Unit 5 and 6: Combined

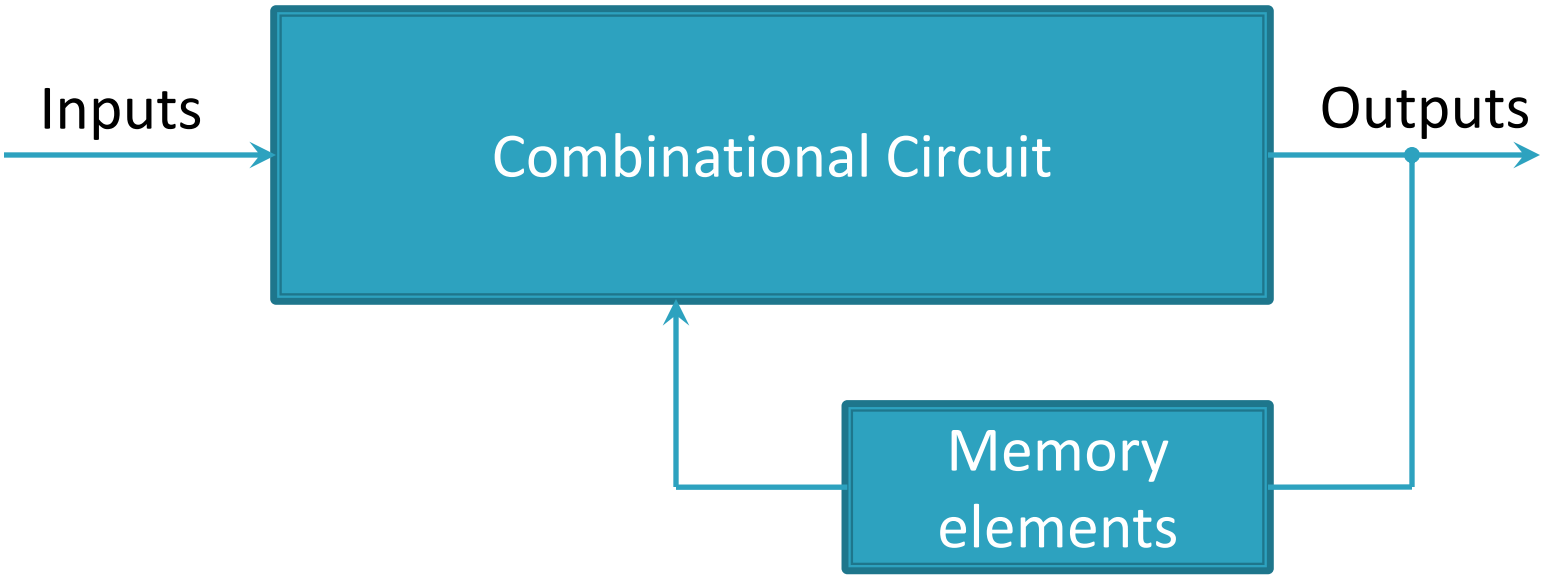
Sequential Circuit

Sequential Circuit

- The logic circuits discussed previously are known as combinational, in that the output depends only on the condition of the latest inputs.
- However, we will now introduce a type of logic where the output depends not only on the latest inputs, but also on the condition of earlier inputs. These circuits are known as sequential, and implicitly they contain memory elements.
- Flip-flops and latches are used as memory elements.

- **Sequential logic** is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs, the input history as well.
- This is in contrast to *combinational logic*, whose output is a function of only the present input. That is, sequential logic has *state* (*memory*) while combinational logic does not.

Sequential Circuits



Types of sequential circuit

- There are two types of sequential circuit :

  Synchronous and **Asynchronous** sequential circuit

- In synchronous sequential circuits, the state of the device changes only at discrete times in response to a clock signal.
- In asynchronous circuits the state of the device can change at any time in response to changing inputs.

Synchronous Sequential Logic

- These circuit uses **clock signal**
- Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit **slower** compared to asynchronous
- Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.
- We use synchronous sequential circuit in synchronous counters, flip flops, and in the design of MOORE-MEALY state management machines.

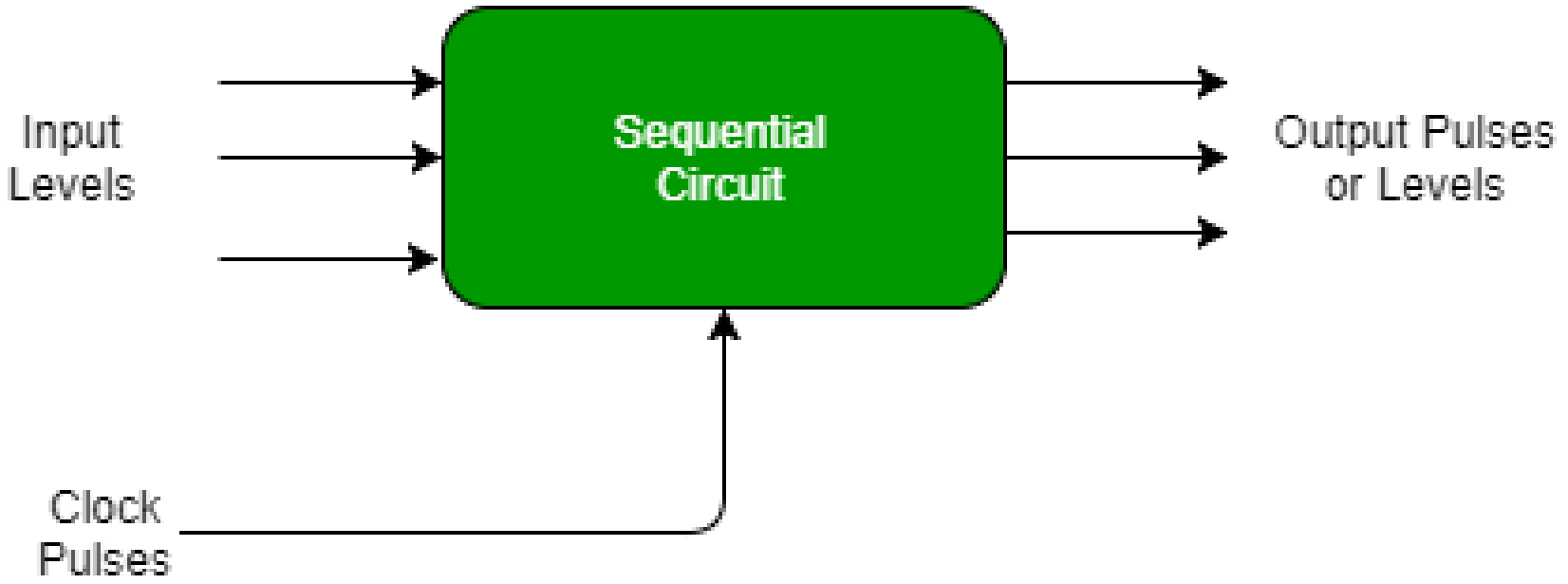


Figure: Synchronous Sequential Circuit

Asynchronous Sequential Logic

- These circuit **do not use a clock signal** but uses the pulses of the inputs which means the state of the circuit changes when the inputs change.
- These circuits are **faster** than synchronous sequential circuits because there is no clock pulse as the circuit doesn't have to wait for a clock signal to process inputs thus, change their state immediately when there is a change in the input signal.
- The speed of the device is potentially limited only by the propagation delays of the logic gates used.
- We use asynchronous sequential circuits when speed of operation is important and **independent** of internal clock pulse.

- But these circuits are more **difficult** to design and their output is **uncertain**.
- Asynchronous logic is more difficult to design and it has some problems compared to synchronous logic. The main problem is that the digital memory is sensitive to the order that their input signals arrive them, like, if two signals arrive at a flip-flop at the same time, which state the circuit goes into can depend on which signal gets to the logic gate first.
- Asynchronous circuits are used in critical parts of synchronous systems where the speed of the system is a priority, like as in microprocessors and digital signal processing circuits.

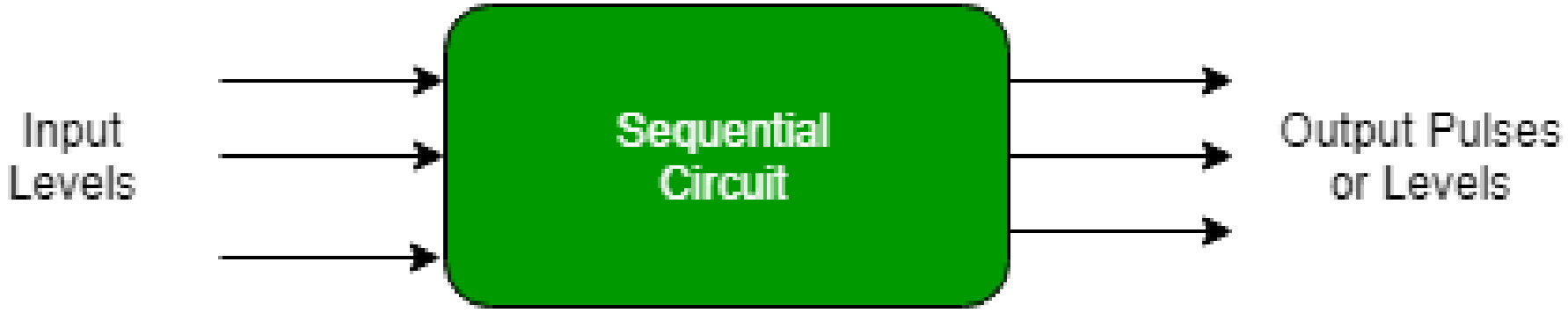


Figure: Asynchronous Sequential Circuit

Difference Between Sequential and Combinational Logic Circuits

Following table shows the **differences** between combinational circuits and sequential circuits.

Combinational Circuits	Sequential Circuits
Outputs depend only on present inputs.	Outputs depend on both present inputs and present state.
Feedback path is not present.	Feedback path is present.
Memory elements are not required.	Memory elements are required.
Clock signal is not required.	Clock signal is required.
Easy to design.	Difficult to design.

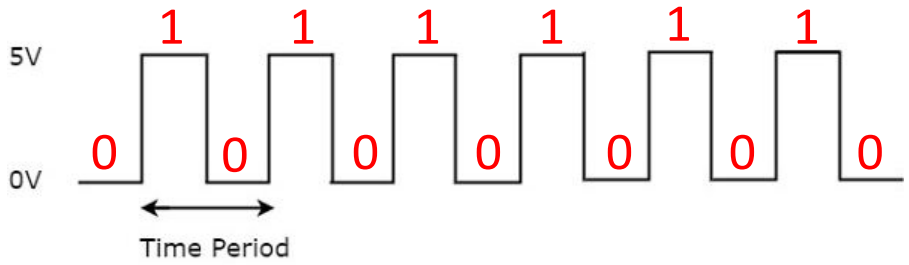
SC Exercise

- Advantages and disadvantages of synchronous and asynchronous sequential circuit.

Clock Signal and Triggering

Clock Signal

- Clock signal is a periodic signal that decides the time of input and its ON time and OFF time need not be the same.
- We can represent the clock signal as a **square wave**, when both its ON time and OFF time are same.
- This clock signal is shown in the following figure.



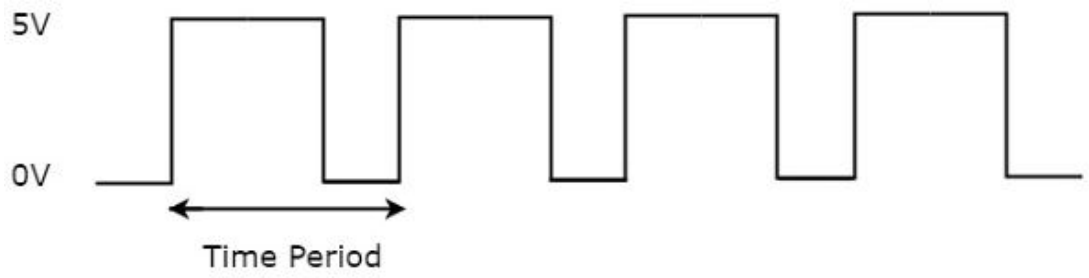
Note

1 : Logic '1'/ ON/ 5V / Set

0 : Logic '0'/ OFF/ 0V / Reset

- In the below figure, square wave is considered as clock signal. This signal stays at logic High (5V) for some time and stays at logic Low (0V) for equal amount of time. This pattern repeats with some time period. In this case, the **time period** will be equal to either twice of ON time or twice of OFF time.

- We can represent the clock signal as **train of pulses**, when ON time and OFF time are not same. This clock signal is shown in the following figure.



- In the above figure, train of pulses is considered as clock signal. This signal stays at logic High (5V) for some time and stays at logic Low (0V) for some other time. This pattern repeats with some time period. In this case, the **time period** will be equal to sum of ON time and OFF time.
- The reciprocal of the time period of clock signal is known as the **frequency** of the clock signal. All sequential circuits are operated with clock signal. So, the frequency at which the sequential circuits can be operated accordingly the clock signal frequency has to be chosen.
- A clock signal is produced by a clock generator and crystal oscillator.

Types of Triggering

- Triggering basically means switching. The way in which a switch or an event is triggered can be different, based on the user's requirements
- Following are the two possible types of triggering that are used in sequential circuits.
- **Level** triggering
- **Edge** triggering

Level triggering

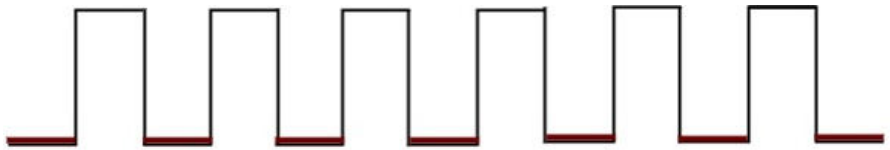
- There are two levels, namely logic High and logic Low in clock signal. Following are the two **types of level triggering**.
- **Positive level** triggering
- **Negative level** triggering

Positive Level Triggering

- If the sequential circuit is operated with the clock signal when it is in **Logic High**, then that type of triggering is known as **Positive level triggering**.
- It is highlighted in below figure.

Negative Level Triggering

- If the sequential circuit is operated with the clock signal when it is in **Logic Low**, then that type of triggering is known as **Negative level triggering**.
- It is highlighted in the following figure.

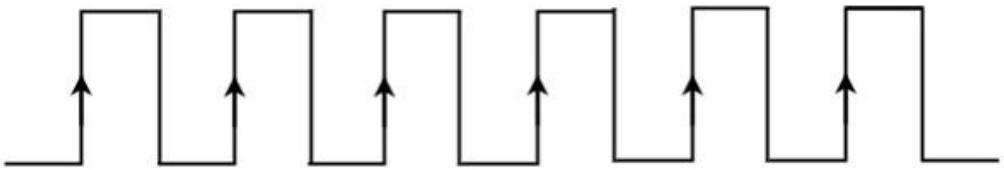


Edge triggering

- There are two types of transitions that occur in clock signal. That means, the clock signal transitions either from Logic Low to Logic High or Logic High to Logic Low.
- Following are the two **types of edge triggering** based on the transitions of clock signal.
- **Positive edge** triggering
- **Negative edge** triggering

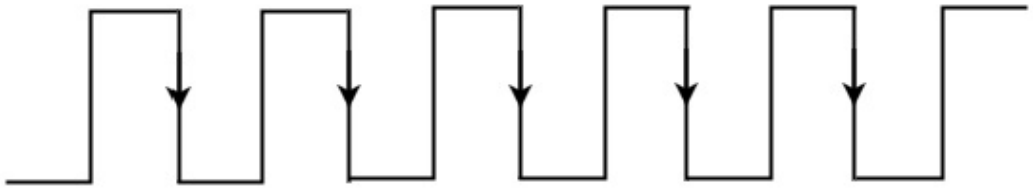
Positive Edge Triggering

- If the sequential circuit is operated with the clock signal that is transitioning from Logic Low to Logic High, then that type of triggering is known as **Positive edge triggering**.
- It is also called as rising edge triggering.
- It is shown in the following figure.

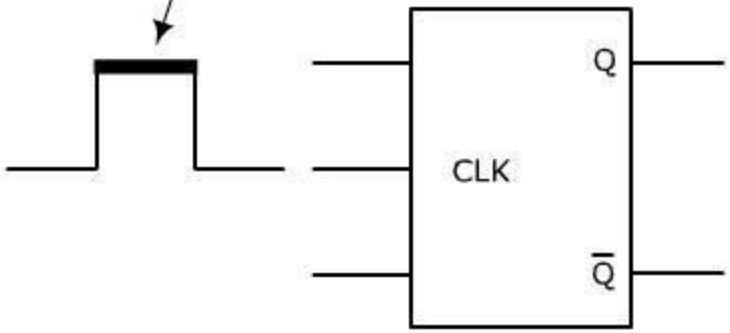


Negative Edge Triggering

- If the sequential circuit is operated with the clock signal that is transitioning from Logic High to Logic Low, then that type of triggering is known as **Negative edge triggering**.
- It is also called as falling edge triggering.
- It is shown in the following figure.

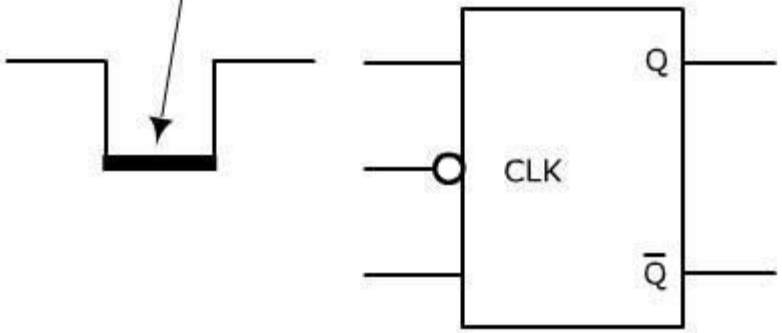


Triggers on high clock level



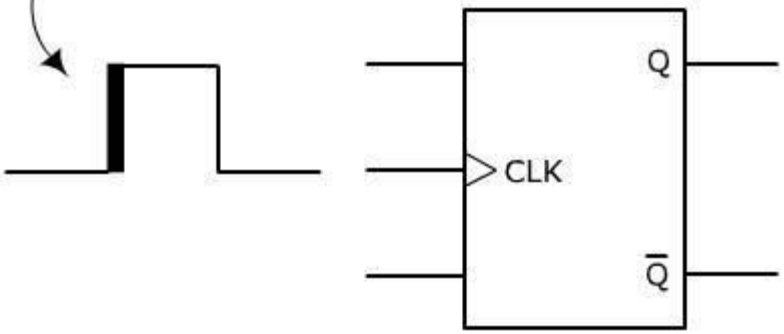
High Level Triggering

Triggers on low clock level



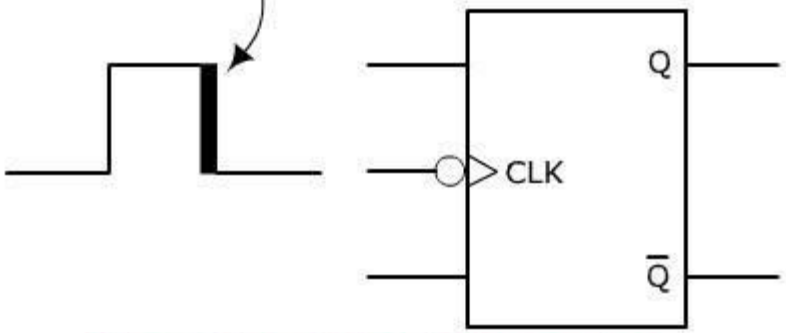
Low Level Triggering

Triggers on this edge of the clock pulse



Positive Edge Triggering

Triggers on this edge of the clock pulse



Negative Edge Triggering

Latch & Flip-Flop

Latch & Flip FLop

- Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information.
- The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted.
- In other words, when they are enabled, their content changes immediately when their inputs change.
- Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal.
- This enable signal is usually the controlling clock signal.
- After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

- There are basically four main types of latches and flip-flops: SR, D, JK, and T.
- The major differences in these flip-flop types are the number of inputs they have and how they change state.
- For each type, there are also different variations that enhance their operations.

Latch VS Flip Flop

Latches	Flip Flops
Latches are building blocks of sequential circuits and these can be built from logic gates	Flip flops are also building blocks oof sequential circuits. But, these can be built from the latches.
Latch continuously checks its inputs and changes its output correspondingly.	Flip flop continuously checks its inputs and changes its output correspondingly only at times determined by clocking signal
The latch is sensitive to the duration of the pulse and can send or receive the data when the switch is on	Flipflop is sensitive to a signal change. They can transfer data only at the single instant and data cannot be changed until next signal change. Flip flops are used as a register.
It is based on the enable function input	It works on the basis of clock pulses
It is a level triggered, it means that the output of the present state and input of the next state depends on the level that is binary input 1 or 0.	It is an edge triggered, it means that the output and the next state input changes when there is a change in clock pulse whether it may a +ve or -ve clock pulse.

S-R Flip-Flop (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.

Gated S-R Flip-Flop

Fig: Logic Symbol

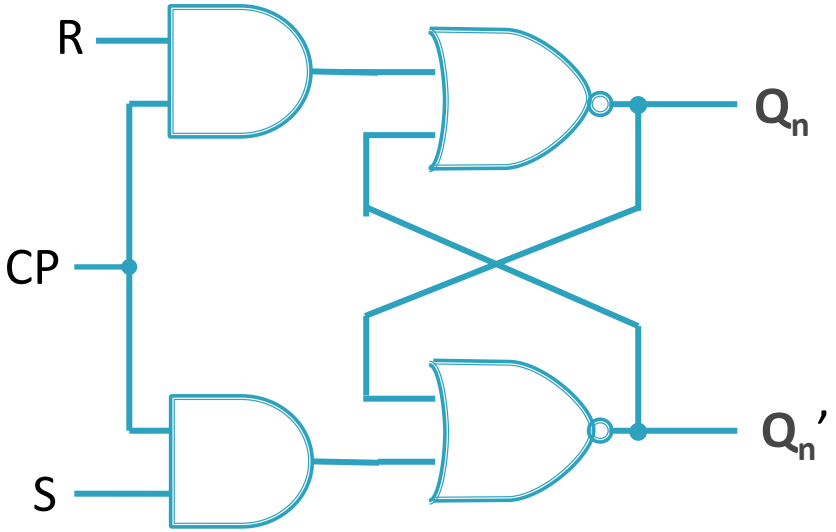


Fig: Logic diagram

CP	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No Change
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
0	x	x	1	1	

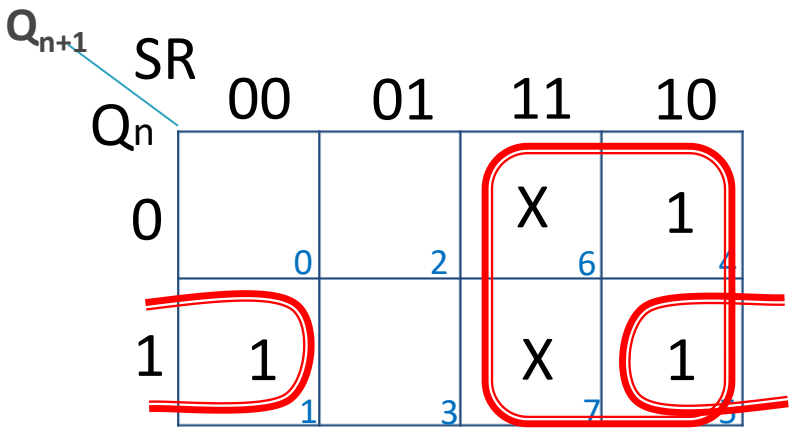
Fig: Characteristics Table

CP	S	R	Q_{n+1}	State
1	0	0	Q_n	Memory
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	x	Invalid
0	x	x	Q_n	Memory

Fig: Function Table/Truth Table

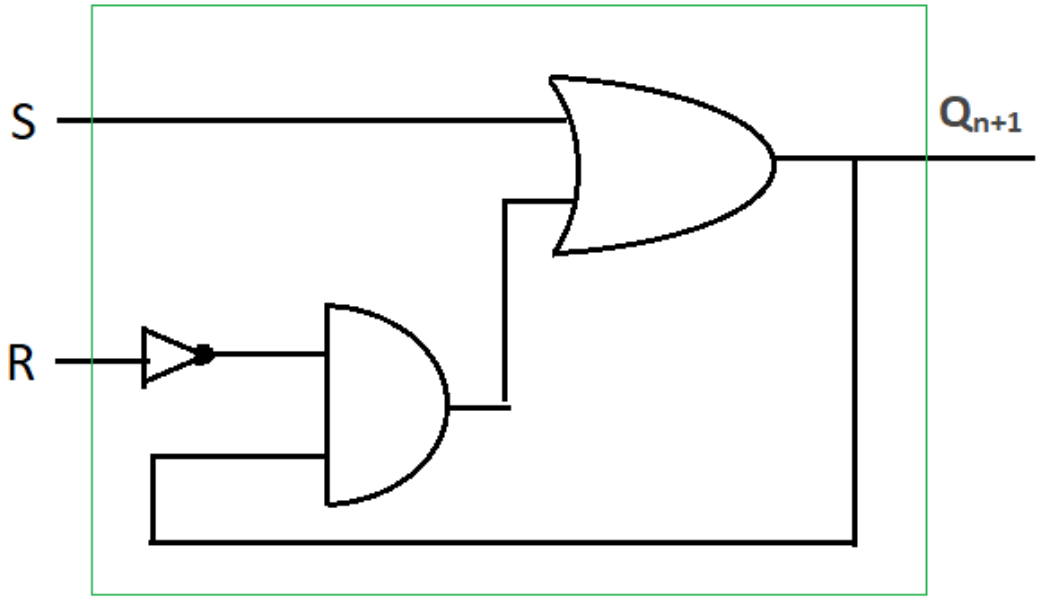
Q_n	Q_{n+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Fig: Excitation Table



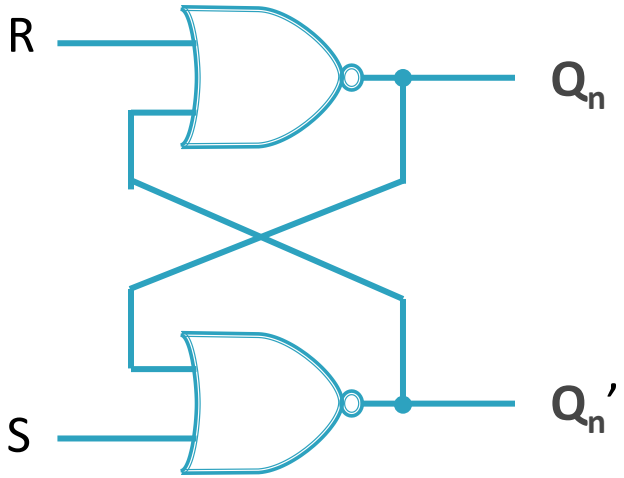
$$Q_{n+1} = S + R'Q_n$$

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



NOR Gate S-R Flip-Flop (High Level/Active High)

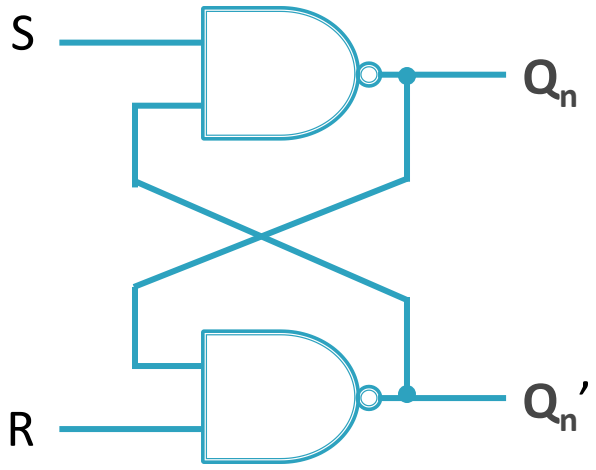
Logic Symbol



Logic diagram

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

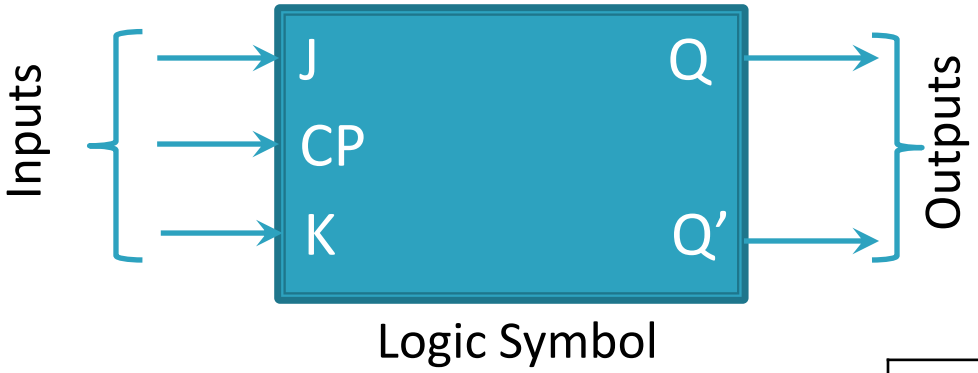
NAND Gate S-R Flip-Flop (Low Level/Active Low)



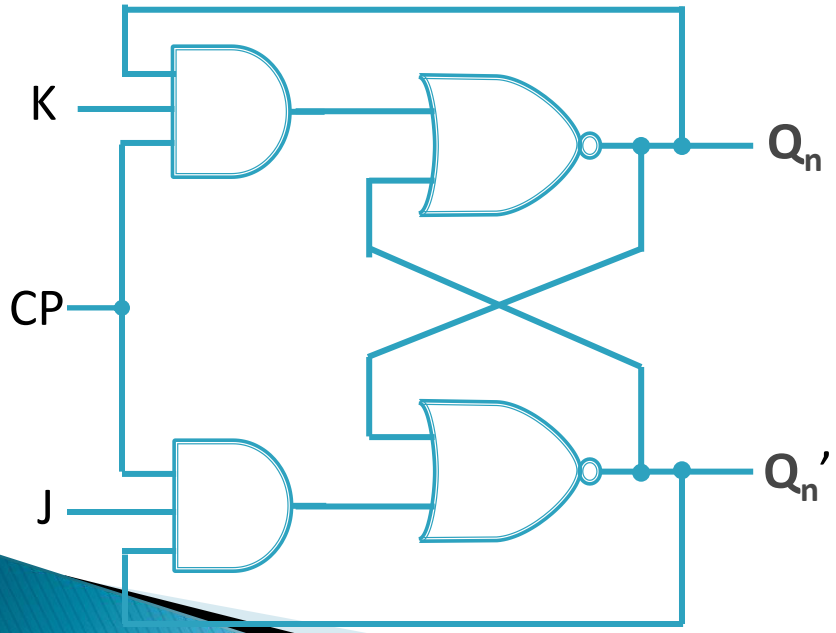
Logic diagram

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

J-K Flip-Flop



Logic Symbol



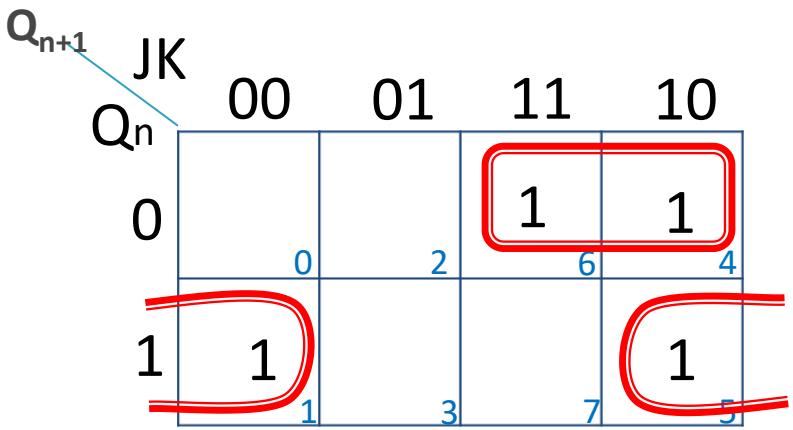
J	K	Q_n	Q_{n+1}	State
0	0	0	0	No Change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

J	K	Q_{n+1}	State
0	0	Q_n	Memory
0	1	0	Reset
1	0	1	Set
1	1	Q_n'	Toggle

Fig: Function Table/Truth Table

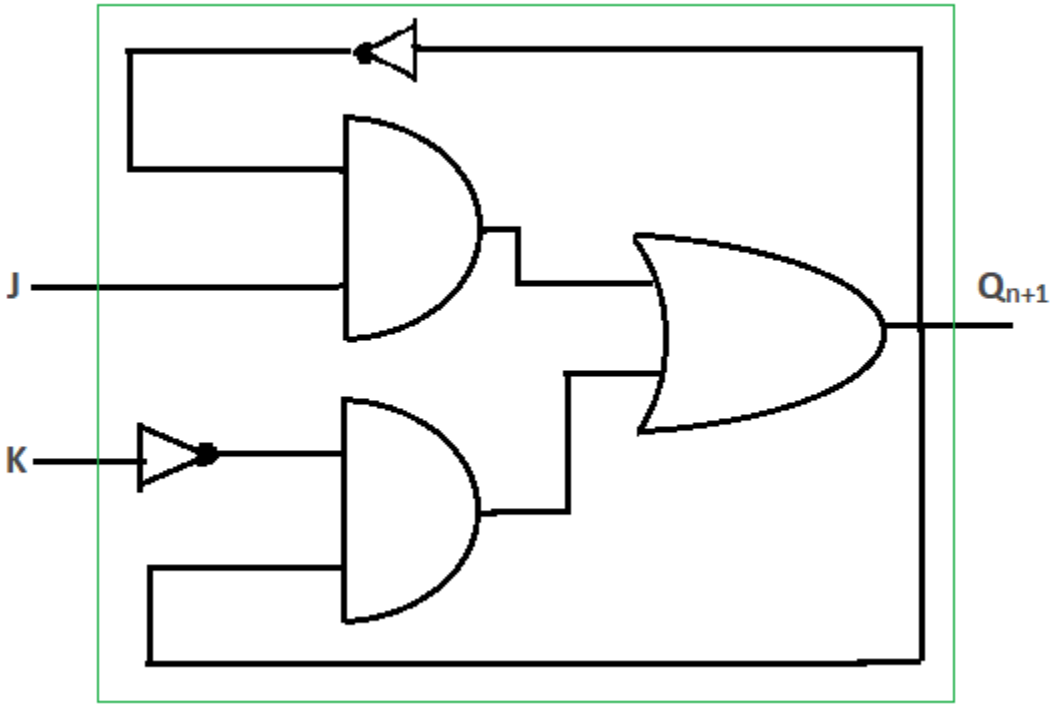
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Fig: Excitation Table



$$Q_{n+1} = JQ_n' + K'Q_n$$

J	K	Q_n	Q_{n+1}	State
0	0	0	0	No Change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	



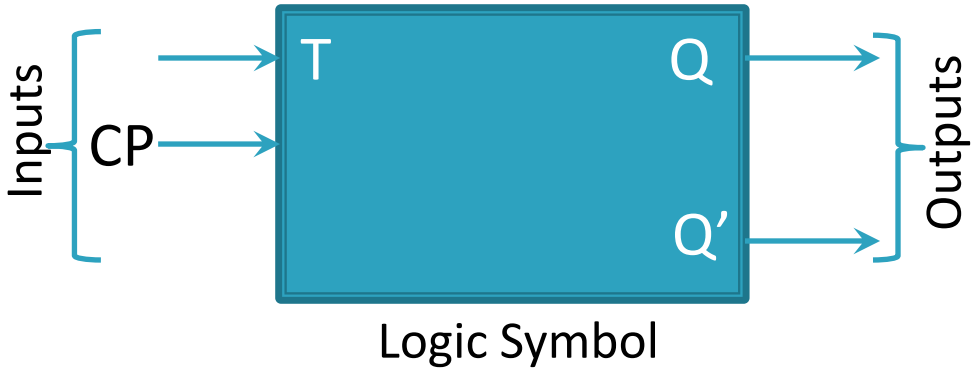
T Flip-Flop

Logic Symbol

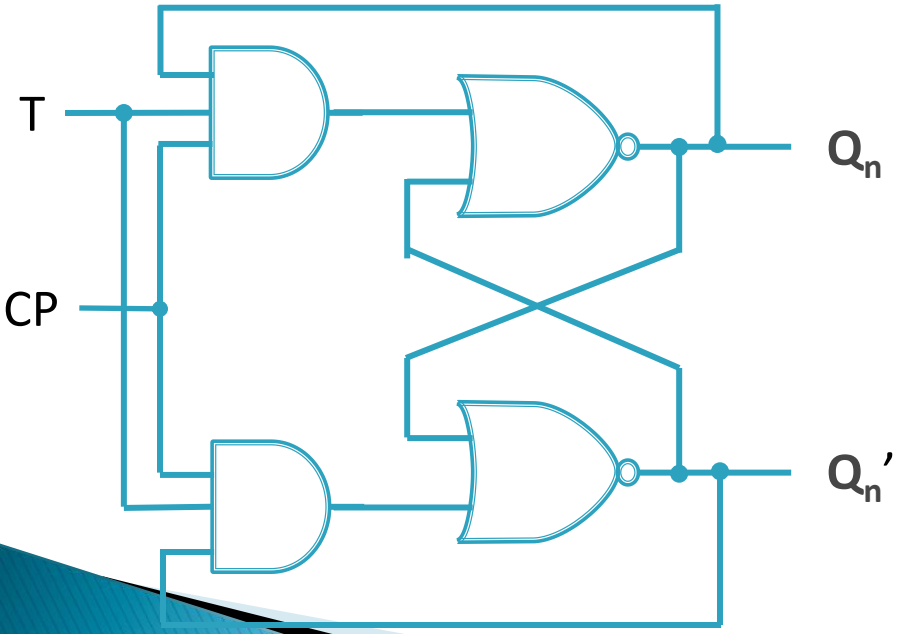
T	Q_n	Q_{n+1}	State
0 0	0 1	0 1	No Change
1 1	0 1	1 0	Toggle

J	K	Q_n	Q_{n+1}	State
0 0	0 0	0 1	0 1	No Change
0 0	1 1	0 1	0 0	Reset
1 1	0 0	0 1	1 1	Set
1 1	1 1	0 1	1 0	Toggle

T Flip-Flop



Logic Symbol



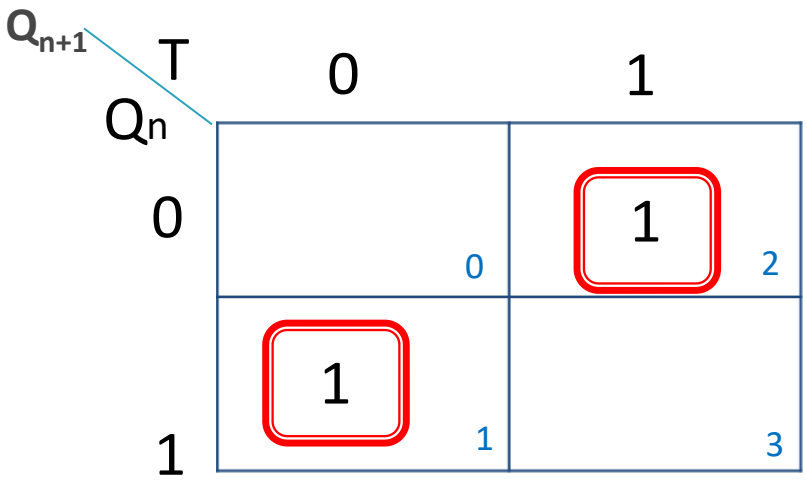
T	Q_n	Q_{n+1}	State
0 0	0 1	0 1	No Change
1 1	0 1	1 0	Toggle

T	Q_{n+1}	State
0	Q_n	Memory
1	Q_n'	Toggle

Fig: Function Table/Truth Table

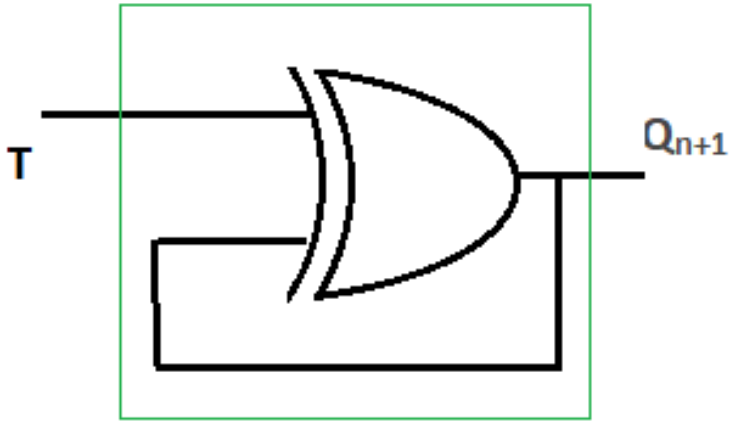
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Fig: Excitation Table



$$Q_{n+1} = TQ_n' + T'Q_n$$
$$= T \oplus Q_n$$

T	Q_n	Q_{n+1}	State
0 0	0 1	0 1	No Change
1 1	0 1	1 0	Toggle



Gated D Flip-Flop

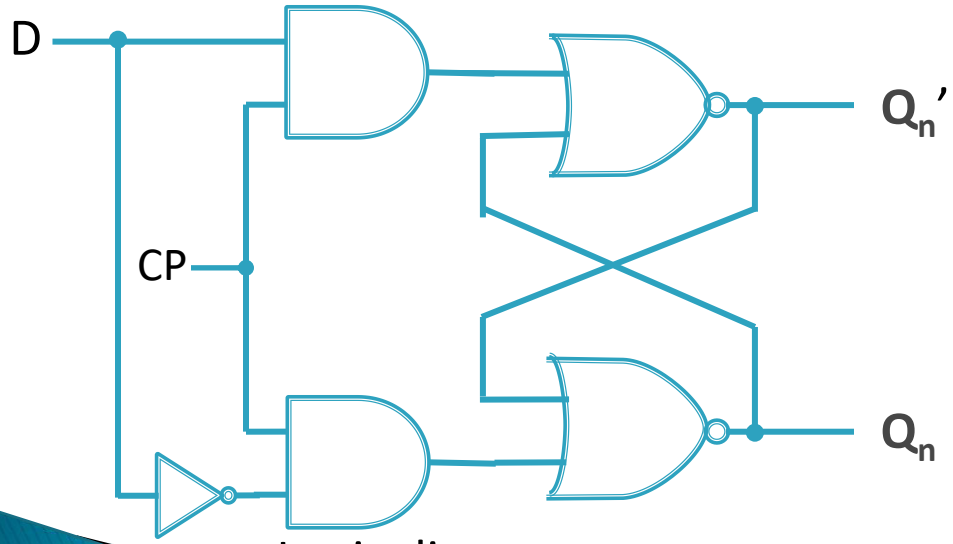
Logic Symbol

D	Q_n	Q_{n+1}	State
0	0	0	Reset
0	1	0	
1	0	1	Set
1	1	1	

S	R	Q_n	Q_{n+1}	State
0	0	0	0	No Change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Invalid
1	1	1	x	

Gated D Flip-Flop

Logic Symbol



Logic diagram

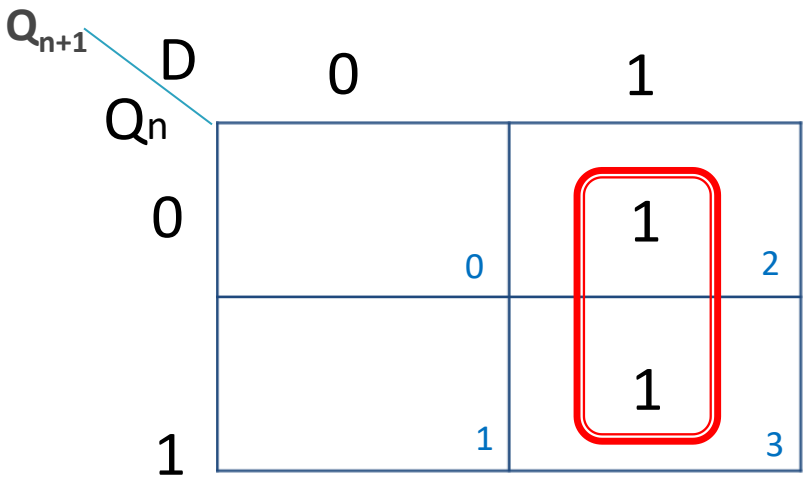
D	Q_n	Q_{n+1}	State
0	0	0	Reset
0	1	0	
1	0	1	Set
1	1	1	

D	Q_{n+1}	State
0	0	Memory
1	1	Memory

Fig: Function Table/Truth Table

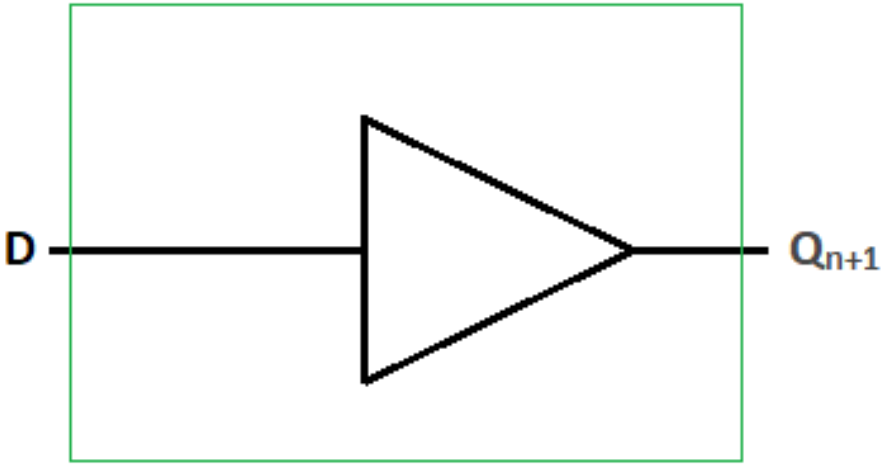
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Fig: Excitation Table



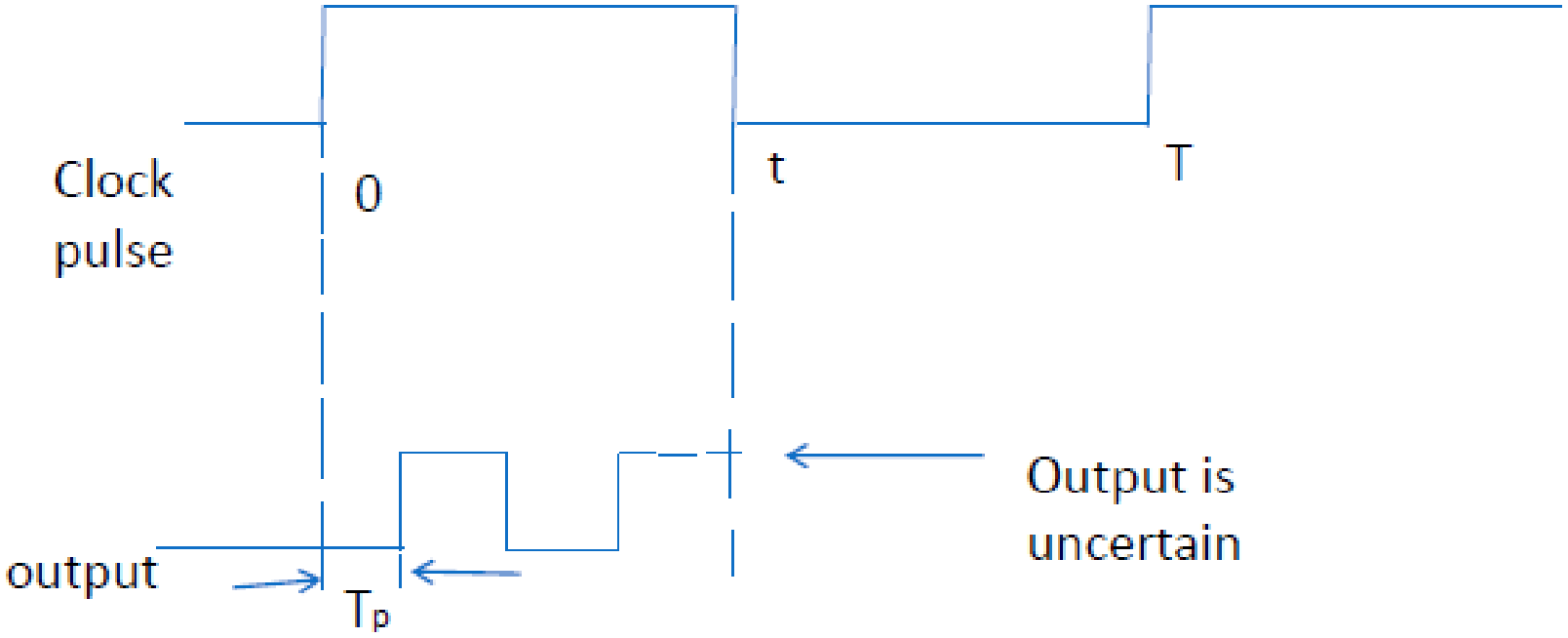
$$Q_{n+1} = D$$

D	Q_n	Q_{n+1}	State
0	0	0	Reset
0	1	0	
1	0	1	Set
1	1	1	



Race – around Condition

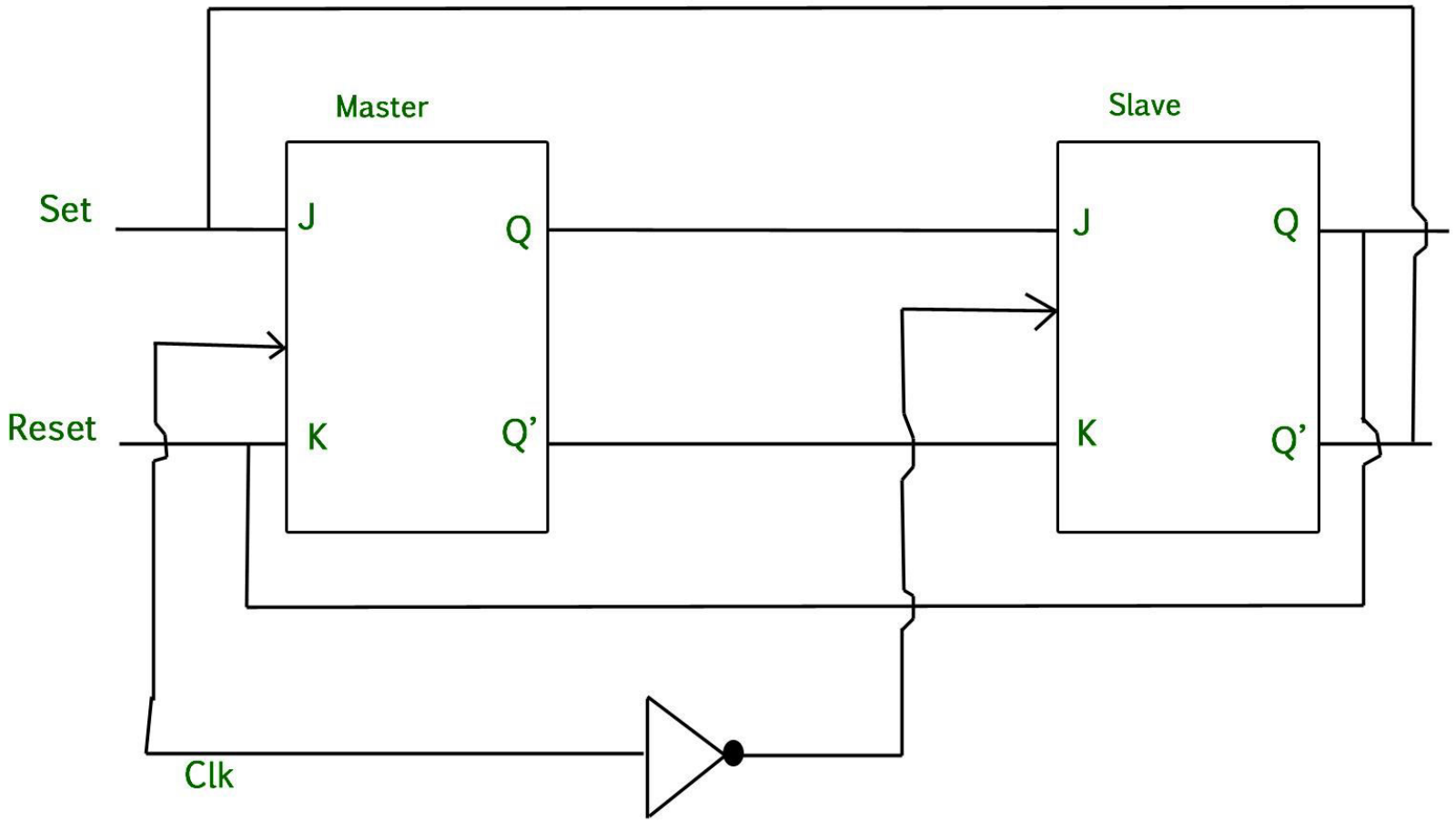
- For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain.
- This problem is called race around condition in J-K flip-flop. This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic “1” only for a very short time, which is practically impossible.
- This introduced the concept of **Master Slave JK** flip flop.



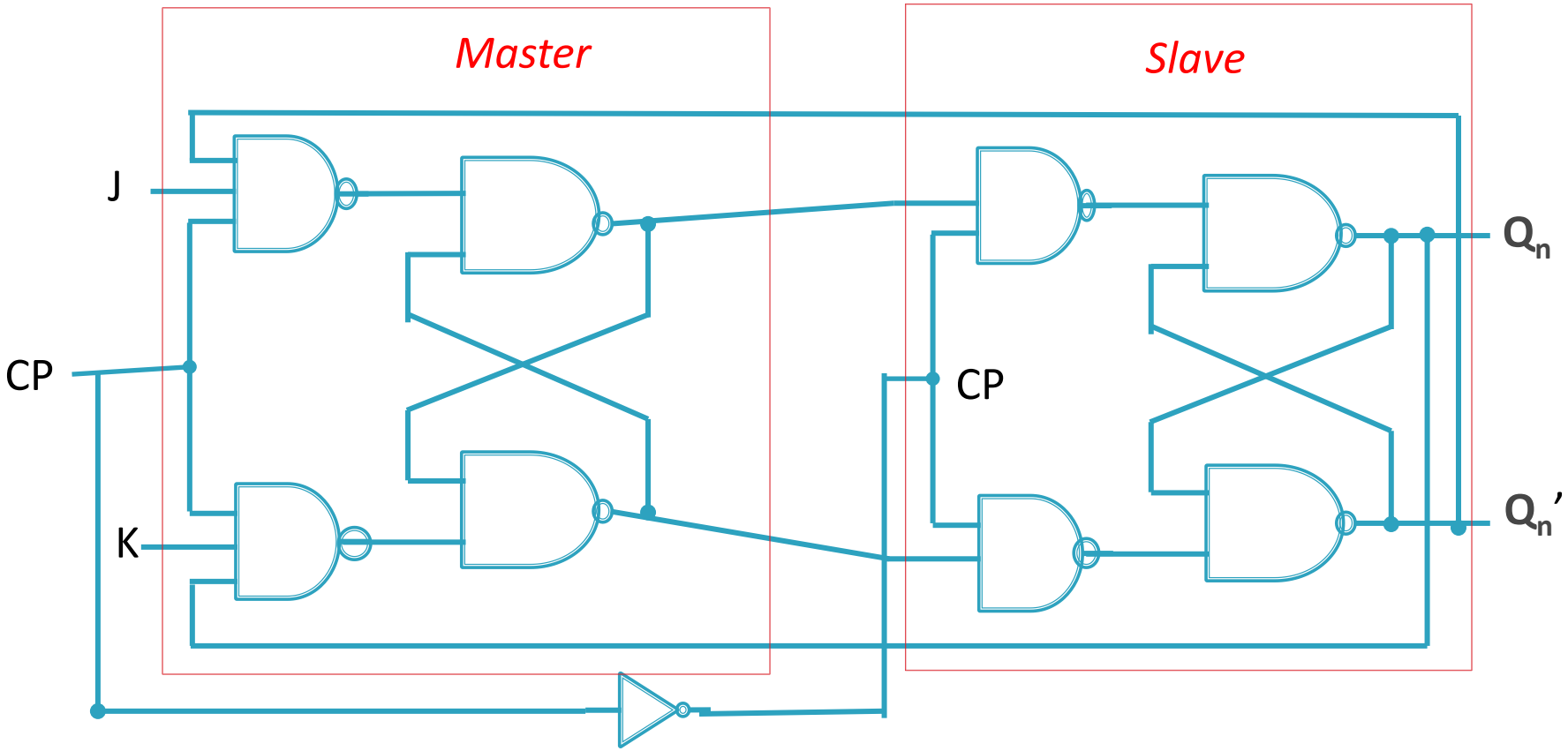
Master Slave JK flip flop

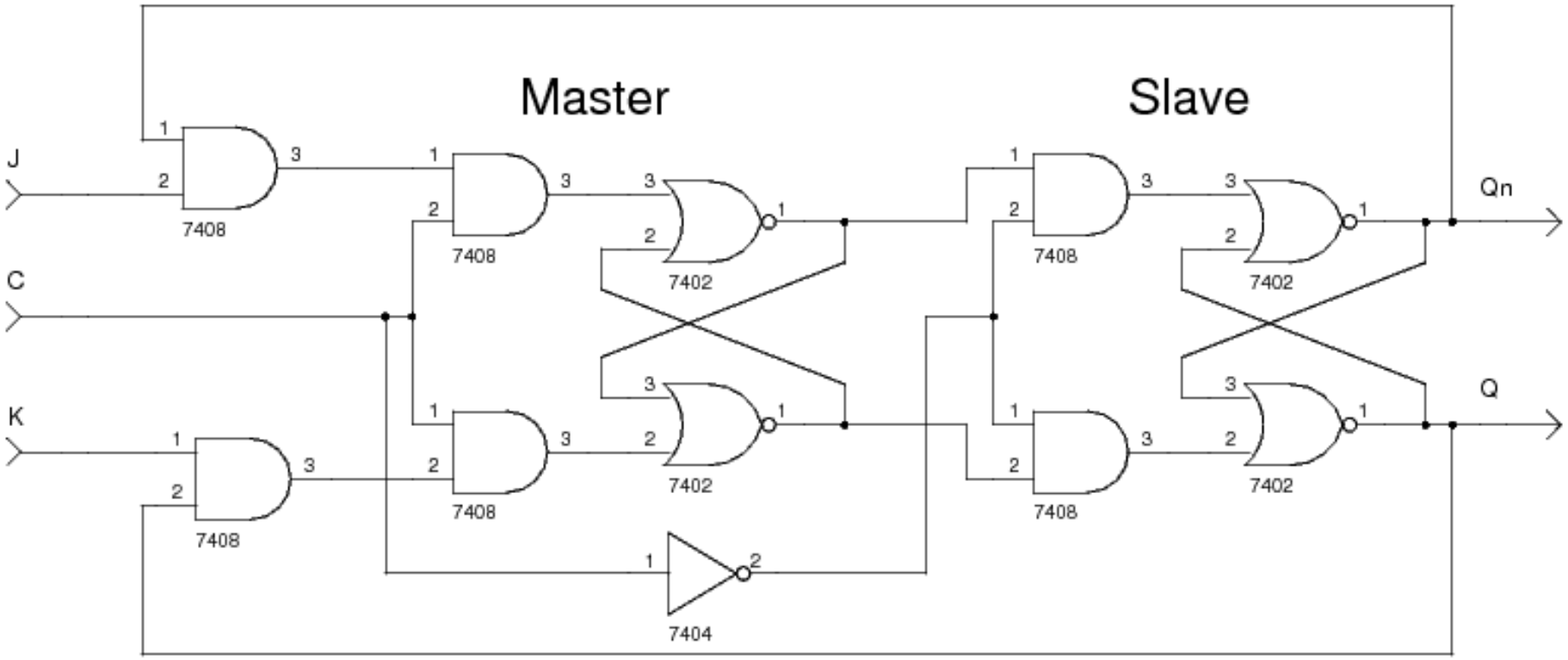
- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration. Out of these, one acts as the **“master”** and the other as a **“slave”**.
- The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.

- In addition to these two flip-flops, the circuit also includes an **inverter**.
- The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop. In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.



Master-slave JK Flip-Flop




Master
Slave
J
C
K
Qn
Q
7408
7402
7404

Clk	J	K	Q_{n+1}	State
0	X	X	Q_n	No Change (Memory)
1	0	0	Q_n	No Change (Memory)
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	Q_n'	Toggle

Fig: Function Table/Truth Table of MS JK Flip Flop

SC Exercise

- Implement gated SR, JK, D and T flip-flops using NAND gate only.

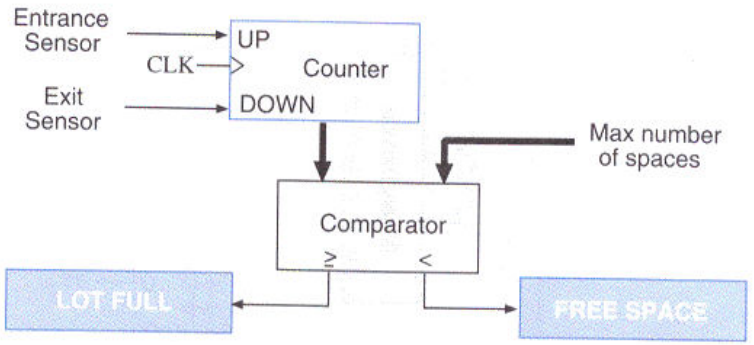
Counter

- A counter is a sequential machine that produces a specified count sequence. The count changes whenever the input clock is asserted.
- There is a great variety of counter based on its construction.
    1. **Clock:** Synchronous or Asynchronous
    2. **Clock Trigger:** Positive edged or Negative edged
    3. **Counts:** Binary, Decade
    4. **Count Direction:** Up, Down, or Up/Down
    5. **Flip-flops:** JK or T or D

- A counter can be constructed by a synchronous circuit or by an asynchronous circuit. With a synchronous circuit, all the bits in the count change synchronously with the assertion of the clock. With an asynchronous circuit, all the bits in the count do not all change at the same time.
- A counter may count up or count down or count up and down depending on the input control.
- Because of limited word length, the count sequence is limited.
- For an n-bit counter, the range of the count is [0, 2n-1]. The count sequence usually repeats itself. When **counting up**, the count sequence goes in this manner: 0, 1, 2, … 2n-2, 2n-1, 0, 1, …etc. When **counting down** the count sequence goes in the same manner: 2n-1, 2n-2, … 2, 1, 0, 2n- 1, 2n-2, … etc.

Uses of Counters

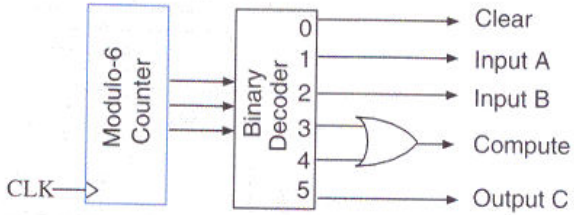
- The most typical uses of counters are
  - To count the number of times that a certain event takes place; the occurrence of event to be counted is represented by the input signal to the counter (Fig. 1.1a)
  - To control a fixed sequence of actions in a digital system (Fig. 1.1b)
  - To generate timing signals (Fig. 1.2a)
  - To generate clocks of different frequencies (Fig. 1.2b)



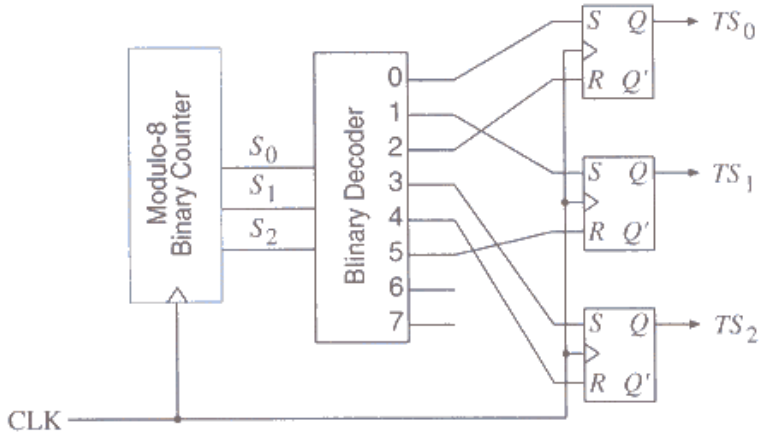
(a)

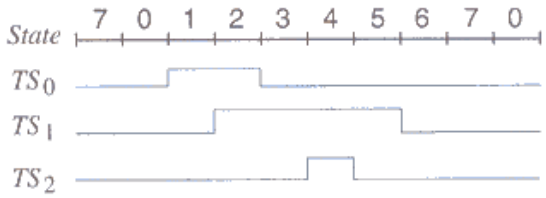
Sequence of actions:

0: Clear all registers
1: Input A
2: Input B
3: Compute
4: Compute
5: Output C

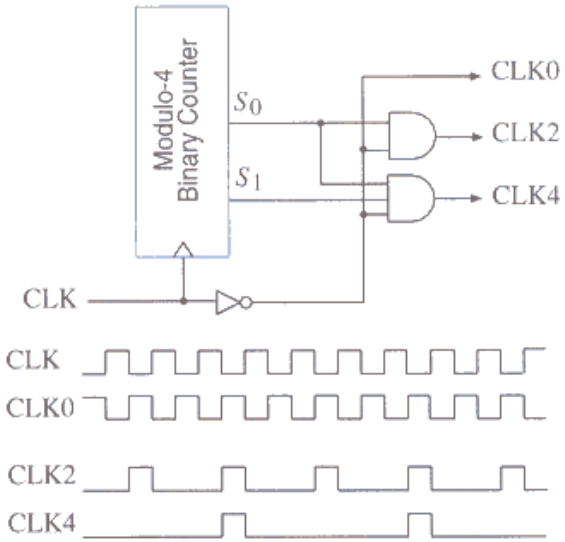


(b)





(a)



(b)

Two Classes of Counters

- Counters are classified into two categories:
  1. Asynchronous Counters (Ripple counters)
  2. Synchronous Counters

- ***Asynchronous:*** The events do not have a fixed time relationship with each other and do not occur at the same time.
- ***Synchronous:*** The events have a fixed time relationship with each other and do occur at the same time

Asynchronous Counters

- An asynchronous counter is one in which the flip-flop within the counter do not change states at exactly the same time because they do not have a common clock pulse.
  - 2 Bit asynchronous binary counter
  - 3 Bit asynchronous binary counter
  - 4 Bit asynchronous binary counter

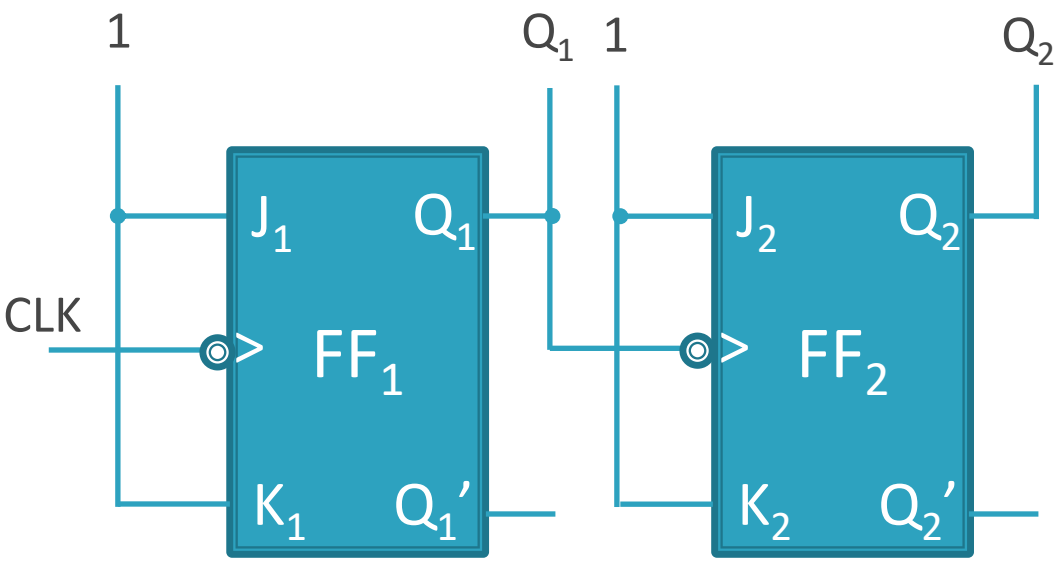
Example

3-bit Up Counter	3-bit Down Counter
000	000
001	111
010	110
011	101
100	100
101	011
110	010
111	001

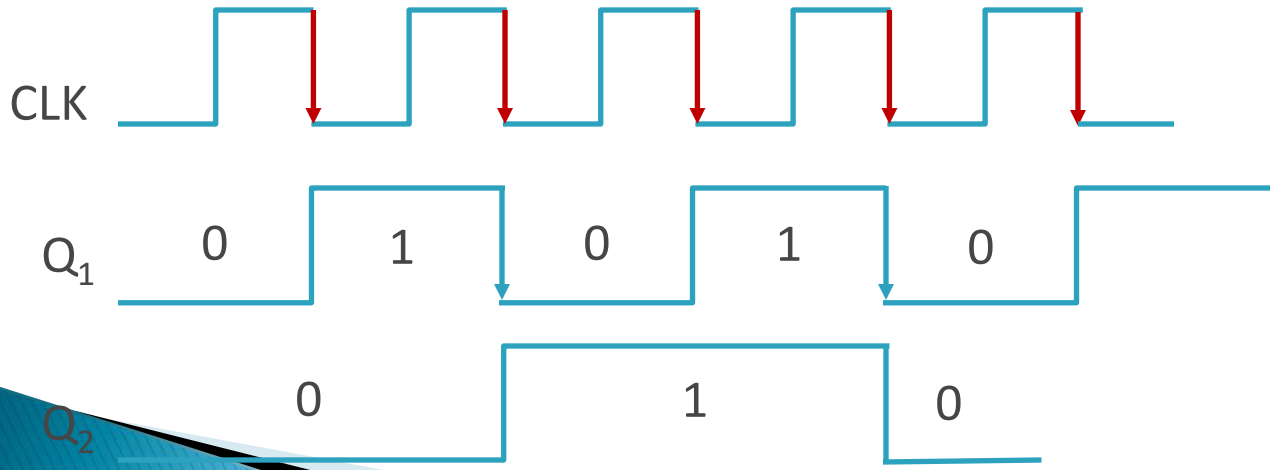
- The complement of the count sequence counts in reverse direction. If the uncomplemented output counts up, the complemented output counts down. If the uncomplemented output counts down, the complemented output counts up.

3-bit Up Counter	Complement of the Count
000	111
001	110
010	101
011	100
100	011
101	010
110	001
111	000

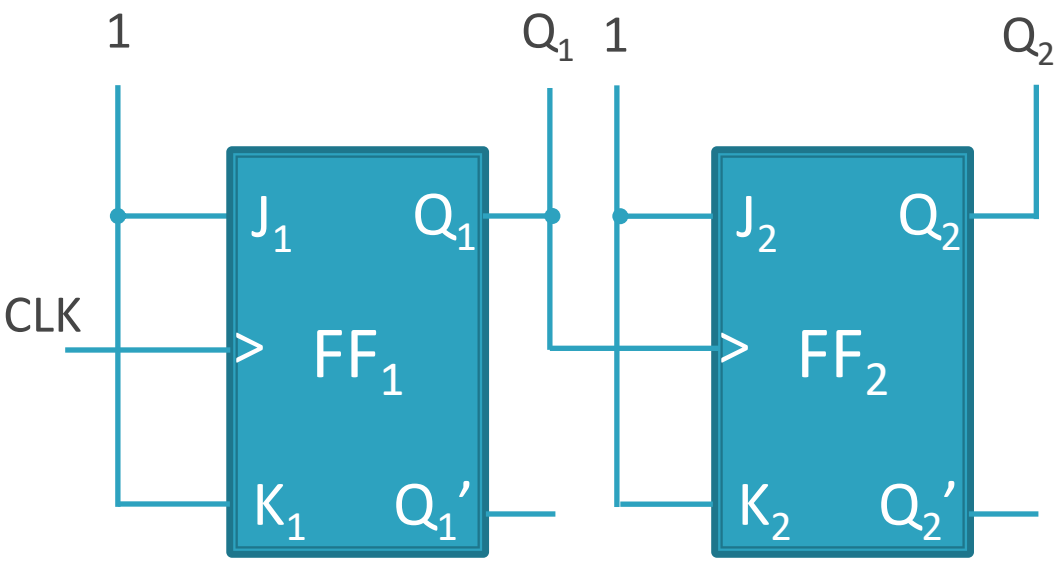
2-bit Ripple Up-Counter using Negative Edge-triggered Flip-Flop



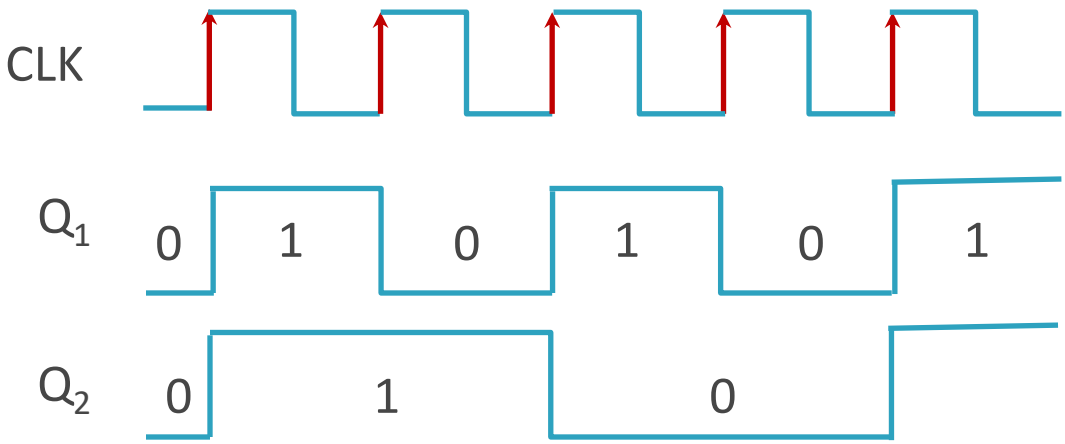
CLK	Present State		Next State	
	Q_2	Q_1	Q_2	Q_1
0	0	0	0	1
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0



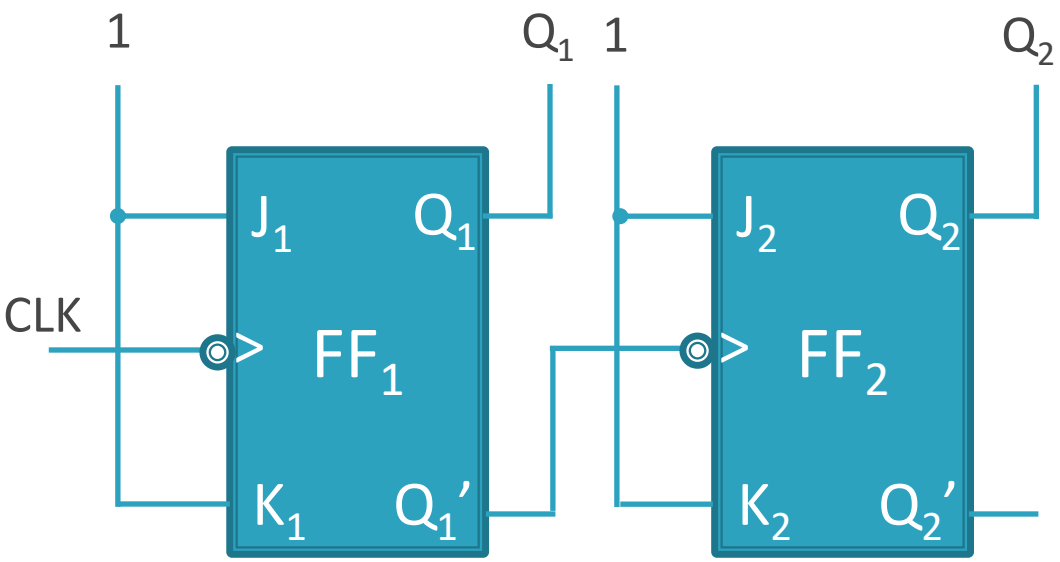
2-bit Ripple Down-Counter using Positive Edge-triggered Flip-Flop



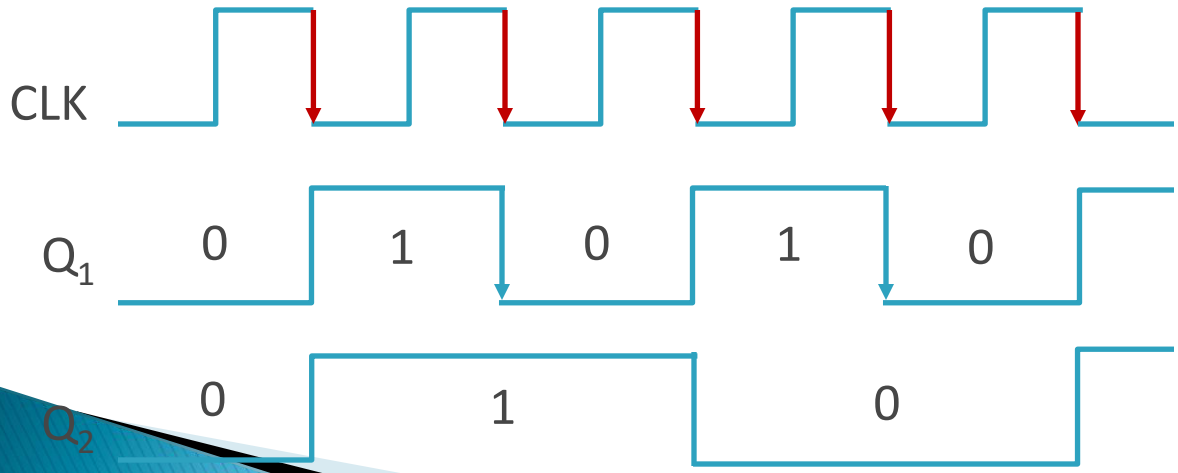
CLK	Present State		Next State	
	Q_2	Q_1	Q_2	Q_1
1	0	0	1	1
1	1	1	1	0
1	1	0	0	1
1	0	1	0	0



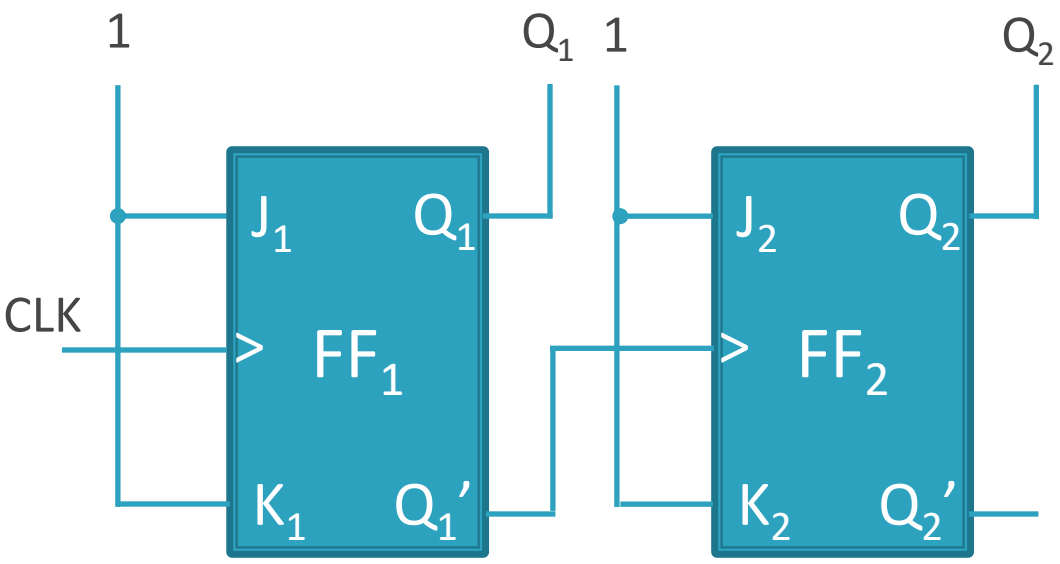
2-bit Ripple Down-Counter using Negative Edge-triggered Flip-Flop



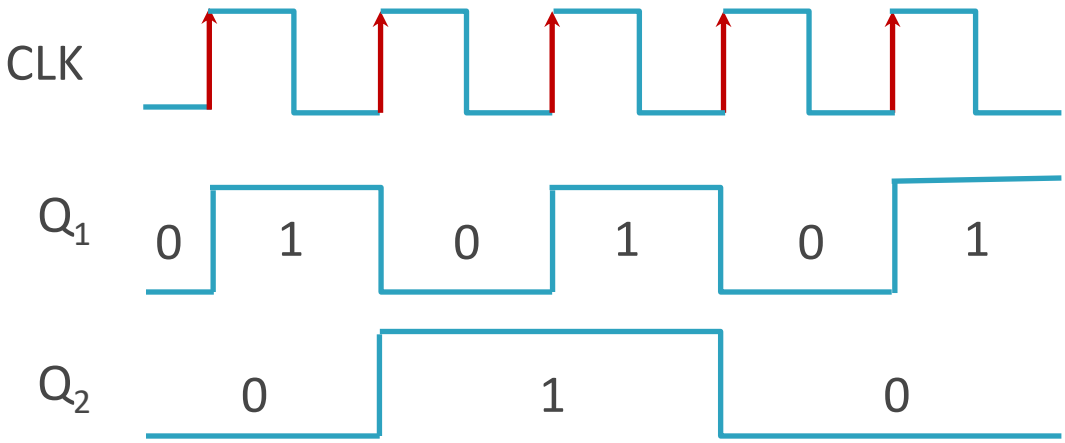
CLK	Present State		Next State	
	Q_2	Q_1	Q_2	Q_1
0	0	0	1	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	0



2-bit Ripple Up-Counter using Positive Edge-triggered Flip-Flop



CLK	Present State		Next State	
	Q_2	Q_1	Q_2	Q_1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0



Mod-6 Asynchronous Counter

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

R = 0 for 000 to 101

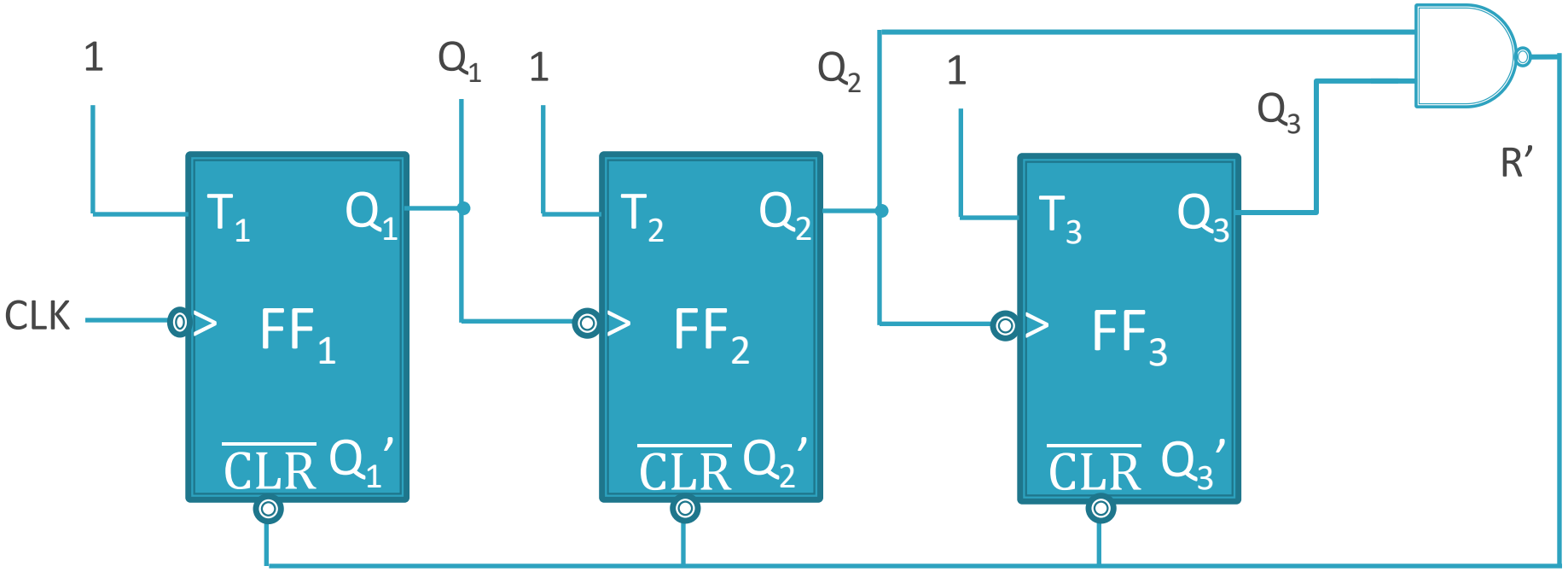
R = 1 for 110

R = x for 111

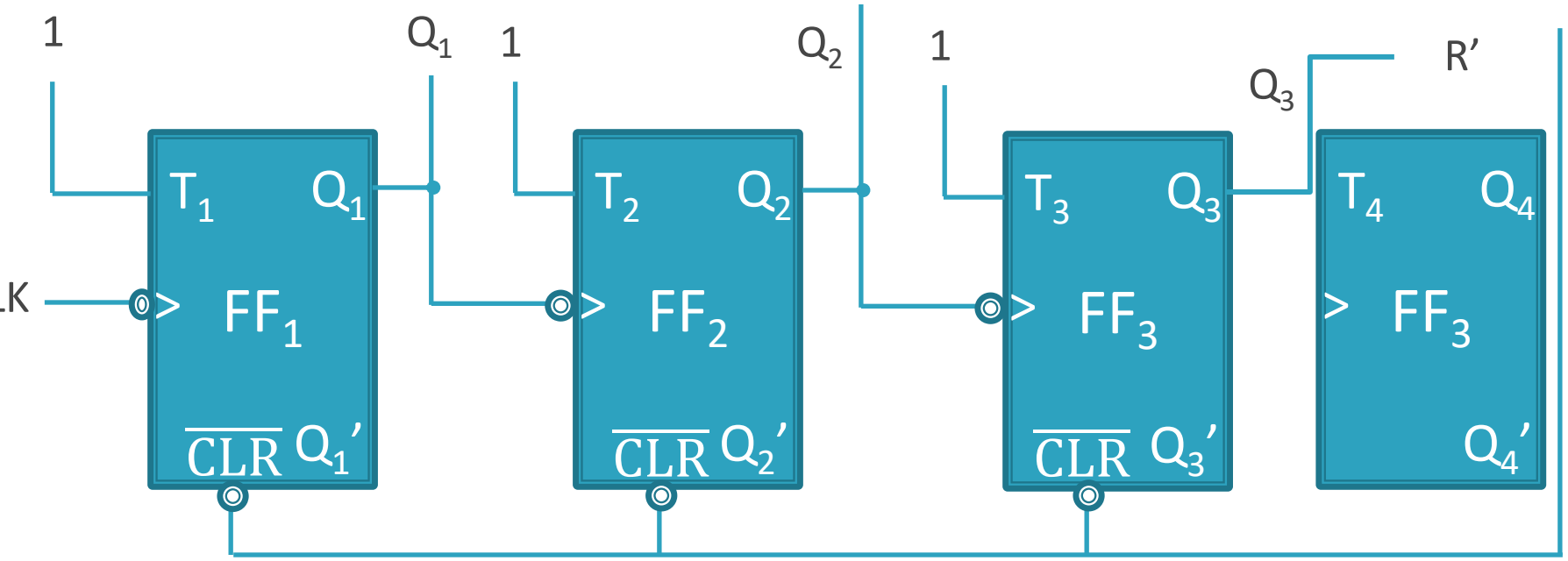
$R = Q_3Q_2Q_1' + Q_3Q_2Q_1$

$R = Q_3Q_2$

Mod-6 Asynchronous Counter



Mod-6 Asynchronous Counter



Truth Table, Characteristics Table and Excitation Table of JK Flip-Flop

Truth Table

CLK	J	K	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Q'_n

Characteristics Table

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

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

Excitation Tables

S-R Flip-Flop

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

J-K Flip-Flop

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

Excitation Tables

D Flop-Flop

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

T Flop-Flop

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

Design of Synchronous Counters

- 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.

Design of Synchronous Counters

- 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:

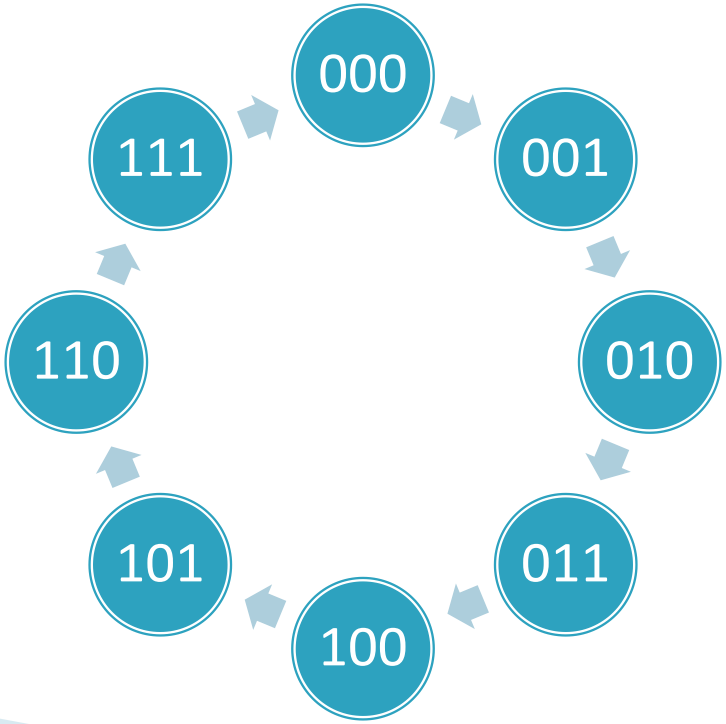
  Draw the logic diagram based on the minimal expressions.

Design of Synchronous 3-bit Up Counters

- 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:



Design of Synchronous 3-bit Up Counters

- 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.

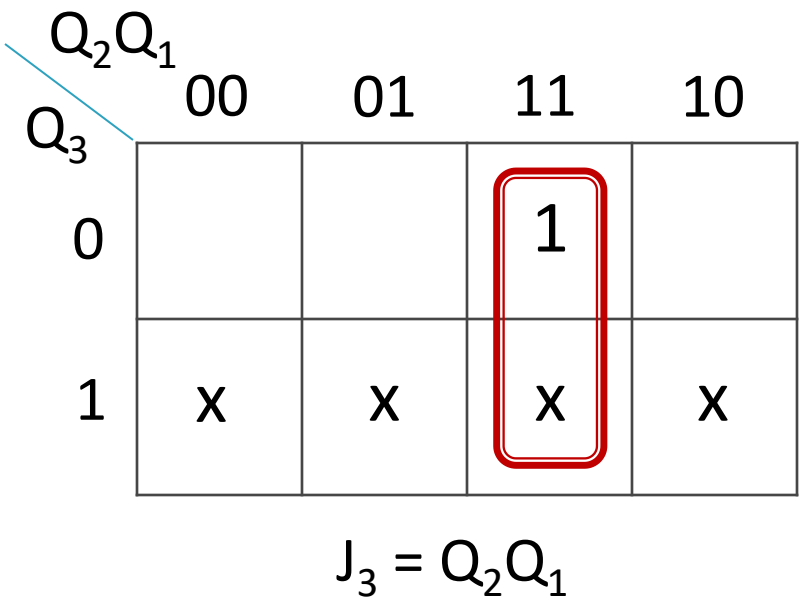
PS			NS			Required excitations					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1

Design of Synchronous 3-bit Up Counters

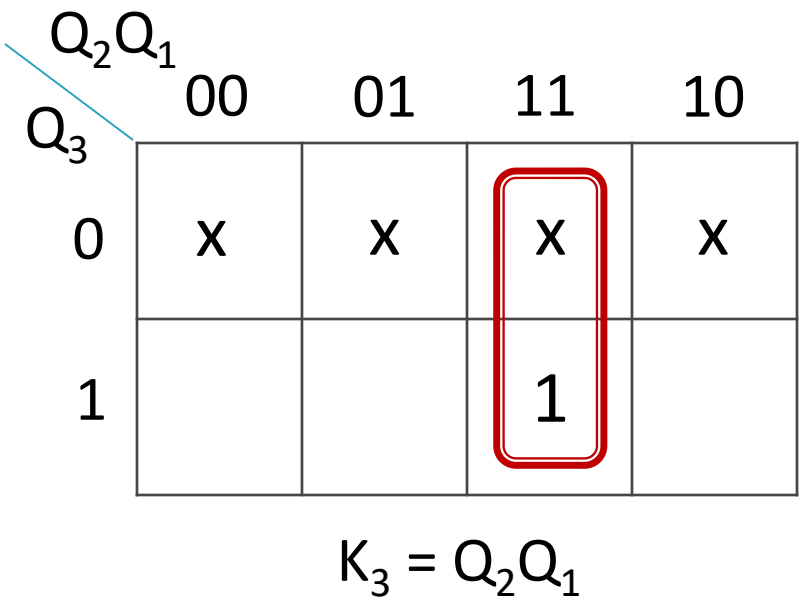
- Step 4. Obtain the minimal expressions

  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:

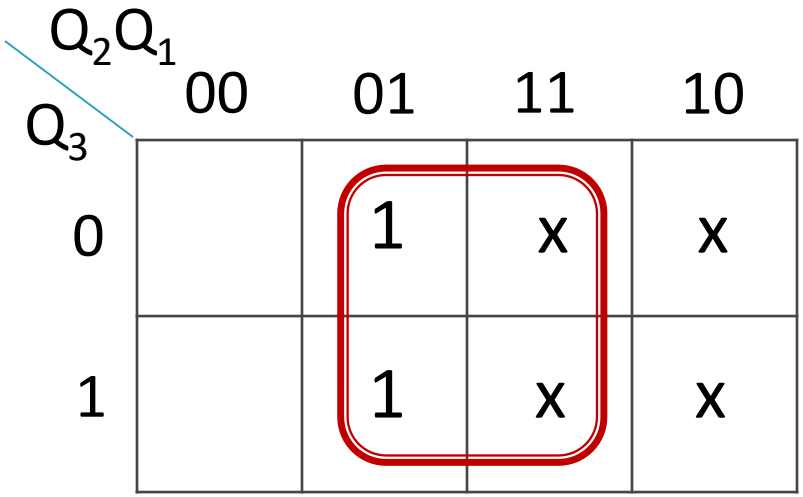


$J_3 = Q_2Q_1$

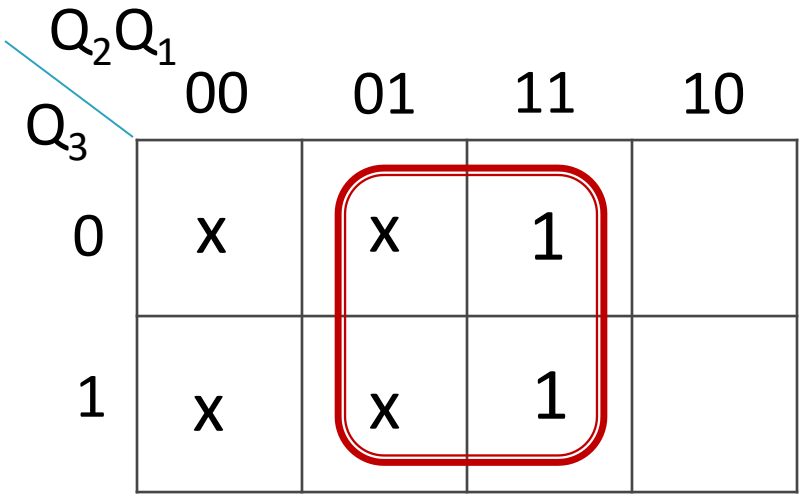


$K_3 = Q_2Q_1$

Design of Synchronous 3-bit Up Counters



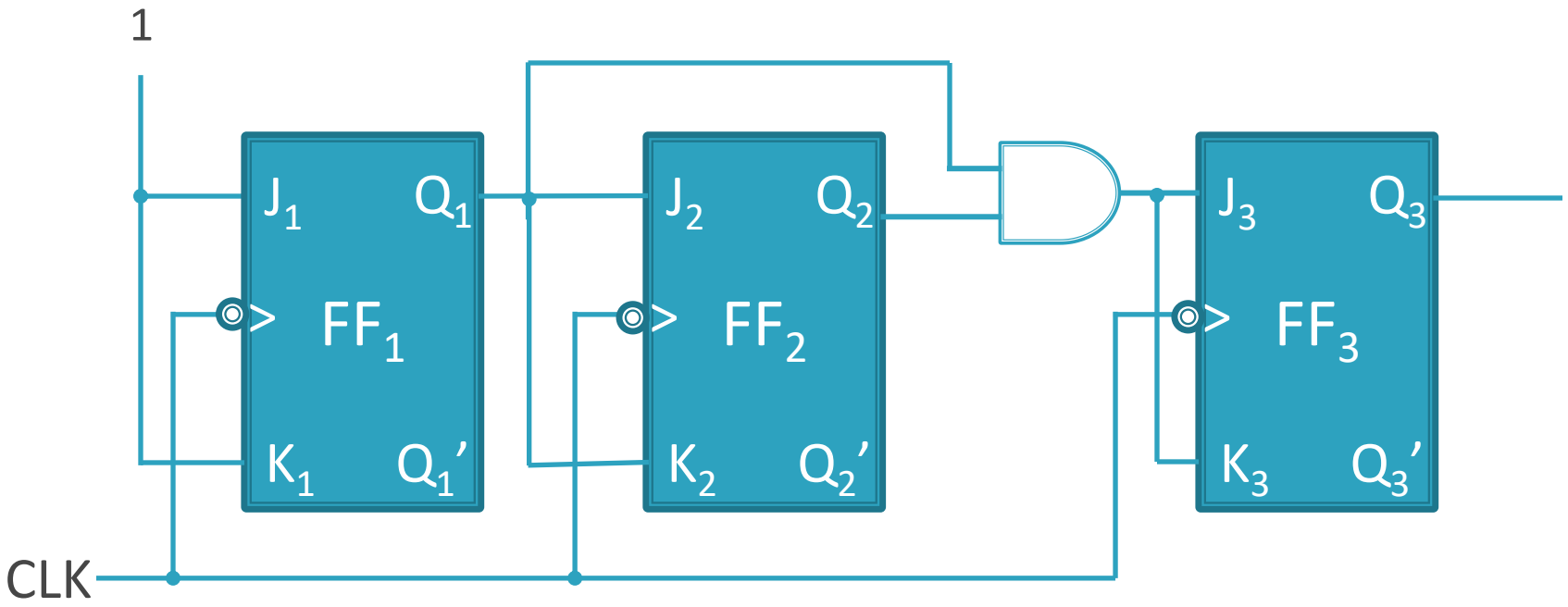
$J_2 = Q_1$



$K_2 = Q_1$

Design of Synchronous 3-bit Up Counters

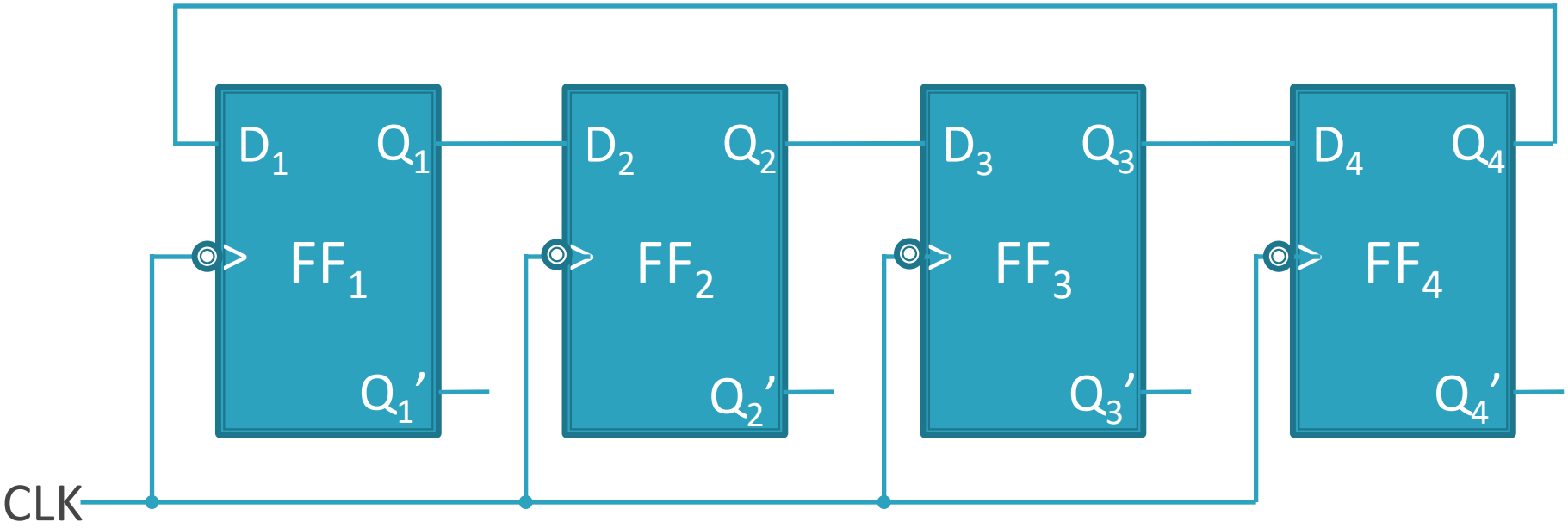
- Step 5. Draw the logic diagram



Shift Register Counter

- A shift register counter is basically a shift register with the serial output connected back to the serial input to produce special sequences.
- These devices are often classified as counters because they exhibit a specified sequence of states.
- Two of the most common types of shift register counters, the **Johnson** counter and the **ring** counter, are introduced in this section

Ring Counter



Ring Counter

After pulses	State			
	Q_1	Q_2	Q_3	Q_4
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1

Table I Truth Table of 4-bit Ring Counter

Clock Cycle	Q_1	Q_2	Q_3	Q_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Bit-pattern repeats for every 4 clock cycles

- **Merit**
  Ring counters have only one bit high at any instant of time. This makes them readily decodable in nature unlike other counters which make use of additional logic circuitry.
- **Demerit**
  A **ring counter** of n-bits has only n valid states instead of 2^n. This makes them inefficient in terms of state-usage.

Ring Counter

- A ring counter is basically a shift register counter in which the output of the first flip flop is connected to the next flip flop and so on and the output of the last flip flop is again fed back to the input of the first flip flop, thus the name ring counter.
- The data pattern within the shift register will circulate as long as clock pulses are applied.
- *Number of states = Number of flip-flops*

10-bit Ring Counter

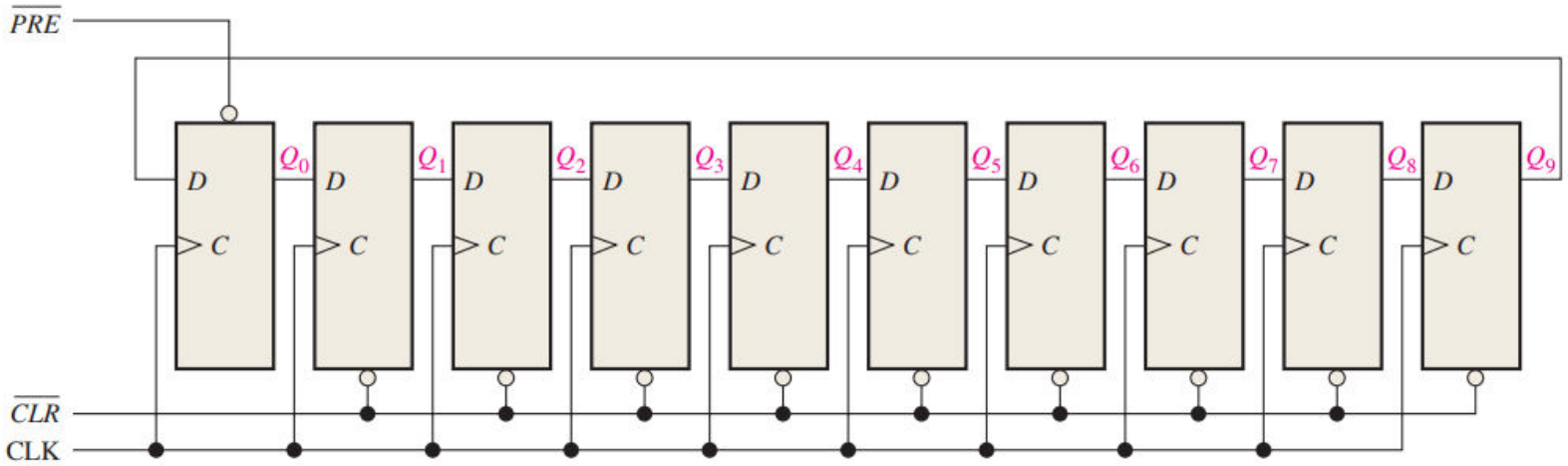


FIGURE 8–24 A 10-bit ring counter. Open file F08-24 to verify operation.

MultiSim

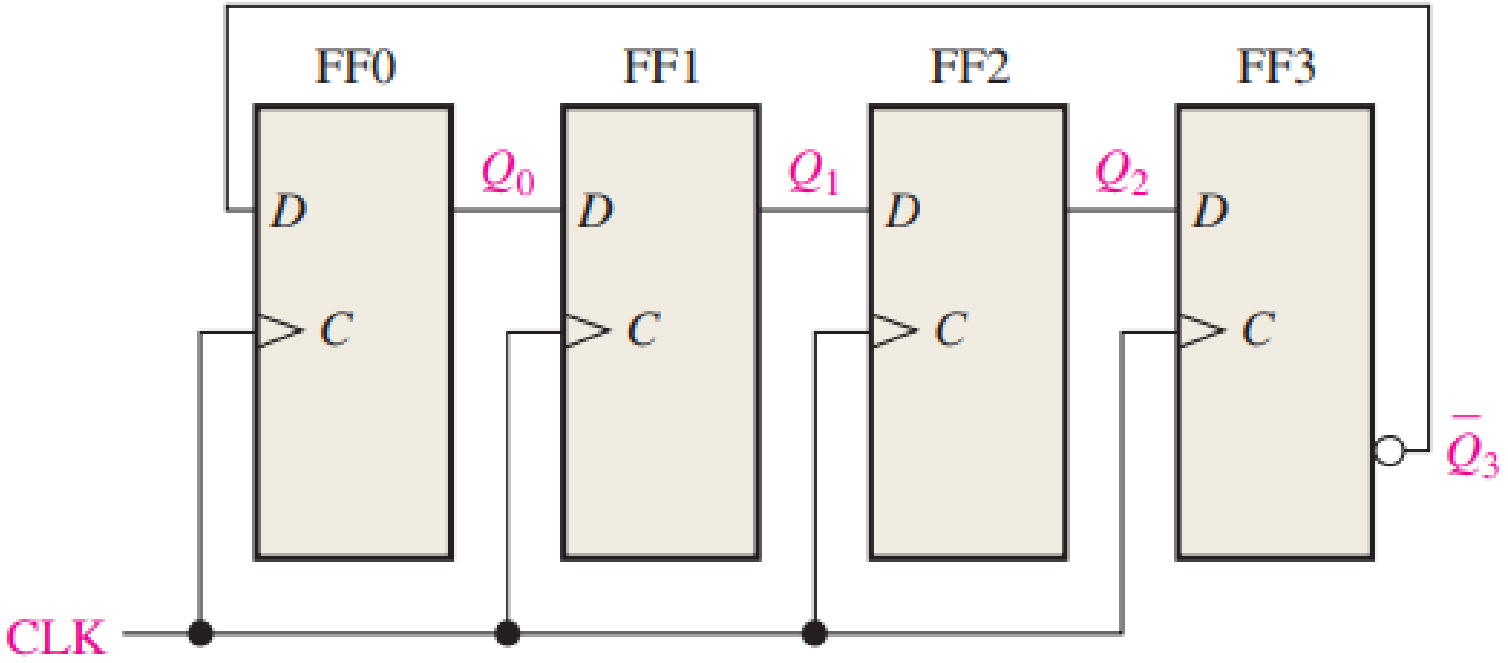
Ten-bit ring counter sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

Johnson Counter

- A Johnson counter is basically a shift register counter in which the output of the first flip flop is connected to the next flip flop and so on and the inverted output of the last flip flop is again fed back to the input of the first flip flop.
- They are also known as twisted ring counters.
- It is also known as switch-tail ring counter, walking ring counter or Johnson counter.
- *Number of states = 2 * Number of flip-flops*

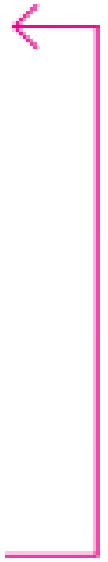
4-bit Johnson Counter



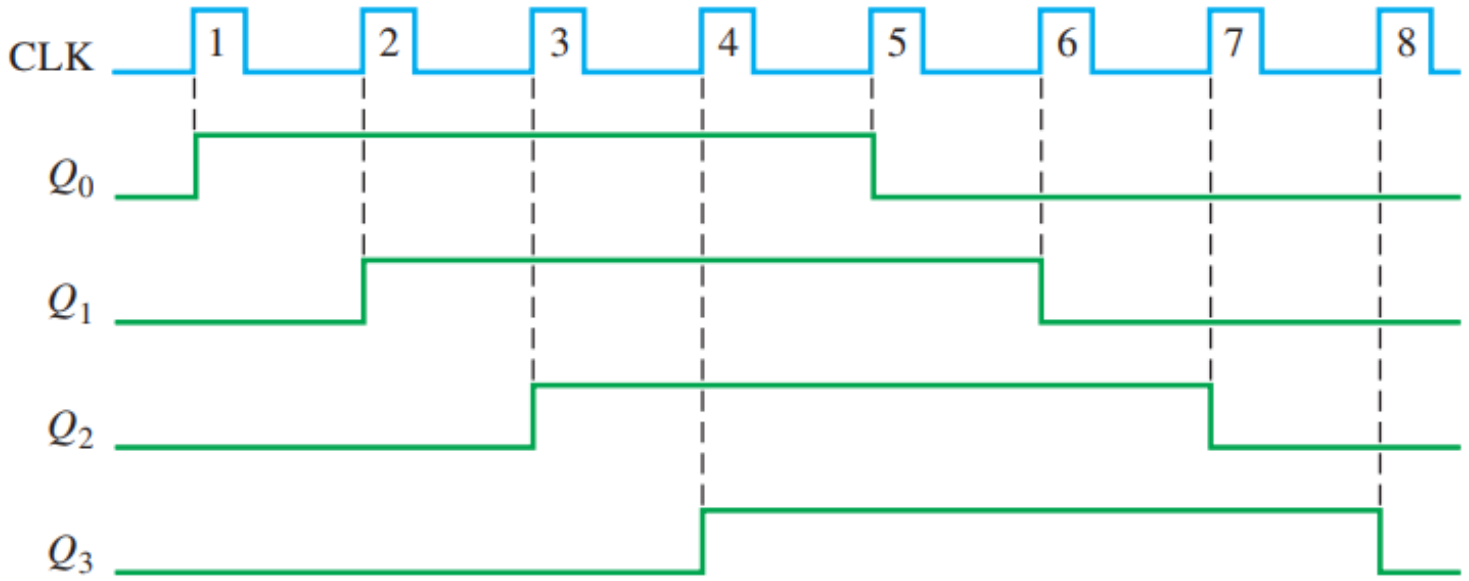
(a) Four-bit Johnson counter

Four-bit Johnson sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1



Timing Diagram



Self Check Exercise

- Construct and explain the logic circuit diagram for Digital Watch and Frequency counter.
- Design Mod-10 ripple counter.

Registers

- Flip-flop is a 1 bit memory cell which can be used for storing the digital data.
- To increase the storage capacity in terms of number of bits, we have to use a group of flip-flop.
- Such a group of flip-flop is known as a **Register**.
- The **n-bit register** will consist of **n** number of flip-flop and it is capable of storing an **n-bit** word.
- A register is a digital circuit with two basic functions: data storage and data movement. The storage capability of a register makes it an important type of memory device.

Registers

- 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

- The information stored within the registers can be transferred with the help of **shift registers**.
- Shift Register is a group of flip flops used to store multiple bits of data. The bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses.
- The registers which will shift the bits to left are called "**Shift left registers**".
- The registers which will shift the bits to right are called "**Shift right registers**".

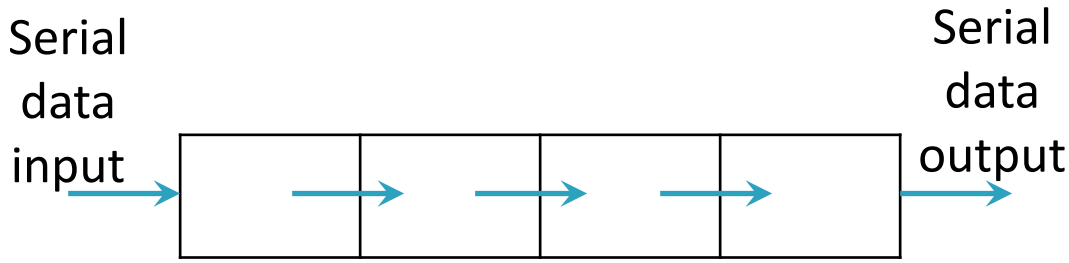
Shift Register

- 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.

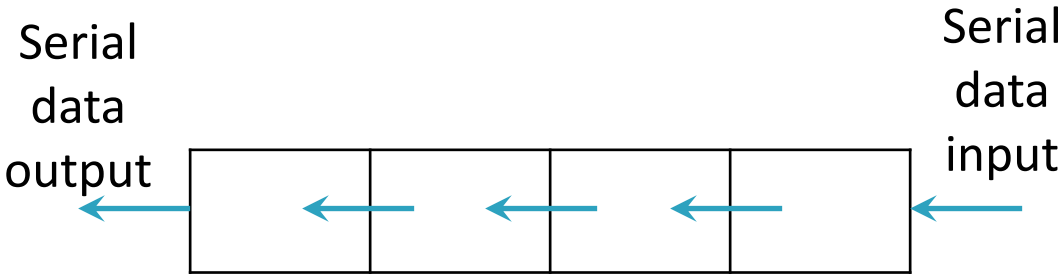
Types of Shift Register

- Shift registers are basically of 4 types. These are:
  1. Serial In Serial Out shift register(SISO)
  2. Serial In parallel Out shift register(SIPO)
  3. Parallel In Serial Out shift register(PISO)
  4. Parallel In parallel Out shift register(PIPO)

Data transmission in shift register

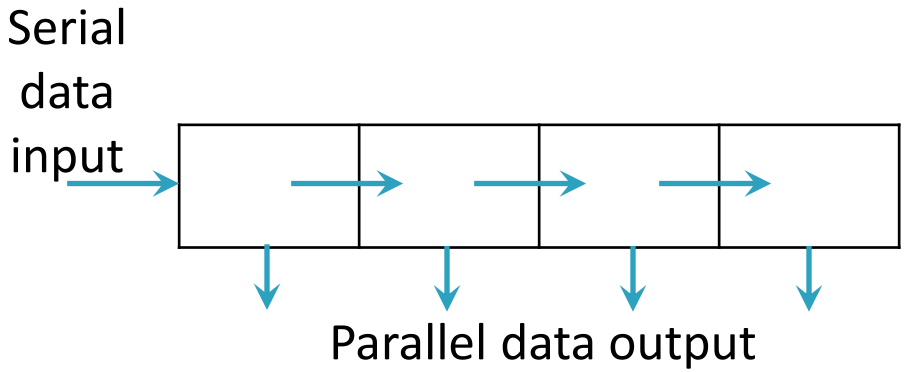


Serial-in, serial-out shift-right, shift register

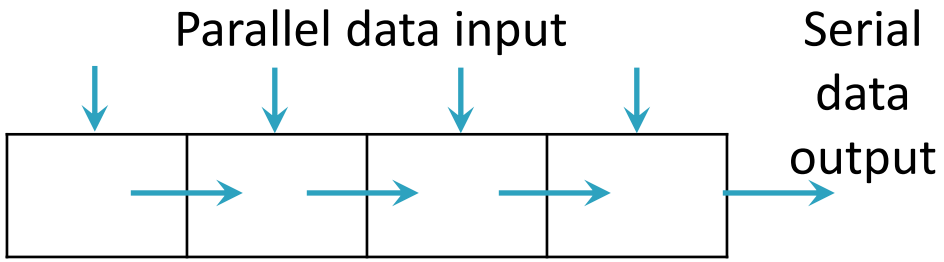


Serial-in, serial-out shift-left, shift register

Data transmission in shift register

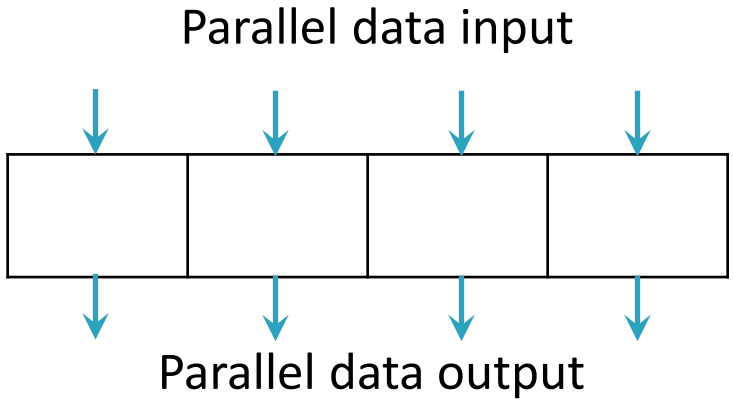


Serial-in, parallel-out, shift register



Parallel-in, serial-out, shift register

Data transmission in shift register

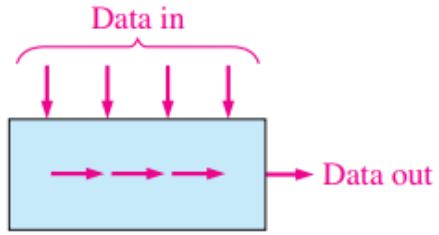


Parallel-in, parallel-out, shift register

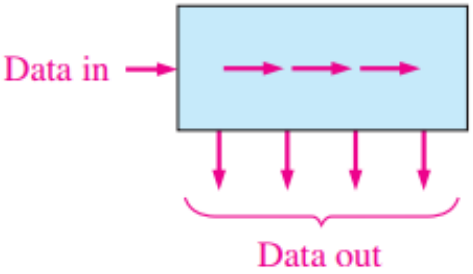
Data transmission Summary

(a) Serial in/shift right/serial out

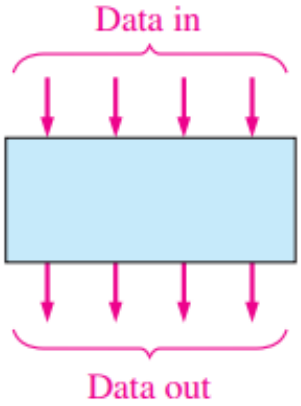
(b) Serial in/shift left/serial out



(c) Parallel in/serial out



(d) Serial in/parallel out

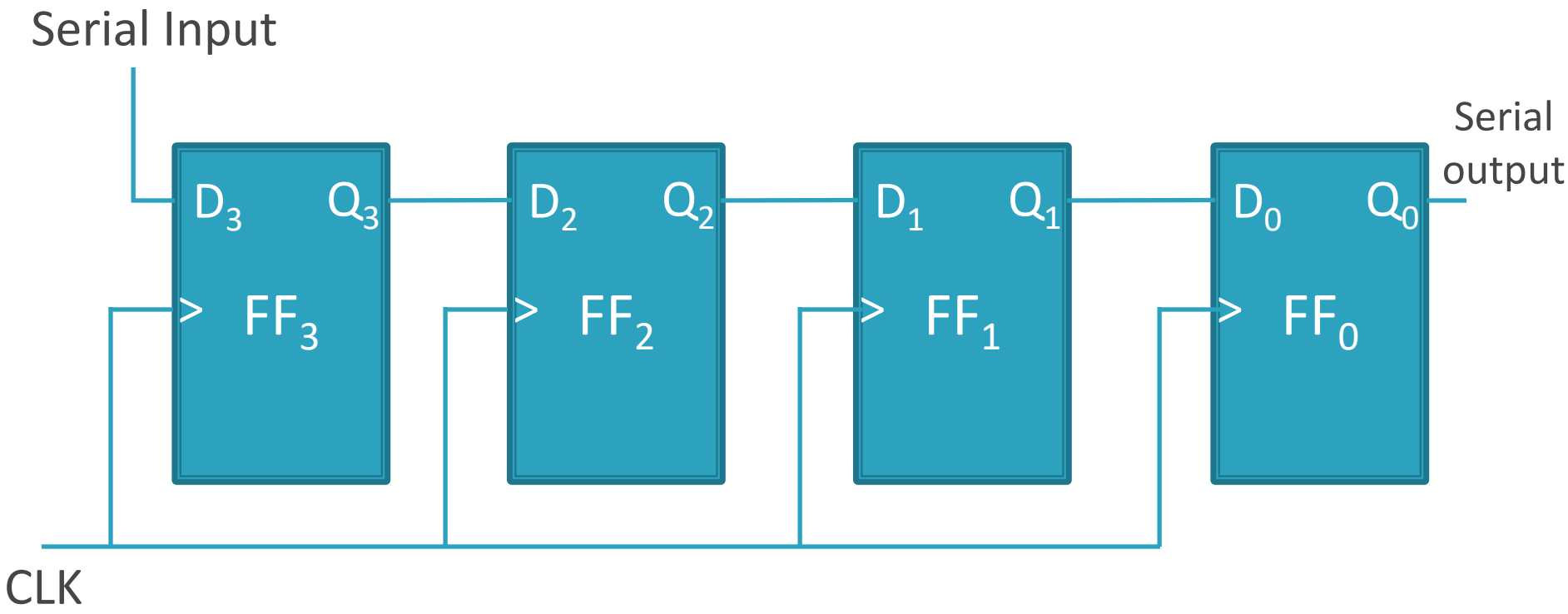


(e) Parallel in/parallel out

Serial-In Serial-Out Shift Register (SISO)

- The shift register, which allows serial input (one bit after the other through a single data line) and produces a serial output is known as Serial-In Serial-Out shift register.
- Since there is only one output, the data leaves the shift register one bit at a time in a serial pattern, thus the name Serial-In Serial-Out Shift Register.
- In this type of register, N number of clock pulses is required to load data into the flipflop and further (N-1) number of clock pulses are required to retrieve data from register.

Serial-in, Serial-out, Shift Right, Shift register

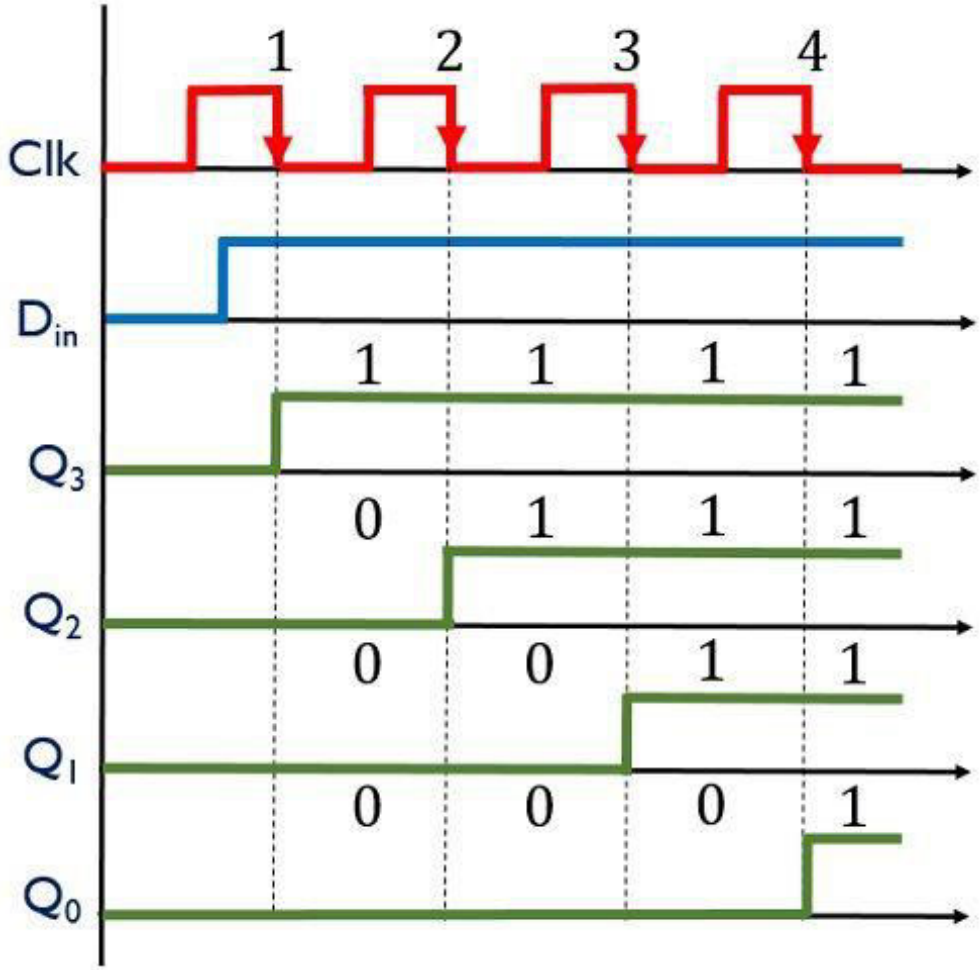


Truth Table

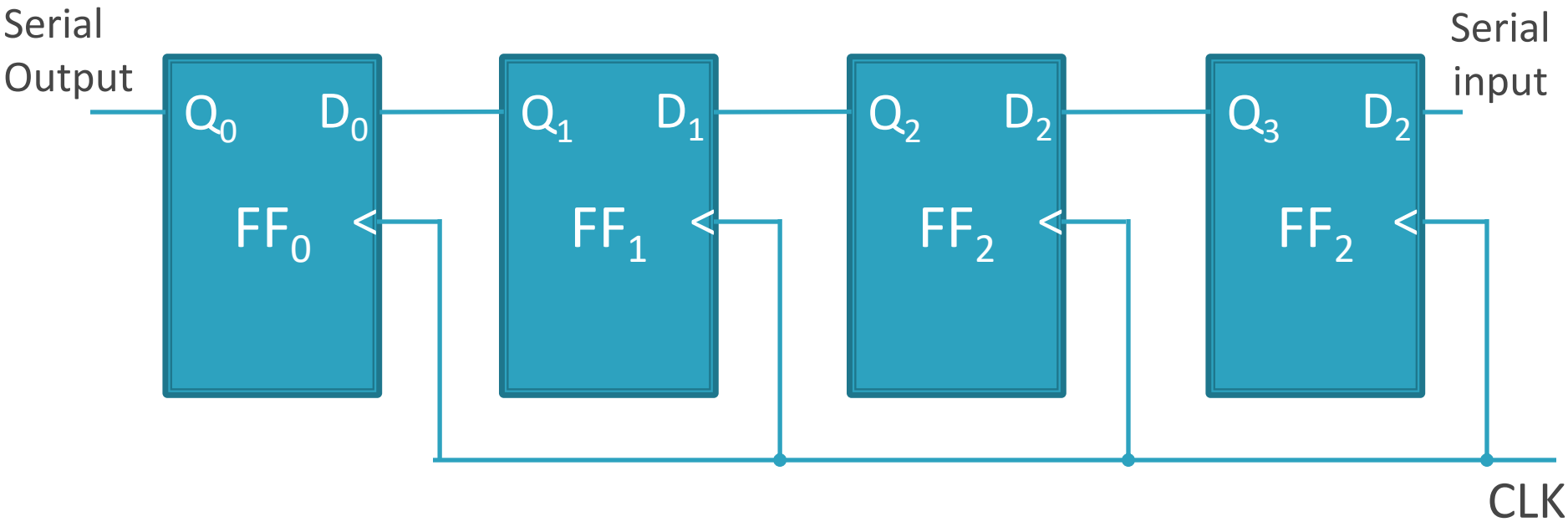
CLK	Q3	Q2	Q1	Q0
Initially	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1

Waveform of SISO shift register

CLK	Q3	Q2	Q1	Q0
Initially	0	0	0	0
1st falling edge	1	0	0	0
2nd falling edge	1	1	0	0
3rd falling edge	1	1	1	0
4th falling edge	1	1	1	1

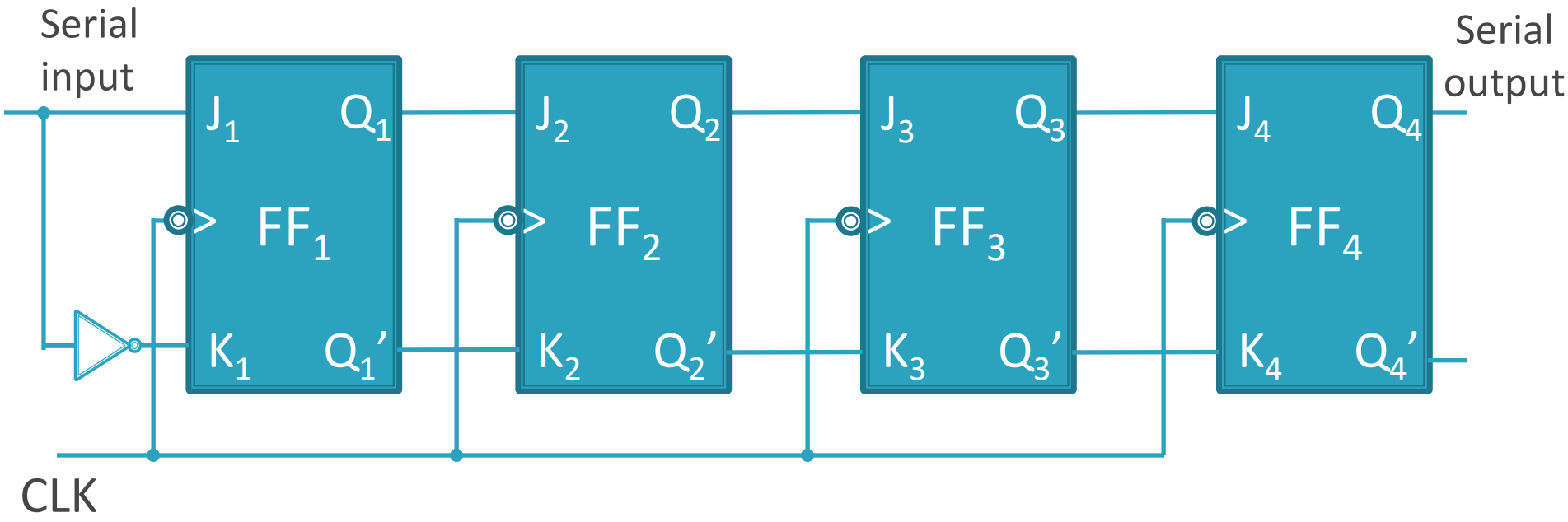


Serial-in, Serial-out, Shift-left, Shift register



Serial-in, Serial-out, Shift register

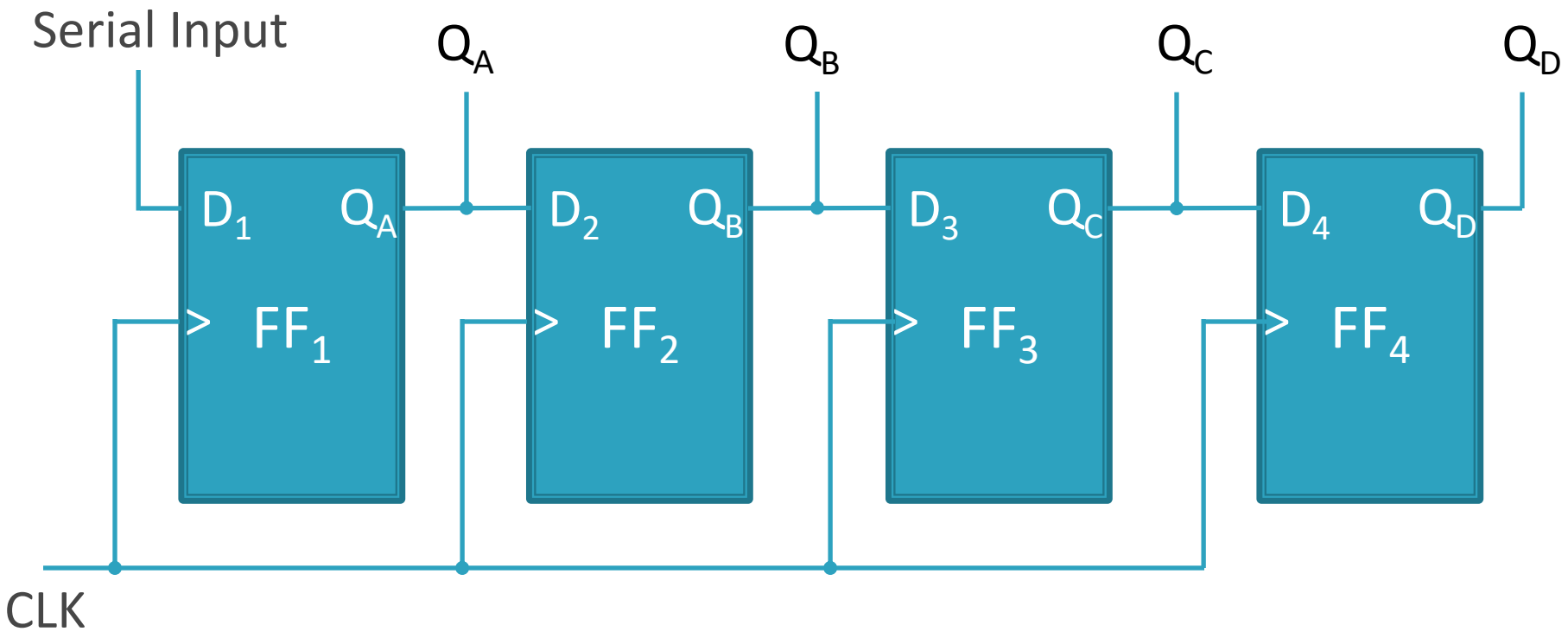
Using J-K Flip Flop



Serial-In Parallel-Out shift Register (SIPO)

- The shift register, which allows serial input (one bit after the other through a single data line) and produces a parallel output is known as Serial-In Parallel-Out shift register.
- In this type of register, **N** number of clock pulses is required to load data into the flipflop and further no more clock pulses are required to retrieve data from register.

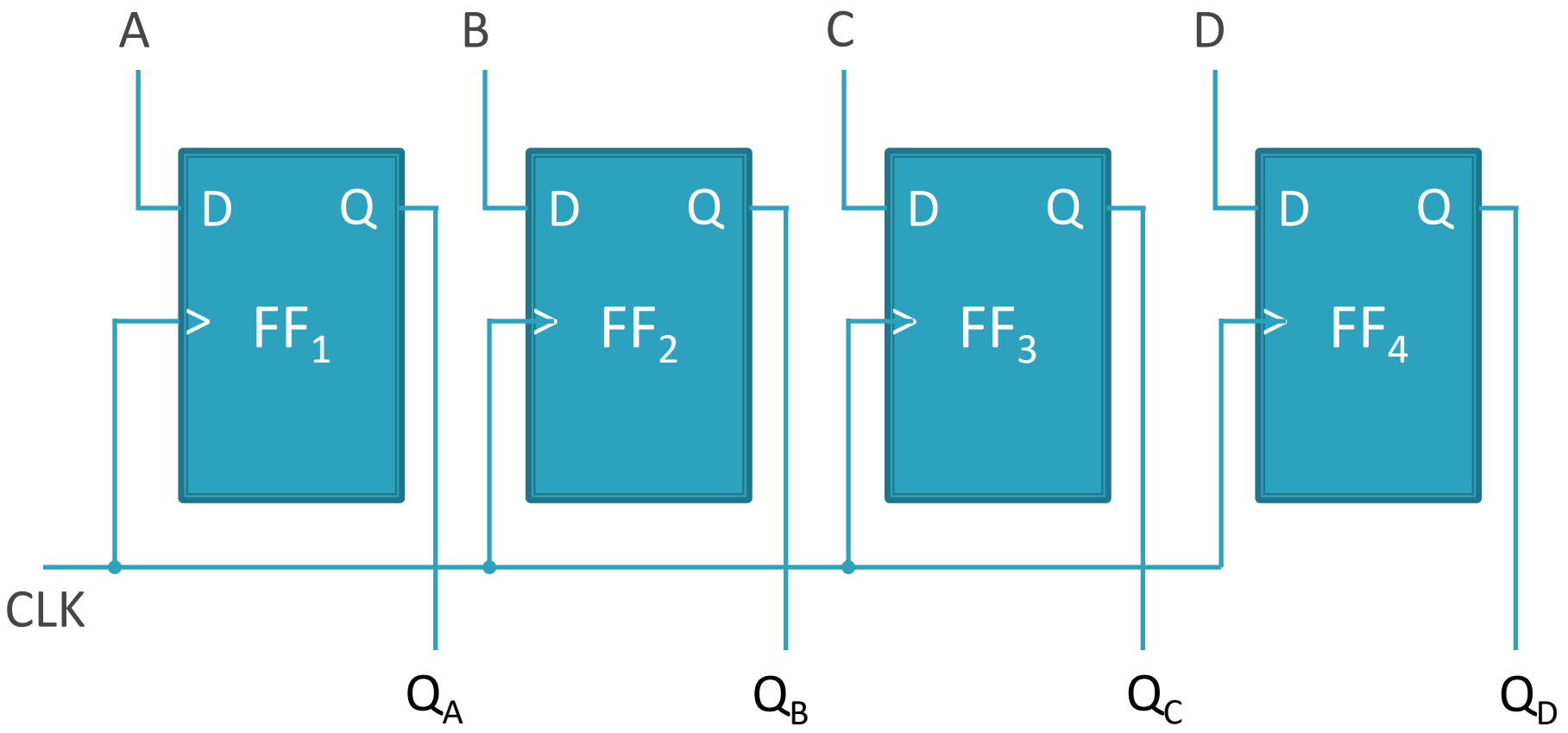
Serial-in, Parallel-out, Shift register



Parallel-In Parallel-Out Shift Register (PIPO)

- The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and also produces a parallel output is known as Parallel-In parallel-Out shift register.
- In this type of register, **1** number of clock pulse is required to load data into the flipflop and further **no more** clock pulses are required to retrieve data from register.

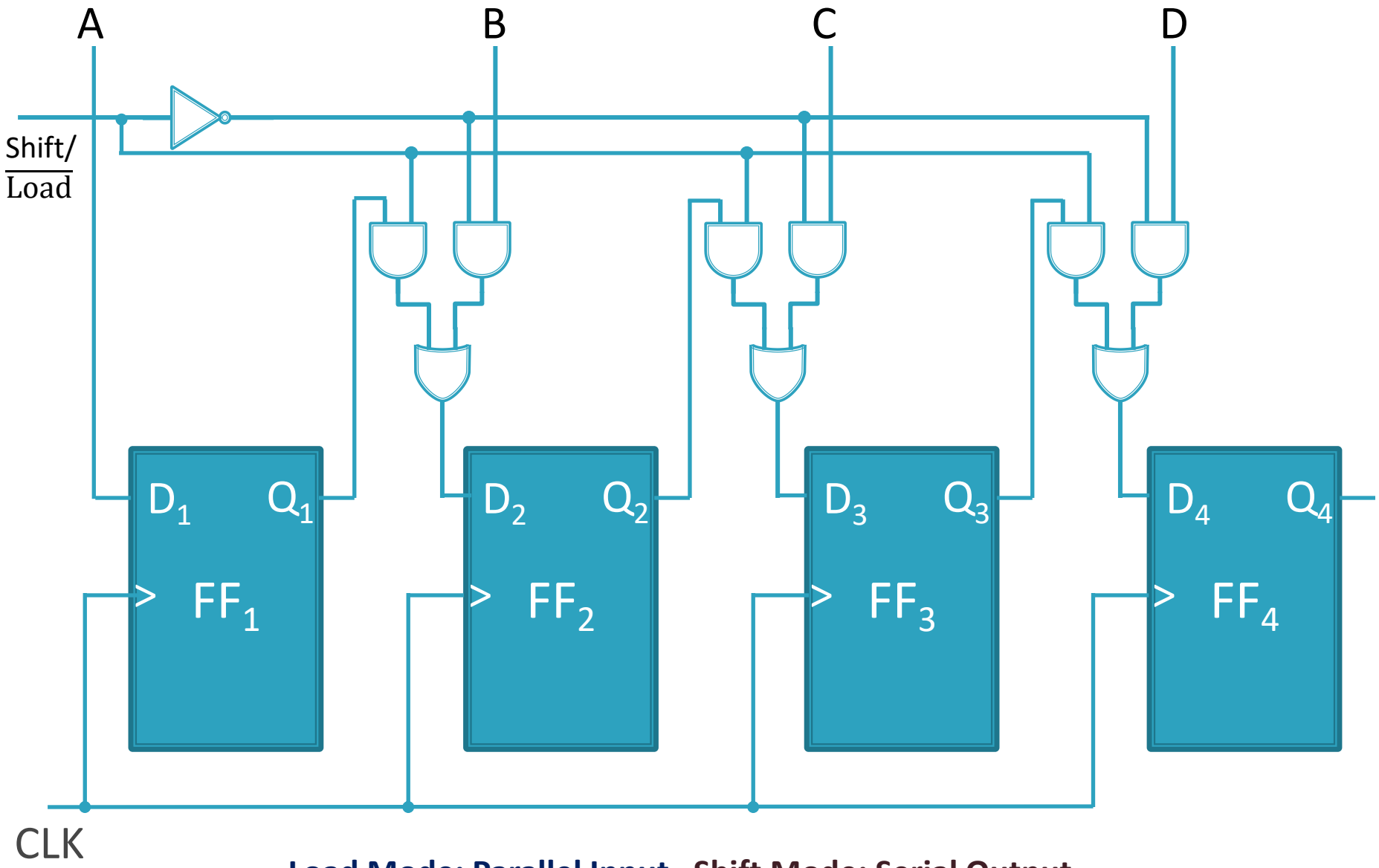
Parallel-in, Parallel-out, Shift register



Parallel-In Serial-Out Shift Register (PISO)

- For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs.
- The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and produces a serial output is known as Parallel-In Serial-Out shift register.
- In this type of register, **1** number of clock pulses is required to load data into the flipflop and further (**N-1**) number of clock pulses are required to retrieve data from register.

Parallel-in, Serial-out, Shift register



Load Mode: Parallel Input **Shift Mode: Serial Output**

Assignment

- Explain the application of counter in the following:
  - Digital Watch
  - Frequency Diagram
- Explain Bidirectional Shift Register with circuit diagram.
- Explain Universal Shift Register with circuit diagram.

Clock Needed

Mode	Clocks needed for n-bit shift register		
	Loading	Reading	Total
SISO	n	n-1	2n-1
SIPO	n	0	n
PISO	1	n-1	n
PIPO	1	0	1

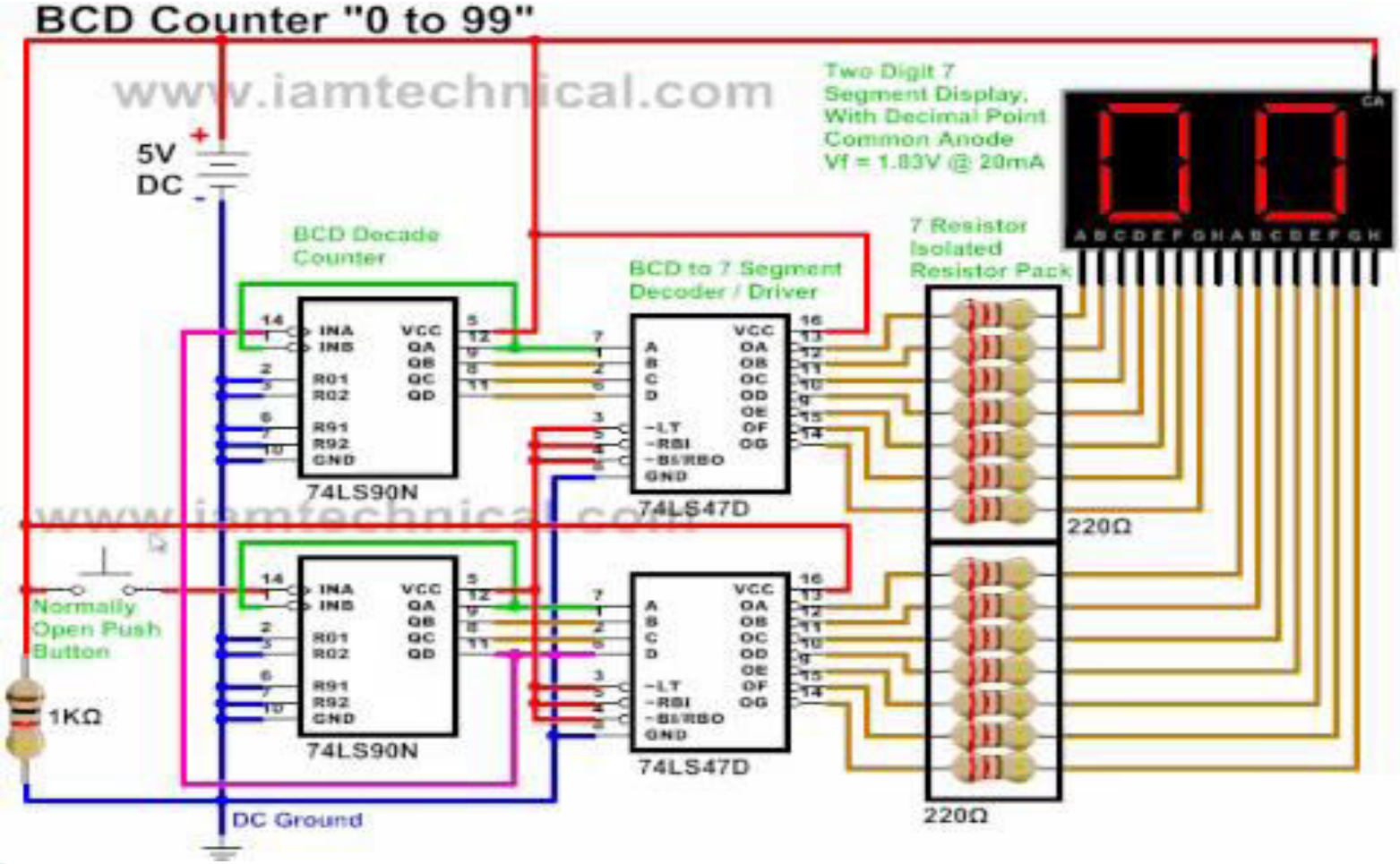
Self Check Exercise

- Application of shift register

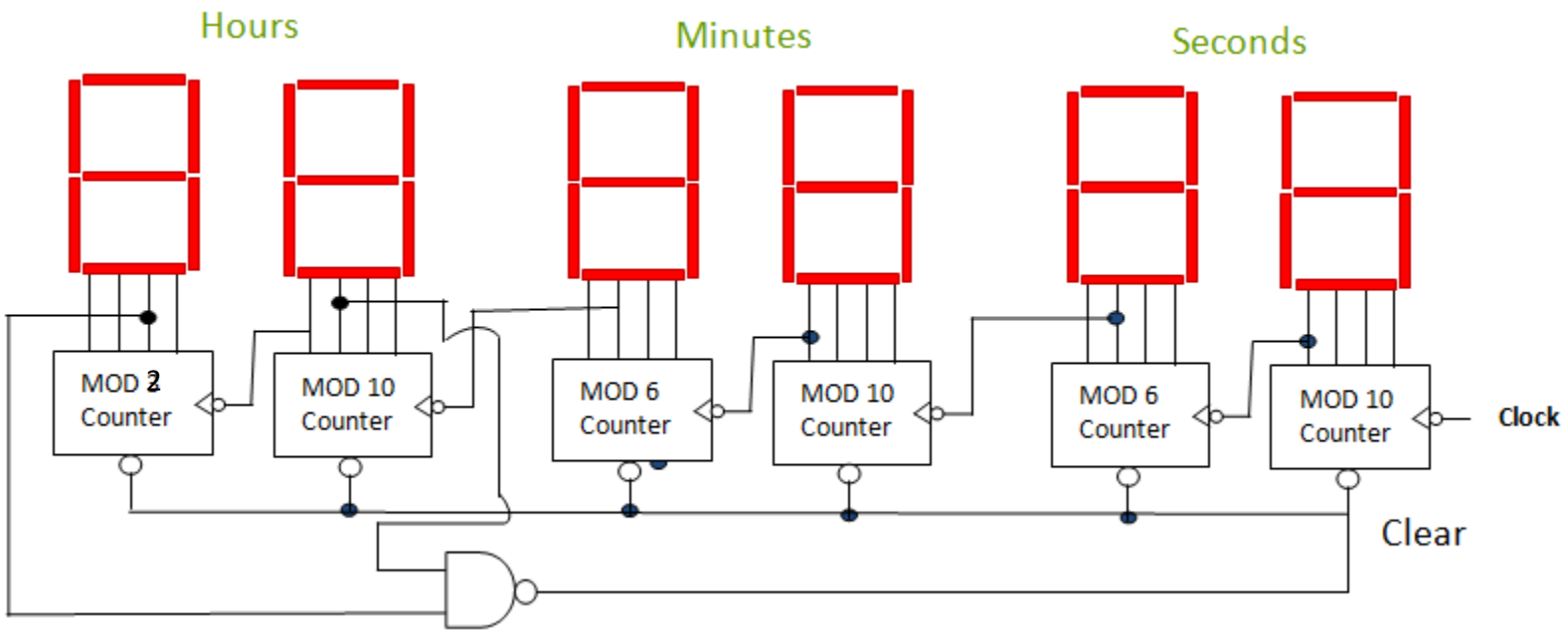
Some Applications of Counter

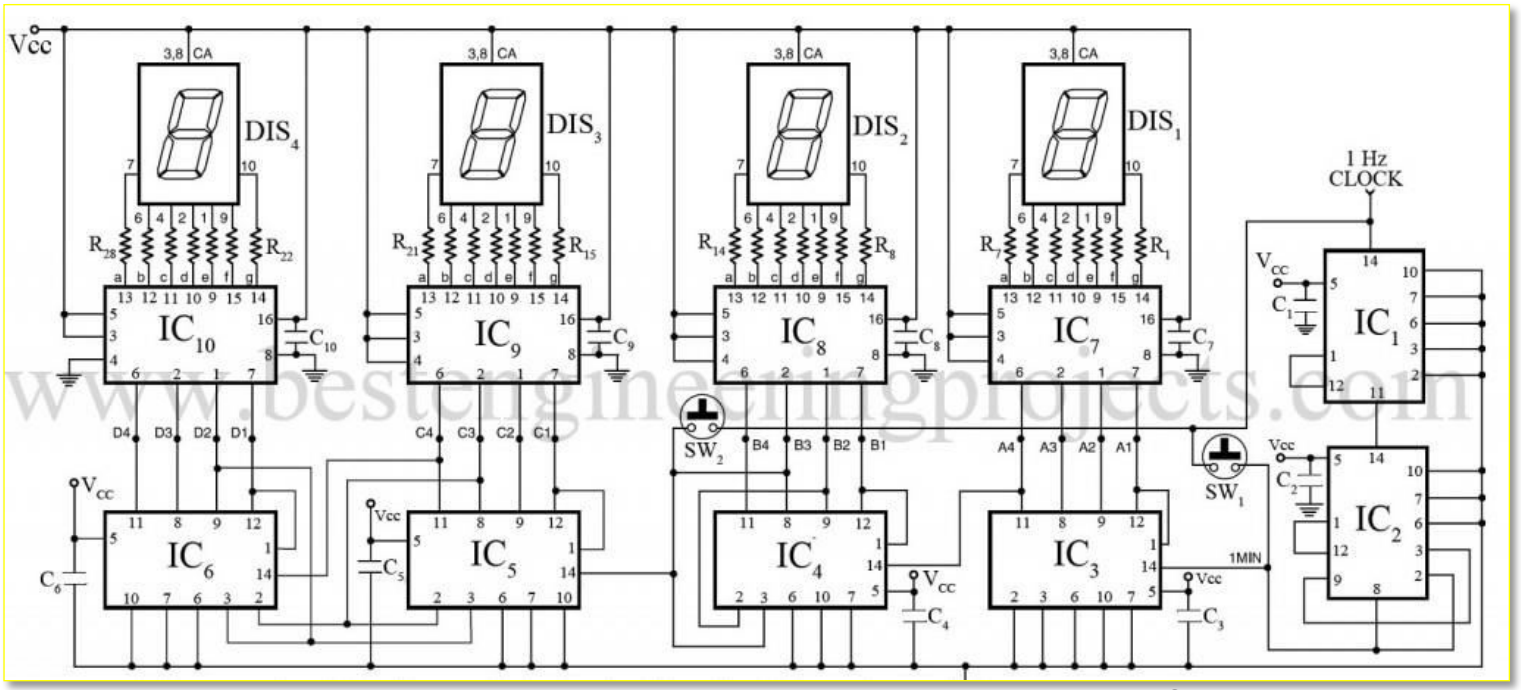
1. BCD Counter (0 to 99)
2. Digital Clock
3. Digital Multimeter
4. Frequency Counter
5. Frequency Divider
6. Analog to Digital Converters

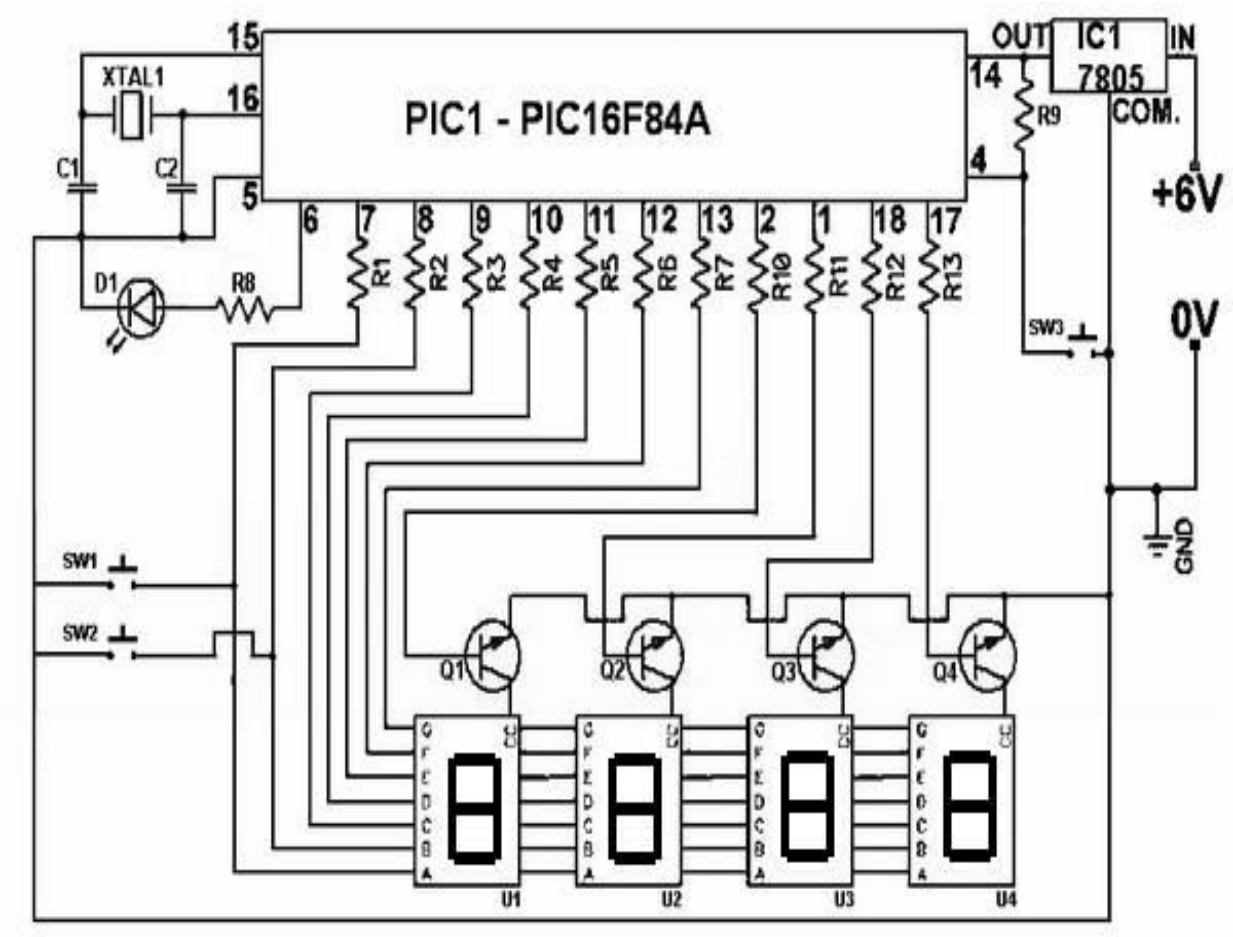
BCD counter (0 through 99) :



Digital Clock :

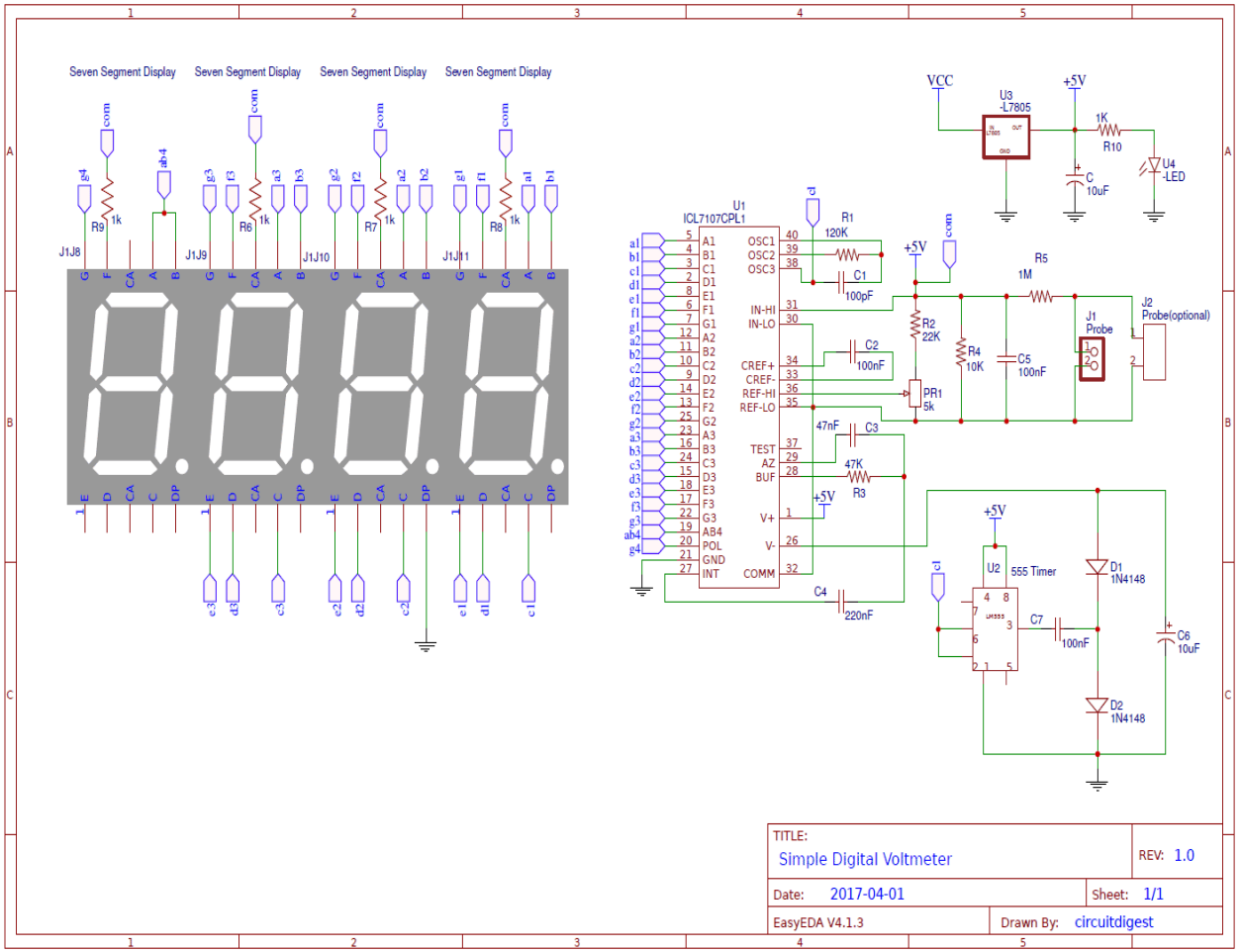






http://electronics-course.com/digital-clock

Digital Multimeter :

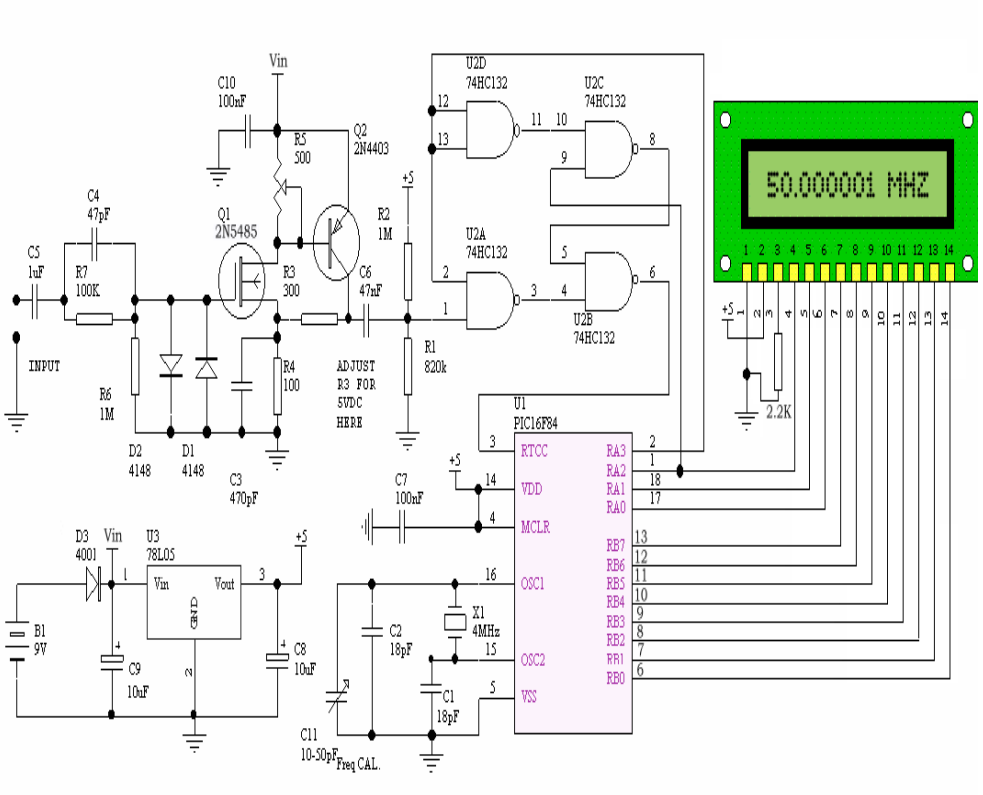


DMM Circuit Diagram .

Frequency Counter

- Frequency counters are test instruments used in many applications associated with radio frequency engineering to measure the frequency of signals very accurately.
- Most frequency counters work by using a counter that accumulates the number of events (oscillations) occurring with in a specific period of time (say, one second). After the preset time period, the value in the counter is transferred to a display and the counter is reset to zero.

Frequency Counter :



Frequency counter circuit diagram .

Synchronous State Machines

- We have seen how we can use FFs (D-types in particular) to design synchronous counters.
- We will now investigate how these principles can be extended to the design of synchronous state machines (of which counters are a subset)

Some Definitions

- **Finite State Machine (FSM)** – a deterministic machine (circuit) that produces outputs which depend on its internal state and external inputs
- **States** – the set of internal memorized values, shown as circles on the state diagram
- **Inputs –** External stimuli, labelled as arcs on the state diagram
- **Outputs –** Results from the FSM

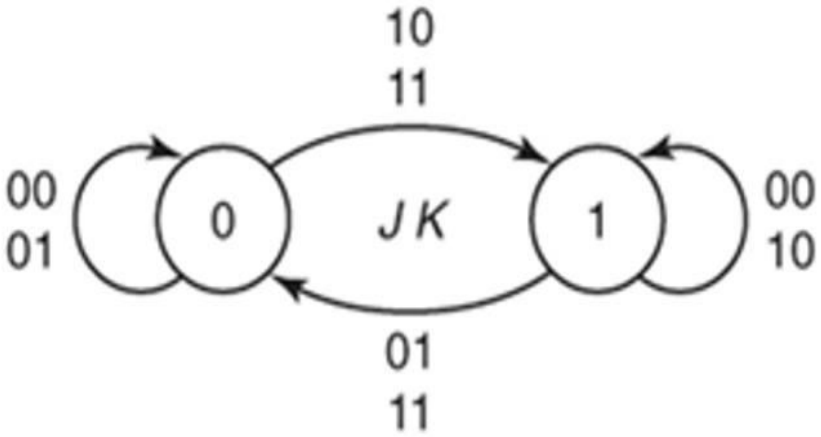
State Machine

- A state machine, also called a sequential machine, is a system that can be described in terms of a set of states that the system may enter.
- Once in a particular state, the system must be capable of remaining in that state for some finite period of time even if the system inputs change.
- This requirement dictates memory capability for the state machine.
- Furthermore, the state machine must have a set of inputs and a set of outputs.
- As the system progresses from one state to another (from present state to next state), the next state reached depends on the inputs and the present state.
- The outputs also depend on inputs and the present state.

JK Flip Flop State Diagram

PS	Input		NS
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

Characteristics Table of JK FF



Advantages of Finite State Machine

- Many electronic systems require the type of sequential operation exhibited by state machines.
- State machine design methods lead to minimal design.
- State machine methods eliminate the problems of narrow unwanted pulse or glitches on output lines or oscillation problems and reduce the time taken to debug the implemented hardware.

Advantages of Finite State Machine

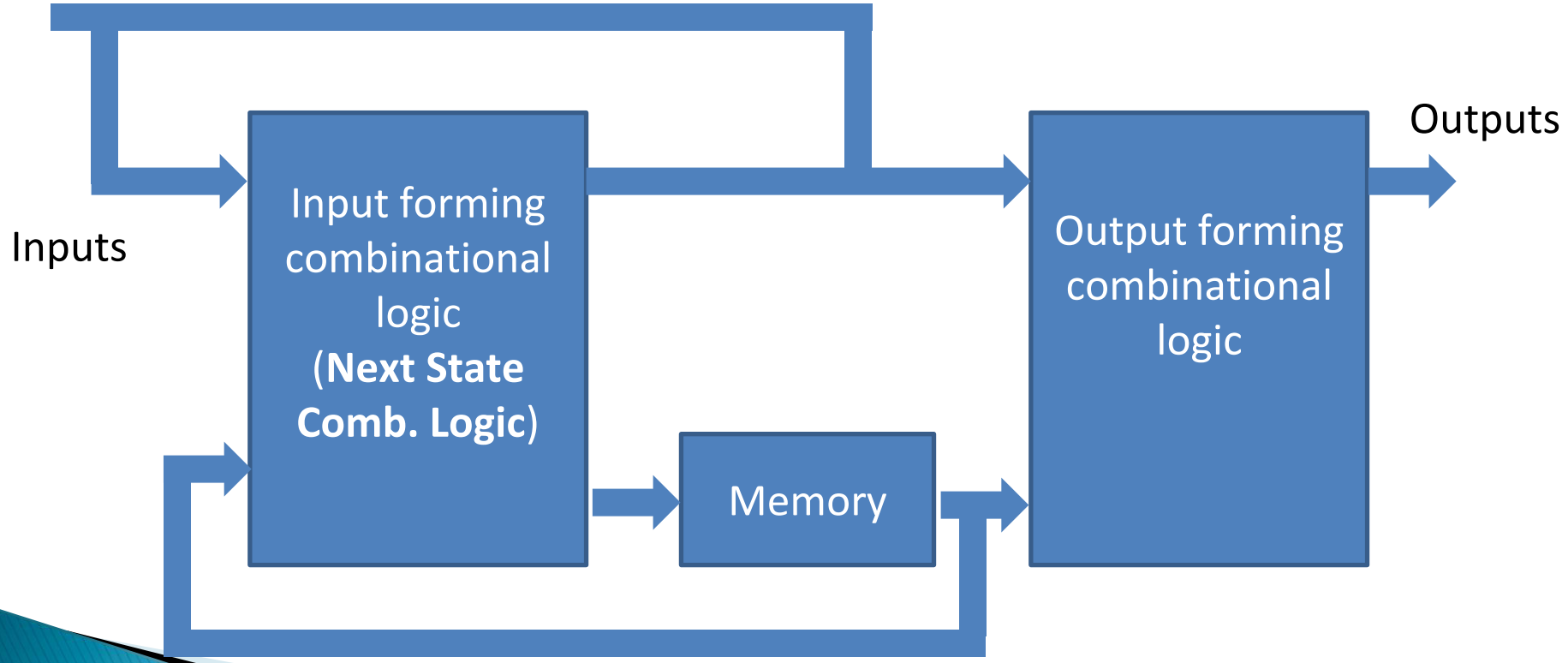
- Many electronic systems require the type of sequential operation exhibited by state machines.
- State machine design methods lead to minimal design. The state machine design procedure relates to sequential circuit design in the same way the K-map relates to combinational circuit design. This method results in the minimum number of required flip-flops and can minimize other circuitry in the system as well.
- It is a well-developed, orderly procedure that anticipates and solves commonly occurring problems of sequential circuits. State machine methods eliminate the problems of narrow unwanted pulse or glitches on output lines or oscillation problems and reduce the time taken to debug the implemented hardware.

Types of Finite State Machine

1. Mealy Model
2. Moore Model

1. Mealy Model

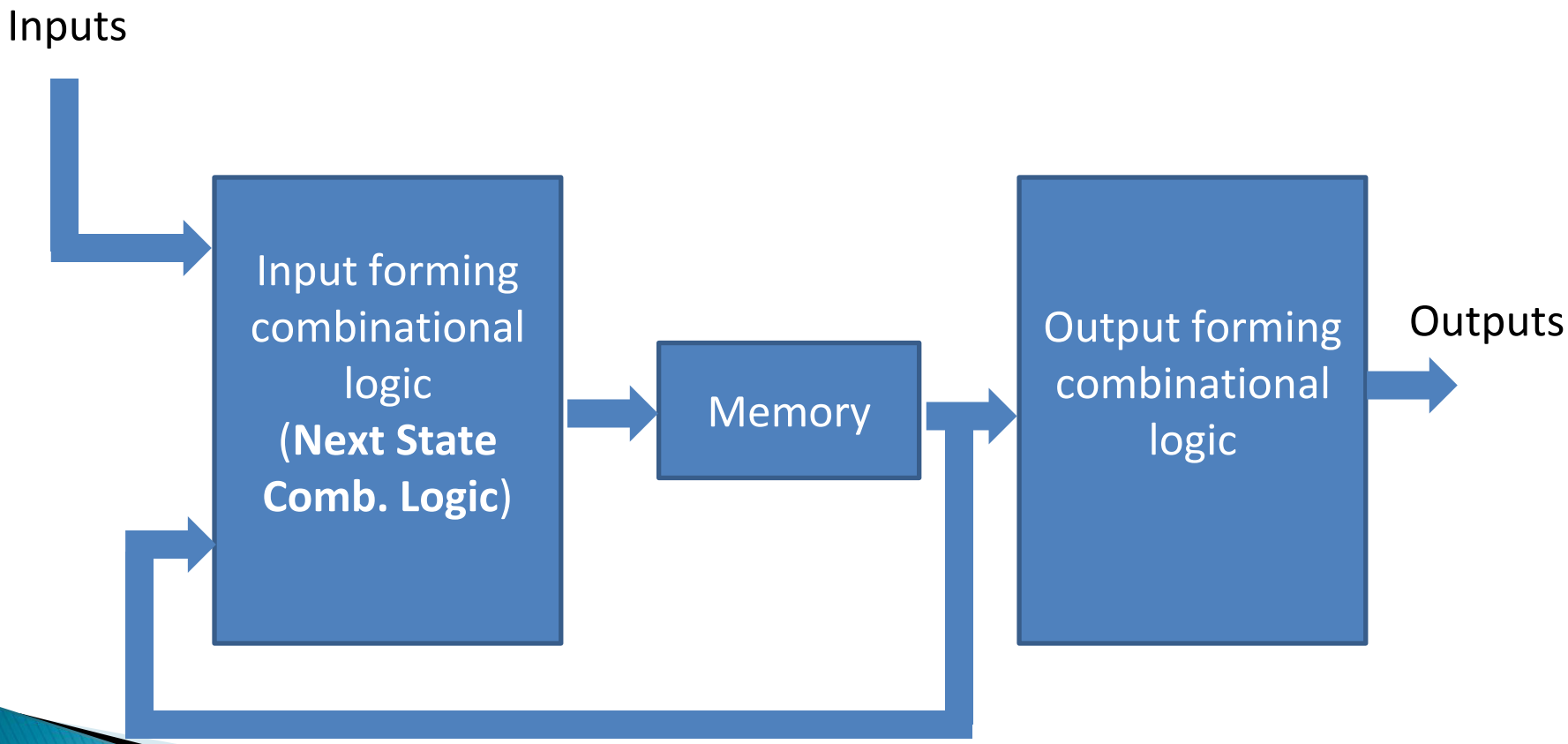
- When the output of the sequential circuit depends on both the present state of the flip-flops and on the inputs, the sequential circuit is referred to as **Mealy circuit or Mealy machine**.



- This model is also called General Model of Sequential Machine
- The input forming logic **(IFL)** and the output forming logic **(OFL)** sections are made up of **combinational logic circuits**.
- The memory section contains the state of the system.
- A path is provided from memory output to the IFL. Both input signals and present state signals drive the IFL to determine the next state of the system.
- The outputs are determined by the present state and the system inputs.

2. Moore Model

- When the output of the sequential circuit depends only on the present state of the flip-flop, the sequential circuit is referred to as **Moore circuit or the Moore machine.**



- A slight variation of the Mealy machine is the Moore machine, which uses only the memory to drive the OFL. In this case, the output is a function only of the state of the system.

Sequential Machine Design Process

1. **Specification**: A description of the sequential circuit. Should include a detailing of the inputs, the outputs, and the operation. Possibly assumes that you have knowledge of digital system basics.
2. **Formulation**: Generate a state diagram and/or a state table from the statement of the problem.
3. **State Assignment**: From a state table assign binary codes to the states.
4. **Flip-flop Input Equation Generation**: Select the type of flip-flop for the circuit and generate the needed input for the required state transitions

5. **Output Equation Generation**: Derive output logic equations for generation of the output from the inputs and current state.
6. **Optimization**: Optimize the input and output equations. Today, CAD systems are typically used for this in real systems.
7. **Technology Mapping**: Generate a logic diagram of the circuit using ANDs, ORs, Inverters, and F/Fs.
8. **Verification**: Use a HDL(Hardware Description Language) to verify the design.