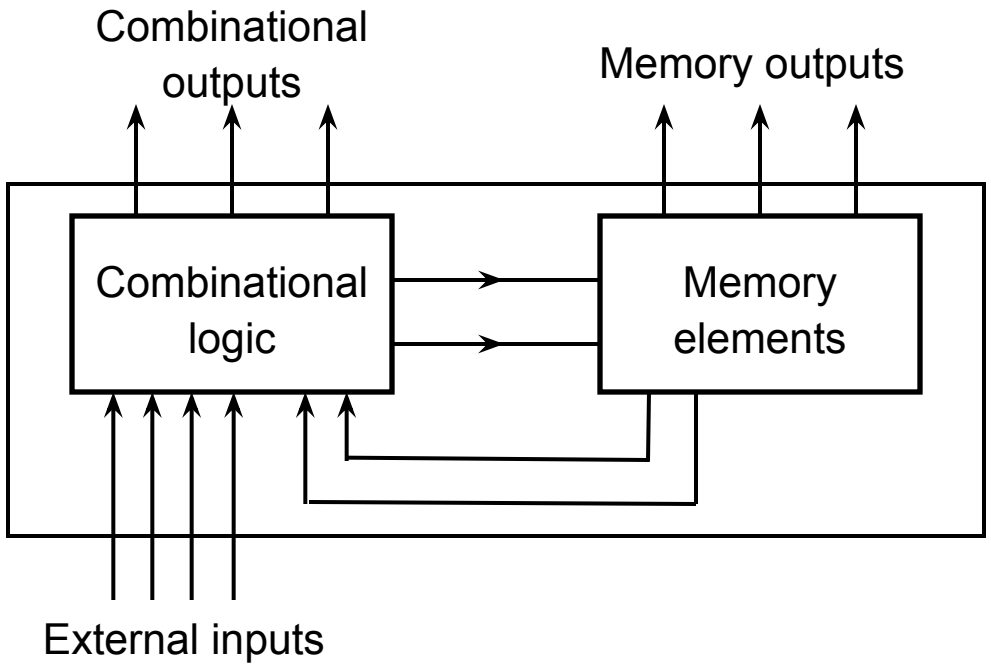
DIGITAL LOGIC DESIGN

Sequential Logic, RS Flip-Flop,
D Flip-Flop, JK Flip-Flop, T Flip-Flop

Introduction

- A **sequential circuit** consists of a *feedback path*, and employs some *memory elements*.



Sequential circuit = Combinational logic + Memory Elements

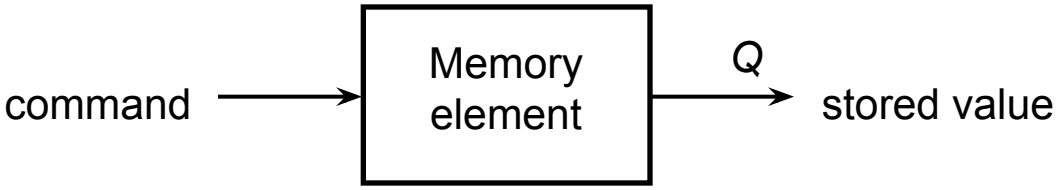
output= external input + present state of memory element

Introduction

- There are two types of sequential circuits:
  - *synchronous*: outputs change only at specific time (i.e. with clock input)
  - *asynchronous*: outputs change at any time (i.e. without clock input)
- *Multivibrator*: a class of sequential circuits. They can be:
  - *bistable* (2 stable states)
  - *monostable* or *one-shot* (1 stable state)
  - *astable* (no stable state)
- Bistable logic devices: *flip-flops*.
- Flip-flops differ in the method used for changing their state.

Memory Elements

- **Memory element**: a device which can remember value indefinitely, or change value on command from its inputs.



Q(t): current state

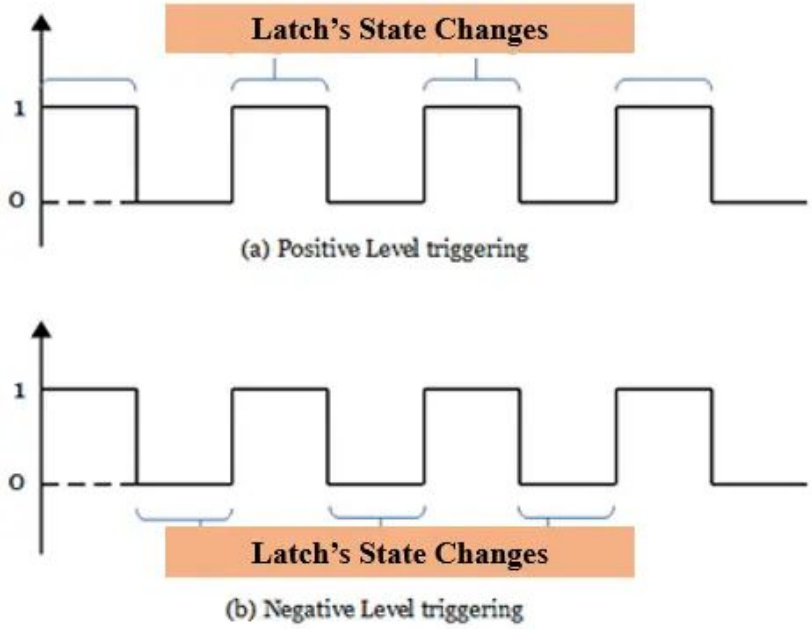
Q(t+1) or Q^+: next state

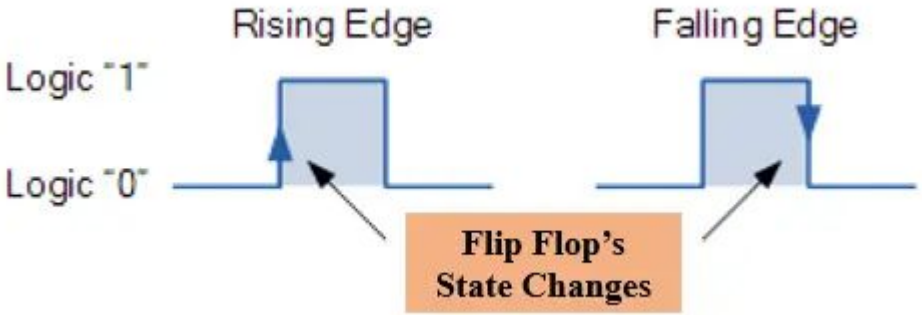
Types of tables in sequential circuit

- Characteristic table
- Truth/Criteria Table

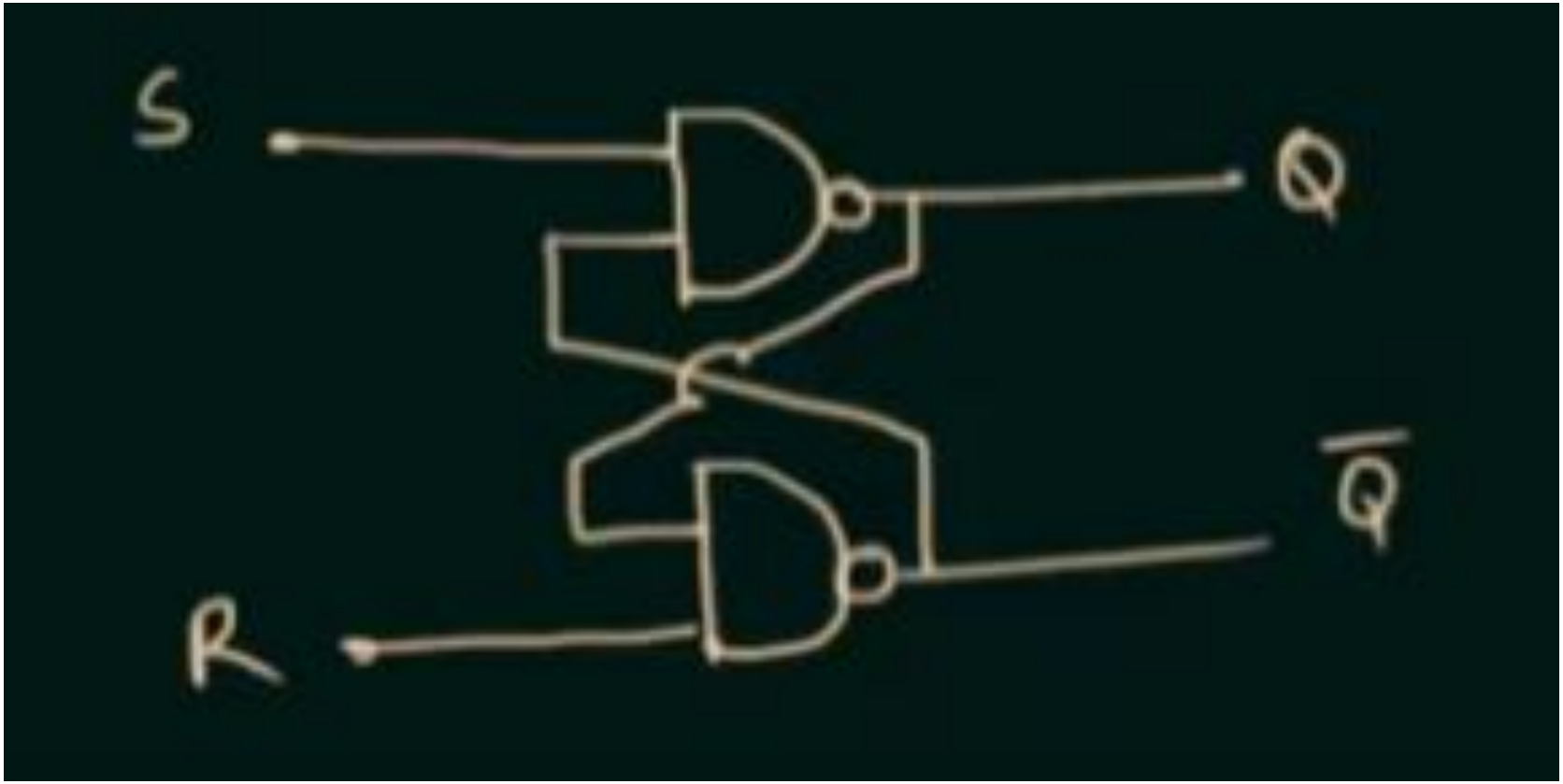
Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

J	K	Q(t+1)	Comments
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q(t)'	Toggle


Latch's State Changes
1
0
(a) Positive Level triggering
1
0
Latch's State Changes
(b) Negative Level triggering


Rising Edge
Falling Edge
Logic "1"
Logic "0"
Flip Flop's
State Changes

SR Latch Using NAND Gate



2 - input NAND gate

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

We will need the knowledge of this truth table later

Case 1:

S=0, R=1, Q=1, Q'=0 [Calculation of Q done first]

S=1, R=1, Q=1, Q'=0

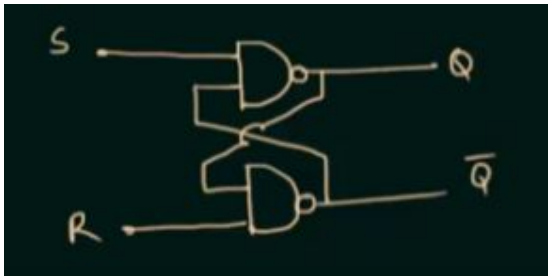
Case 2:

S=1, R=0, Q=0, Q'=1 [Calculation of Q' done first]

S=1, R=1, Q=0, Q'=1

Case 3:

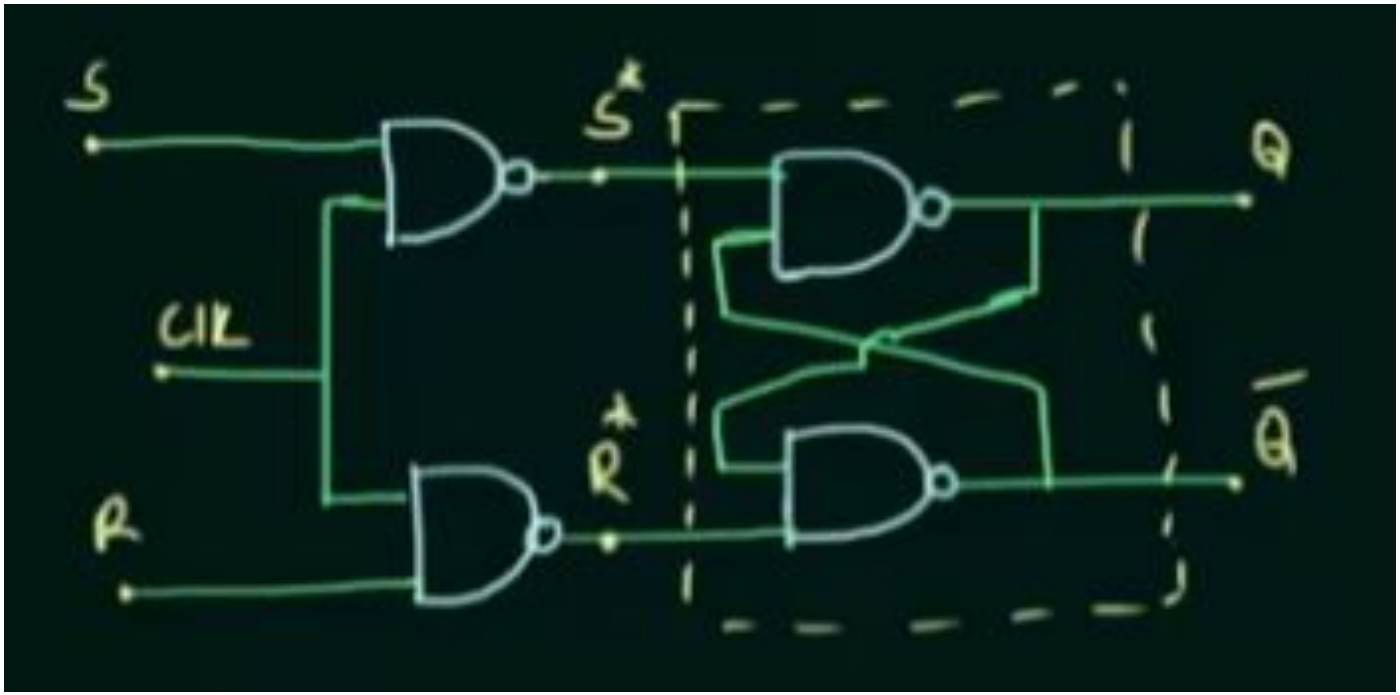
S=0, R=0, Q=1, Q'=1 (Impossible)



Truth Table for SR Latch (NAND Gate)

S	R	Q	Q'
0	0	Not Used	
0	1	1	0
1	0	0	1
1	1	Memory/ No change	

S-R Flip-Flop



Now we will use the knowledge of SR latch to build the truth table for SR FF. But before that, we need to do some calculation.

S-R Flip-Flop (Truth Table)

S* = (S.CLK)'=S'+CLK'

R* = (R.CLK)'=R'+CLK'

Now, for a FF to work, CLK must be 1.

If CLK=1, then,

S*= S'+0 = S'

R*=R'+0=R'

If,

S=0, R=0, then S*=1, R*=1

S=0, R=1, then S*=1, R*=0

S=1, R=0, then S*=0, R*=1

S=1, R=1, then S*=0, R*=0

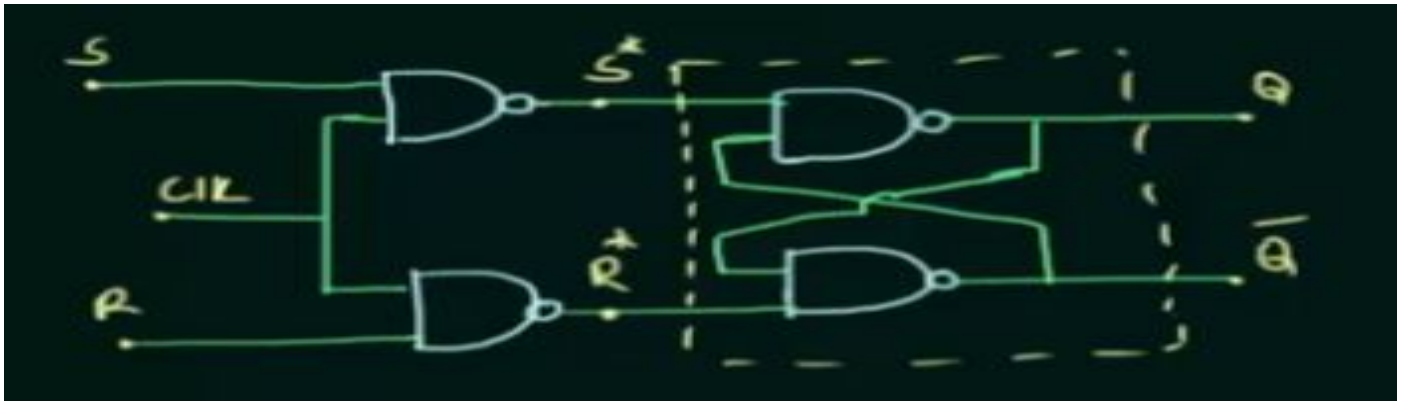
Let's build the truth table now [this table also includes latch inputs which are S* and R*. In the final truth table, you can ignore these two columns as shown in the right side.

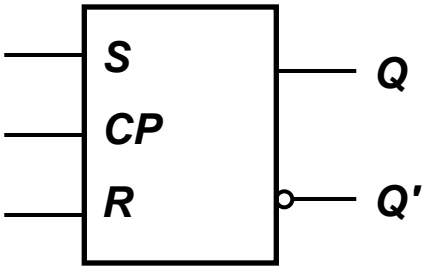
S	R	S*	R*	Q	Q'
0	0	1	1	Memory/ No change	
0	1	1	0	0	1
1	0	0	1	1	0
1	1	0	0	Not Used	

S	R	Q	Q'
0	0	Memory/ No change	
0	1	0	1
1	0	1	0
1	1	Not Used	

Clocked S-R FF Diagram

- S-R FF + Clock Pulse (CP) and 2 NAND gates → Clocked *S-R FF*.





Block Diagram

S-R Flip-Flop Truth + Characteristics Table

S	R	Q	Q'
0	0	Memory/ No change	
0	1	0	1
1	0	1	0
1	1	Not Used	

While building the Characteristics table, always consider Q from the truth table as Q' is just the complement of Q.

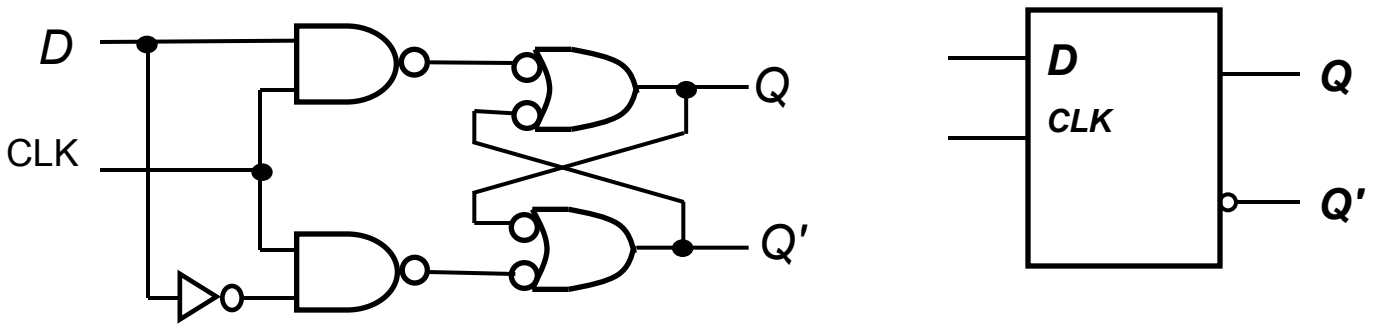
You always need to remember the truth table to build the characteristics table. Meaning, truth table comes first, characteristics table comes second. You can't build characteristics table before truth table

Characteristics Table

Q(t)	*S*	*R*	*Q(t+1)*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

Clocked D Flip-Flop

- Make *R* input equal to *S′* → *D FF.*
- *D* FF eliminates the undesirable condition of invalid state in the *S*-*R* FF.



D Flip-flop Truth table

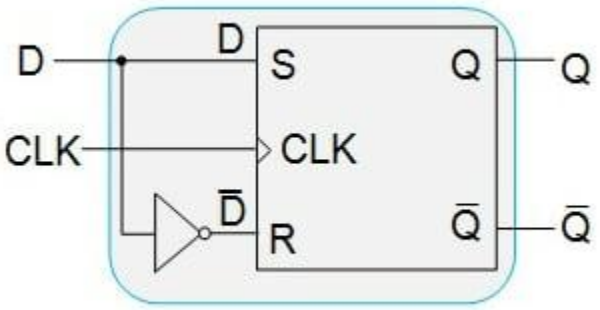
D	Q	Q’
0	0	1
1	1	0

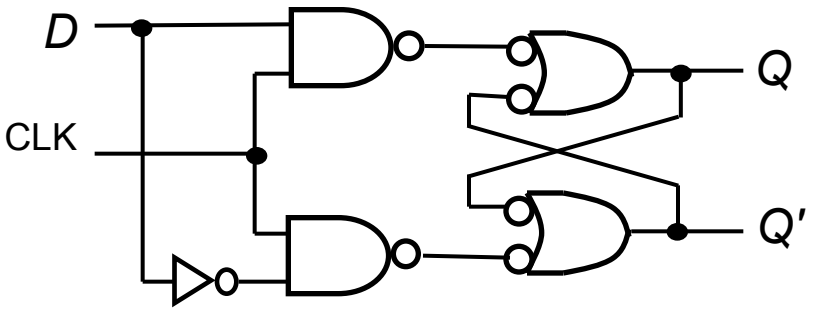
D Flip-flop Characteristic table

Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Try it yourself (Don’t Ignore)

- Design a D FF using RS FF (Learn it yourself) (Draw both circuit and block diagram)

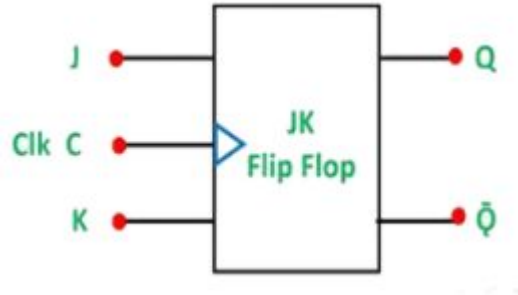




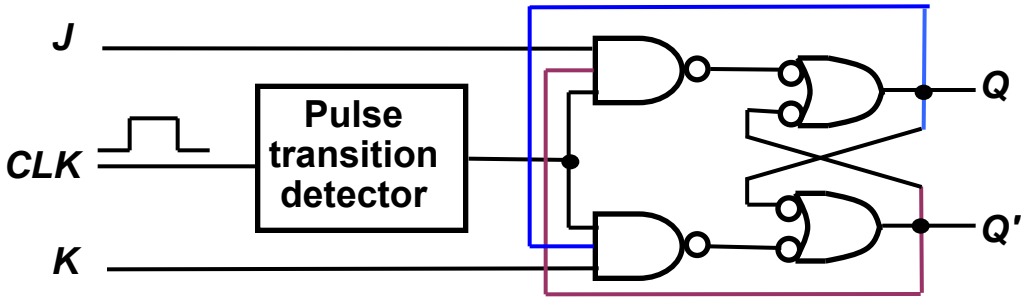
J-K Flip-flop

- J-K flip-flop: Q and Q' are fed back to the NAND gates.
- No invalid state.
- Include a *toggle* state.
  - *J*=HIGH (and *K*=LOW) SET state
  - *K*=HIGH (and *J*=LOW) RESET state
  - both inputs LOW no change
  - both inputs HIGH toggle

J-K Flip-flop



- J-K flip-flop.



- Characteristic table.

Q	J	K	Q(t+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

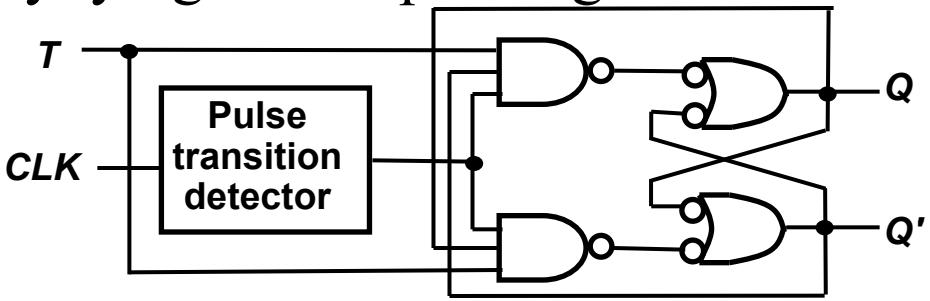
Truth Table

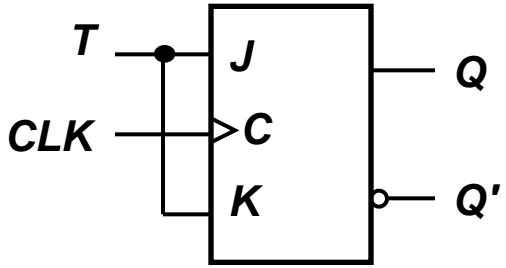
J	K	Q	Q'
0	0	Memory/ No change	
0	1	0	1
1	0	1	0
1	1	Toggle	

This truth table is same as SR except the last row
This is because, J-K FF solves only that "Not Used" problem of SR FF and keeps everything else same as before

T Flip-flop

- T flip-flop: single-input version of the J-K flip flop, formed by tying both inputs together.





Truth Table

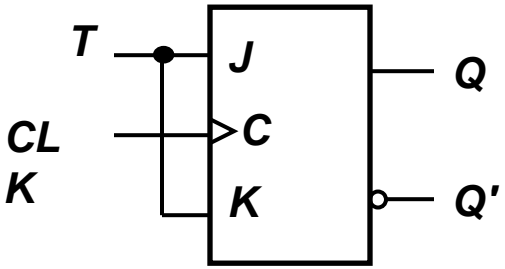
T	Q	Q'
0	No Change	
1	Toggle	

Characteristics Table

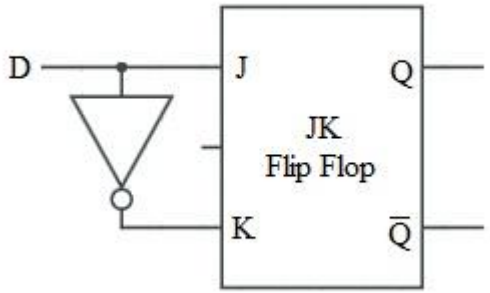
Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Try it yourself (Don’t Ignore)

- Design a T FF using JK FF



- Design a D FF using JK FF



Flip-flop Excitation Tables

- *Excitation tables*: given the required transition from present state to next state, determine the flip-flop input(s).

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR Flip-flop

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-flop

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

T Flip-flop