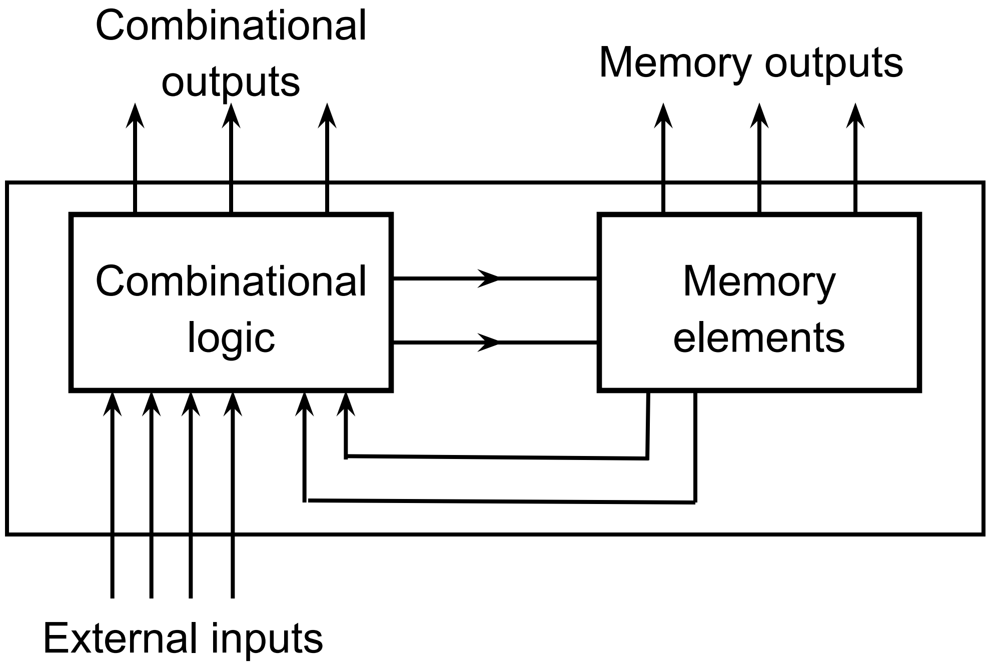
# CSE 260
# DIGITAL LOGIC DESIGN

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

# Introduction

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



Sequential circuit = Combinational logic + Memory Elements

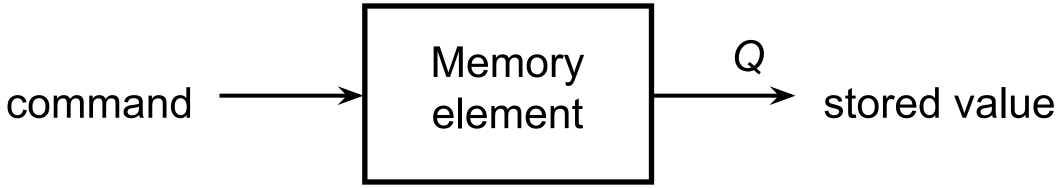
output= external input + present state of memory element

# 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.



- Characteristic table:

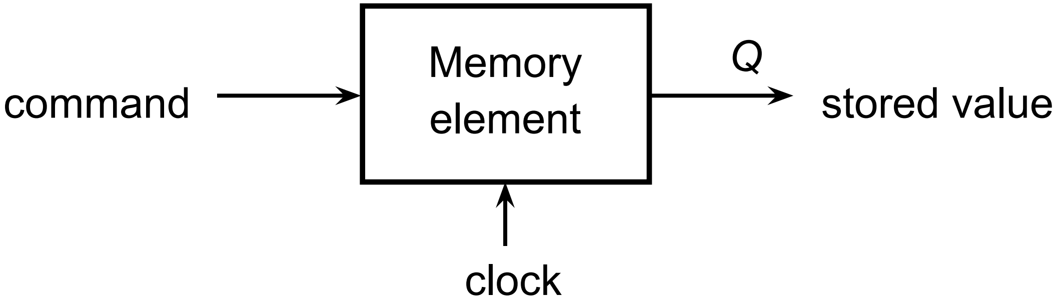
| Command (at time *t*) | *Q(t)* | *Q(t+1)* |
|---|---|---|
| Set | X | 1 |
| Reset | X | 0 |
| Memorise / No Change | 0 | 0 |
| | 1 | 1 |

*Q(t)*: current state

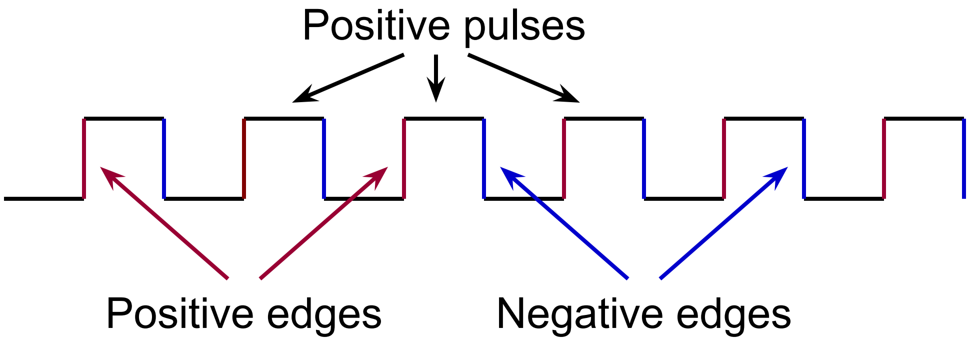
*Q(t+1)* or $Q^+$: next state

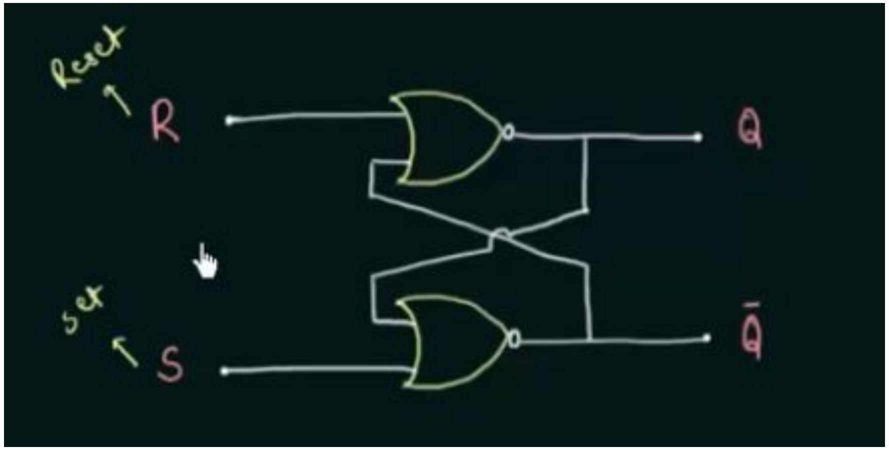
# Memory Elements

- Memory element with clock. Flip-flops are memory elements that change state on clock signals.

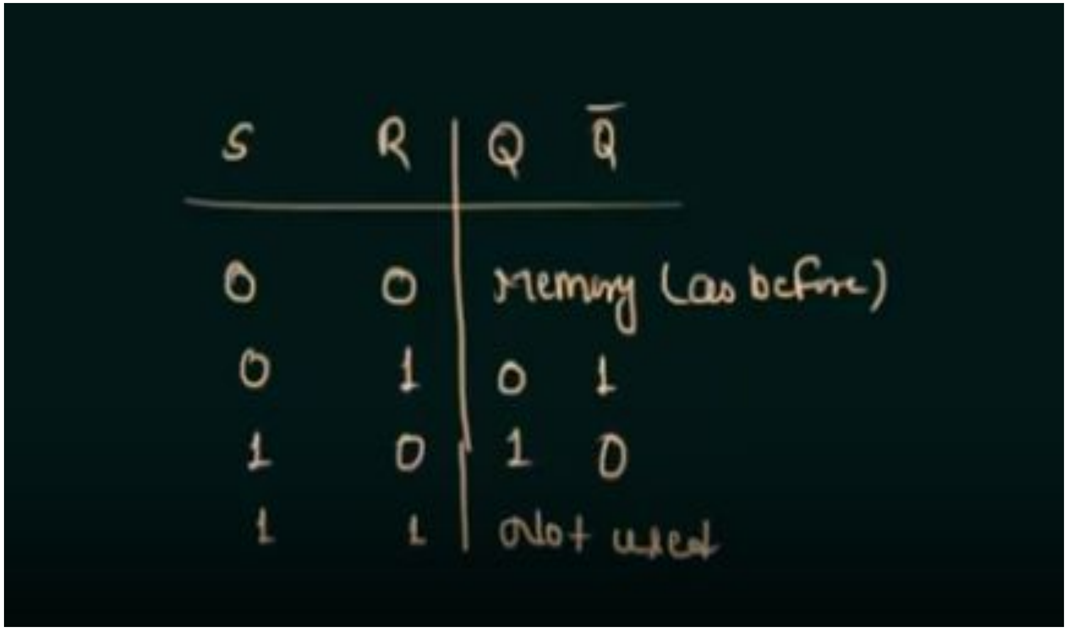


- Clock is usually a square wave.



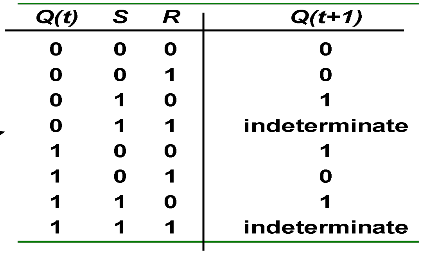


https://www.youtube.com/watch?v=kt8d3CYWGH4



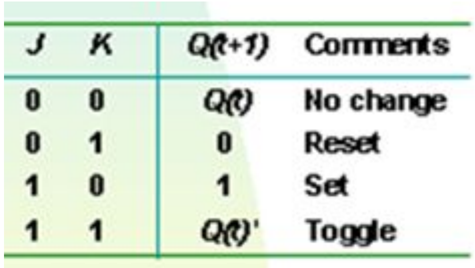
| S | R | Q | $\bar{Q}$ |
|---|---|---|---|
| 0 | 0 | memory (as before) | |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Not used | |

# Types of tables in sequential circuit

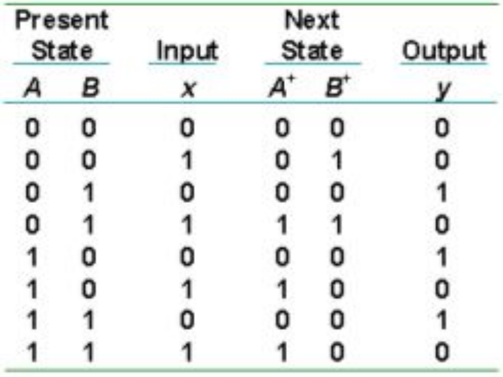


| Q(t) | S | R | Q(t+1) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | indeterminate |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | indeterminate |

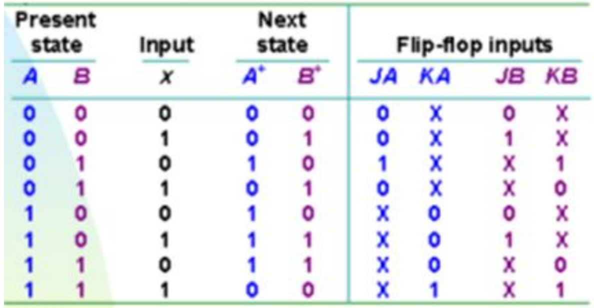
- Characteristic table
- Criteria Table
- State Table
- Excitation table



| 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 |



| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| $A$ | $B$ | $x$ | $A^+$ | $B^+$ | $y$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |



| Present state | | Input | Next state | | Flip-flop inputs | | | |
|---|---|---|---|---|---|---|---|---|
| $A$ | $B$ | $x$ | $A^+$ | $B^+$ | $JA$ | $KA$ | $JB$ | $KB$ |
| 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | X |
| 0 | 0 | 1 | 0 | 1 | 0 | X | 1 | X |
| 0 | 1 | 0 | 1 | 0 | 1 | X | X | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | X | X | 0 |
| 1 | 0 | 0 | 1 | 0 | X | 0 | 0 | X |
| 1 | 0 | 1 | 1 | 1 | X | 0 | 1 | X |
| 1 | 1 | 0 | 1 | 1 | X | 0 | X | 0 |
| 1 | 1 | 1 | 0 | 0 | X | 1 | X | 1 |

# S-R Flip-Flop

- *Complementary* outputs: $Q$ and $Q'$.
- When $Q$ is HIGH, the FF is in *SET* state.
- When $Q$ is LOW, the FF is in *RESET* state.
- For *active-HIGH input* S-R FF (also known as NOR gate FF),

  $R$=HIGH (and $S$=LOW) □ RESET state

  $S$=HIGH (and $R$=LOW) □ SET state

  both inputs LOW □ no change

  both inputs HIGH □ $Q$ and $Q'$ both LOW (invalid)!

# S-R FF

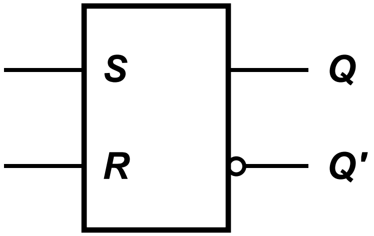
- For *active-LOW input* S'-R' FF (also known as NAND gate FF),
  *R′*=LOW (and *S′*=HIGH) □ RESET state
  *S′*=LOW (and *R′*=HIGH) □ SET state
  both inputs HIGH □ no change
  both inputs LOW □ *Q* and *Q′* both HIGH (invalid)!
- Drawback of S-R FF: invalid condition exists and must be avoided.

# S-R FF

Use this mostly

- Characteristics table for **active-high** input S-R FF:

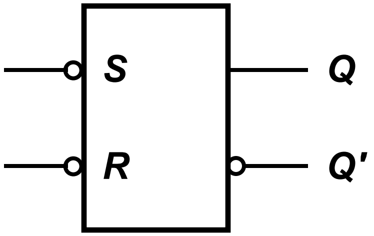
| *S* | *R* | *Q* | *Q'* | |
|---|---|---|---|---|
| 0 | 0 | NC | NC | No change. FF remained in present state. |
| 1 | 0 | 1 | 0 | FF SET. |
| 0 | 1 | 0 | 1 | FF RESET. |
| 1 | 1 | 0 | 0 | Invalid condition. |



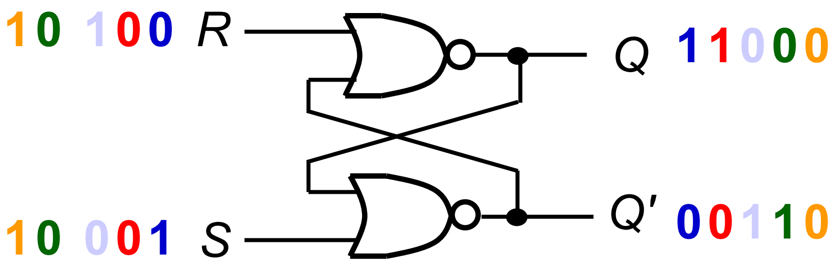
- Characteristics table for **active-low** input S'-R' FF:

| *S'* | *R'* | *Q* | *Q'* | |
|---|---|---|---|---|
| 1 | 1 | NC | NC | No change. FF remained in present state. |
| 0 | 1 | 1 | 0 | FF SET. |
| 1 | 0 | 0 | 1 | FF RESET. |
| 0 | 0 | 1 | 1 | Invalid condition. |

Notice the difference



# S-R FF: Active-HIGH input S-R FF



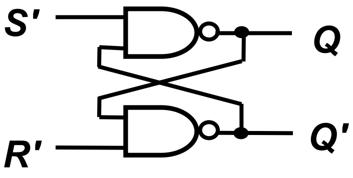
| *S* | *R* | *Q* | *Q'* | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | initial |
| 0 | 0 | 1 | 0 | (afer *S*=1, *R*=0) |
| 0 | 1 | 0 | 1 | |
| 0 | 0 | 0 | 1 | (after *S*=0, *R*=1) |
| 1 | 1 | 0 | 0 | invalid! |

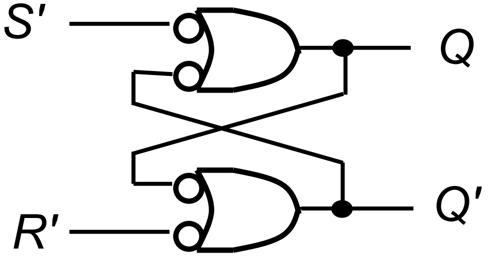
**NOR truth table**

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**Presence of '1' in input, leads to '0' in output**

# S-R FF: Active-LOW input
# S’-R’ FF





| *S'* | *R'* | *Q* | *Q'* | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | initial |
| 1 | 1 | 0 | 1 | (afer *S'*=1, *R'*=0) |
| 0 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 0 | (after *S'*=0, *R'*=1) |
| 0 | 0 | 1 | 1 | invalid! |

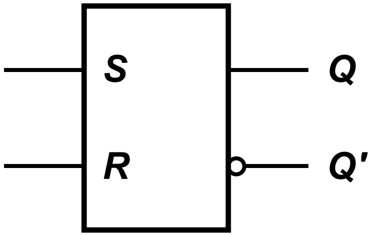
Note: Here output=0 means high!

| NAND truth table | | |
|---|---|---|
| A | B | Output |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Presence of ‘0’ in input, leads to ‘1’ in output**

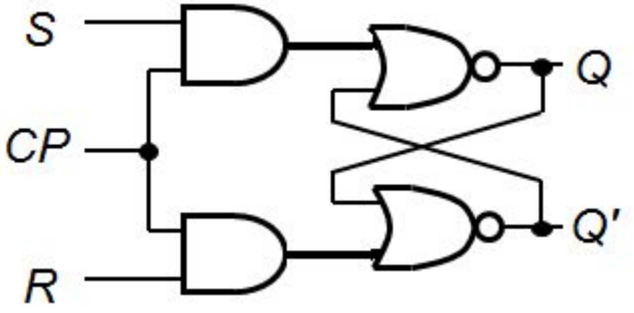
# S-R FF

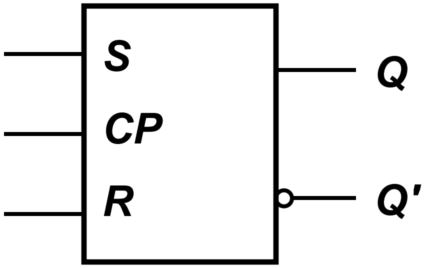
| R | S | Operation | Q(t) | Q(t+1) | Q’(t) | Q’(t+1) |
|---|---|---|---|---|---|---|
| 0 | 0 | No Change | 0 | 0 | 1 | 1 |
| | | | 1 | 1 | 0 | 0 |
| 0 | 1 | Set | x | 1 | x | 0 |
| 1 | 0 | Reset | x | 0 | x | 1 |
| 1 | 1 | Invalid | - | - | - | - |



# Clocked S-R FF

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





# Clocked S-R FF

- Outputs change (if necessary) only when **CP is HIGH**.
- Under what condition does the invalid state occur?
- Characteristic table:

Characteristic Eq$^n$:

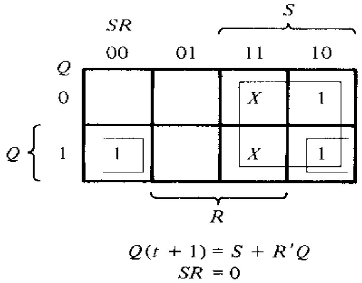
***CP*=1**

| *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 |

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

$S.R = 0$

| *S* | *R* | *Q(t+1)* | |
|---|---|---|---|
| 0 | 0 | *Q(t)* | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | indeterminate | |



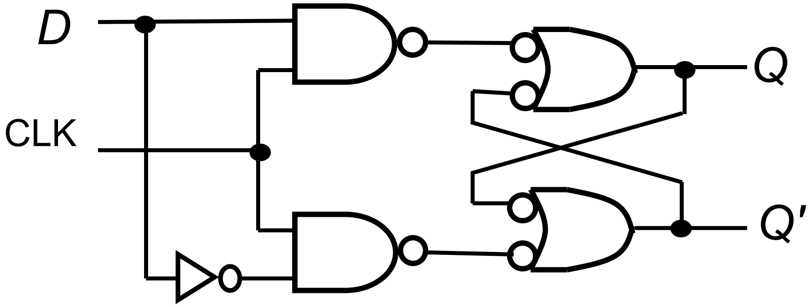
| Q \ SR | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | X | 1 |
| 1 | 1 | | X | 1 |

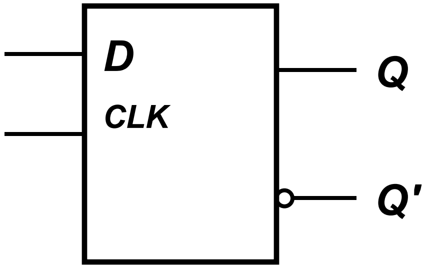
$Q(t + 1) = S + R'Q$

$SR = 0$

# Clocked D Flip-Flop

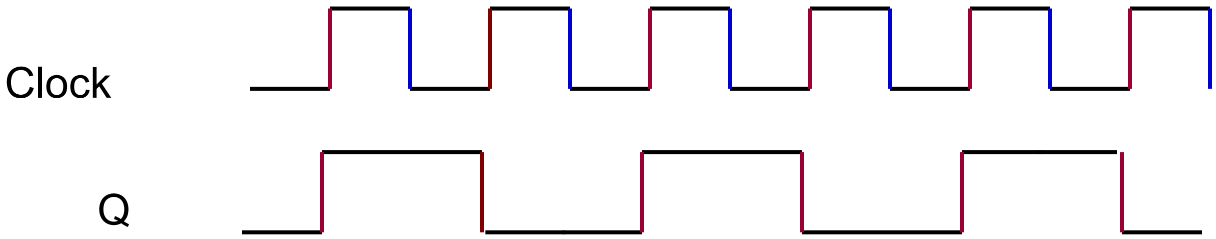
- Make *R* input equal to *S′* → *D FF.*
- *D* FF eliminates the undesirable condition of invalid state in the *S*-*R* FF.





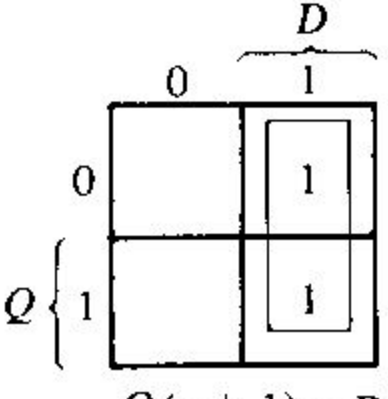
# D Flip-flop Characteristic table

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



# Clocked D Flip-Flop

- When CLK is HIGH,
  - ❖ *D*=HIGH → FF is SET
  - ❖ *D*=LOW → FF is RESET
- Hence when CLK is HIGH, *Q* ‘follows’ the *D* (data) input.
- Characteristic table:



$Q(t+1) = D$

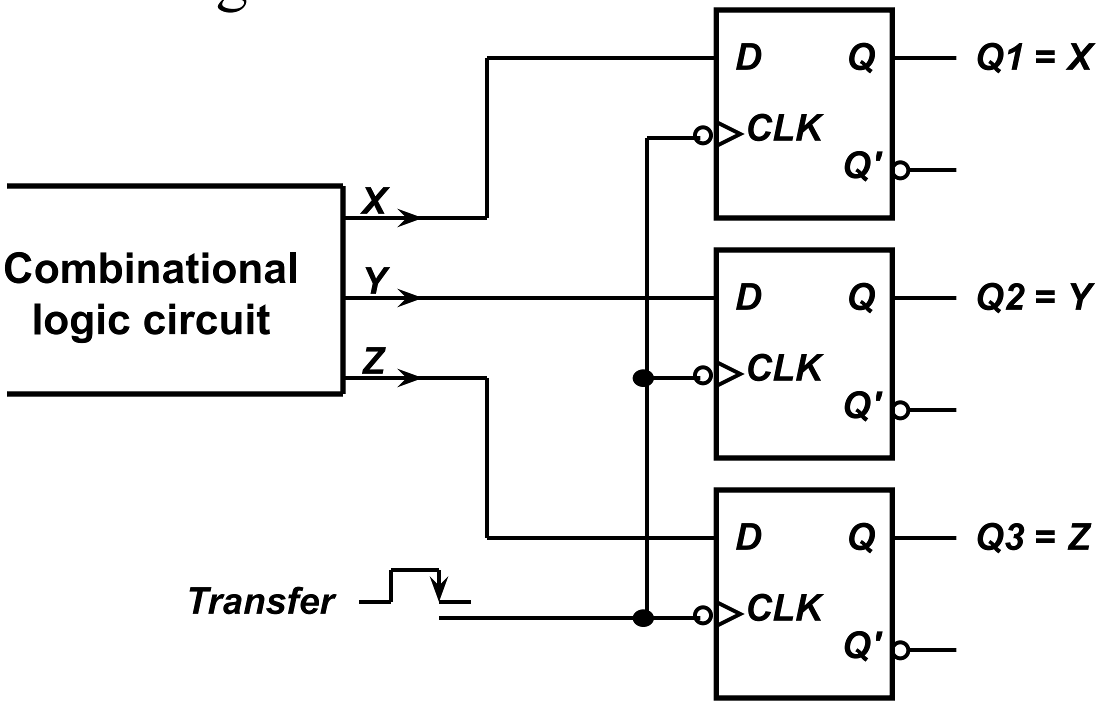
| *CLK* | *D* | *Q(t+1)* | |
|---|---|---|---|
| 1 | 0 | 0 | Reset |
| 1 | 1 | 1 | Set |
| 0 | X | *Q(t)* | No change |

When CLK=1, *Q(t+1) = D*

# D Flip-flop

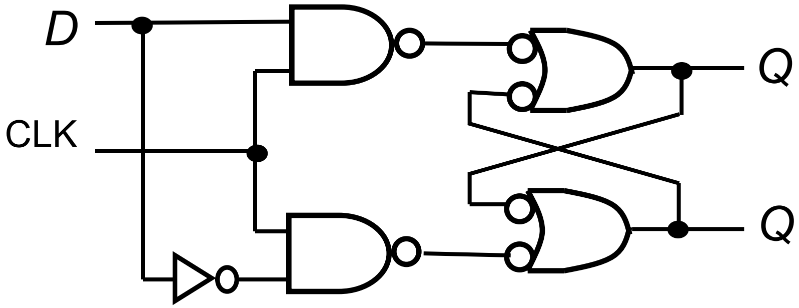
- Application: *Parallel data transfer*.

  To transfer logic-circuit outputs *X*, *Y*, *Z* to flip-flops $Q_1$, $Q_2$ and $Q_3$ for storage.



# Try it yourself

- Design a D FF using RS FF



# 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

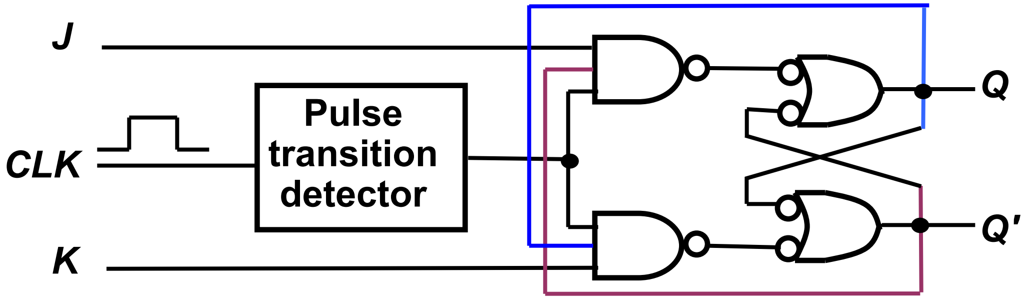
# J-K FF

*SET* *RESET*

| Clock | J | K | Operation | Q(t) | Q(t+1) | Q’(t) | Q’(t+1) |
|---|---|---|---|---|---|---|---|
| ↑ | 0 | 0 | No Change | 0 | 0 | 1 | 1 |
| | | | | 1 | 1 | 0 | 0 |
| ↑ | 1 | 0 | Set | x | 1 | x | 0 |
| ↑ | 0 | 1 | Reset | x | 0 | x | 1 |
| ↑ | 1 | 1 | Toggle | 1 | 0 | 0 | 1 |

# J-K Flip-flop

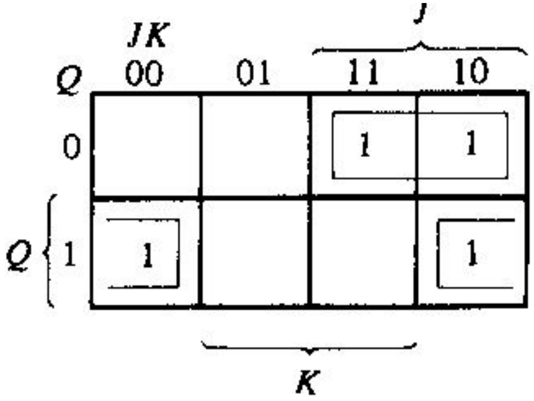
- J-K flip-flop.



| J | K | CLK | Q(t+1) | Comments |
|---|---|---|---|---|
| 0 | 0 | ↑ | *Q(t)* | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | *Q(t)'* | Toggle |

- 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 |

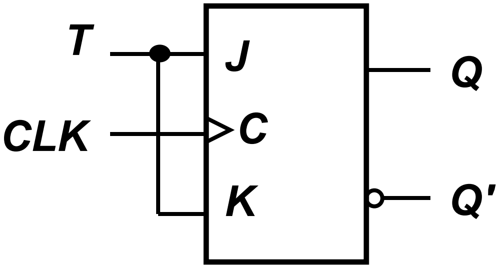


| Q \ JK | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | 1 |
| 1 | 1 | | | 1 |

*Q(t+1) = J.Q' + K'.Q*

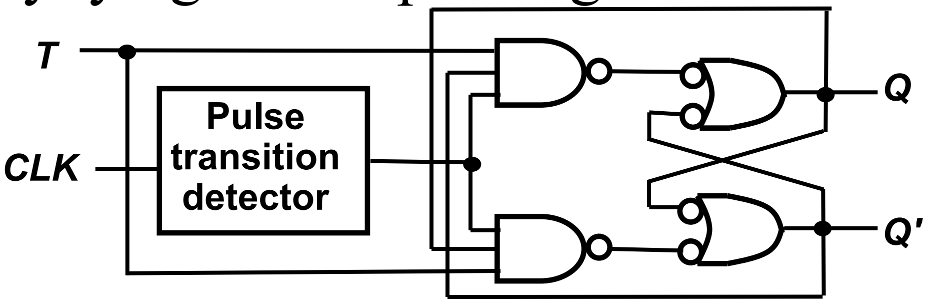
# T Flip-flop

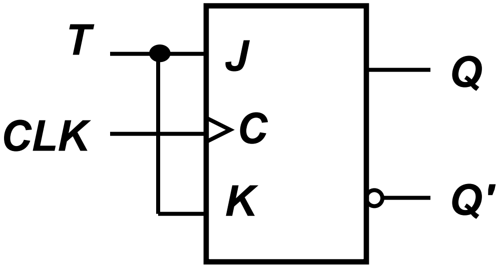
| T | Operation | Q(t) | Q(t+1) |
|---|---|---|---|
| 0 | No change | 0 | 0 |
| | | 1 | 1 |
| 1 | Toggle | 0 | 1 |
| | | 1 | 0 |



# T Flip-flop

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

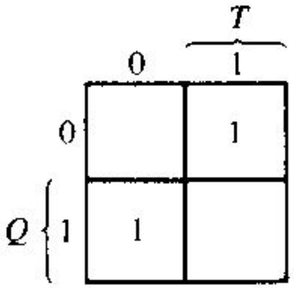




- Characteristic table.

| T | CLK | Q(t+1) | Comments |
|---|---|---|---|
| 0 | ↑ | Q(t) | No change |
| 1 | ↑ | Q(t)' | Toggle |

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

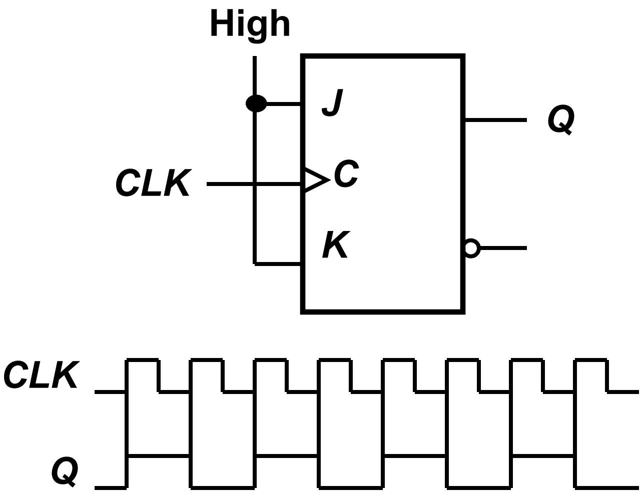


|  |  | T |
|---|---|---|
|  | 0 | 1 |
| 0 |  | 1 |
| Q 1 | 1 |  |

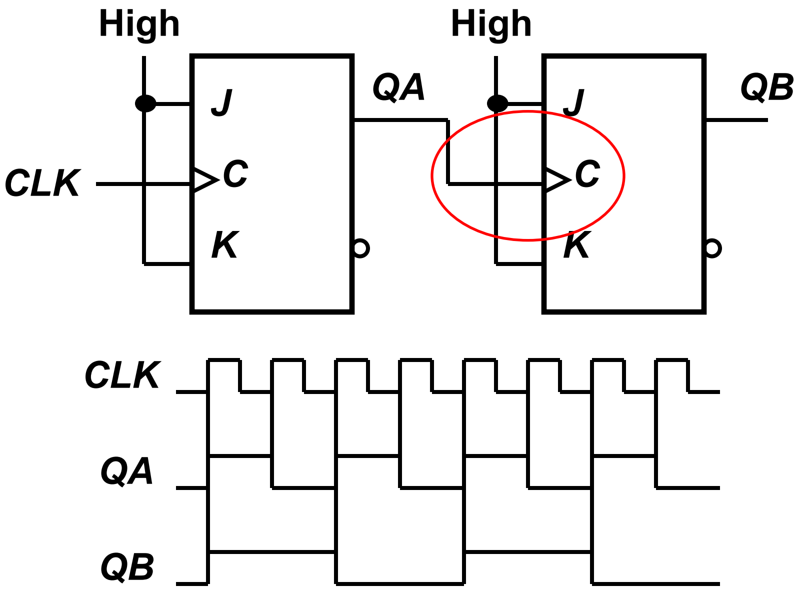
$Q(t+1) = T.Q' + T'.Q$

# T Flip-flop

- Application: *Frequency division*.



Divide clock frequency by 2.

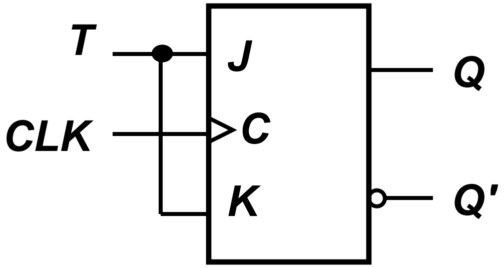


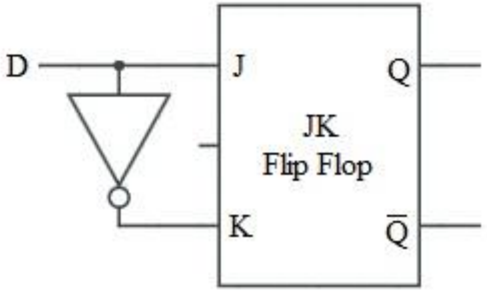
Divide clock frequency by 4.

- Application: *Counter*

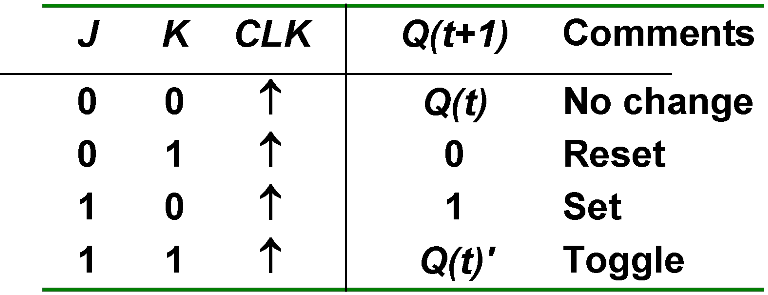
# Try it yourself

- Design a T FF using JK FF
- Design a D FF using JK FF


T
CLK
J
C
K
Q
Q'

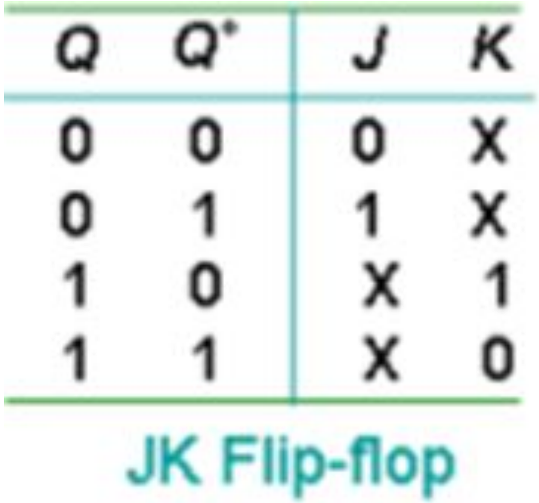

D
J
K
JK
Flip Flop
Q
$\overline{Q}$

# How to build excitation table: Example: JK Flip-Flop



| *J* | *K* | *CLK* | *Q(t+1)* | Comments |
|---|---|---|---|---|
| 0 | 0 | ↑ | *Q(t)* | No change |
| 0 | 1 | ↑ | 0 | Reset |
| 1 | 0 | ↑ | 1 | Set |
| 1 | 1 | ↑ | *Q(t)'* | Toggle |

| Q | Q+ | Actually what happens | | Final Combined Result | |
|---|---|---|---|---|---|
| | | J | K | J | K |
| 0 | 0 | 0 | 1 | 0 | x |
| | | 0 | 0 | | |
| 0 | 1 | 1 | 0 | 1 | x |
| | | 1 | 1 | | |
| 1 | 0 | 0 | 1 | x | 1 |
| | | 1 | 1 | | |
| 1 | 1 | 0 | 0 | x | 0 |
| | | 1 | 0 | | |

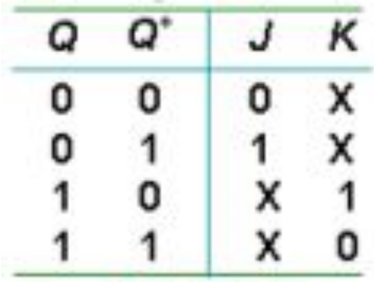


| Q | Q' | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

JK Flip-flop

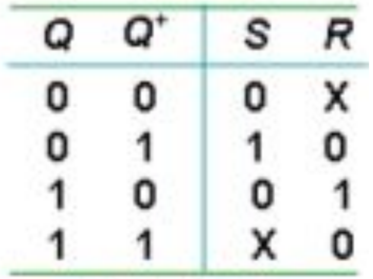
# Flip-flop Excitation Tables

- Excitation tables : it give transition characteristic between current condition and next condition  to determine flip-flop input



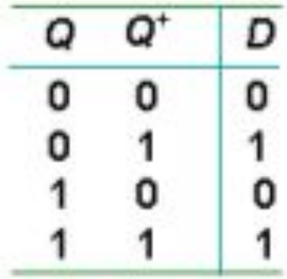
| Q | Q⁺ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

JK Flip-flop



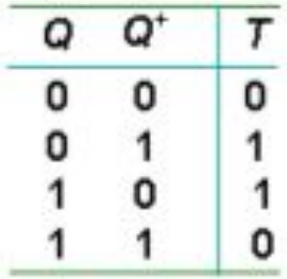
| Q | Q⁺ | S | R |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

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