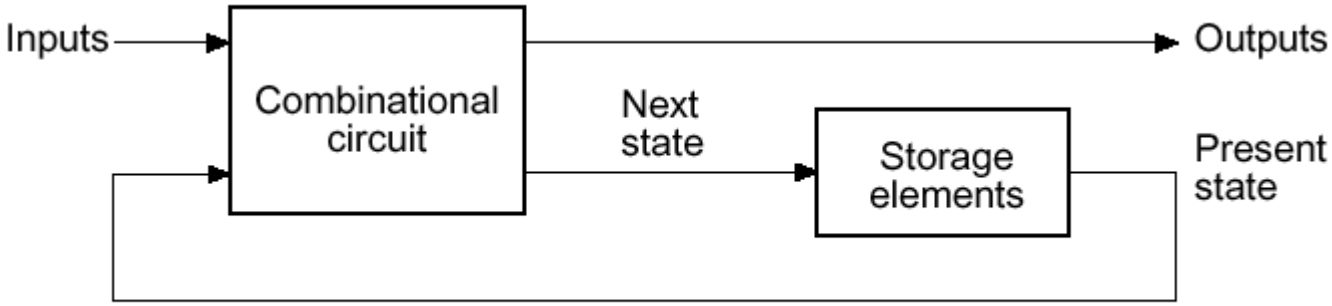
Introduction to Sequential Design

Sequential Logic circuits

- Remembers past inputs and past circuit state.
- Outputs from the system are “fed back” as new inputs.
- The storage elements are circuits that are capable of storing binary information: memory.

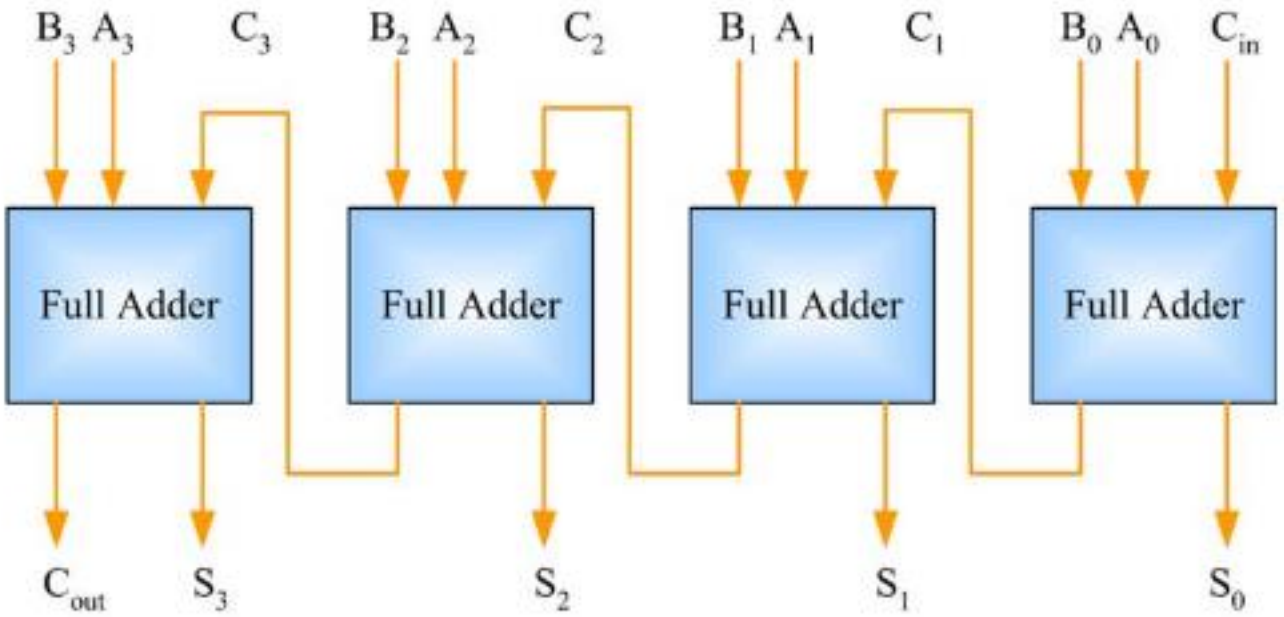


- A combinational circuit with **feedback** through **memory**
  - The stored information at any time defines a **state**
- Outputs depends on inputs and previous inputs
  - Previous inputs are stored as binary information into memory
- Next state depends on inputs and present state

Examples of sequential systems

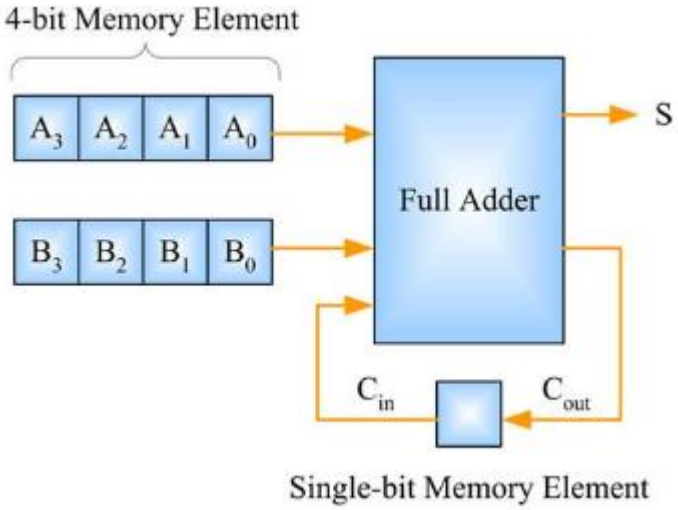
- Traffic light
- ATM
- Vending machine

Combinational Adder



- 4-bit adder (ripple-carry)
  - Notice how carry-out propagates
  - One adder is active at a time
- 4 full adders are needed

Sequential Adder

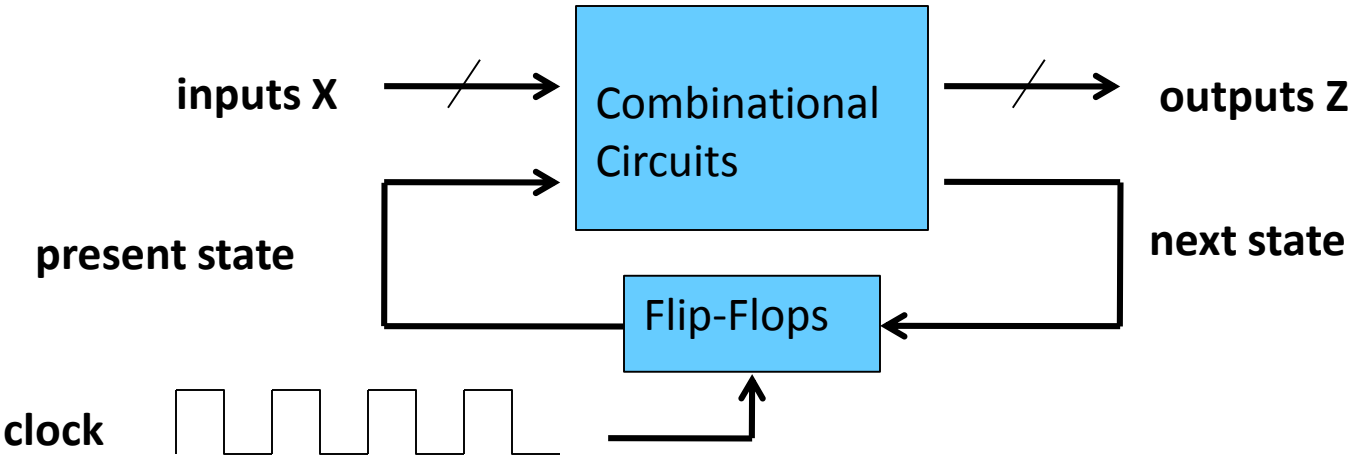


- 1-bit memory and 2 4-bit memory
- Only one full-adder
- 4 clocks to get the output
- The 1-bit memory defines the circuit state (0 or 1)

Types of Sequential Circuits

- Two types of sequential circuits:
  - **Synchronous**: The behavior of the circuit depends on the input signal at discrete instances of time *(also called clocked)*
  - **Asynchronous**: The behavior of the circuit depends on the input signals at any instance of time and the order of the inputs change
    - A combinational circuit with feedback

Synchronous Sequential Circuits



- Synchronous circuits employs a synchronizing signal called **clock** (a periodic train of pulses; 0s and 1s)
- A clock determines **when** computational activities occur

- The storage elements (memory) used in clocked sequential circuits are called **flip-flops**
  - Each flip-flop can store one bit of information 0,1
  - A circuit may use many flip-flops; together they define the circuit state
- Flip-Flops (memory/state) update **only** with the clock

Storage Elements (Memory)

- A storage element can maintain a binary state (0,1) indefinitely, until directed by an input signal to switch state
- Main difference between storage elements:
  - Number of inputs they have
  - How the inputs affect the binary state
- Two main types:
  - Latches
  - Flip-Flops

- Latches are useful in asynchronous sequential circuits
- Flip-Flips are built with latches
- The difference between a latch and a flip-flop is that a latch does not have a clock signal, whereas a flip-flop always does.

Latches

- A **latch** is binary storage element
- Can store a 0 or 1
- The most basic memory
- Easy to build
  - Built with gates (NORs, NANDs, NOT)

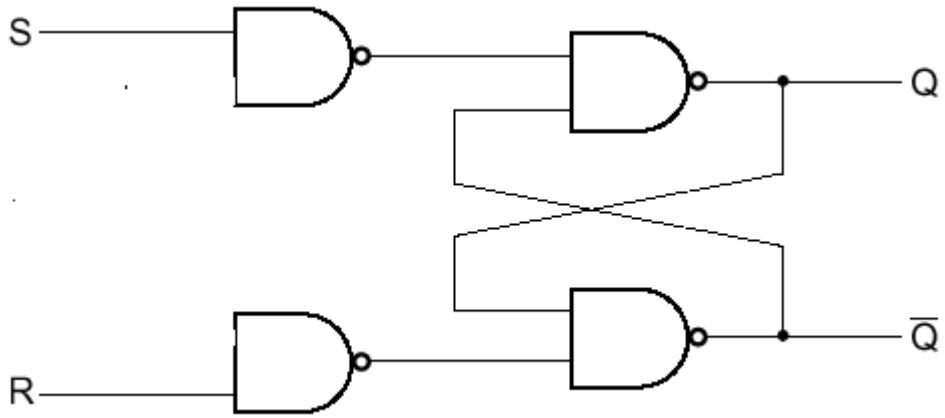
Flip-Flop

- Latches are a synchronous, which means that the output changes very soon after the input changes. Most computers today, on the other hand, are synchronous, which means that the outputs of all the sequential circuits change simultaneously to the rhythm of a global clock signal.
- A flip-flop is a synchronous version of the latch.

Flip-Flop Applications

- Parallel Data Storage
- Frequency Division
- Counting

Basic flip flop circuit



Flip-Flop

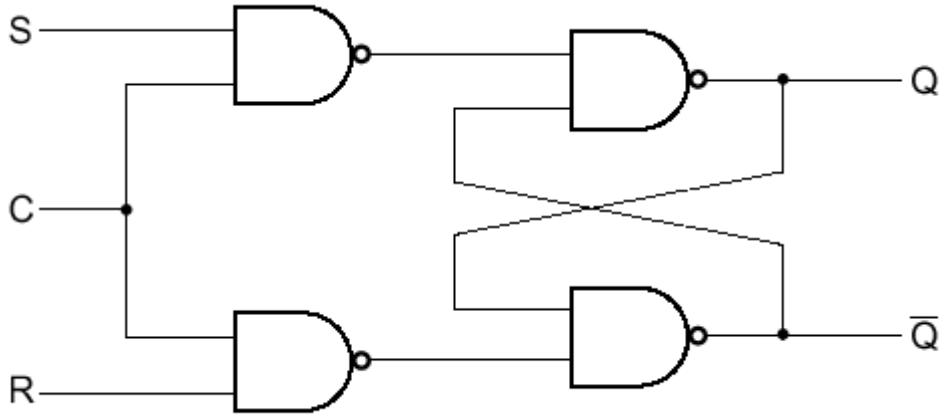
- A flip-flop circuit can be constructed from two NAND gates or two NOR gates.
- Each flip-flop has two outputs, Q and Q’, and two inputs, set and reset.
- This type of flip-flop is referred to as an SR flip-flop or SR latch.

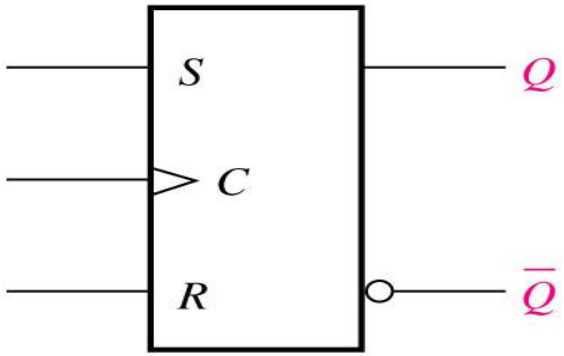
- The flip-flop has two useful states.
  - When Q=1 and Q’=0, it is in the set state (or 1 -state).
  - When Q =0 and Q’=1, it is in the clear state (or 0 -state).
- The outputs Q and Q’ are complements of each other and are referred to as the normal and complement outputs, respectively. The binary state of the flip-flop is taken to be the value of the normal output.

Types of Flip flops/latch

- SR
- JK
- D
- T
- Master Slave

SR Flip flop





Characteristic Tables

- Defines the logical properties of a flip-flop (such as a truth table does for a logic gate).
- Q(t) – present state at time t
- Q(t+1) – next state at time t+1

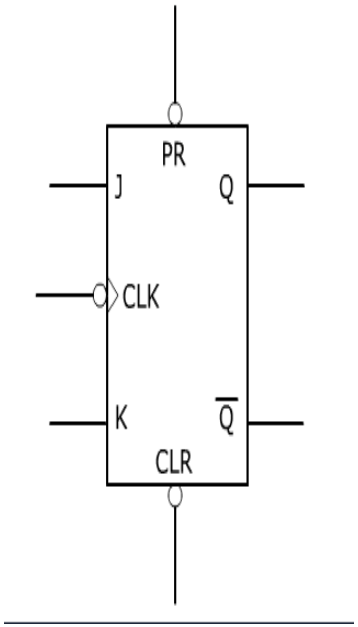
Characteristic Table of SR

S	R	Q(t+1)	Operation
0	0	Q(t)	No change/Hold
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined/Invalid

X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

JK Flip flop

- The uncertainty in the state of SR flip flop when S=R=1, can be eliminated by converting it in to JK flip flop



Has 5 inputs named:

J(set),K(reset), PR, CLR, and CLK

Has 2 outputs: Q and Q'

PR = Preset

CLR = Clear

CLK = Clock

Set: when it stores a binary 1

Cleared (reset): when it stores a binary 0

The PR and CLR inputs always **override** the J,K inputs.

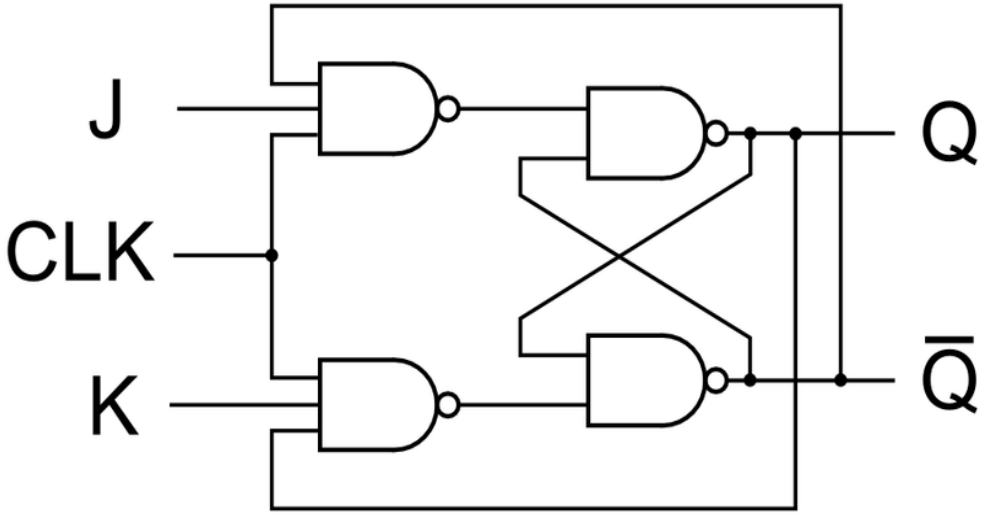
- A low at the PR input sets Q = 1
- A low at the CLR input sets Q = 0

Inputs: J and K

- The logic states applied to the J and K inputs cause the flip-flop to operate 4 different ways.
- The way the logic state is applied to J and K is called **Mode of Operation**.
- The mode of operation refers to the condition of the flip-flop as it prepares for the positive clock pulse.

Four Modes Of Operation

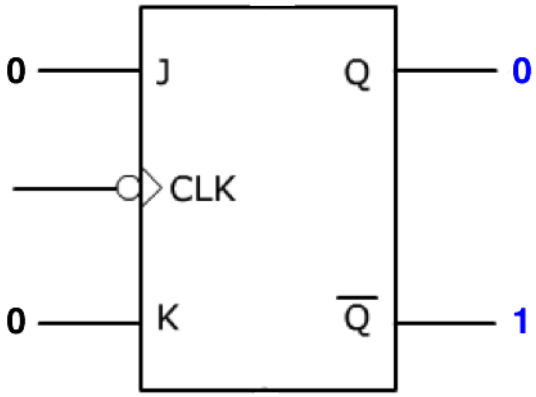
- The 4 modes of operation are: **hold, set, reset, toggle**

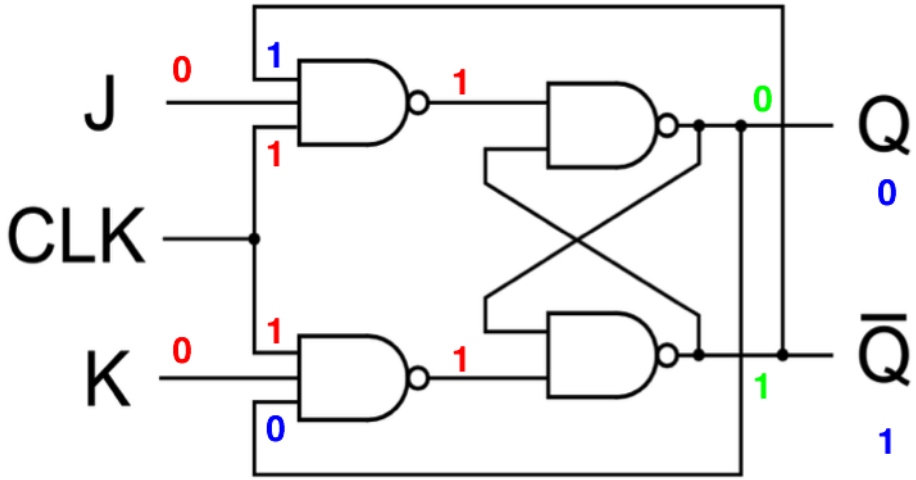


J	K	Q	Q'	Mode
0	0	Q	Q'	Hold
1	0	1	0	Sets
0	1	0	1	Resets
1	1	Q'	Q	Toggle

Mode of Operation: Hold

- **Hold**: no change in Q.





Truth Table for NAND

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

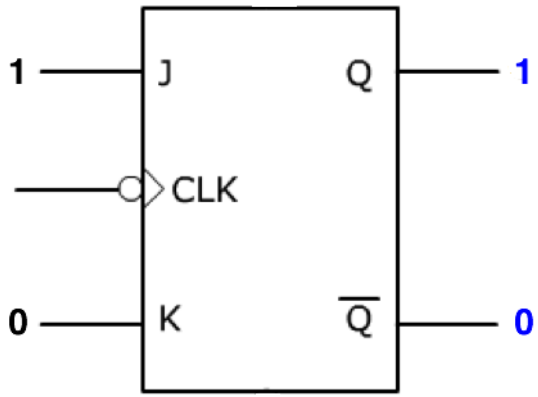
2 Inputs

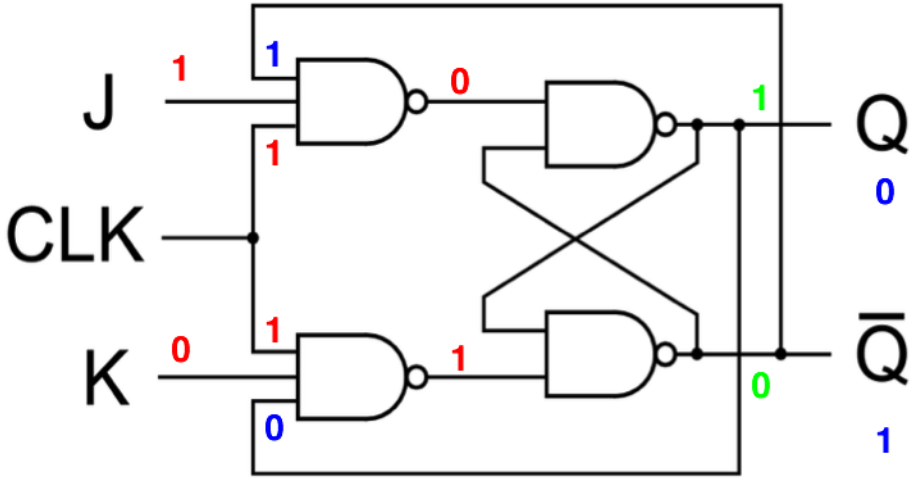
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

3 Inputs

Mode of Operation: Set

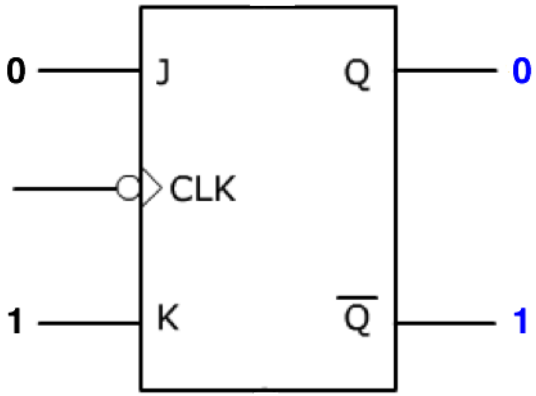
- **Set**: Q = 1.

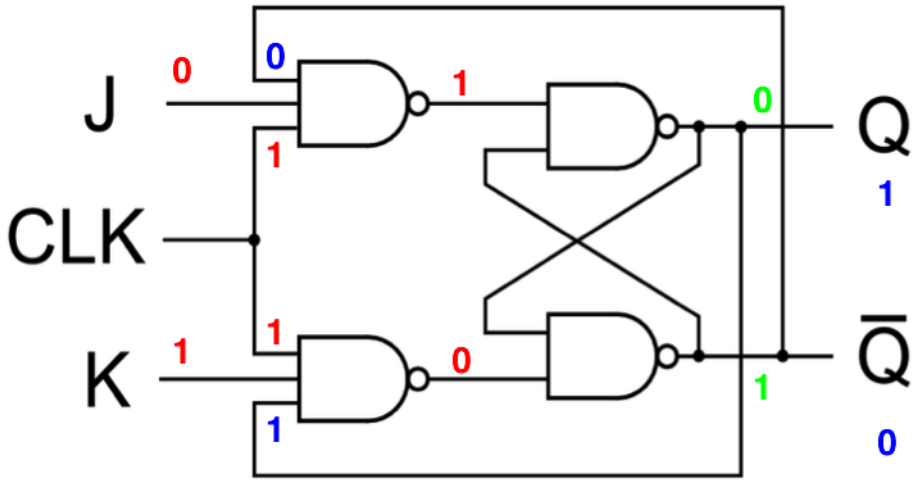




Mode of Operation: Reset

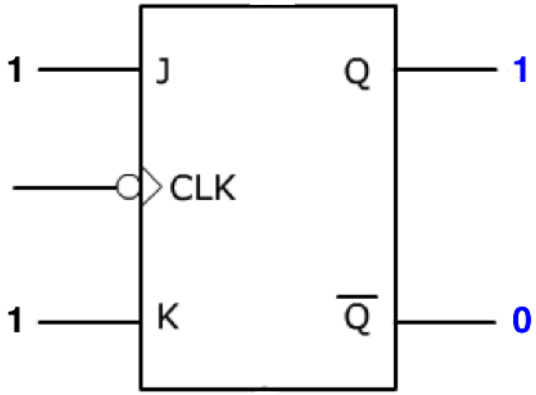
- **Reset**: Q = 0.

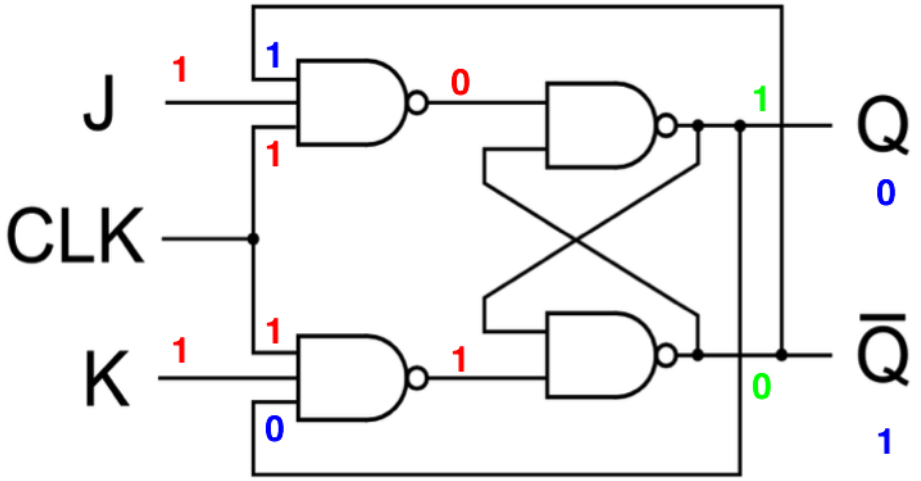




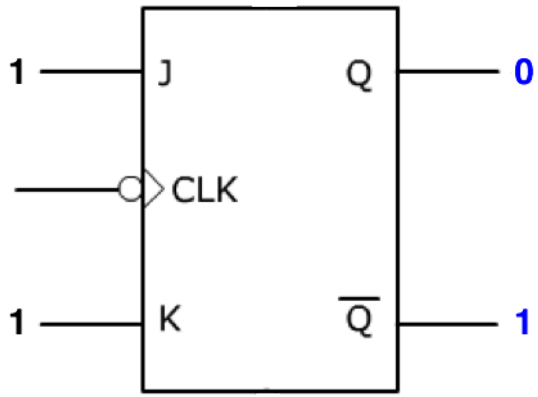
Mode of Operation: Toggle

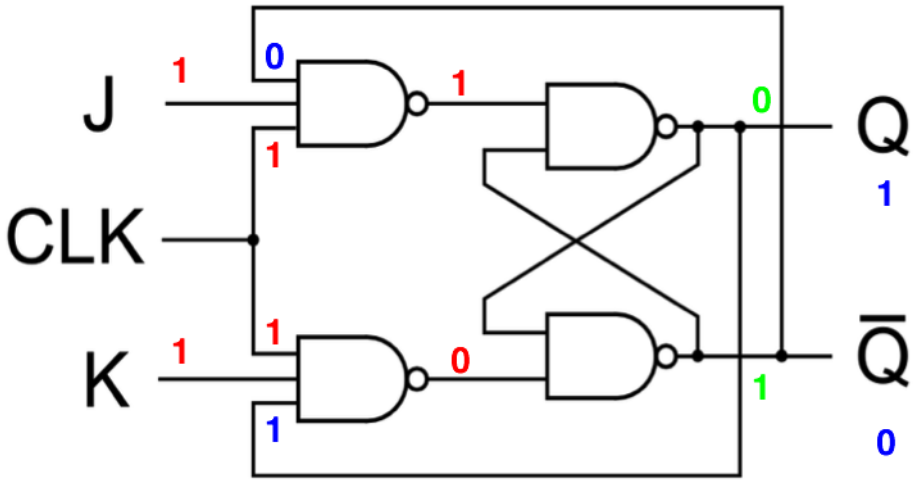
- **Toggle**: Q = Q'.





- **Toggle**: Q = Q’.

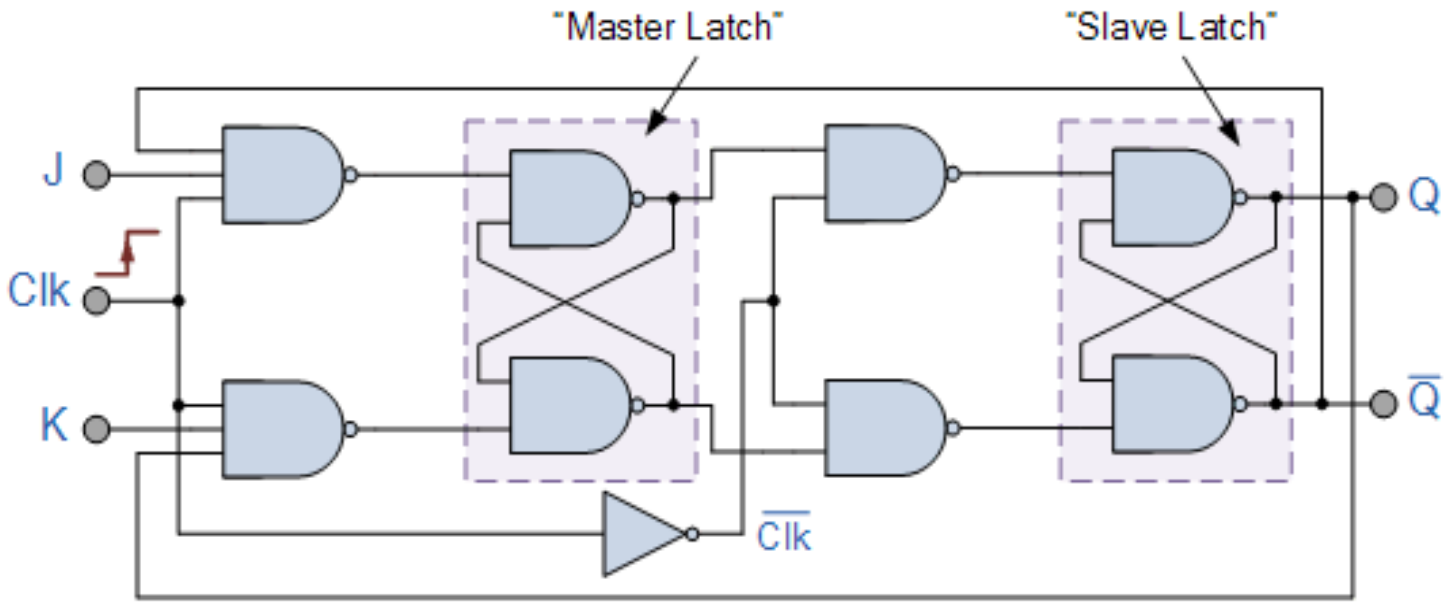




Race Condition

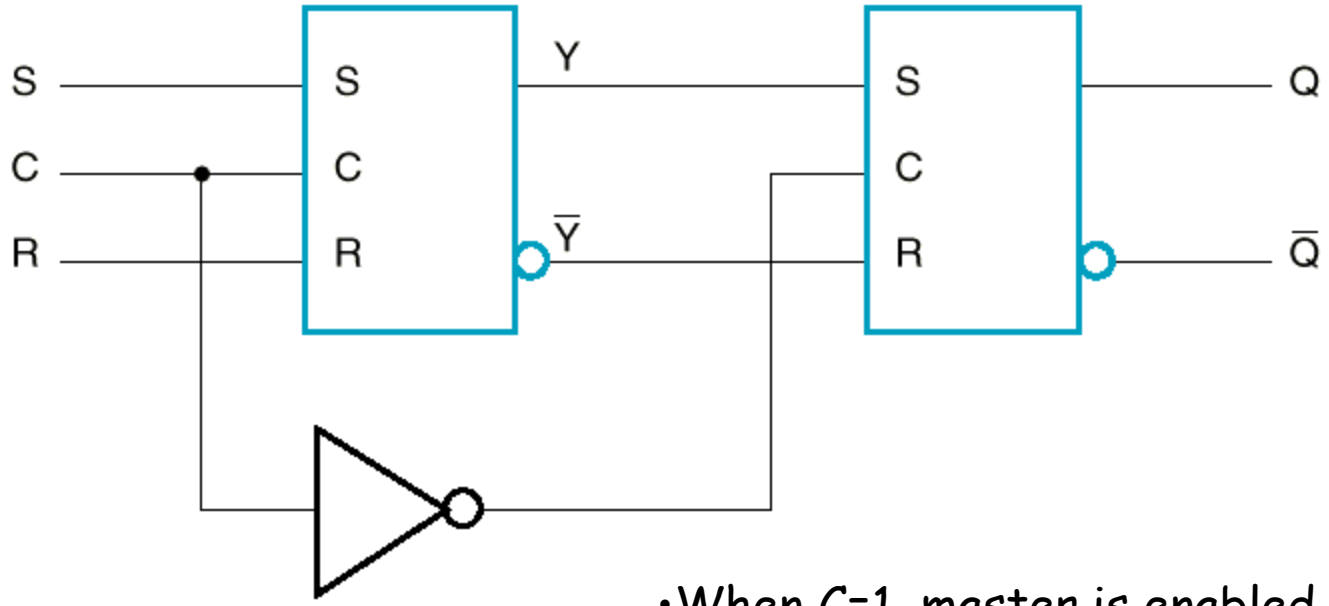
- When both the J and the K inputs are at logic level “1” at the same time, and the clock input is pulsed “HIGH”, the circuit will “toggle” from its SET state to a RESET state, or visa-versa.
- Solution: Master/Slave Flip flop

Master-Slave JK Flip-flop



As the clock input of the “slave” flip flop is the inverse (complement) of the “master” clock input, the “slave” SR flip flop does not toggle.

Master-Slave FF configuration using SR



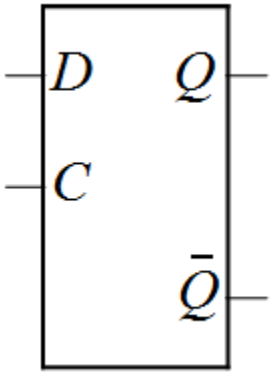
S	R	CLK	Q	Q'	
0	0	1	Q_0	Q_0'	Store
0	1	1	0	1	Reset
1	0	1	1	0	Set
1	1	1	1	1	Disallowed
X	X	0	Q_0	Q_0'	Store

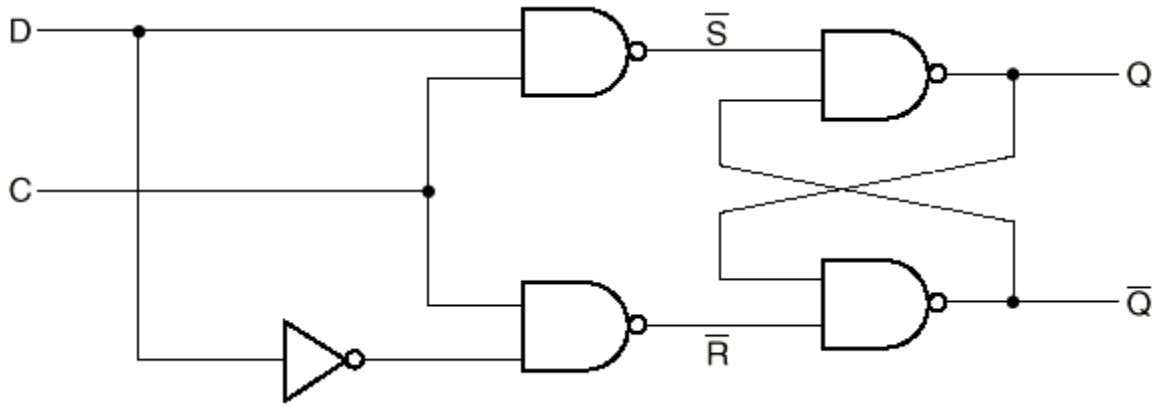
- When *C*=1, master is enabled and stores *new* data, slave stores *old* data.
- When *C*=0, master's state passes to enabled slave, master not sensitive to new data (disabled).

D Flip-Flop

- Forms basic storage element in computers
- One way to eliminate the undesirable indeterminate state in the RS flip flop is to ensure that inputs S and R are never 1 simultaneously. This is done in the *D flip flop*

- D flip flop is simple to understand conceptually
  - When C = 1, data input D stored in latch and output as Q
  - When C = 0, data input D ignored and previous latch value output at Q



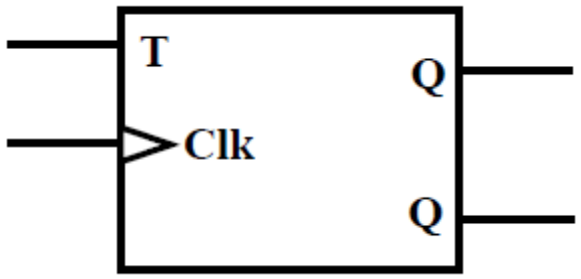


C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

- Also known as Data/Delay flip flop
- **The most widely used flip-flops**: simple to build and design with.
- A **register** comprises several *D* flip-flops, one for each bit to be stored.

Toggle (T) Flip-Flop

- The T flip-flop is a single input version of the JK flip-flop.
- T Flip flop is obtained from the JK type if both inputs are tied together.
- The output of the T flip-flop "toggles" with each clock pulse.



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

- The T input doesn’t specify a value for its output, it specifies only whether or not the output should be changed
- If T = 0 then the output of the flip-flop is unchanged; if T=1, the output is inverted.

Flip Flop Excitation Table

- Characteristic table
  - defines the next state of the flip-flop in terms of flip-flop inputs and current state
  - Used in analysis
- Excitation table
- defines the flip-flop input variable values as function of the current state and next state
- Used in design

S-R Flip-Flop

Characteristic Table

S	R	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined

Excitation Table

Q(t)	Q(t+1)	S	R	Operation
0	0	0	X	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change

J-K Flip-Flop

Characteristic Table

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

Excitation Table

Q(t)	Q(t+1)	J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change

D Flip-Flop

Characteristic Table

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

Excitation Table

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

T Flip-Flop

Characteristic Table

T	Q(t+1)	Operation
0	$Q(t)$	No change
1	$\bar{Q}(t)$	Complement

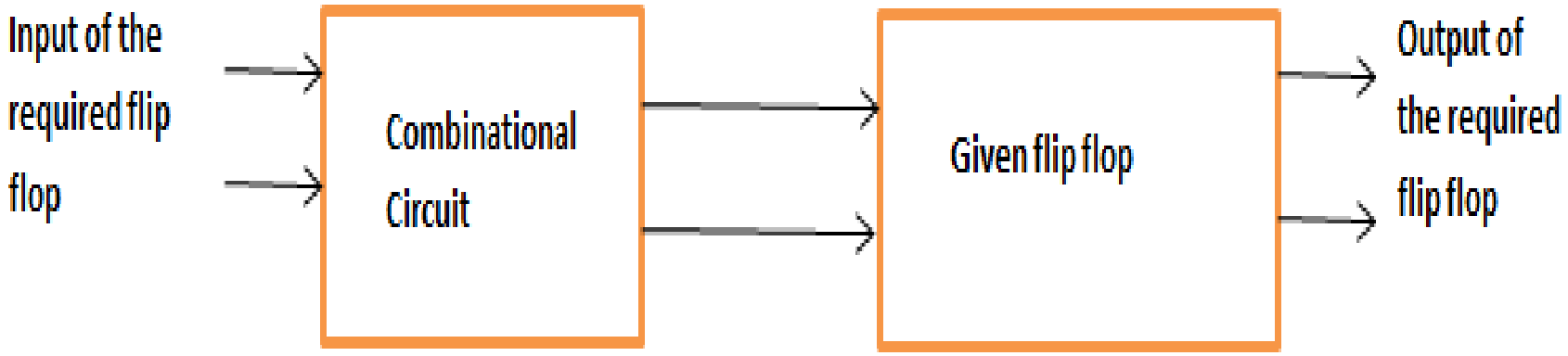
Excitation Table

Q(t+1)	T	Operation
$Q(t)$	0	No change
$\bar{Q}(t)$	1	Complement

Flip flop Conversions

Flip flop Conversions

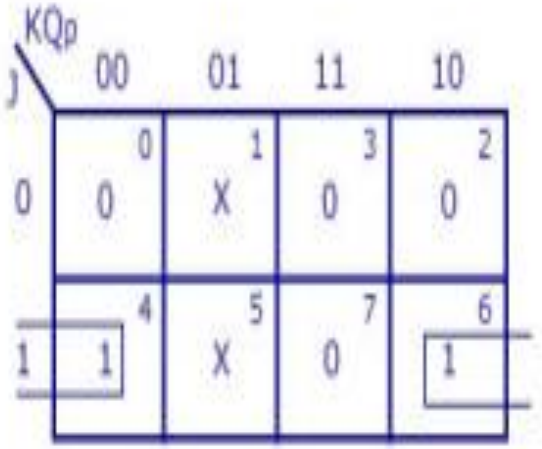
- The purpose is to convert a given type FF to a required type FF using some conversion logic.



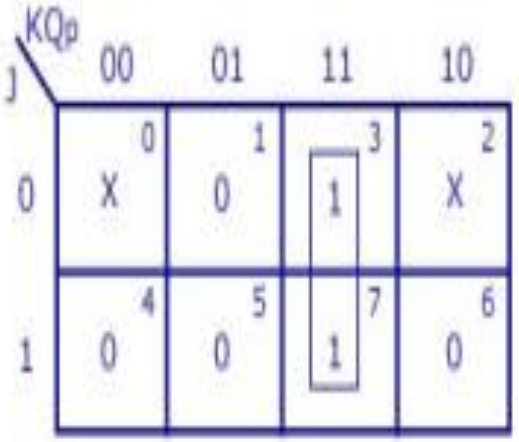
SR Flip flop to JK Flip flop

Conversion Table

J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1



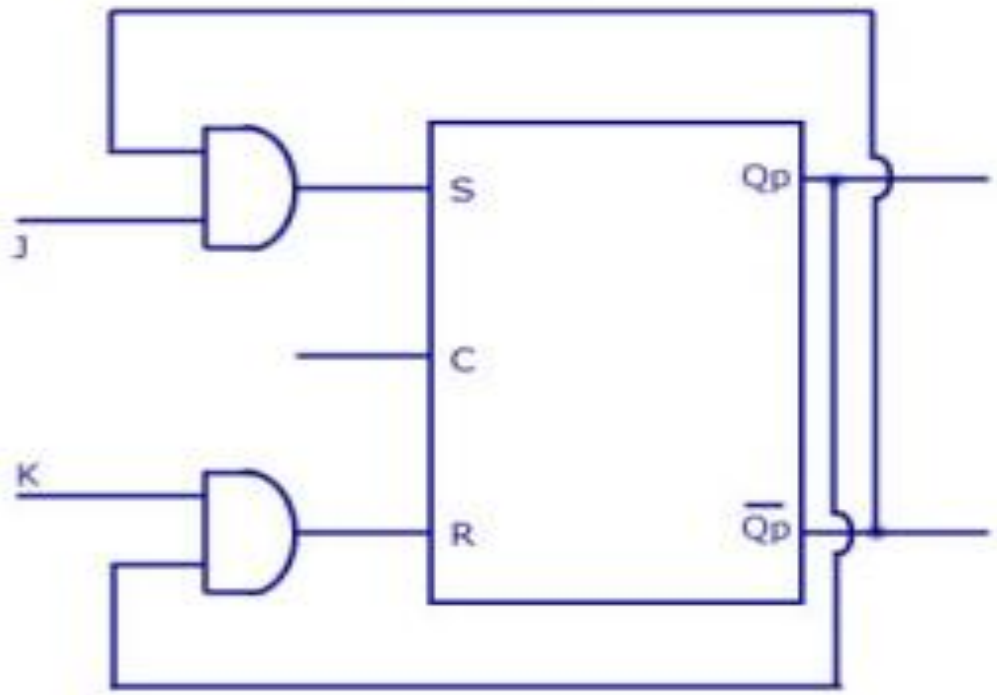
$S = \bar{J}Q_p$



$R = KQ_p$

K-Map

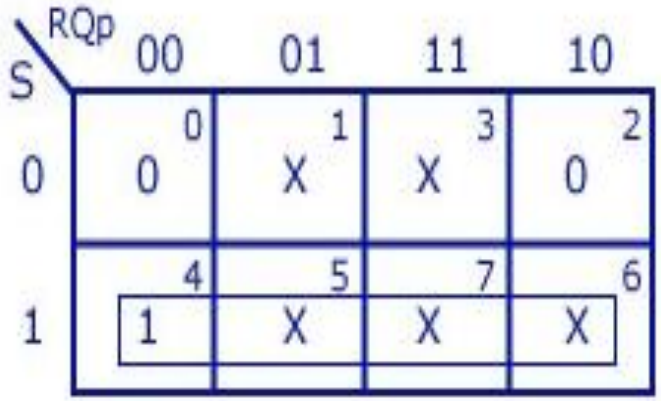
Logic Diagram



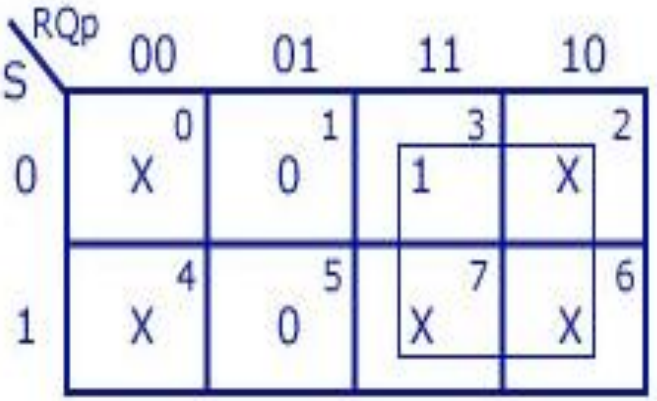
JK Flip Flop to SR Flip Flop

Conversion Table

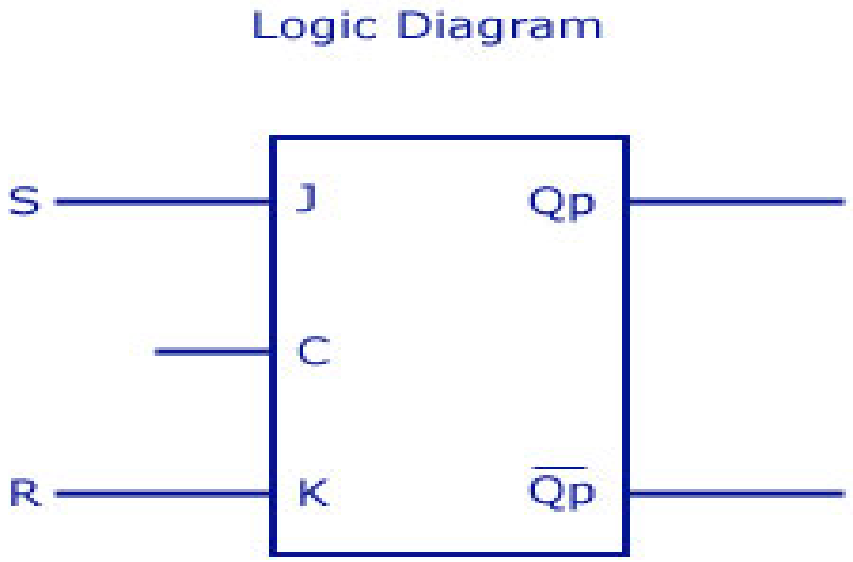
S-R Inputs		Outputs		J-K Inputs	
S	R	Qp	Qp+1	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	Invalid		Dont care	
1	1	Invalid		Dont care	



K-maps



JK Flip Flop to SR Flip Flop

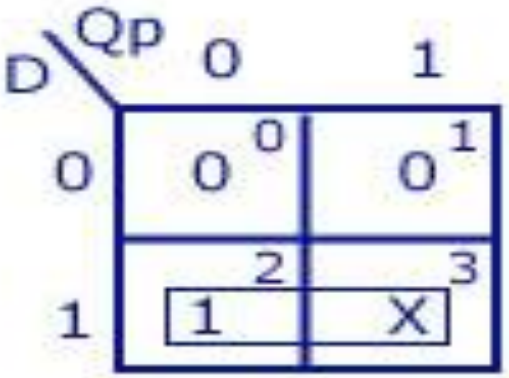


SR Flip Flop to D Flip Flop

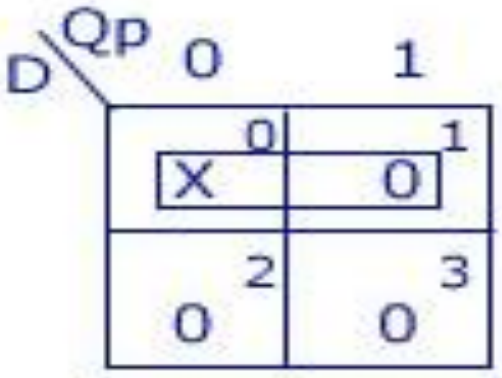
Conversion Table

D Input	Outputs		S-R Inputs	
	Qp	Qp+1	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

K-maps

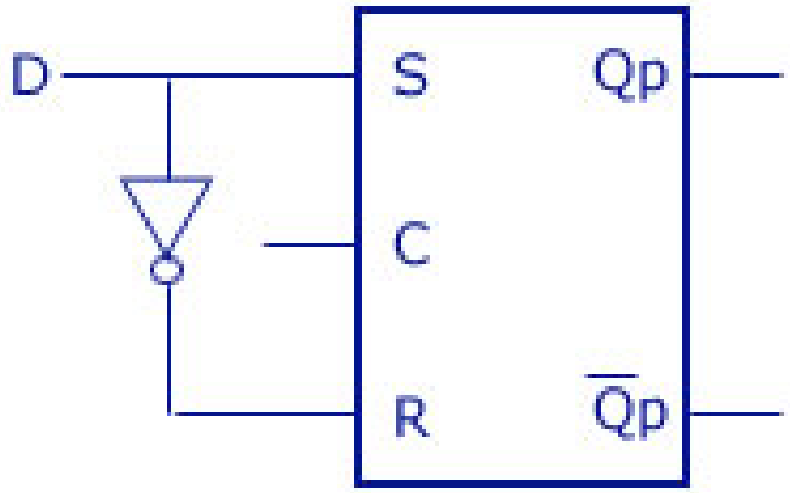


$S = D$



$R = \overline{D}$

Logic Diagram

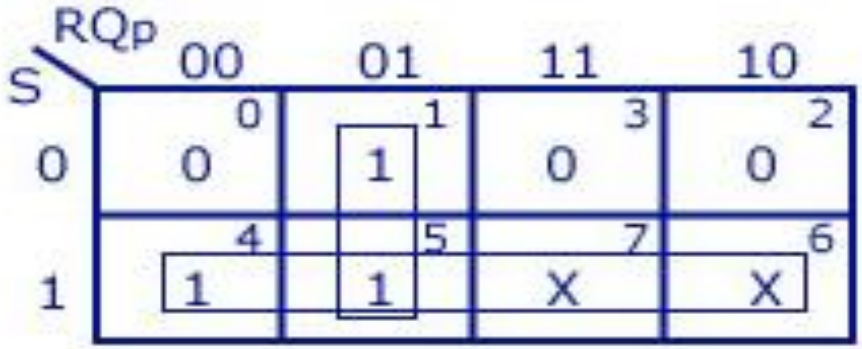


D Flip Flop to SR Flip Flop

Conversion Table

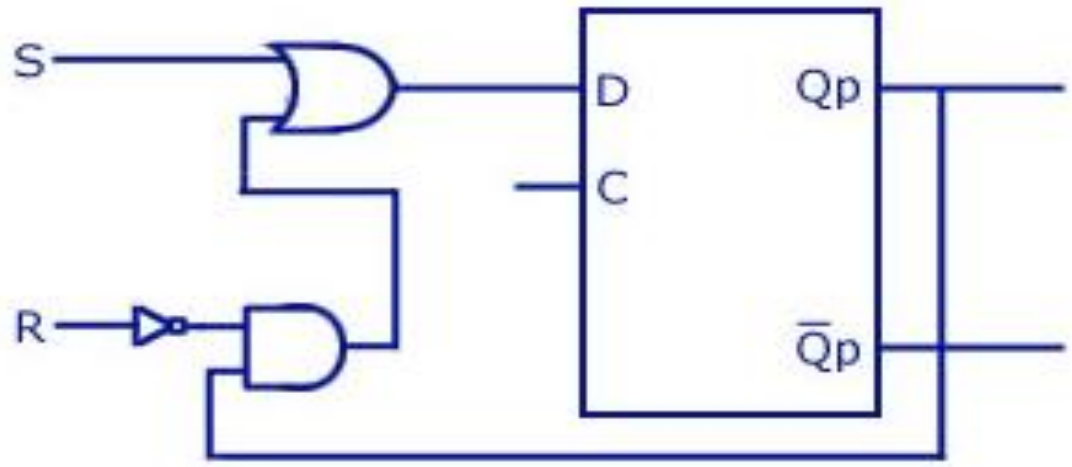
S-R Inputs		Outputs		D Input
S	R	Qp	Qp+1	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	Invalid		Dont care
1	1	Invalid		Dont care

K-map



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

Logic Diagram

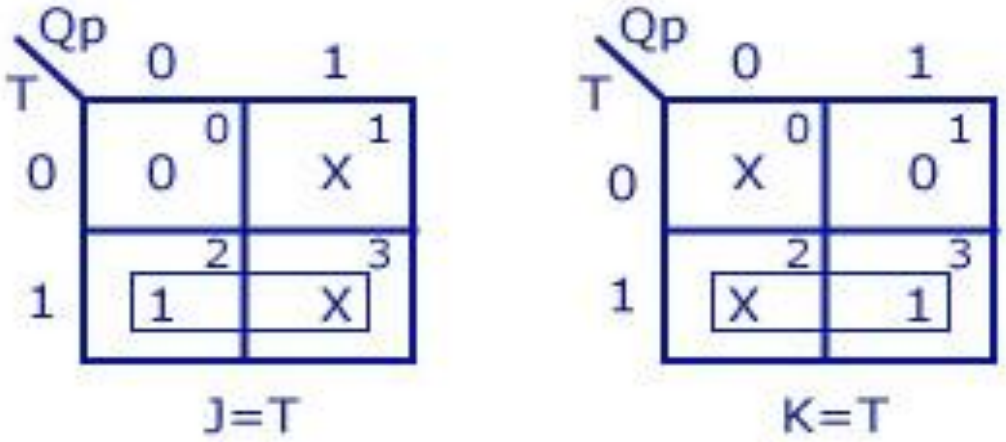


JK Flip Flop to T Flip Flop

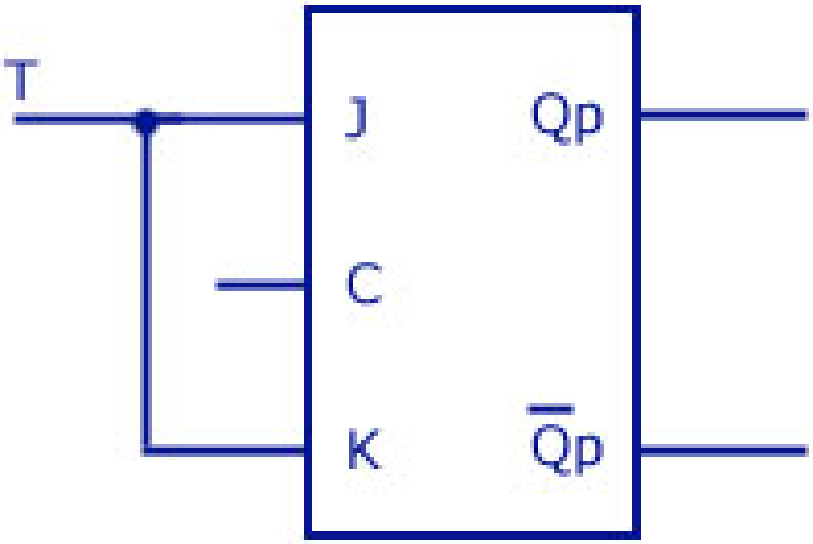
Conversion Table

T Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps



Logic Diagram

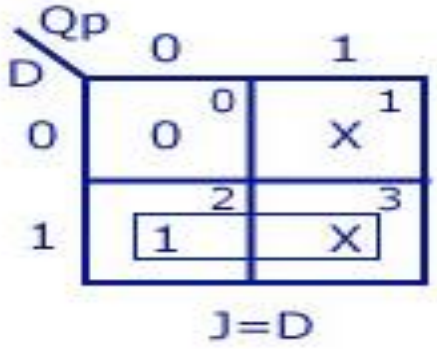


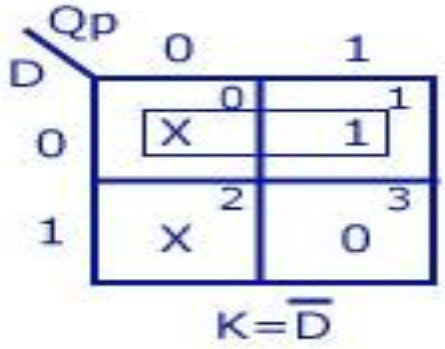
JK Flip Flop to D Flip Flop

Conversion Table

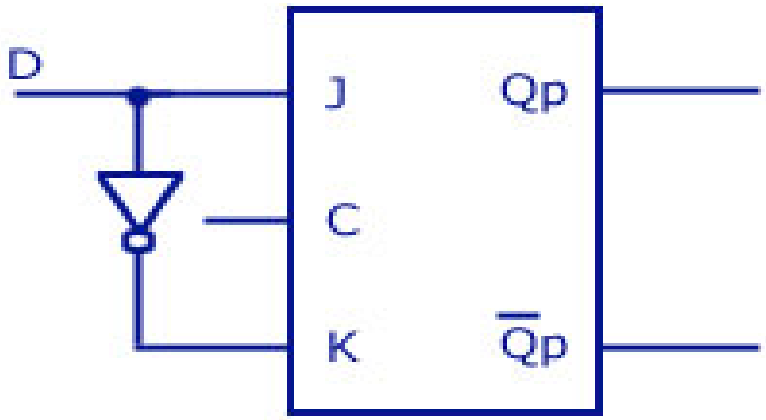
D Input	Outputs		J-K Inputs	
	Qp	Qp+1	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	0	X	0

K-maps





Logic Diagram

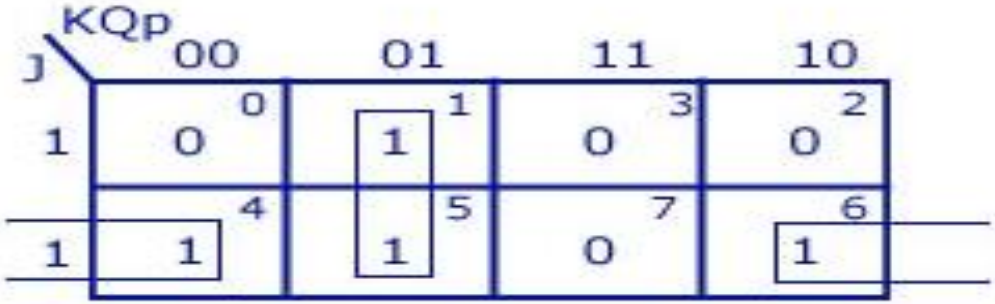


D Flip Flop to JK Flip Flop

Conversion Table

J-K Input		Outputs		D Input
J	K	Qp	Qp+1	
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

K-map



$$D = J\overline{Q}p + \overline{K}Qp$$

Logic Diagram