Sequential Switching Circuits

- Sequential switching circuits are circuits whose output levels at any instant of time are dependent on the levels present at the inputs at that time and 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.
- Made up of combinational circuits and memory elements, e.g. Counters, shift registers, serial adder, etc.

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.

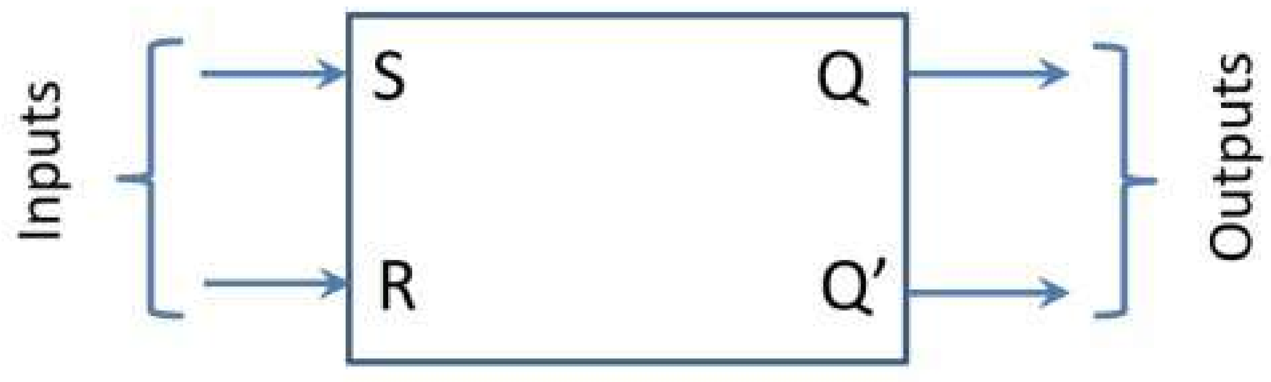
Flip-flop and Latch

- A flip-flop, known formally as 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.
- Latch is used for certain flip-flop which are *non-clocked.*
- These flip-flops 'latch on' to a 1 or a 0 immediately upon receiving the input pulse called SET or RESET.
- The latch is sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

S-R Latch

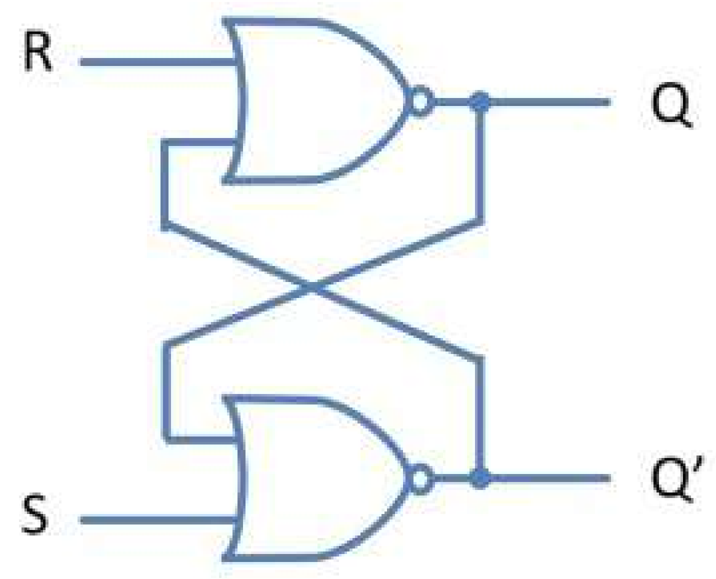
- The simplest type of flip-flop is called an S-R latch.
- It has two outputs labelled Q and Q' and two inputs labelled S and R. The state of the latch corresponds to the level of Q (HIGH or LOW, 1 or 0) and Q' is the complement of that state.
- It can be constructed using either two cross-coupled NAND 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.
- The name of the latch, S-R or SET-RESET, is derived from the names of its inputs.
- Below is the common logic symbol for both latches.



Logic Symbol

NOR gate S-R latch (Active HIGH)

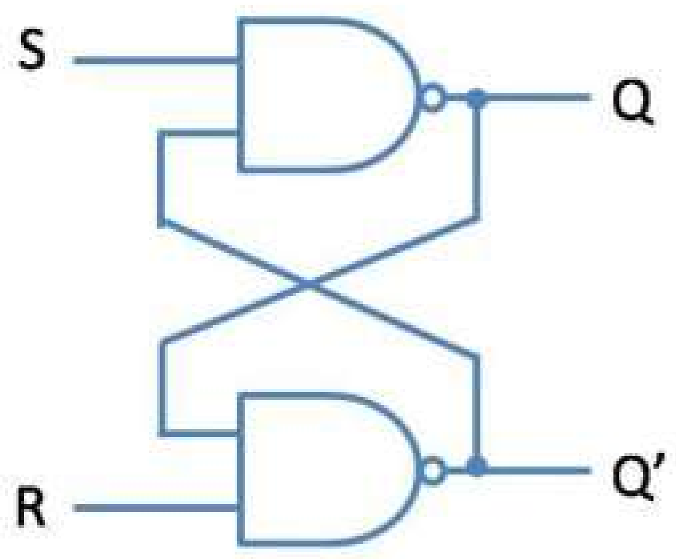


Logic diagram

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	

- When the SET input is made HIGH, Q becomes 1.
- When the RESET input is made HIGH, Q becomes 0.
- If both the inputs S and R made LOW, there is no change in the state of the latch.
- If both the inputs are made HIGH, the output is unpredictable i.e. invalid.
- Figure shows the logic diagram of an active HIGH S-R latch composed of two cross-coupled NOR gates.
- Note that the output of each gate is connected to one of the inputs of the other gate.
  The latch works as per the truth table of figure, where Q_n represents the state of the flip-flop before applying inputs and Q_{n+1} represents the state of the flip-flop after applying inputs.

NAND gate S-R latch (Active LOW)



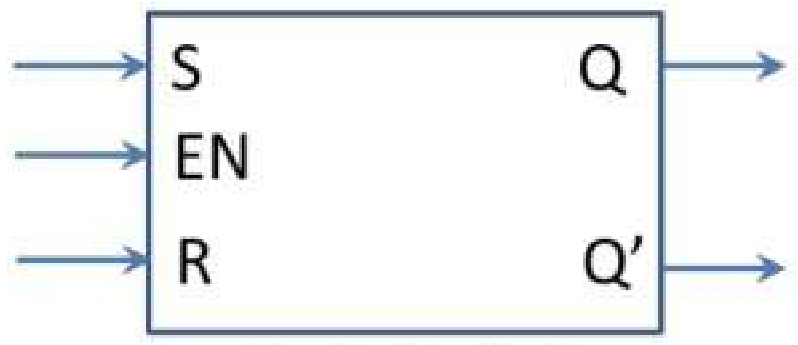
Logic diagram

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

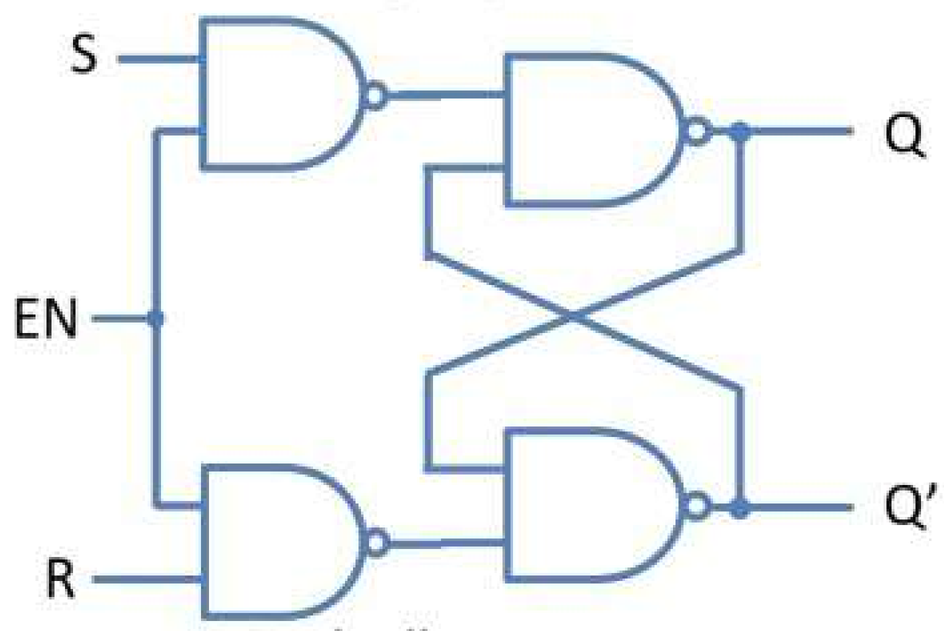
- The NAND gate is equivalent to an active LOW OR gate, am active LOW S-R latch using OR gates may also be represented.
- The operation of this latch is the reverse of the operation of the NOR gate latch.

- If the 0s are replaced by 1s and 1s by 0s, we get the same truth table as that of NOR gate latch.
- 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.

S-R Flip-flop (Gated S-R latch)



Logic Symbol

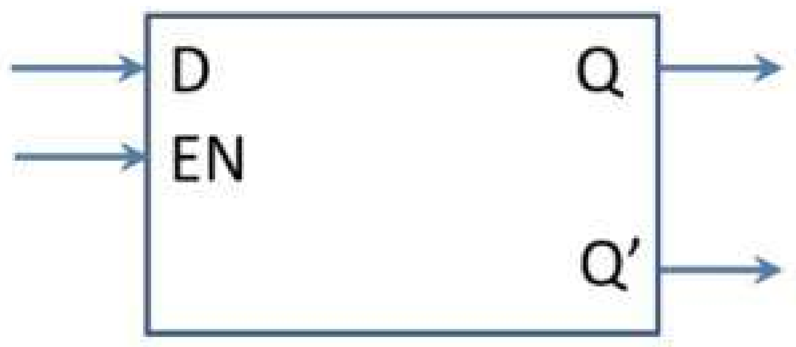


Logic diagram

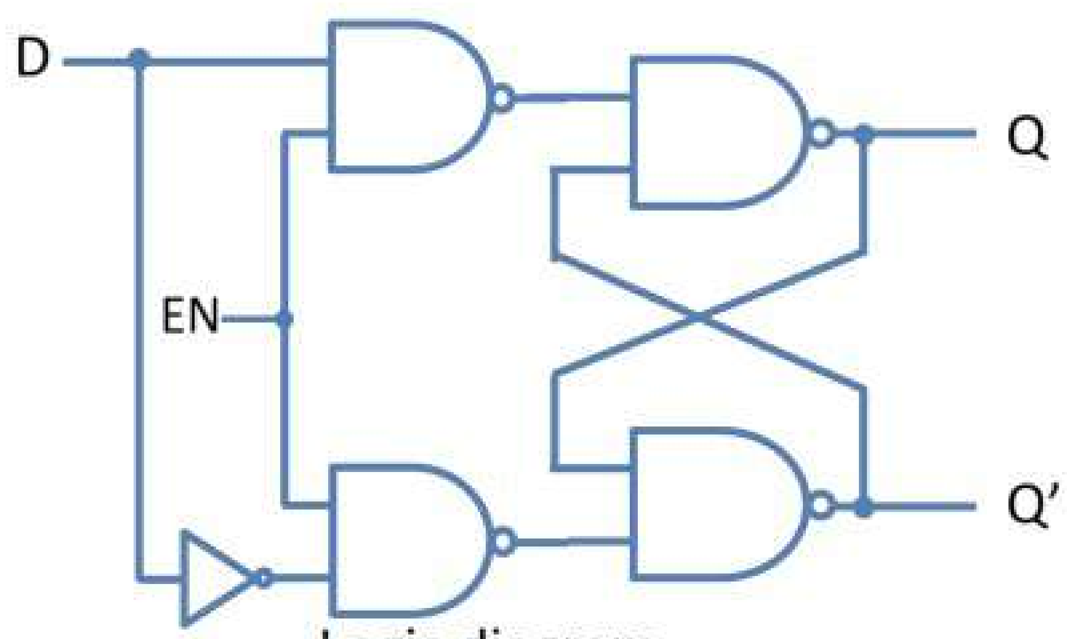
En	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No Change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Indeterminate (Invalid)
1	1	1	1	X	
0	X	X	0	0	No Change (NC)
0	X	X	1	1	

- A gated S-R larch requires an ENABLE (EN) input.
- Its S and R inputs will control the state of the flip-flop only when the EN is HIGH.
- When EN is LOW, the inputs become ineffective and no change of state can take place.
- The EN input may be a clock. So, a gated S-R latch is also called a *clocked S-R latch*.
- Since this type of flip-flop responds to the changes in inputs only as long as the clock is HIGH, these types of flip-flops are also called *level triggered flip-flops*.

D Flip-flop (Gated D latch)



Logic Symbol

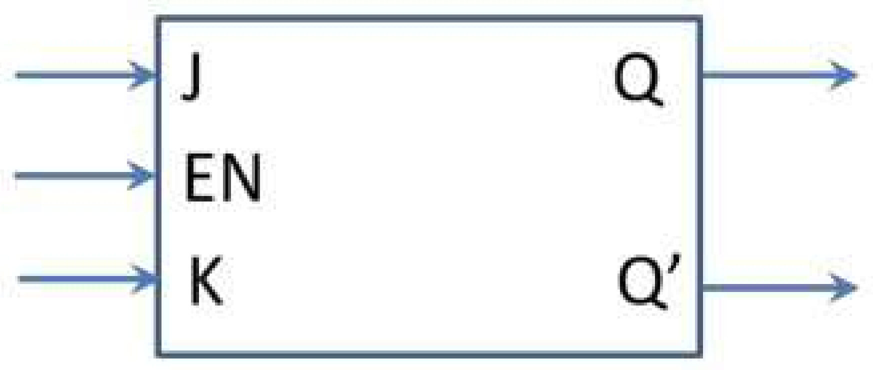


Logic diagram

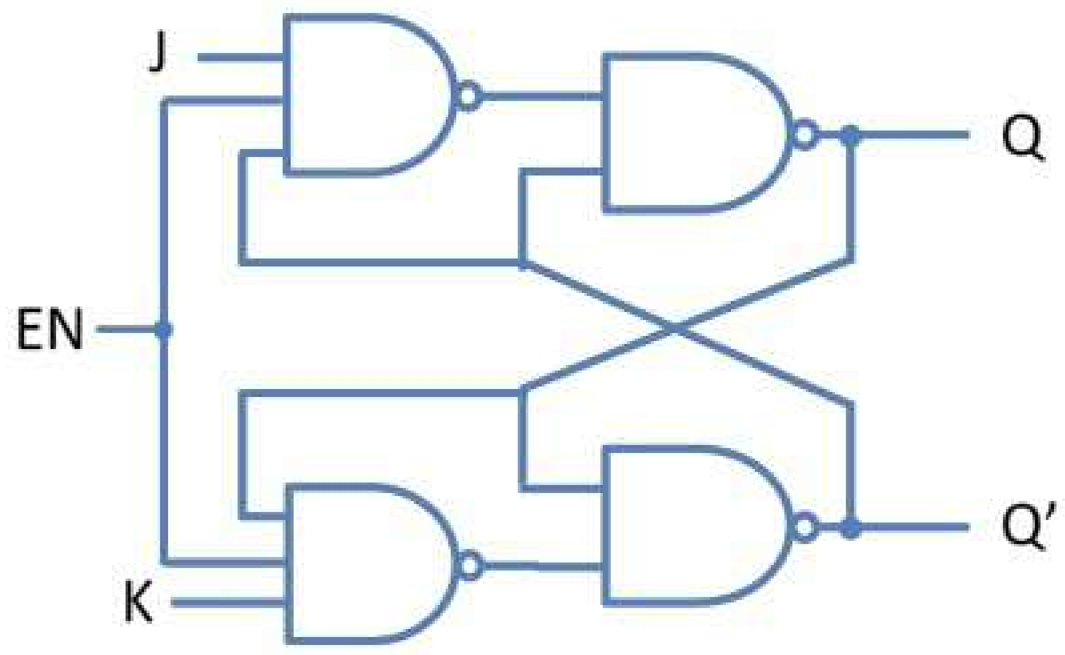
En	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	X	0	0	No Change (NC)
0	X	1	1	

- It differs from the S-R latch in that it has only one input in addition to EN.
- When D=1, we have S=1 and R=0, causing the latch to SET when ENABLED.
- When D=0, we have S=0 and R=1, causing the latch to RESET when ENABLED.
- When EN is LOW, the latch is ineffective, and any change in the value of D input does not affect the output at all.
- When EN is HIGH, a LOW D input makes Q LOW, i.e. resets the flip-flop and a HIGH D input makes Q HIGH, i.e. sets the flip-flop.
  In other words, we can say that the output Q follows the D input when EN is HIGH.

J-K Flip-flop (Gated J-K latch)



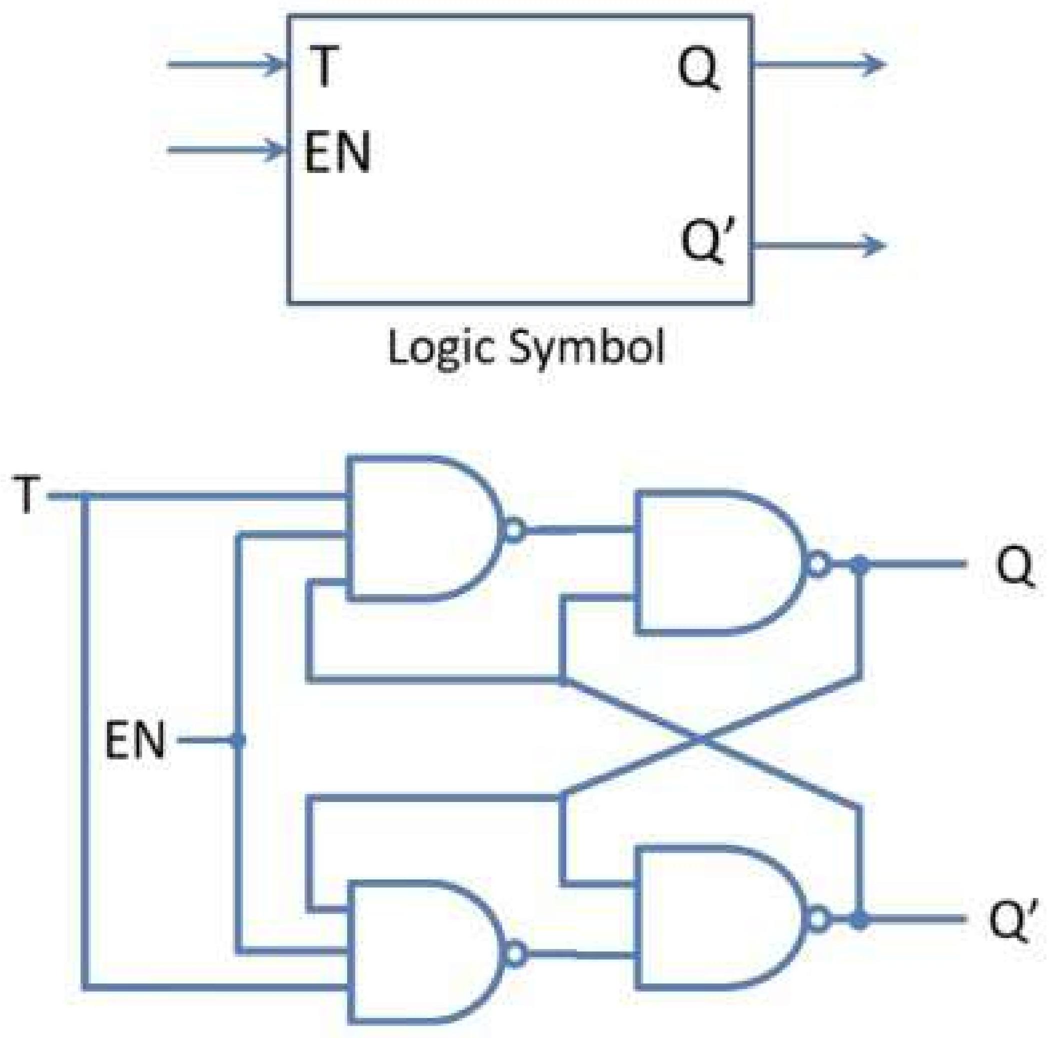
Logic Symbol



En	J	K	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	1	Toggle
1	1	1	1	0	
0	X	X	0	0	No Change (NC)
0	X	X	1	1	

- The J-K flip-flop is very versatile and also the most widely used.
- 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 S-R flip-flop.
- When J=0 and K=0, no change of state place even if a clock pulse is applied.
- When J=0 and K=1, the flip-flop resets at the HIGH level of the clock pulse.
- When J=1 and K=0, the flip-flop sets at the HIGH level of the clock pulse.
- When J=1 and K=1, the flip-flop toggles, i.e. goes to the opposite state at HIGH level of clock pulse.
- We can make edge triggered flip-flop as well by using positive and negative edges of clock instead of HIGH and LOW level.

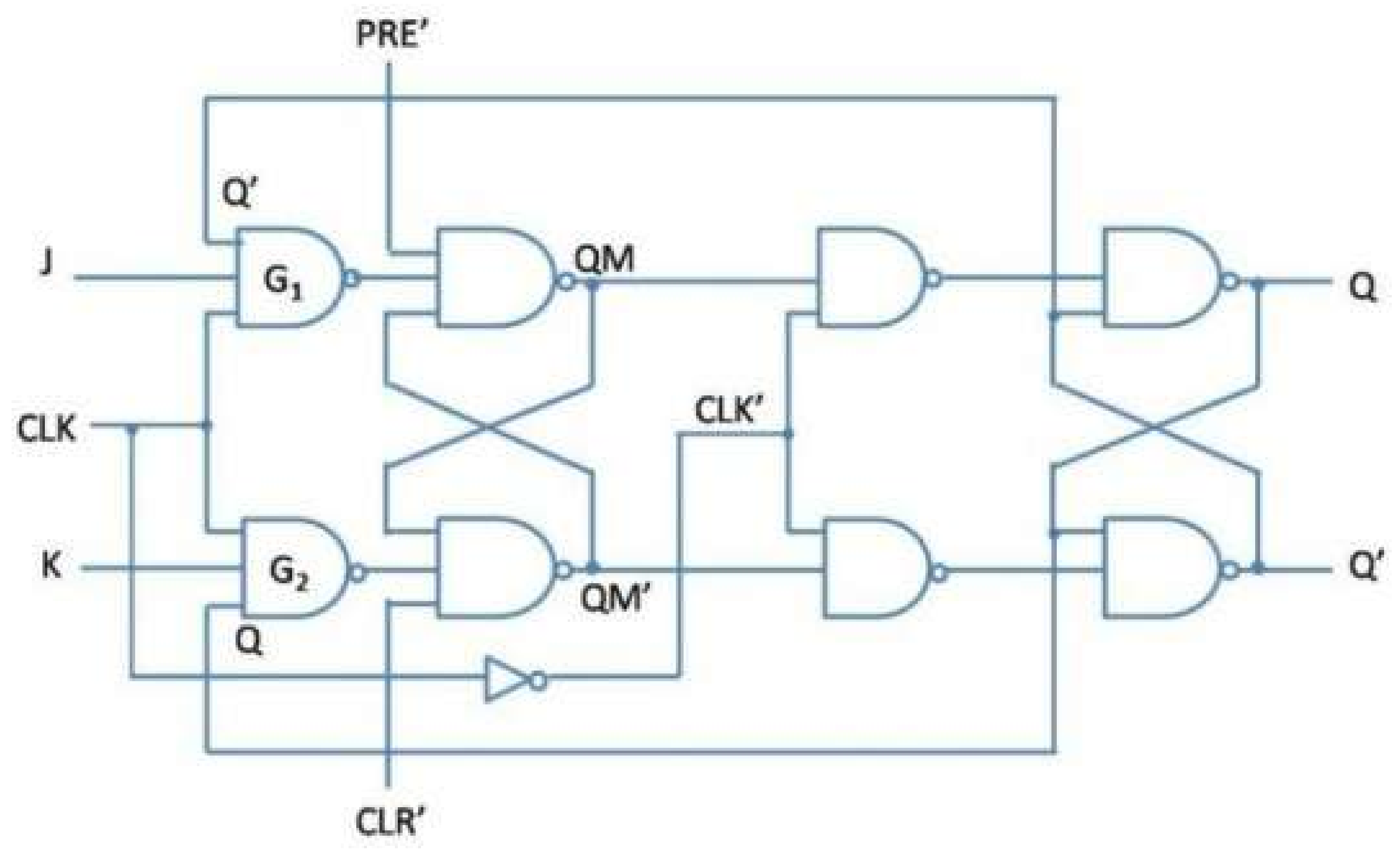
T Flip-flop (Gated T latch)



En	T	Q_n	Q_{n+1}	State
1	0	0	0	No Change (NC)
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	
0	X	0	0	No Change (NC)
0	X	1	1	

- A T flip-flop has a single control input, labeled 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.
- Although T flip-flops are not widely available commercially, 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 labeling the common connection as T.
- Thus, when T = 1, we have J = K = 1, and the flip-flop toggles.
  When T = 0, we have J =K = 0, and so there is no change of state.

Master-slave J-K Flip-flop



Inputs			Output	Comments
J	K	C	Q	
0	0	1	Q_0	No Change
0	1	1	0	Reset
1	0	1	1	Set
1	1	1	Q_0'	Toggle

- As shown in figure, the external control inputs J and K are applied to the master section.
- The master section is basically a gated JK latch, with Q output is connected back to the input of G_2 and Q' output is connected back to the input of G_1 and responds to the external J-K inputs applied to it at the positive 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 at the negative edge of the clock pulse.

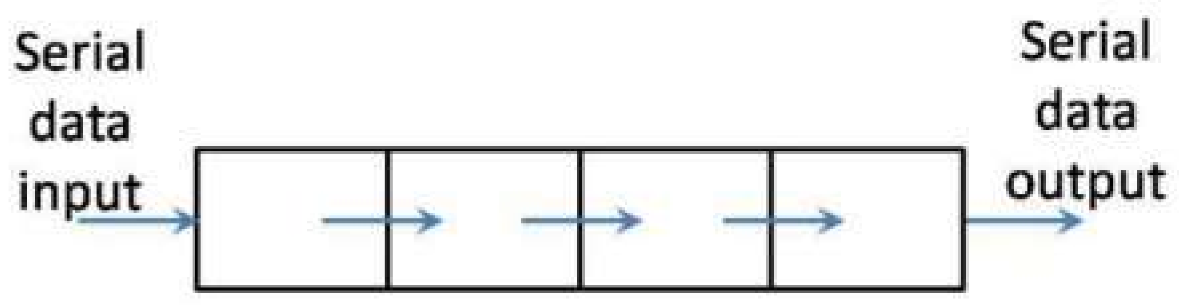
- Thus, the master section assumes the state determined by the J and K inputs at the positive edge of the clock pulse and the slave section copies the state of master section at negative edge of the clock pulse.
  The state of the slave then immediately appears on its Q and Q′ outputs.

Register

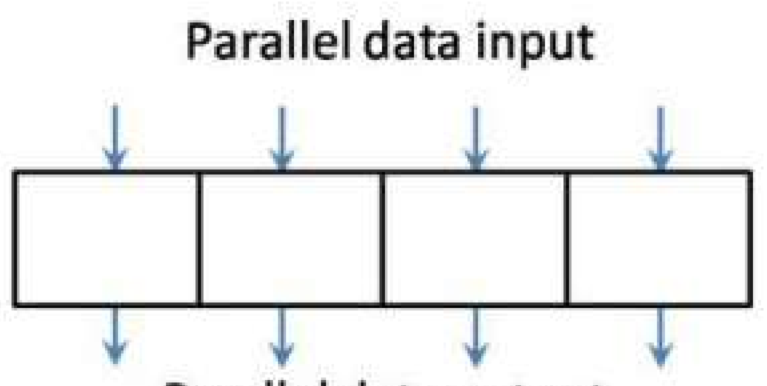
- As a flip-flop (FF) can store only one bit of data, a 0 or a 1, it is referred to as a single-bit register.
- A register is a set of FFs used to store binary data.
- The storage capacity of a register is the number of bits (1s and 0s) of digital data it can retain.
- Loading a register means setting 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.
- 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.
- Types of Registers
  1. Buffer register
  2. Shift register
  3. Bidirectional shift register
  4. Universal shift register

Shift register

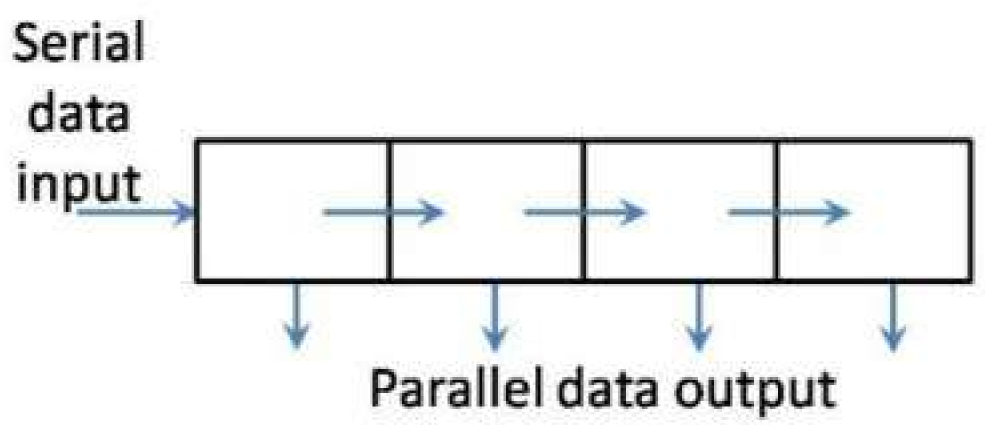
- A number of FFs connected together such that data may be shifted into and shifted out of them is called a shift register.
- Data may be shifted into or out of the register either in serial form or in parallel form.
- So, there are four basic types of shift registers:
  1. serial-in, serial-out
  2. serial-in, parallel out
  3. parallel-in, serial-out
  4. parallel-in, parallel-out
- Data may be rotated left or right. Data may be shifted from left to right or right to left at will, i.e. in a bidirectional way.
- Also, data may be shifted in serially (in either way) or in parallel and shifted out serially (in either way) or in parallel.



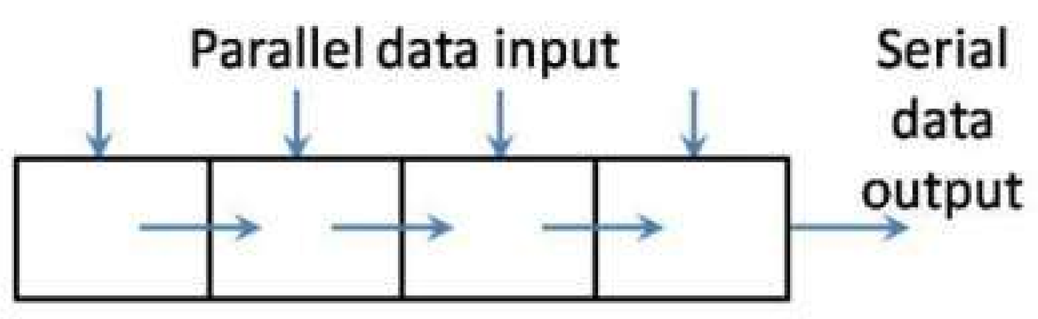
Serial-in, serial-out shift-right, shift register



Parallel-in, parallel-out, shift register

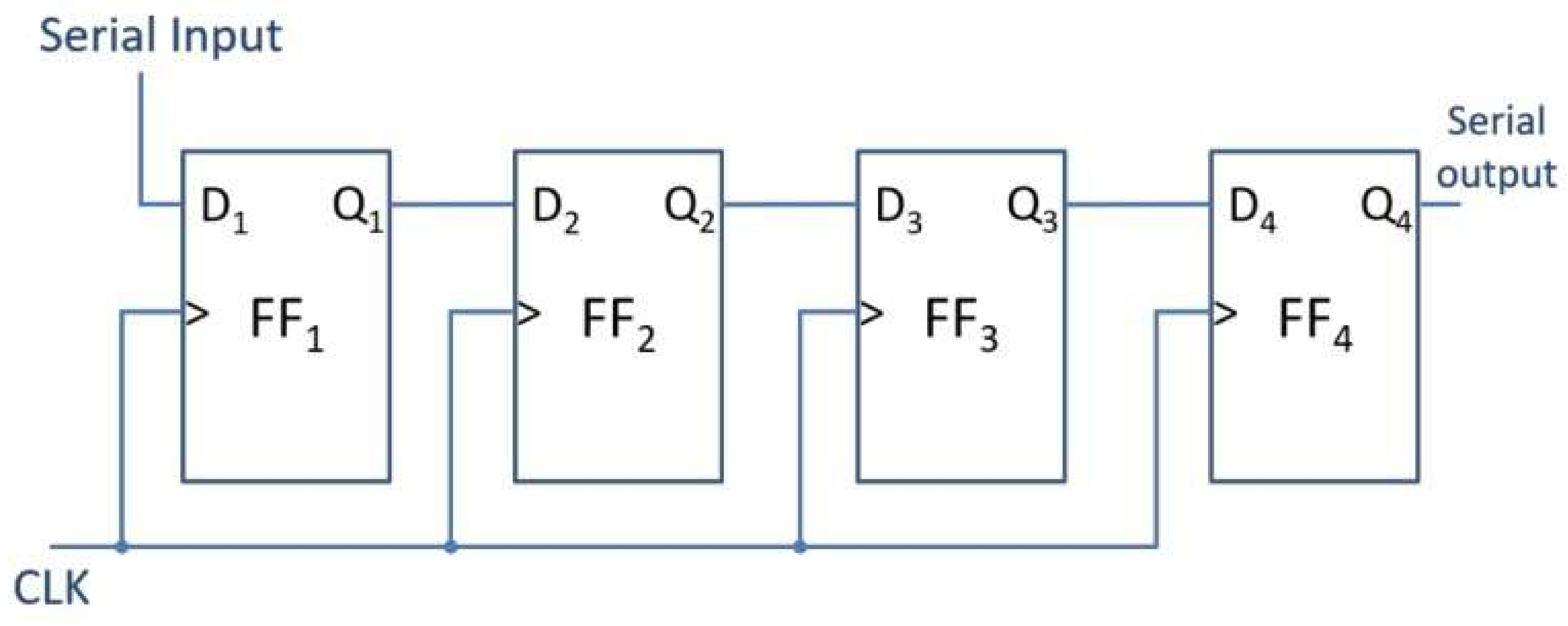


Serial-in, parallel-out, shift register



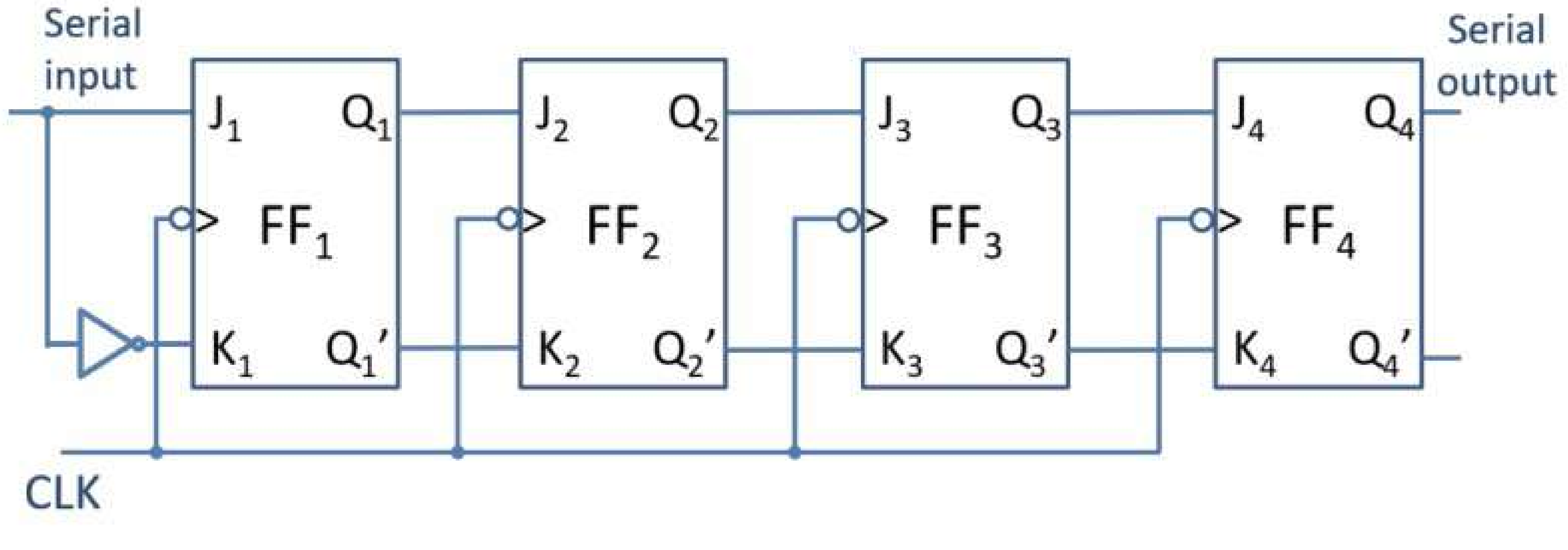
Parallel-in, serial-out, shift register

Serial-in, Serial-out, Shift register



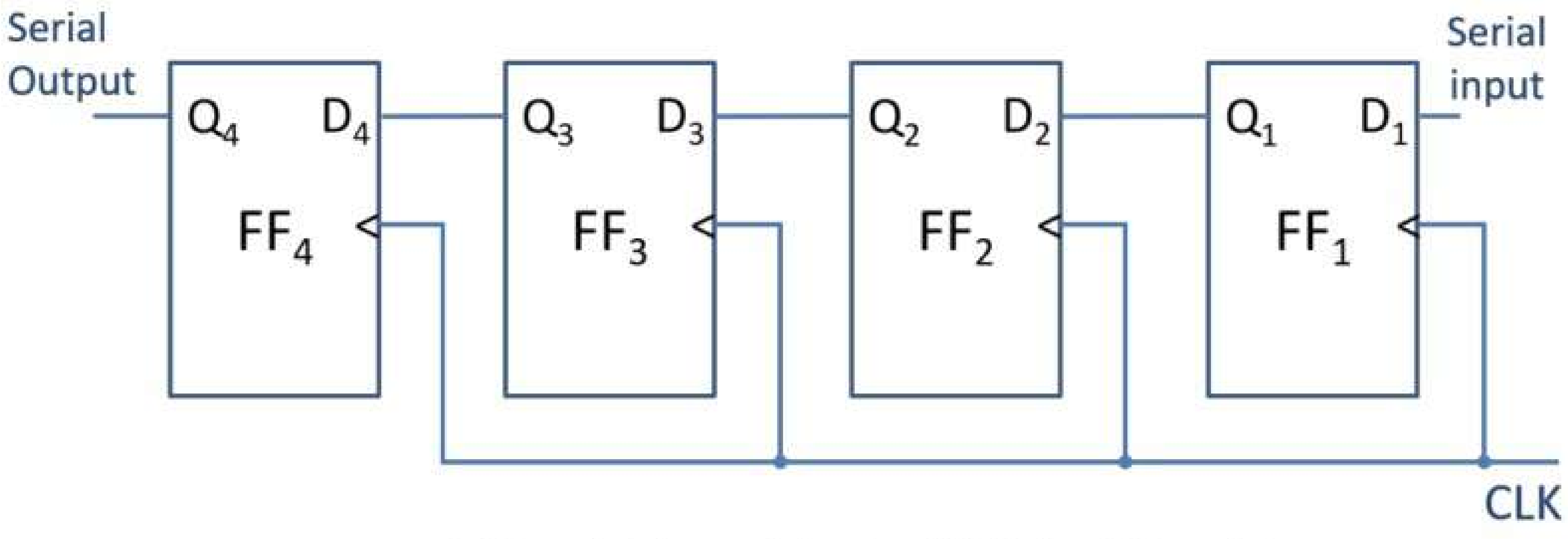
4 bit serial-in, serial-out, shift-right, shift register

- With four stages, i.e. four FFs, the register can store up to four bits.
- 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 fourth FF.
- When serial data is transferred into a register, each new bit is clocked into the first FF at the positive 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.



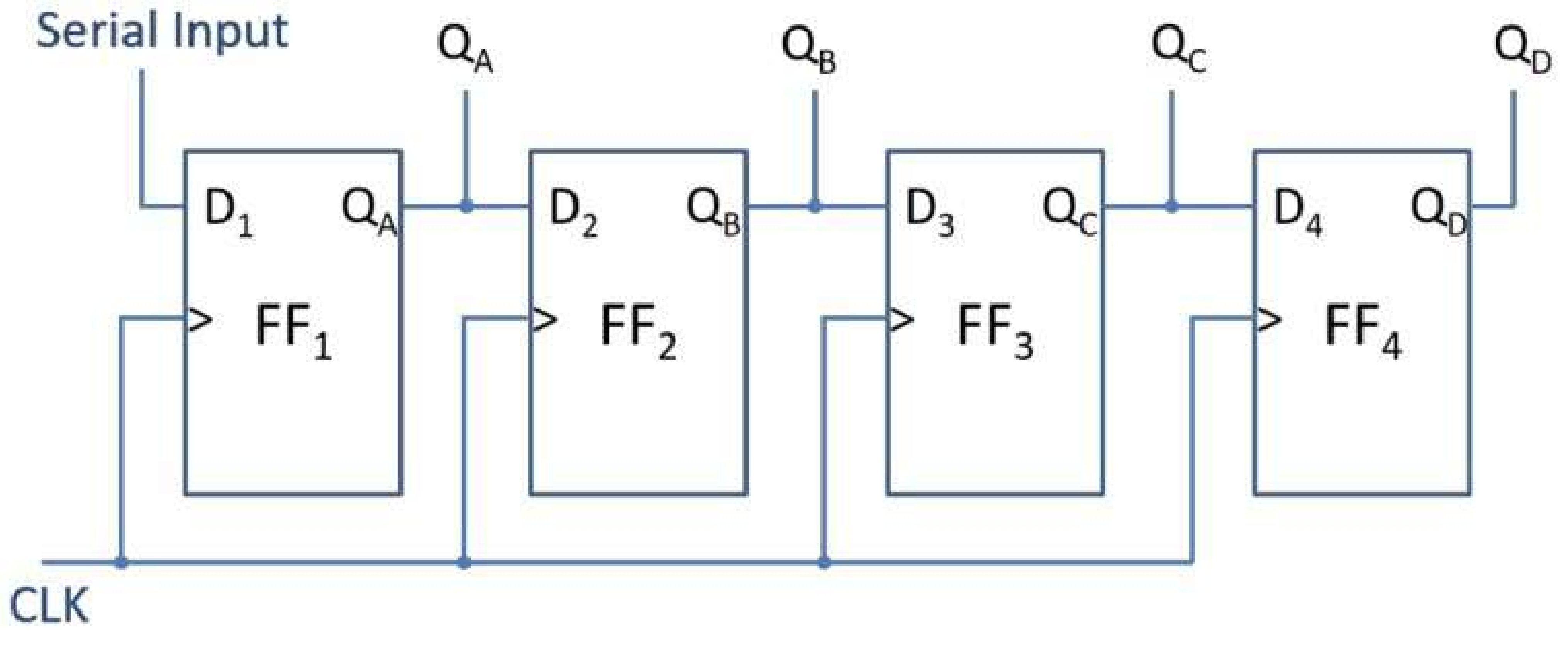
- A shift register can also be constructed using J-K FFs as shown in above figure.
- The data is applied at the J input of the first FF, the complement of this is fed to the K input of FF.

- The Q output of the first FF is connected to J input of the second FF, the Q output of the second FF to J input of the third FF, and so on.
- Also, Q_1' is connected K_2, Q_2' is connected to K_3, and so on.



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

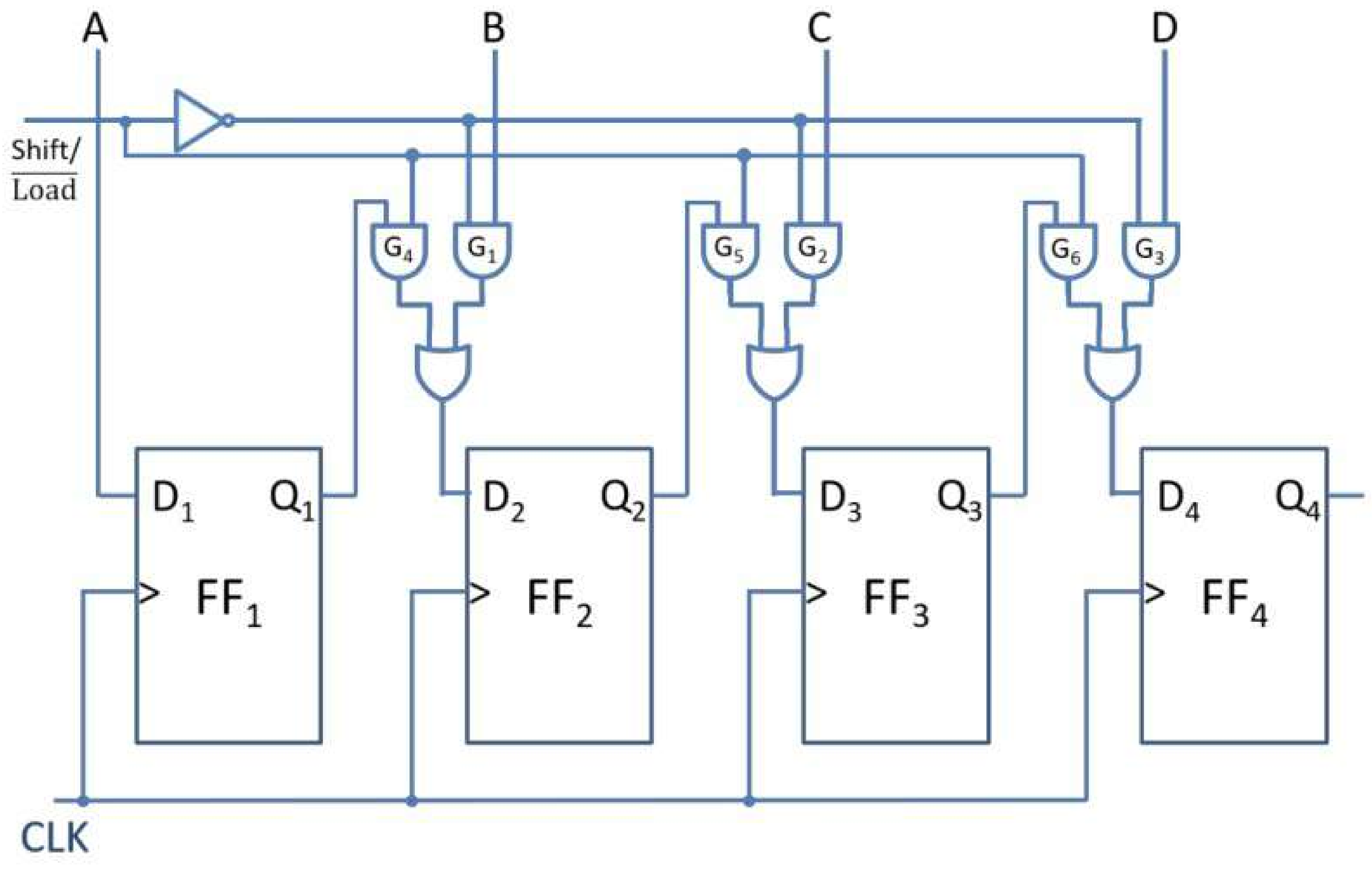
Serial-in, Parallel-out, Shift register



- 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, rather than on a bit-by-bit basis as with the serial output.

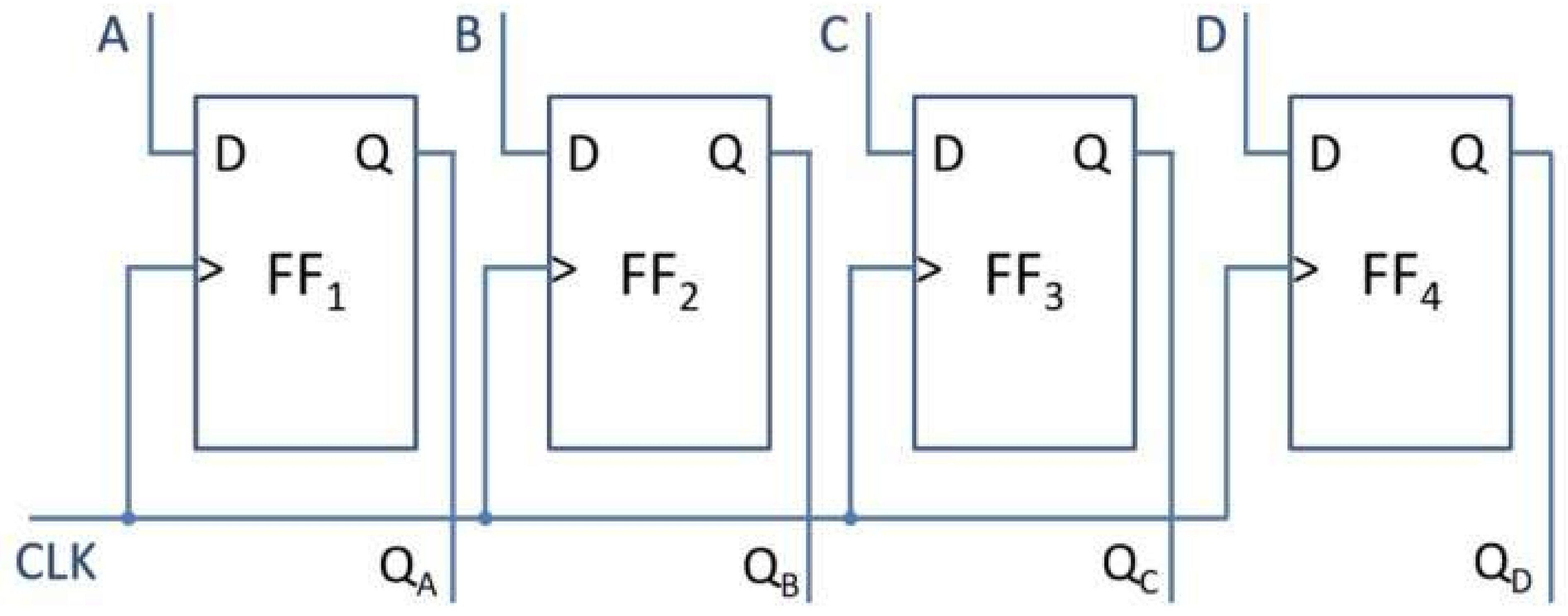
  The serial-in, parallel-out, shift register can be used as a serial-in, serial-out, shift register if the output is taken from Q terminal of the last FF.

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/$\overline{LOAD}$ 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/$\overline{LOAD}$ 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/$\overline{LOAD}$ line is LOW, gates G_4, G_5, and G_6 are disabled, whereas gates G_1, G_2, and G_3 are enabled 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 t the Q output terminals of the FFs and, therefore, data is inputted in one step.
  The OR gate allows wither the normal shifting operation or the parallel data entry depending on which NAD gates are enabled by the level on the Shift/$\overline{LOAD}$ input.

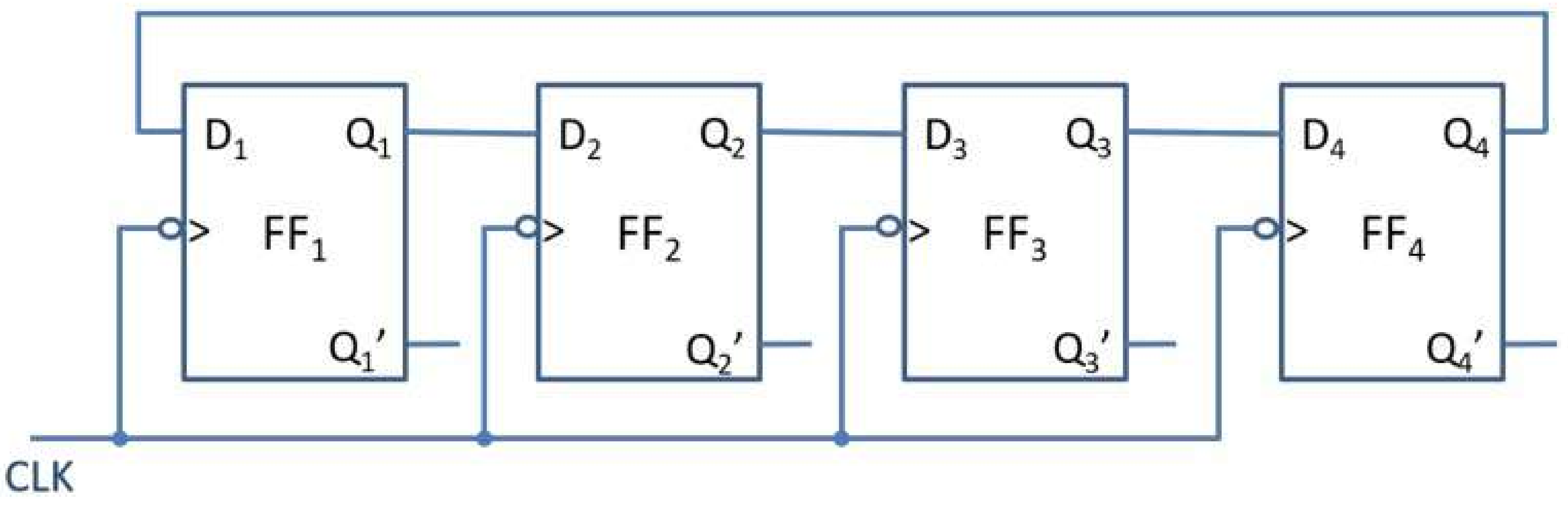
Parallel-in, Parallel-out, Shift register



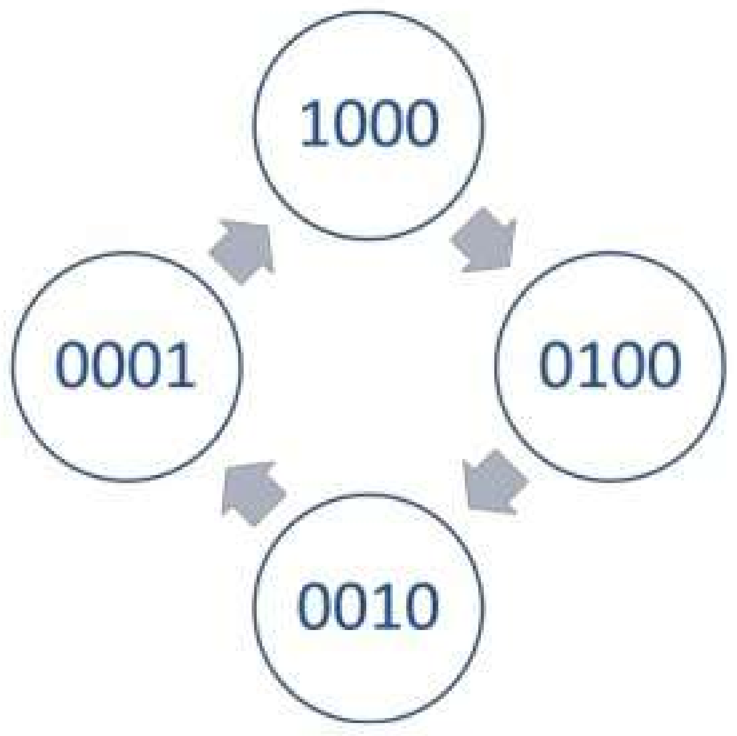
- In a parallel-in, parallel-out, shift register the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form.
- Data is applied to the D input terminals of the FFs.
- When a clock pulse is applied, at the positive-going edge of that pulse, the 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.

Ring counter

- This is the simplest shift register counter. The basic ring counter using D FFs is shown in figure.
- The FFs are arranged as in a normal shift register, i.e. Q output of each stage is connected to the D input of the next stage, but the Q output of the last FF is connected back to the D input of the first FF such that the array of FFs is arranged in a ring, and therefore, the name *ring counter.*



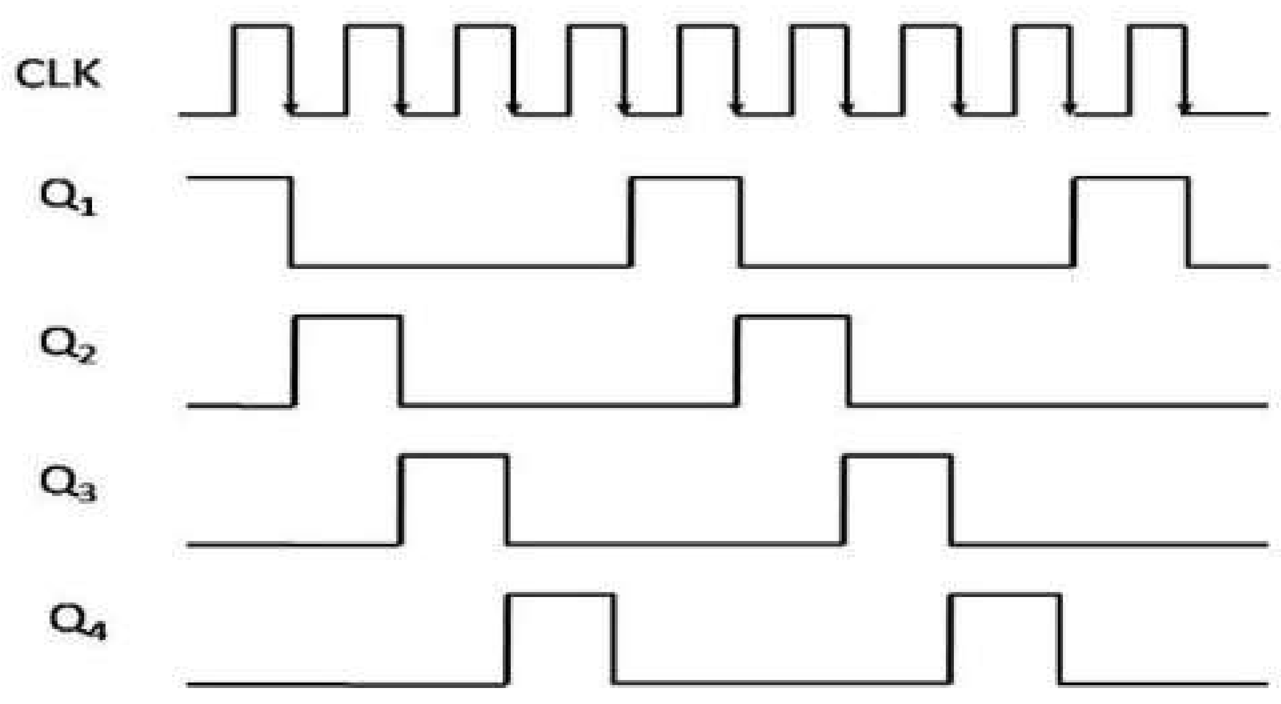
- State diagram:-



- Sequence table

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

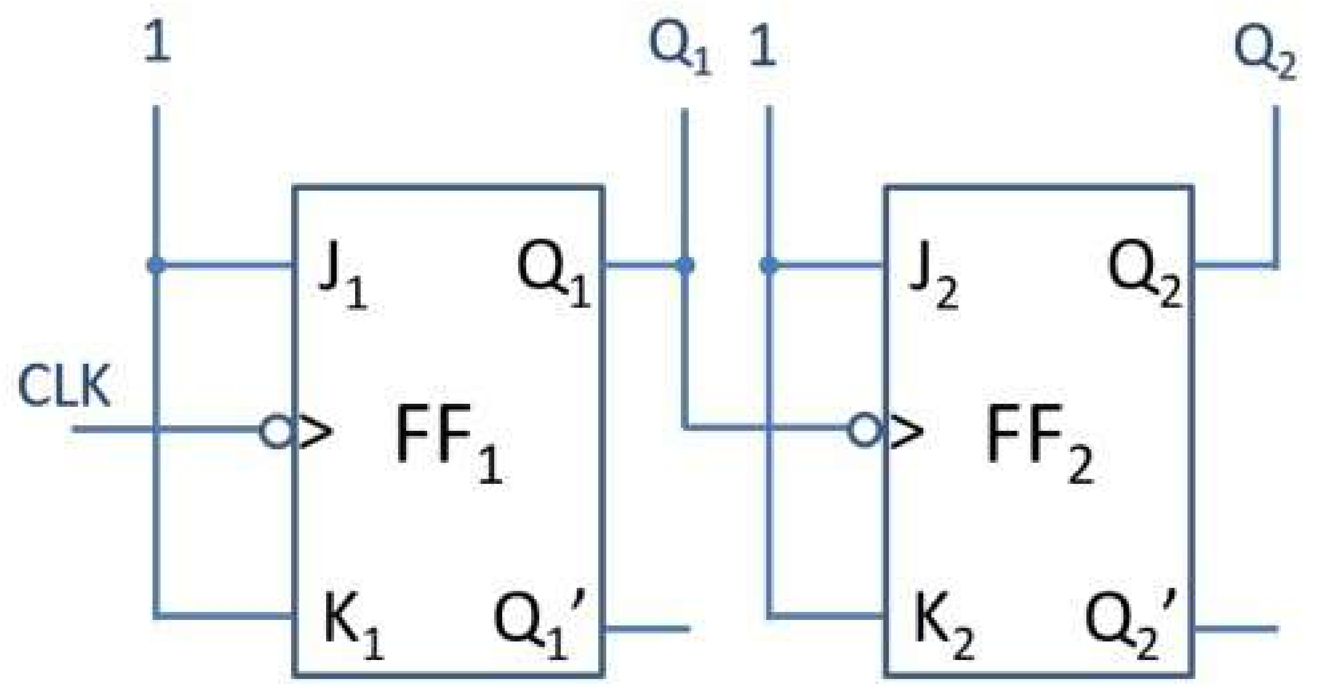
- In most instances, only single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially, the first FF is present to a 1.
- So, the initial state is 1000. After each clock pulse, the contents of the register are shifted to the right by one bit and Q_4 is shifted to Q_1.
- The sequence repeats after four clock pulses.



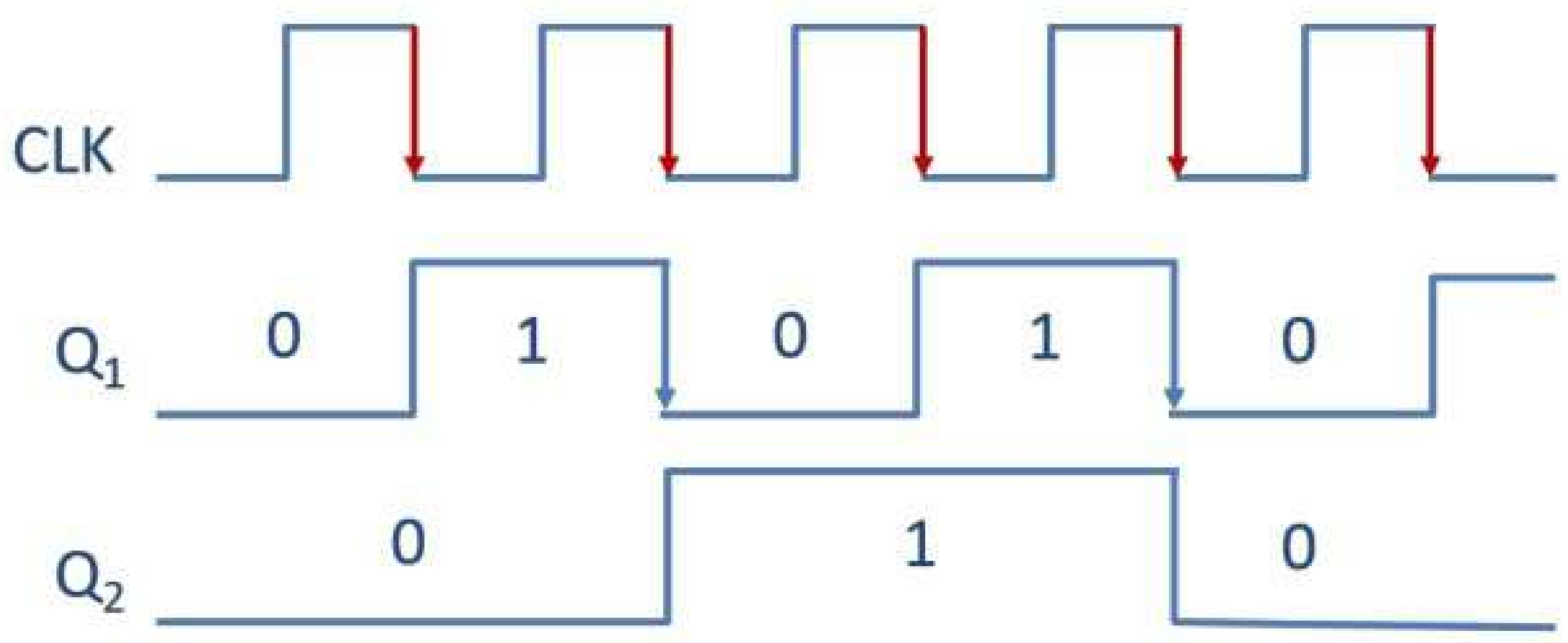
Asynchronous counter V/S Synchronous counter

Asynchronous counter	Synchronous counter
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 drives 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.
5. 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 drives the clock of the third and so on.	5. 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-bit ripple up-counter using negative edge triggered FF

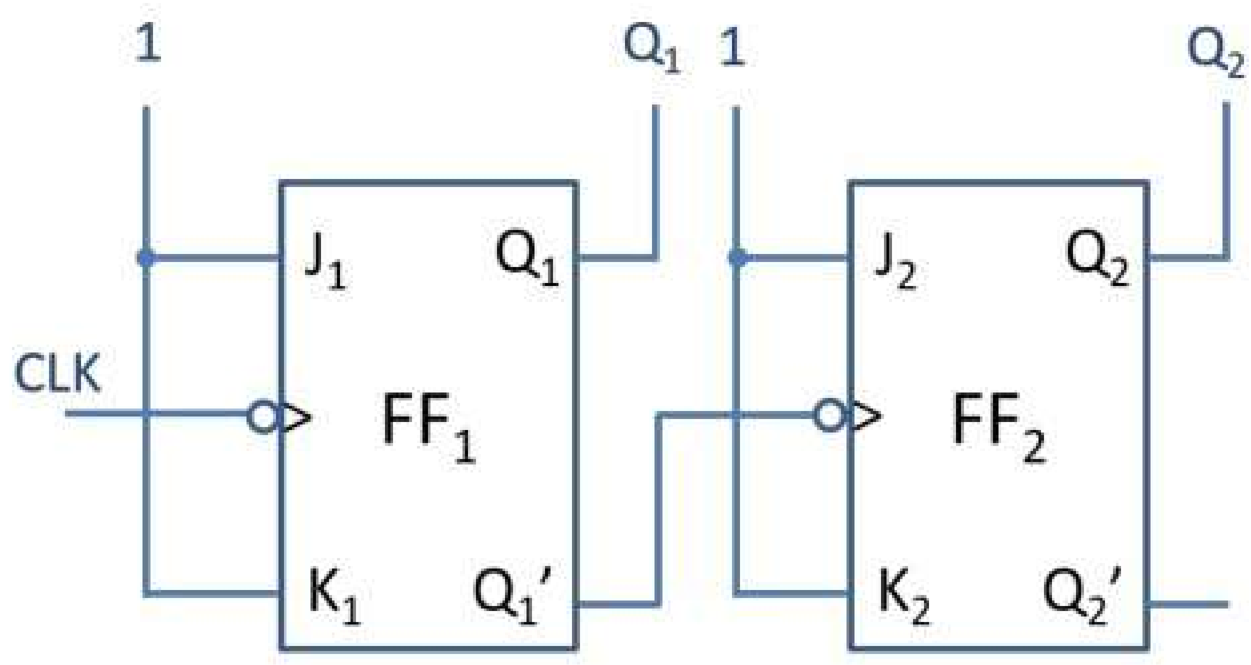


CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↓	0	0	0	1
↓	0	1	1	0
↓	1	0	1	1
↓	1	1	0	0

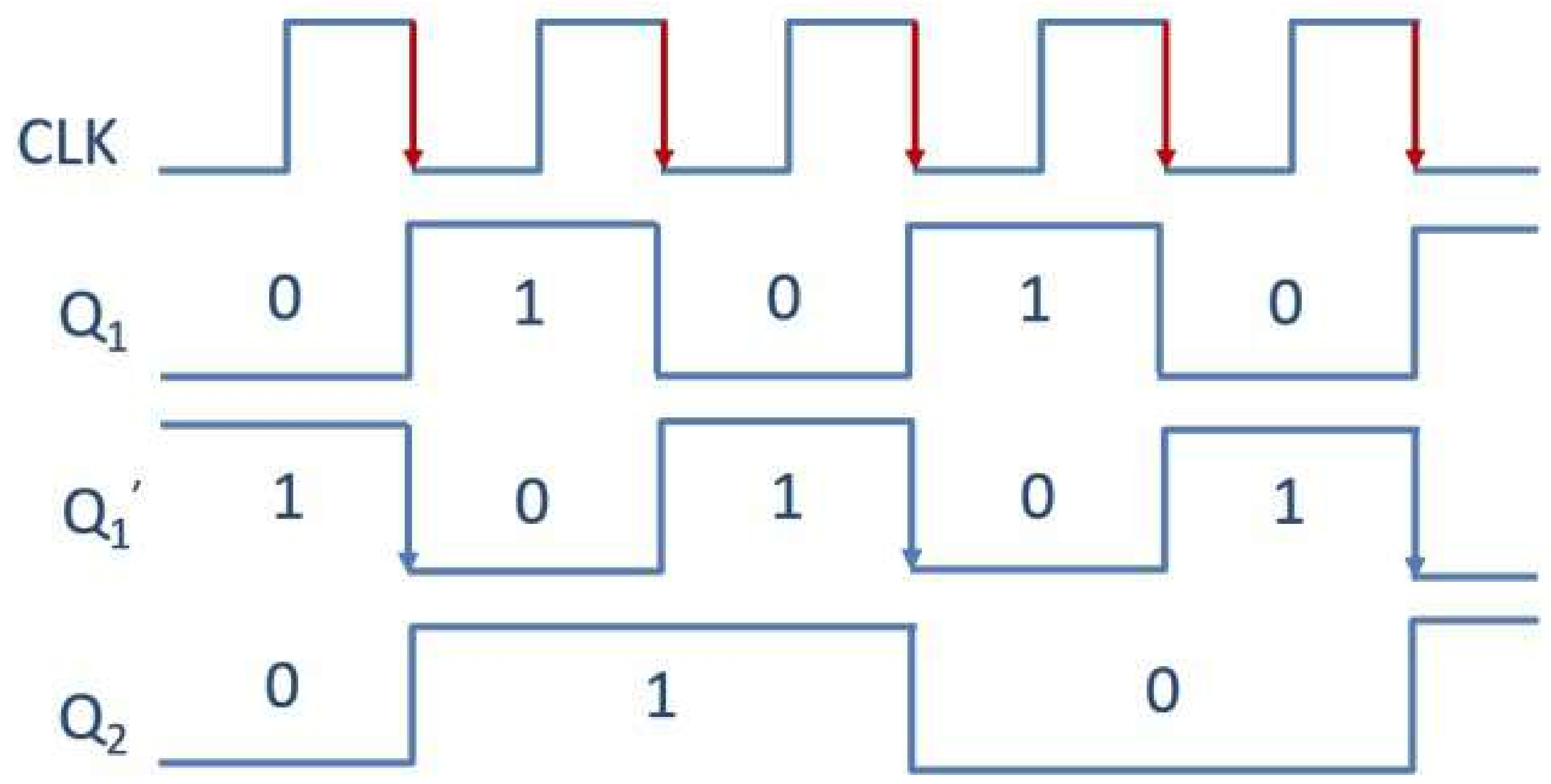


- The 2-bit up-counter counts in the order 0, 1, 2, 3, 0, … etc..
- The counter is initially reset to 00.
- When the first clock pulse is applied, FF_1 toggles at the negative edge of this pulse, therefore, Q_1 goes from LOW to HIGH.
- This becomes a positive edge at the clock input of FF_2. So FF_2 is not affected, and hence the state of the counter after one clock pulse is $Q_1 = 1$ and $Q_2 = 0$, i.e. 01.
- At the negative edge of the second clock pulse, FF_1 toggles.
- So Q_1 changes from HIGH to LOW and this negative edge clock applied to CLK of FF_2 activates FF_2, and hence, Q_2 goes from LOW to HIGH. Therefore, $Q_1 = 0$ and $Q_2 = 1$, i.e. 10 is the state of the counter after the second clock pulse.
- At the negative edge of the third clock pulse, FF_1 toggles.
- So Q_1 changes from a 0 to a 1. This becomes a positive edge to FF_2, hence FF_2 is not affected. Therefore, $Q_2 = 1$ and $Q_1 = 1$, i.e. 11 is the state of the counter after the third clock pulse.
- At the negative edge of the fourth clock pulse, FF_1 toggles.
- So, Q_1 changes from a 1 to a 0. This negative edge at Q_1 toggles FF_2, hence Q_2 also changes from a 1 to a 0. Therefore, $Q_2 = 0$ and $Q_1 = 0$, i.e. 00 is the state of the counter after the fourth clock pulse.
- So, it acts as a mod-4 counter with Q_1 as the LSB and Q_2 as the MSB.

2-bit ripple down-counter using negative edge triggered FF



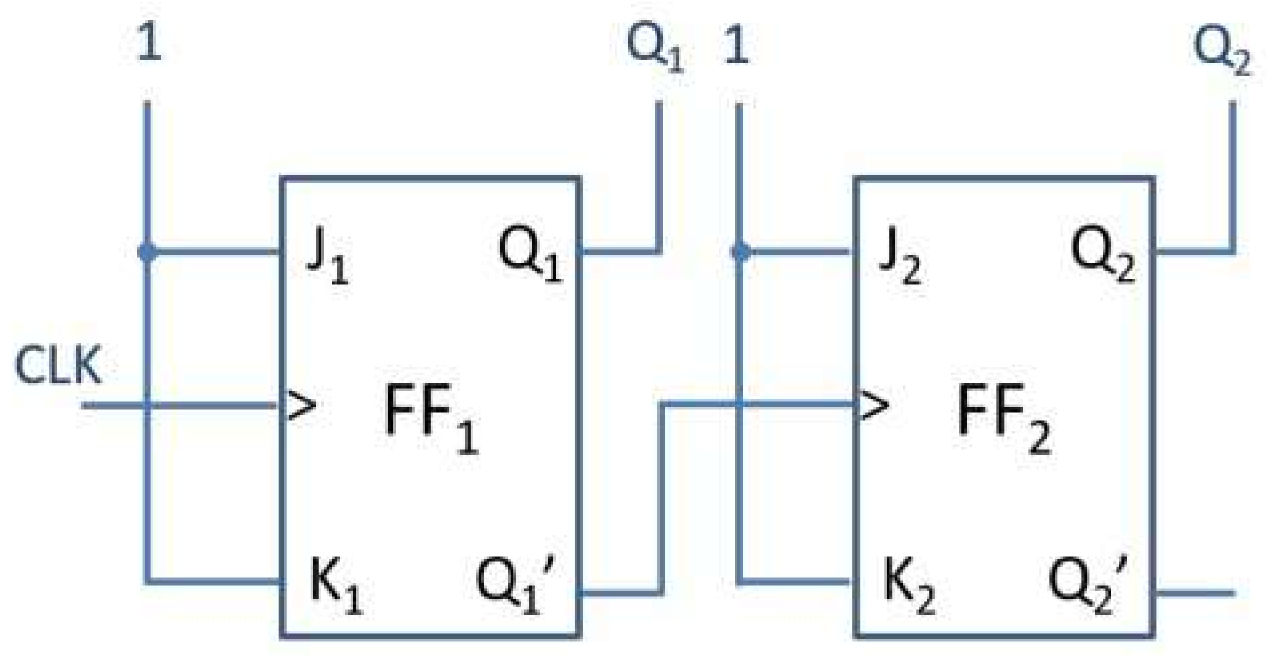
CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↓	0	0	1	1
↓	1	1	1	0
↓	1	0	0	1
↓	0	1	0	0



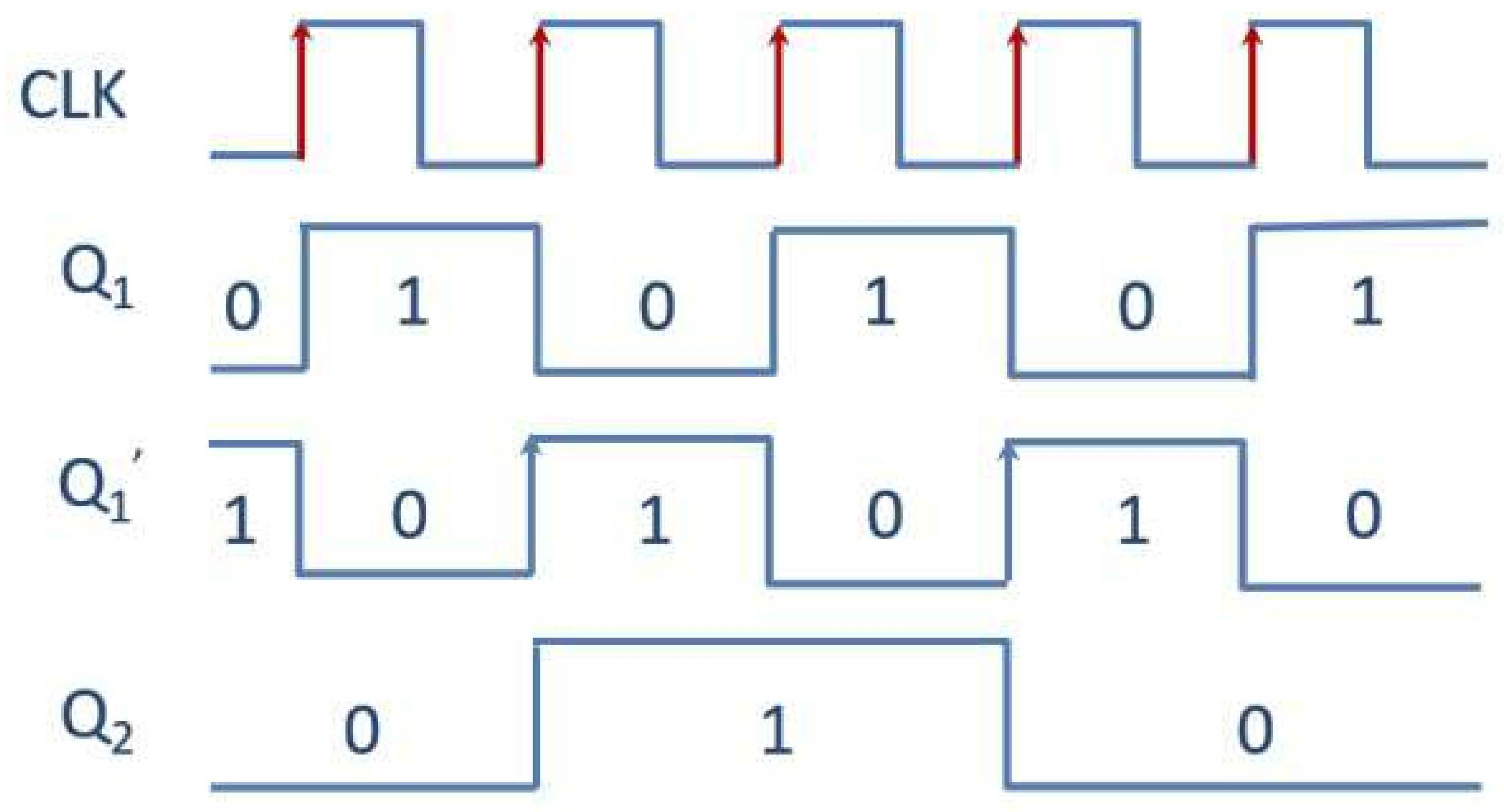
- A 2-bit down-counter counts in the order 0, 3, 2, 1, 0, …etc.
- For down counting, Q_1' of FF_1 is connected to the clock of FF_2. Let initially all the FFs be reset, i.e. 00.

- At the negative edge of the first clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
- This negative edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 0 to a 1. So, after one clock pulse $Q_2 = 1$ and $Q_1 = 1$, i.e. the state of the counter is 11.
- At the negative edge of the second clock pulse, Q_1 changes from a 1 to a 0 and Q_1' from a 0 to a 1.
- This positive edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 1. Hence, the state of the counter after second clock pulse is 10.
- At the negative edge of third clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
- This negative edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 1 to a 0. Hence, the state of the counter is 01.
- At the negative edge of fourth clock pulse, FF_1 toggles. So, Q_1 goes from a 1 to a 0 and Q_1' goes from a 0 to a 1.
- This positive edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 0. Hence, the state of the counter after fourth clock pulse is 00.

2-bit ripple up-counter using positive edge triggered FF



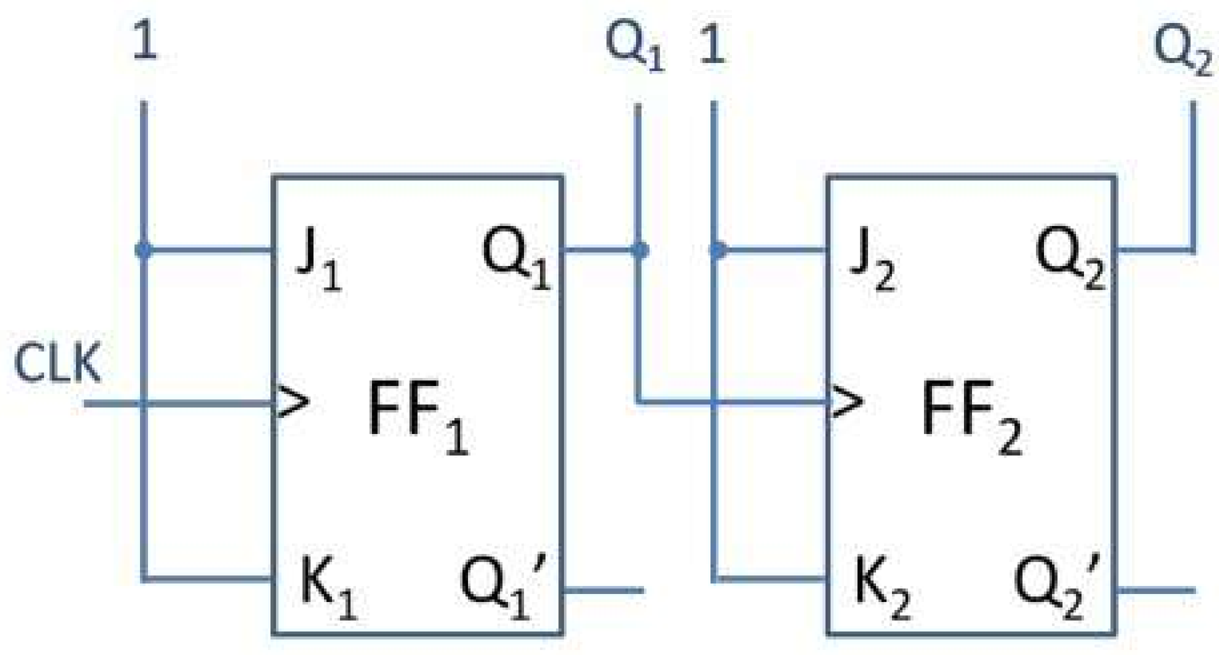
CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↑	0	0	0	1
↑	0	1	1	0
↑	1	0	1	1
↑	1	1	0	0



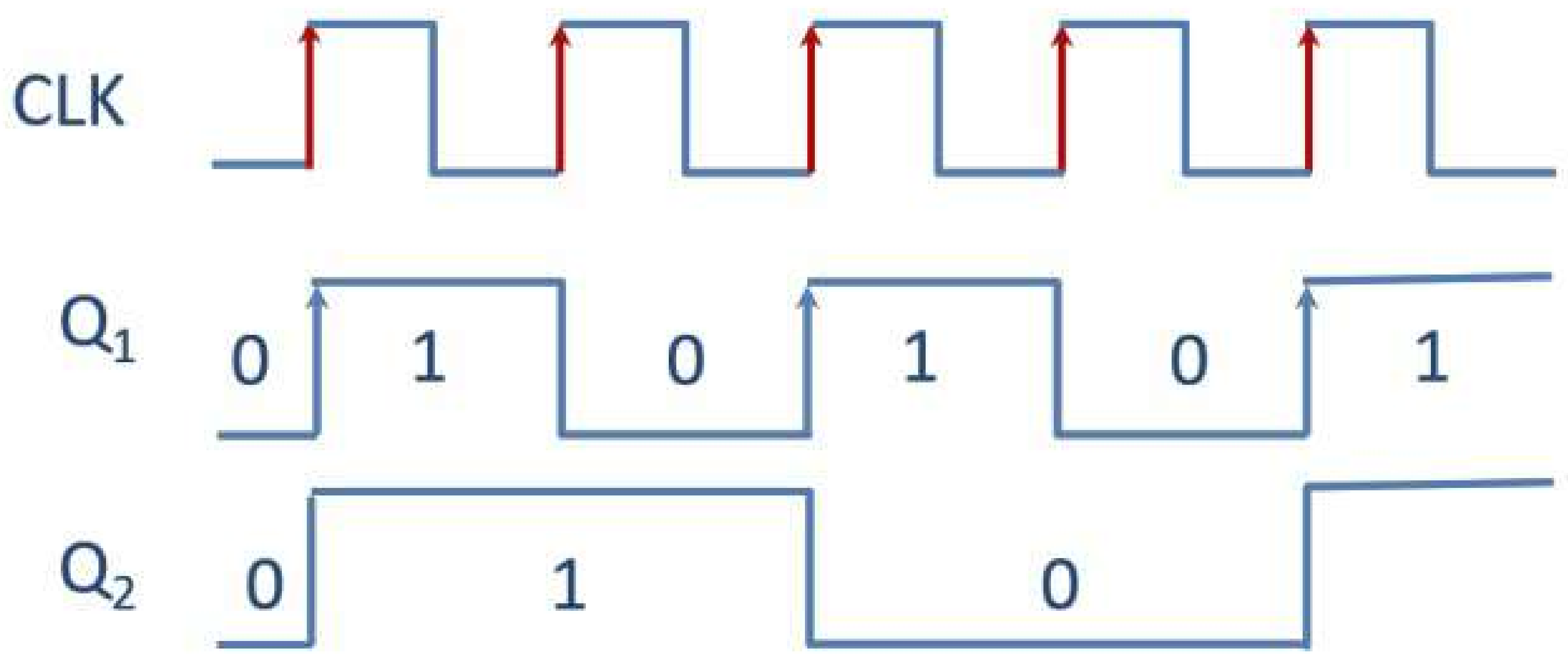
- A 2-bit up-counter counts in the order 0, 1, 2, 3, 0, …etc.
- For up counting in positive edge triggering, Q_1' of FF_1 is connected to the clock of FF_2. Let initially all the FFs be reset, i.e. 00.

- At the positive edge of the first clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
- This negative edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 0. Hence, the state of the counter after first clock pulse is 01.
- At the positive edge of the second clock pulse, Q_1 changes from a 1 to a 0 and Q_1' from a 0 to a 1.
- This positive edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 0 to a 1. Hence, the state of the counter is 10.
- At the positive edge of third clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
- This negative edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 1. Hence, the state of the counter after third clock pulse is 11.
- At the positive edge of fourth clock pulse, FF_1 toggles. So, Q_1 goes from a 1 to a 0 and Q_1' goes from a 0 to a 1.
- This positive edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 1 to a 0. Hence, the state of the counter is 00.

2-bit ripple down-counter using positive edge triggered FF



CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↑	0	0	1	1
↑	1	1	1	0
↑	1	0	0	1
↑	0	1	0	0



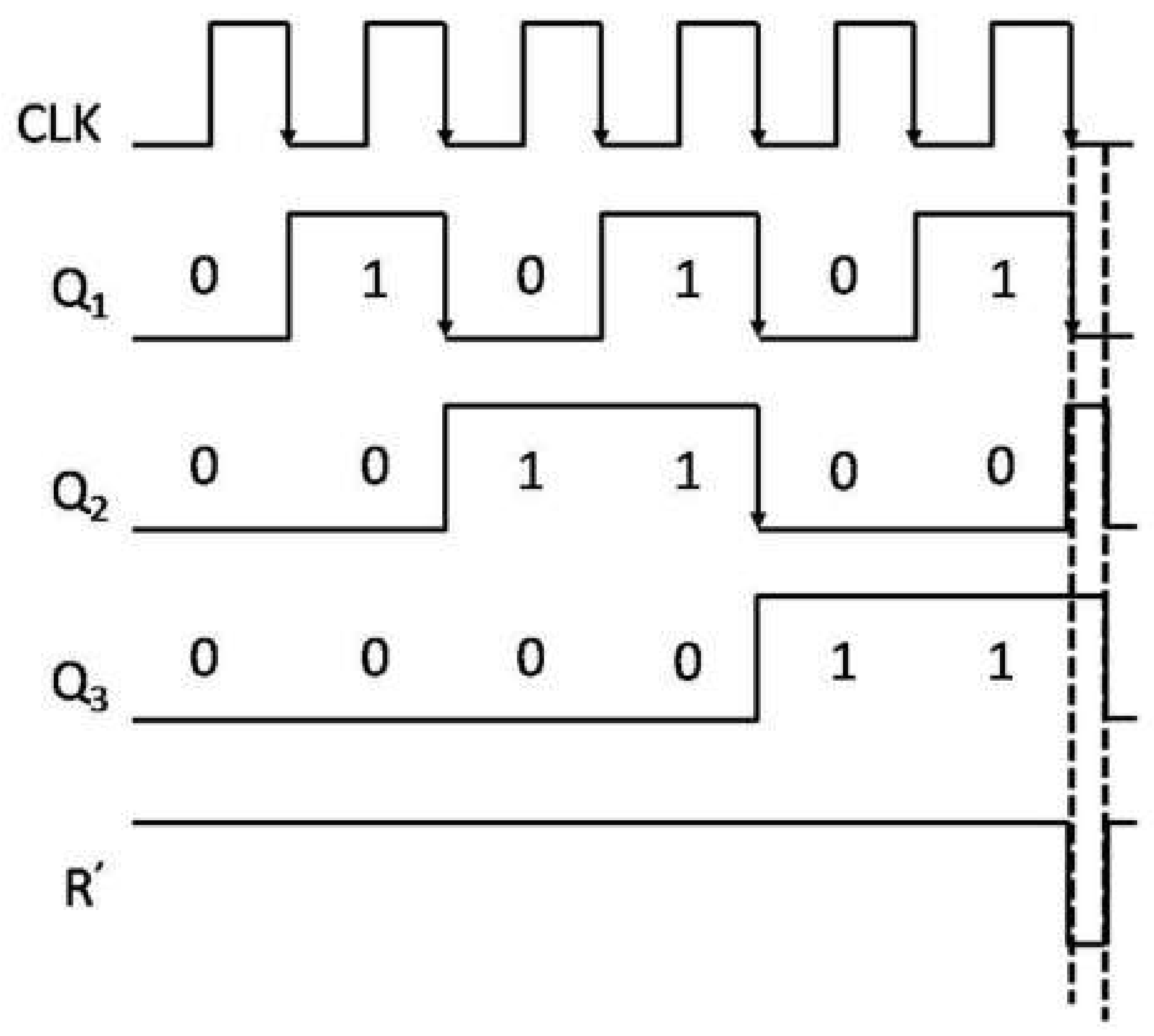
- The 2-bit down-counter counts in the order 0, 3, 2, 1, 0, … etc..
- The counter is initially reset to 00.
- When the first clock pulse is applied, FF_1 toggles at the positive edge of this pulse, therefore, Q_1 goes from LOW to HIGH.
- This positive edge at Q_1 toggles FF_2, hence Q_2 also changes from a 0 to a 1. Therefore, Q_2 = 1 and Q_1 = 1, i.e. 11 is the state of the counter after the first clock pulse.

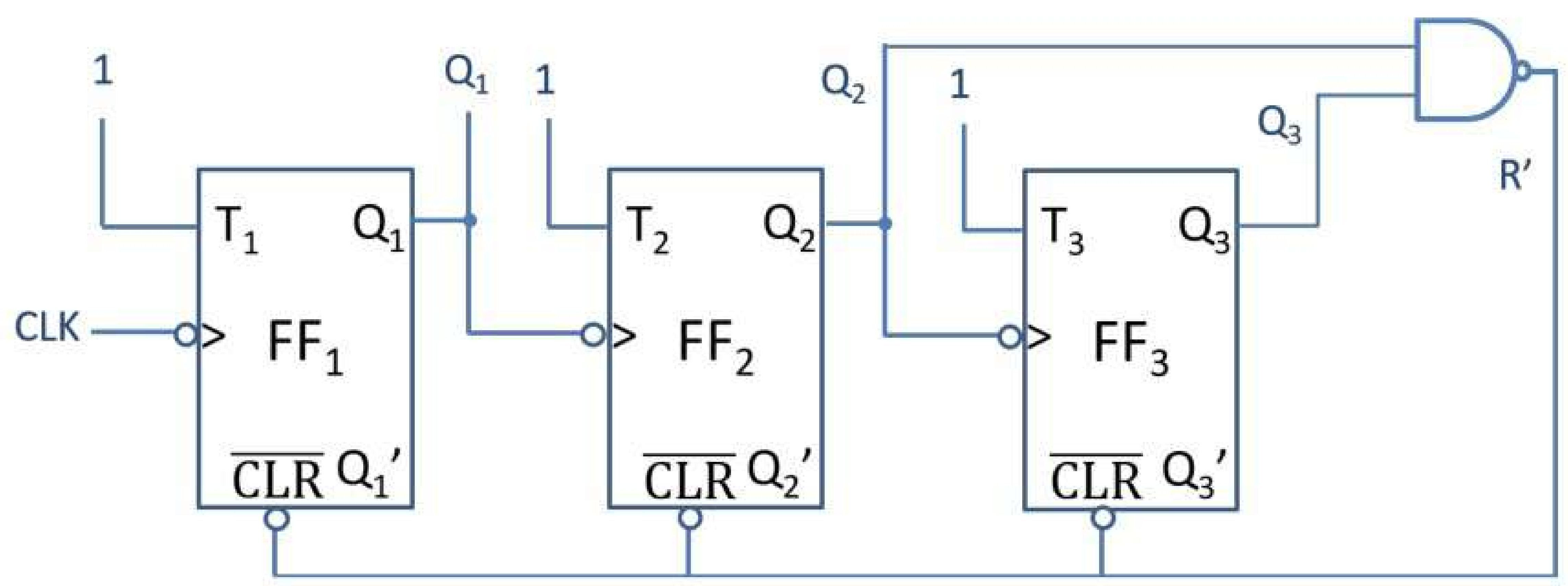
- At the positive edge of the second clock pulse, FF_1 toggles.
- So Q_1 changes from a 1 to a 0. This becomes a negative edge to FF_2, hence FF_2 is not affected. Therefore, $Q_2 = 1$ and $Q_1 = 0$, i.e. 10 is the state of the counter after the second clock pulse.
- At the positive edge of the third clock pulse, FF_1 toggles.
- So Q_1 changes from a 0 to a 1. This positive edge at Q_1 toggles FF_2, hence Q_2 also changes from a 1 to a 0. Therefore, $Q_2 = 0$ and $Q_1 = 1$, i.e. 01 is the state of the counter after the third clock pulse.
- At the positive edge of the fourth clock pulse, FF_1 toggles.
- So Q_1 changes from a 1 to a 0. This becomes a negative edge to FF_2, hence FF_2 is not affected. Therefore, $Q_2 = 0$ and $Q_1 = 0$, i.e. 00 is the state of the counter after the fourth clock pulse.

Modulo-6 asynchronous counter

- A mod-6 counter has six stable states 000, 001, 010, 011, 100, 101.
- It is also known as "Divide by 6" counter and it requires 3 Flip-flops for designing.
- At the 6th clock pulse, it will again reset to 000 via feedback circuit.
- Reset signal R = 1 at time of 110, R = 0 for 000 to 101 and R = X for invalid states i.e. 111.
- Therefore, $R = Q_3Q_2Q_1' + Q_3Q_2Q_1 = Q_3Q_2$.

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	




1
Q_1
1
Q_2
1
Q_3
R′
CLK
T_1 Q_1 FF_1 $\overline{CLR}$ Q_1'
T_2 Q_2 FF_2 $\overline{CLR}$ Q_2'
T_3 Q_3 FF_3 $\overline{CLR}$ Q_3'

Synchronous counter designing

- Step 1. Number of flip-flops:
  Based on the description of the problem, determine the required number n of the FFs - the smallest value of n is such that the number of states $N \leq 2^n$ and the desired counting sequence.
- Step 2. State diagram:
  Draw the state diagram showing all the possible states.
- Step 3. Choice of flip-flops and excitation table:
  Select the type of flip-flops to be used and write the excitation table.
  An excitation table is a table that lists the present state (PS), the next state (NS) and the required excitations.
- Step 4. Minimal expressions for excitations:
  Obtain the minimal expressions for the excitations of the FFs using K-maps for the excitations of the flip-flops in terms of the present states and inputs.
- Step 5. Logic Diagram:
  Draw the logic diagram based on the minimal expressions.

Flip-Flop Excitation Tables

(1) S-R Flip-flop excitation table

PS	NS	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

(2) J-K Flip-flop excitation table

PS	NS	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

(3) D Flip-flop excitation table

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

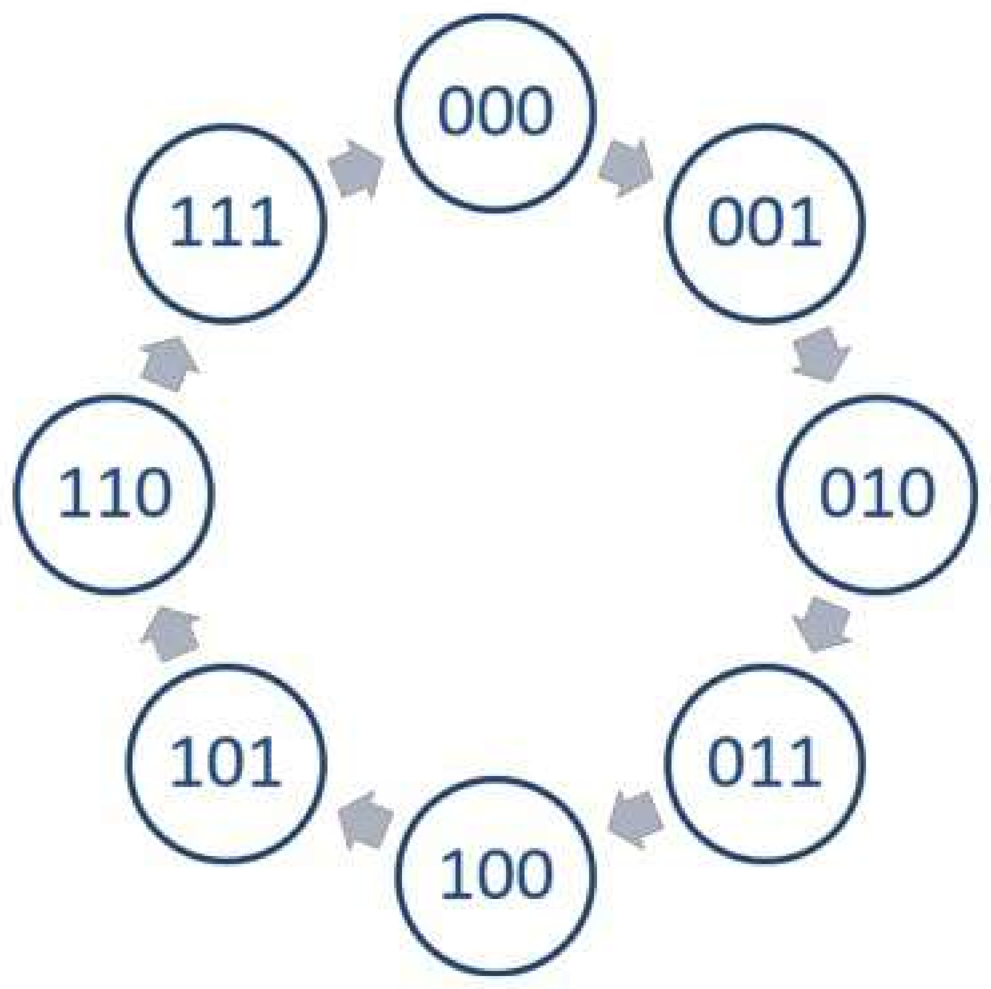
(4) T Flip-flop excitation table

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

Synchronous 3-bit up counter

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

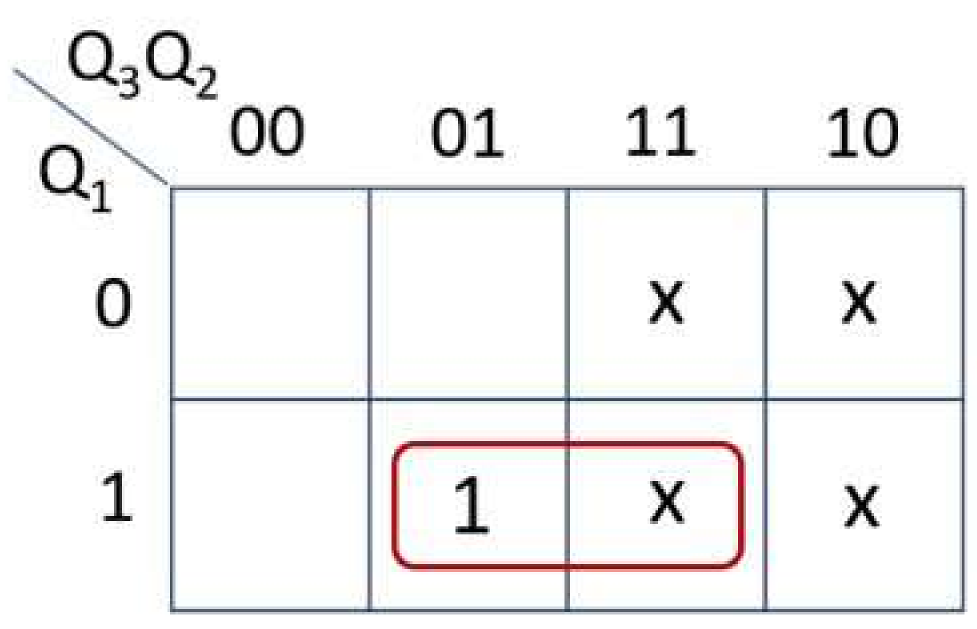
- Step 2. Draw the state diagram:



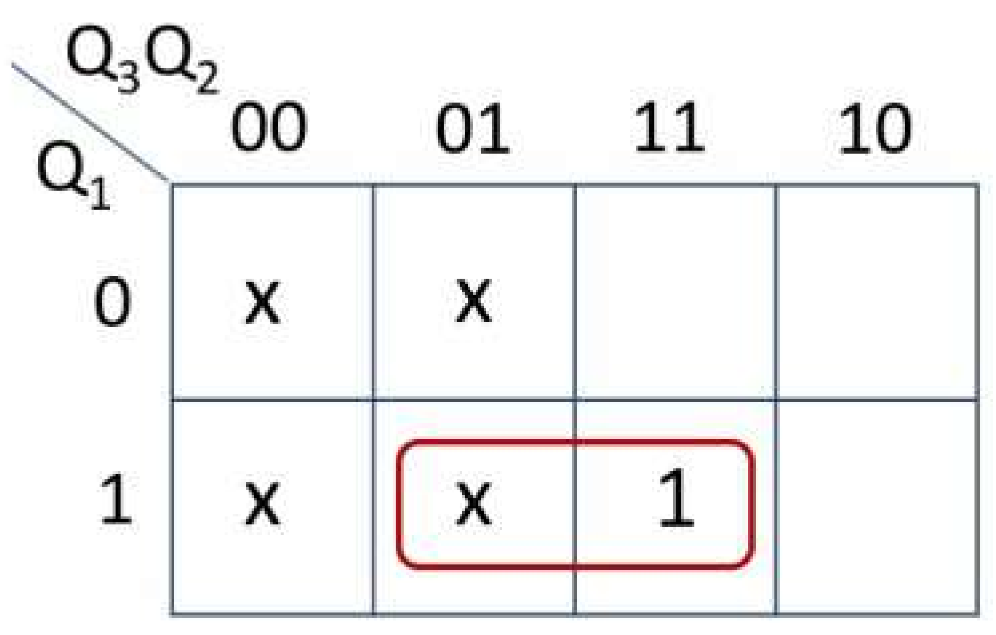
- Step 3. Select the type of flip-flops and draw the excitation table:
  JK flip-flops are selected and the excitation table of a 3-bit up-counter using J-K flip-flops is drawn as shown below.

Present State			Next State			Required Excitation					
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

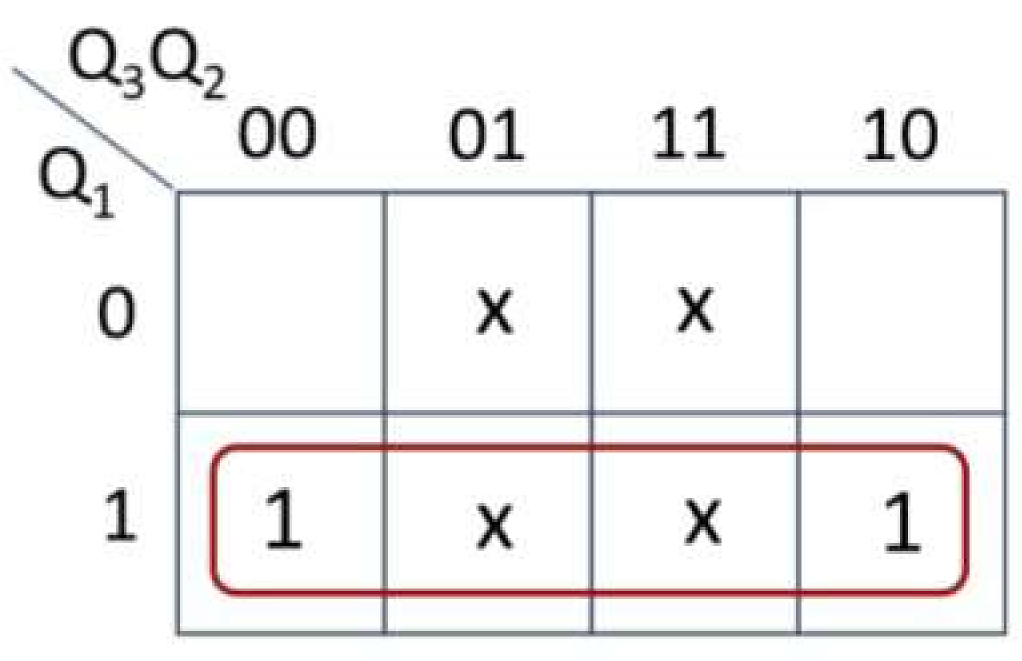
- Step 4. Obtain the minimal expressions
  From excitation table, $J_1 = K_1 = 1$.
  K – Maps for excitations J_3, K_3, J_2 and K_2 and their minimized form are as follows:



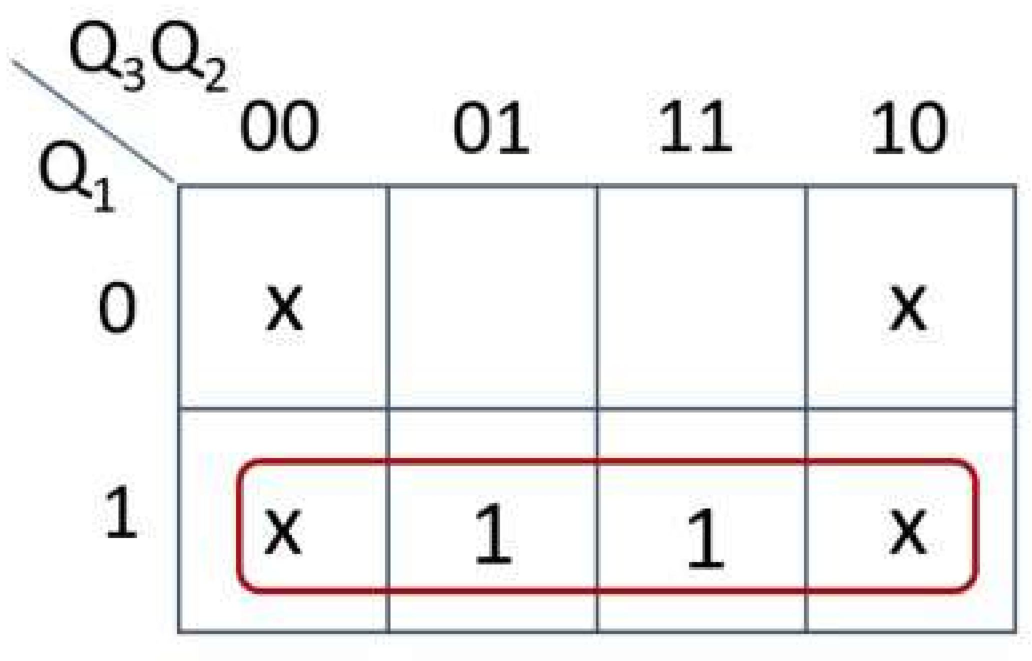
$J_3 = Q_2Q_1$



$K_3 = Q_2Q_1$

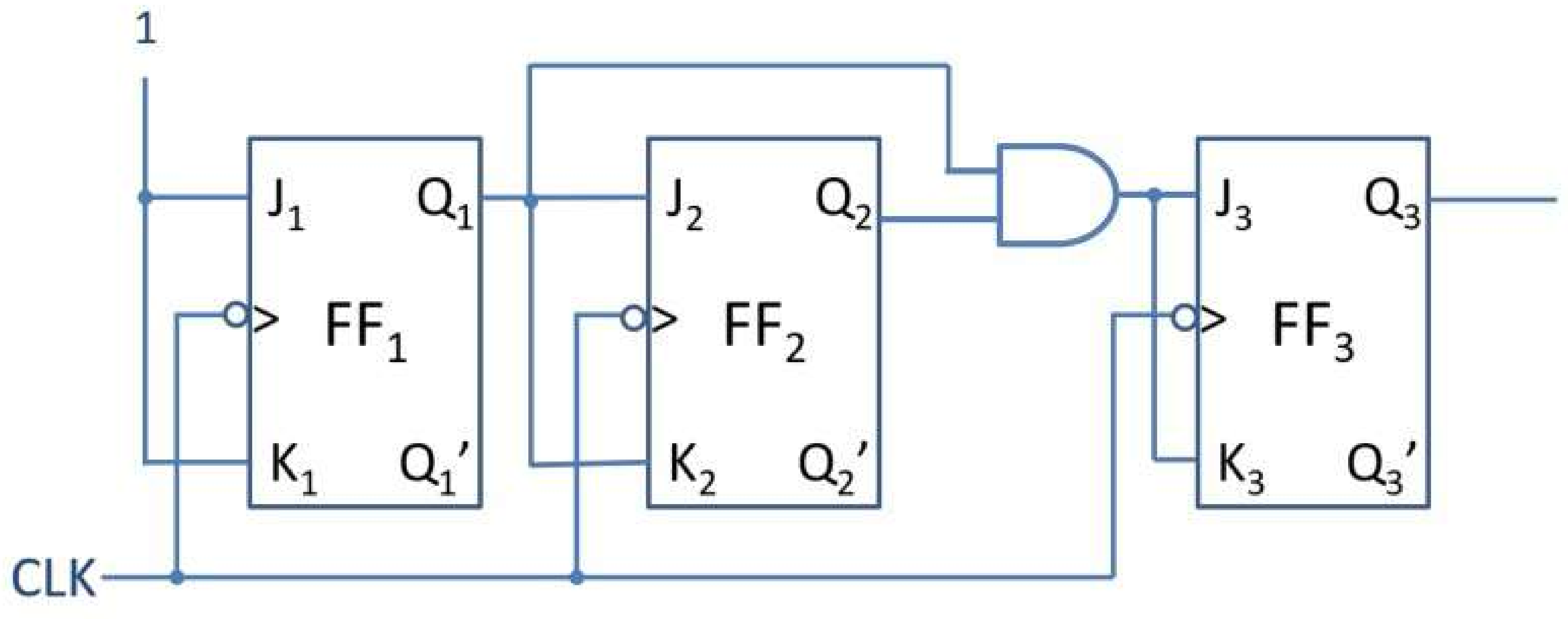


$J_2 = Q_1$



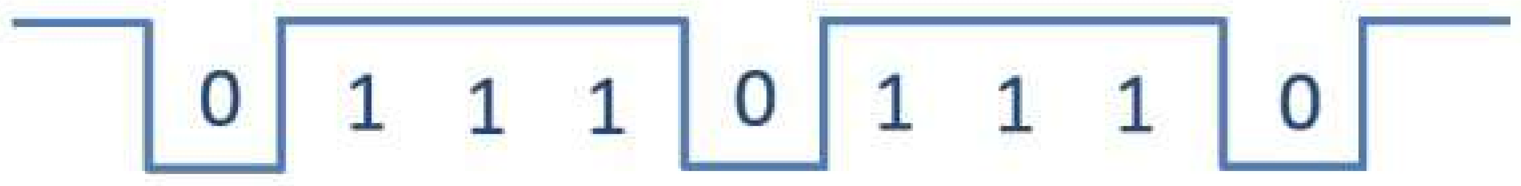
$K_2 = Q_1$

- Step 5. Draw the logic diagram



Sequence generator using direct logic (Example)

- Step 1. Inspect given pule train

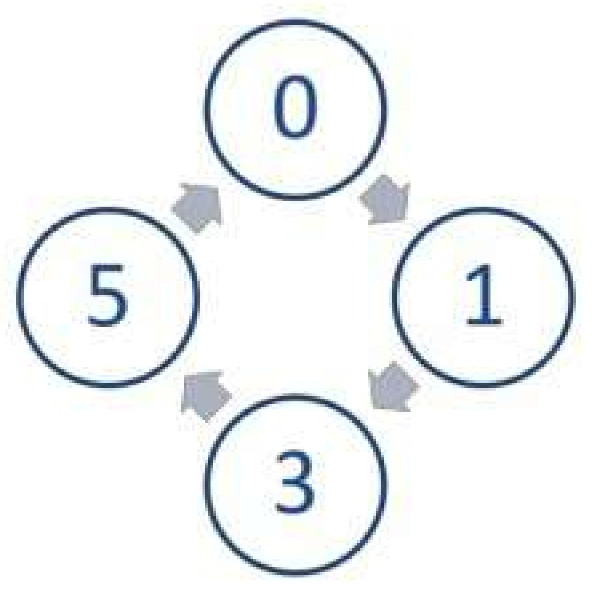


- Step 2. Decide the number of unique states and minimum number of FFs required. If unique states are not possible with the least number of FFs n, then increase the number of FFs by one or more to get the unique states.

FF States	
	LSB
0	0
0	1
1	1
?	1

FF States			Decimal equivalent
		LSB	
0	0	0	0
0	0	1	1
0	1	1	3
1	0	1	5

State assignment



State diagram

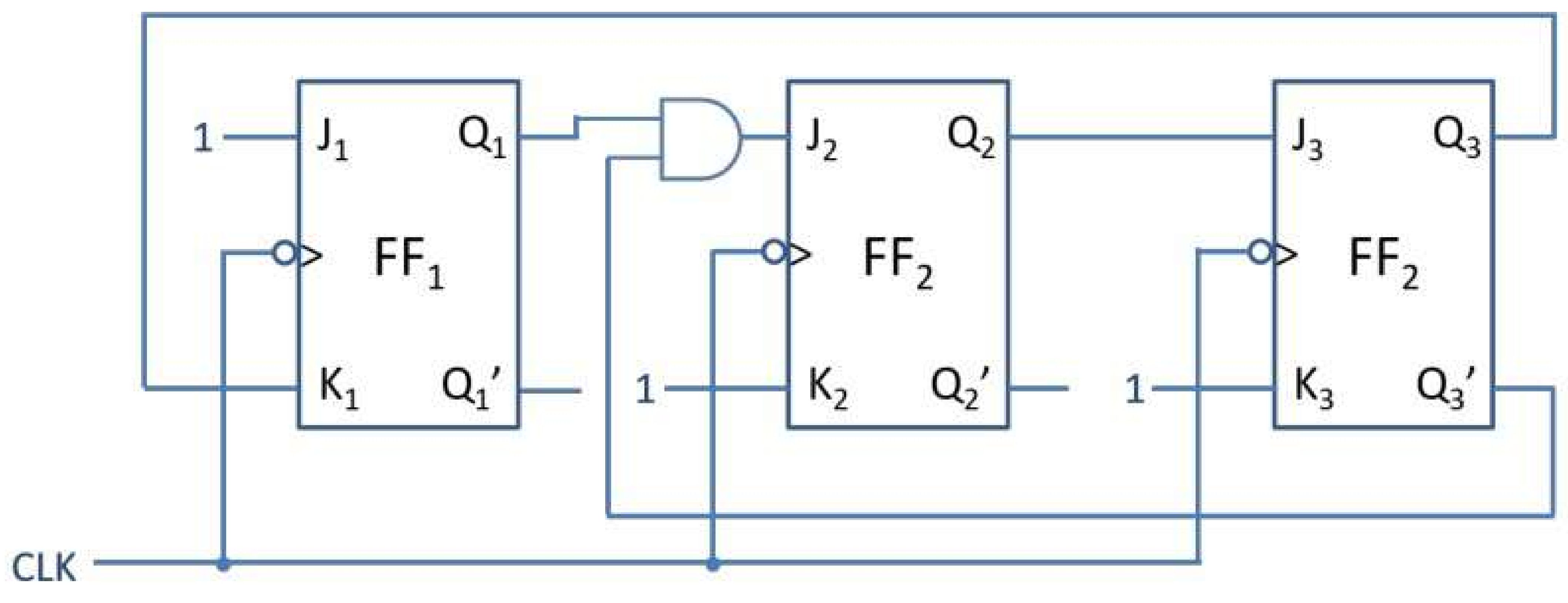
- Step 3. Select the type of flip-flops and draw the excitation table:
  JK flip-flops are selected and the excitation table of a given sequence using J-K flip-flops is drawn as shown below.

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	1	0	X	1	X	X	0
0	1	1	1	0	1	1	X	X	1	X	0
1	0	1	0	0	0	X	1	0	X	X	1

- Step 4. Obtain the minimal expressions
  Using K – Maps, we can obtain minimal expressions as shown in below.

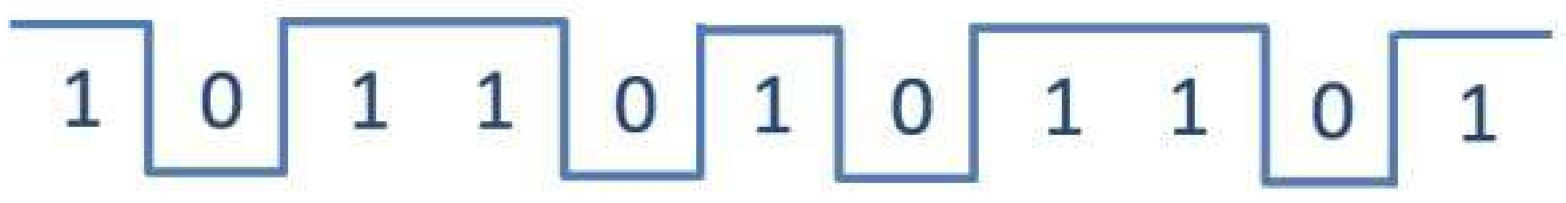
  $J_3 = Q_2$, $K_3 = 1$ $J_2 = Q_3'Q_1$, $K_2 = 1$ $J_1 = 1$, $K_1 = Q_3$

- Step 5. Draw the logic diagram



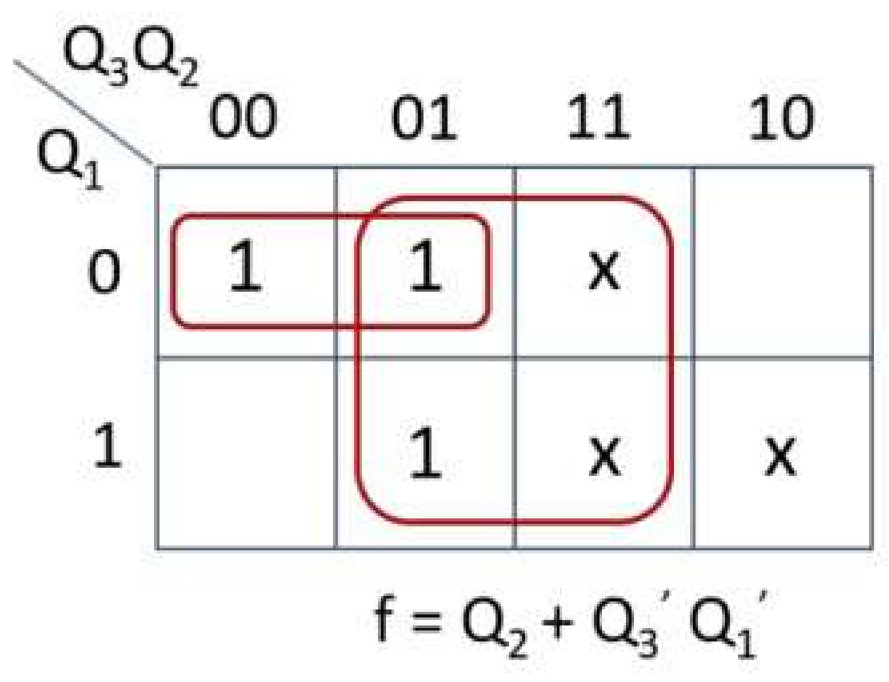
Sequence generator using indirect logic (Example)

- Step 1. Inspect given pule train

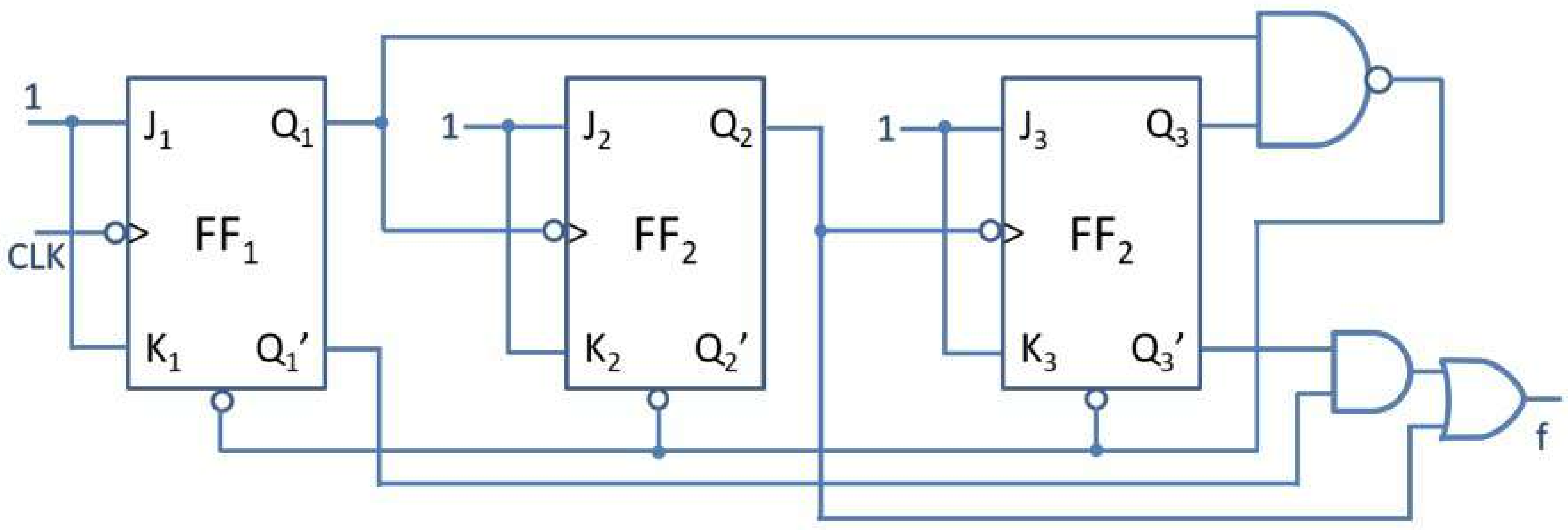


- Step 2. Obtain output function f to generate the inspected sequence and also obtain minimal expression of function f using K – Map.
  We need to reset counter to 000 at the state 101, so expression of R should be Q_3Q_1.

Q_3	Q_2	Q_1	Output(f)	States
0	0	0	1	0
0	0	1	0	1
0	1	0	1	2
0	1	1	1	3
1	0	0	0	4
1	0	1	X	5
1	1	0	X	6
1	1	1	X	7



- Step 3. Draw the logic diagram



1
CLK
J_1
Q_1
FF_1
K_1
Q_1'
1
J_2
Q_2
FF_2
K_2
Q_2'
1
J_3
Q_3
FF_2
K_3
Q_3'
f