clock input.

CLASSIFICATION OF COUNTERS

There are various types of counters, which are used to count binary numbers. Based on application of clock, counter are classified as asynchronous and synchronous counters. Counters may be an up counter or down counter.

Asynchronous (Ripple) Counters : In asynchronous counter, the external clock pulse is applied to the first flip-flop and the output of first flip-flop (either Q or $\overline{Q}$) is connected as clock of the next flip-flop.

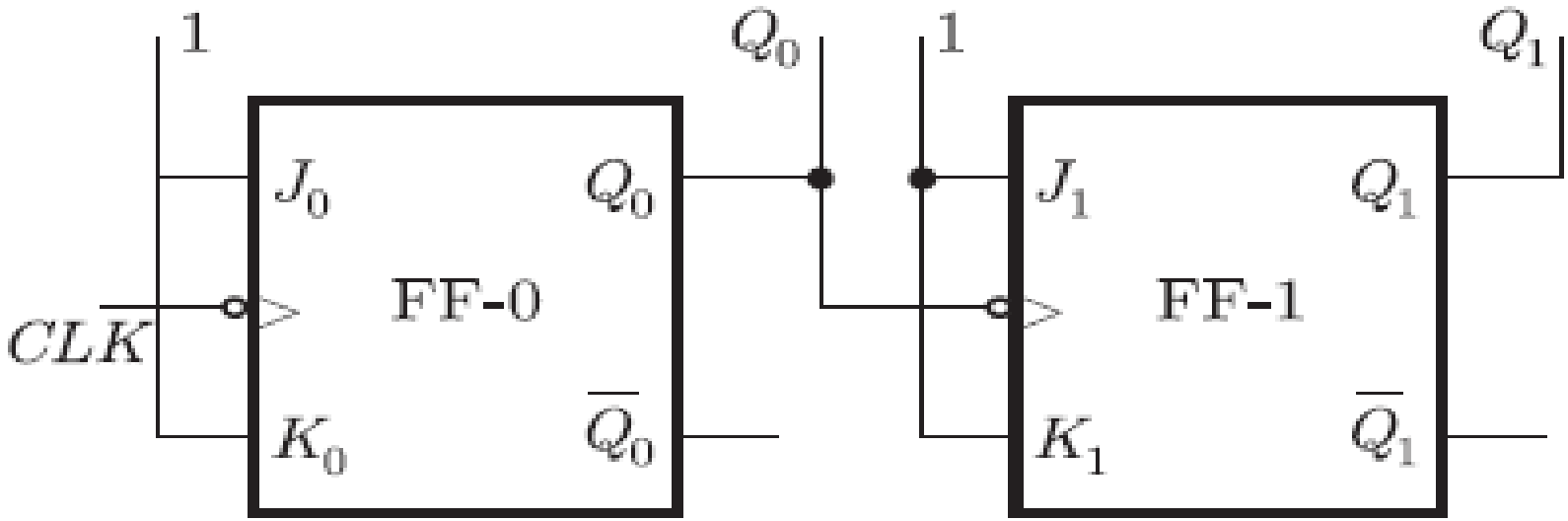
Synchronous Counters: In a synchronous counter, all the flip-flops in the counter change state at the same time in synchronism with the input clock signal.

ASYNCHRONOUS COUNTERS OR BINARY RIPPLE COUNTERS

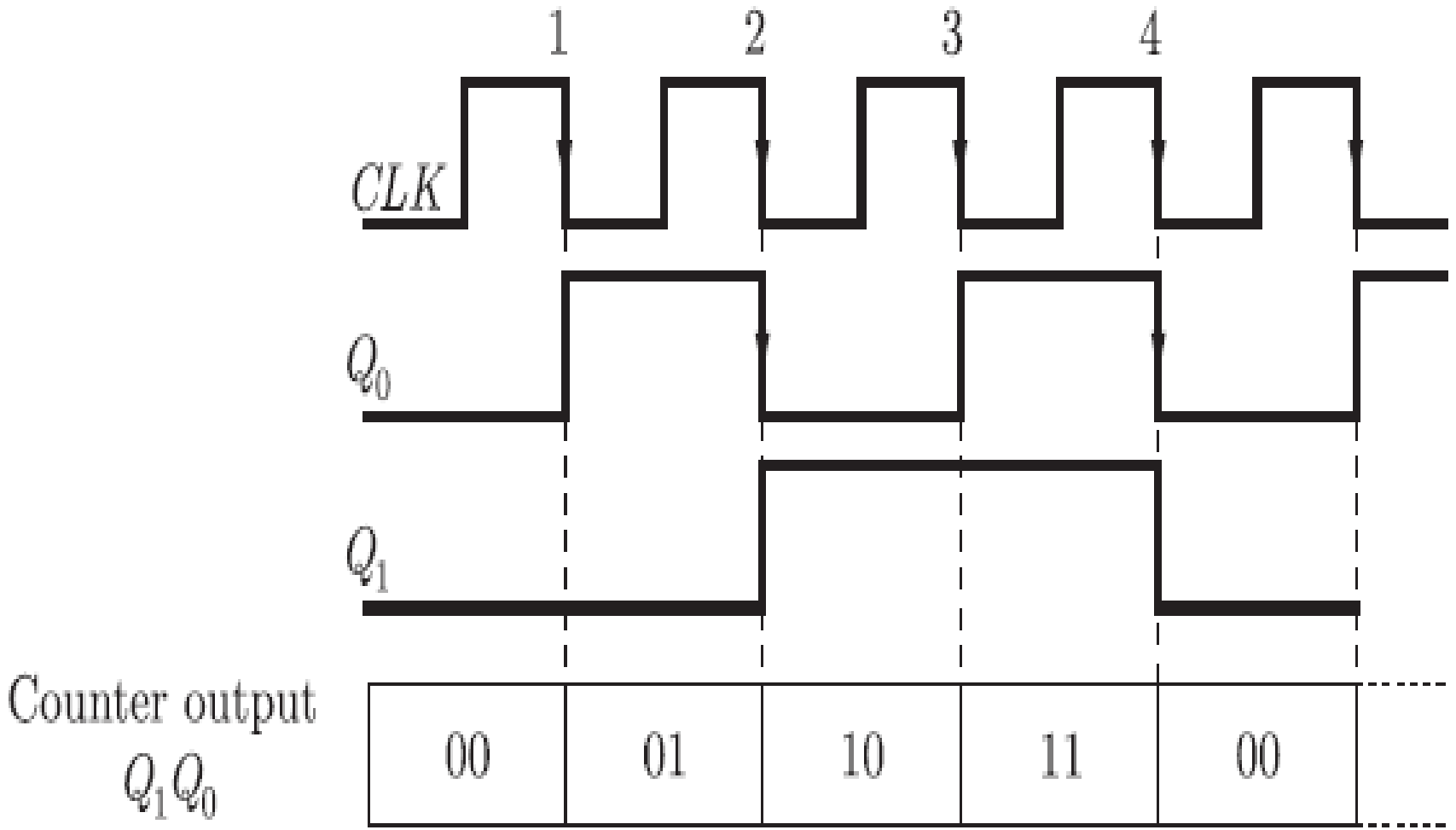
An *n*-bit counter will have *n* flip-flops and 2_n states, and divides the input frequency by 2_n. Hence, it is a divide by 2_n counter. It can count in binary from 0 to through 2_n - 1.

Since flip-flops are used in toggle mode, we can construct counters using *J* -*K* flip-flops with the *J* and *K* inputs tied together or from *T* flip-flops. A third possibility is to use a *D* flip-flop with the complement output connected to the *D* input. In this way, the *D* input is always the complement of the present state, and the next clock pulse will cause the flip-flop to toggle. Also, we may use positive edge-triggered or negative edge triggered flip-flop to design counters. For simplicity, here we consider all counters using negative edge-triggered *J* -*K* flip-flops. The other design using *T* or *D* or positive edge-triggered flip-flops will be explained through examples.

2-bit Ripple Up-counter



Clock Pulse	Q_1	Q_0
0	0	0
1	0	1
2	1	0
3	1	1
4 (repeat)	0	0


1
2
3
4
CLK
Q_0
Q_1
Counter output
Q_1Q_0
00
01
10
11
00

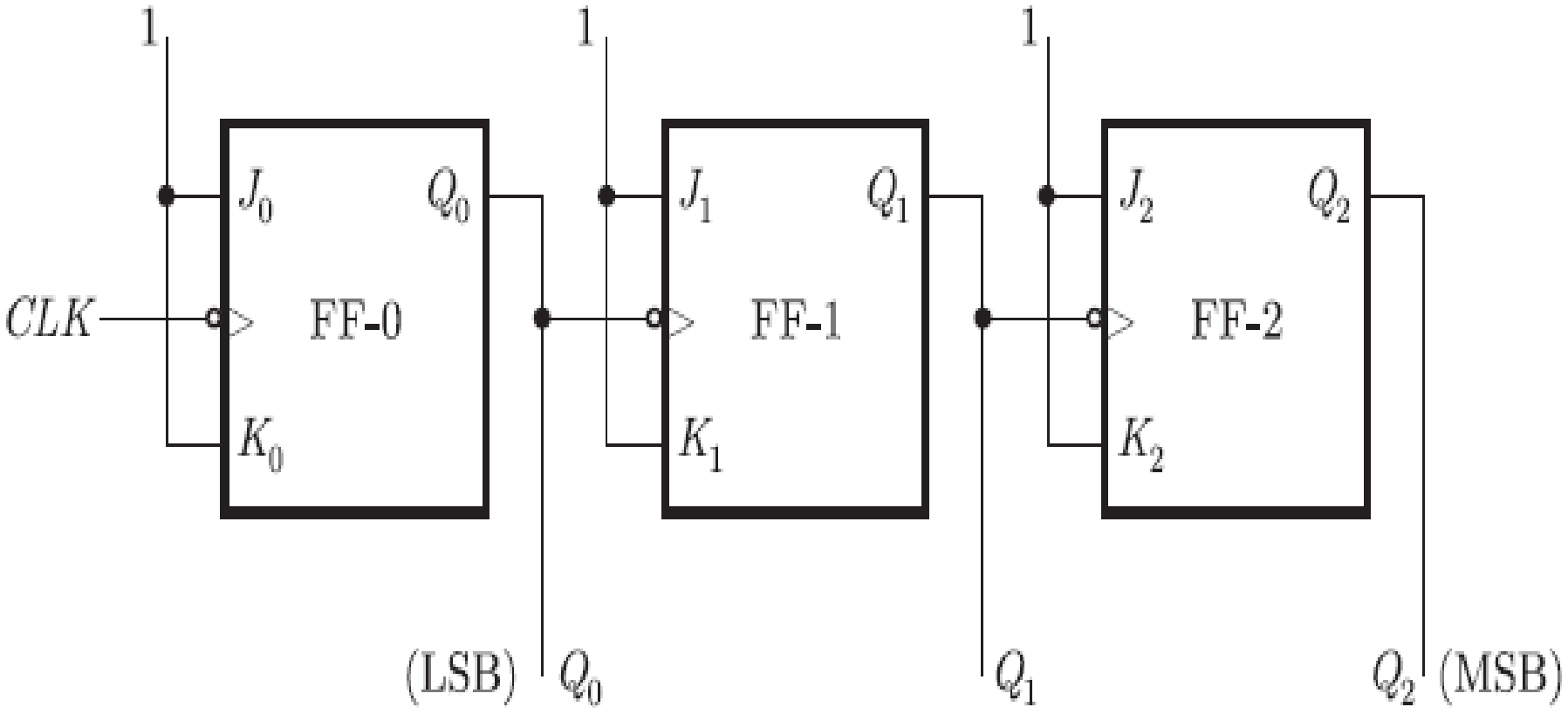
2-bit Ripple Down Counter

A 2-bit down-counter counts in the order 0, 3, 2, 1, 0, 3, ...or, in binary as 00, 11, 10, 01, 00, 11, ..., etc.

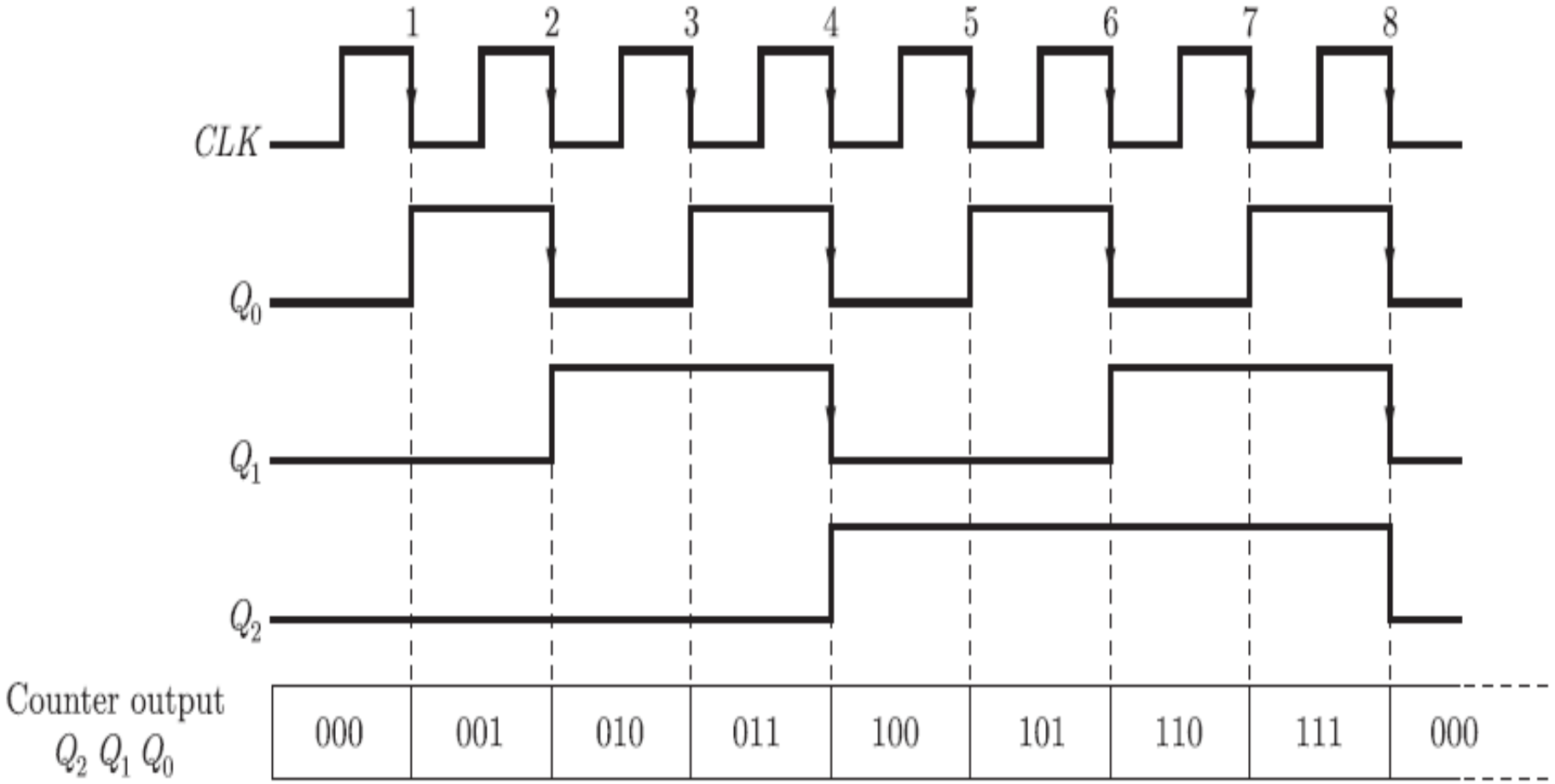
Clock Pulse	Q_1	Q_0
0	1	1
1	1	0
2	0	1
3	0	0
4 (repeat)	1	1

3-bit Ripple up counter

The 3-bit up counter counts in the order 0, 1, 2, 3, 4, 5, 6, 7, 0, 1,...
or in binary as 000, 001, 010, 011, 100, 101, 110, 111,.... etc.



Timing diagram of a 3-bit binary ripple up-counter



UP/DOWN RIPPLE COUNTERS

$M = 0$ **(Down counting mode)**

$M = 1$ **(Up counting mode)**