서강대학교
컴퓨터공학과

## 재미있는 컴퓨터공학실험2

# 11주차 발표

김도영, 윤준서, 이규형

# 목차

# 플립플롭이란?

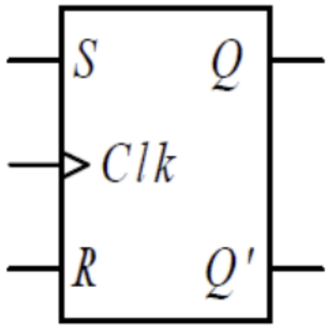
1bit를 기억할 수 있는 순서회로, 피드백

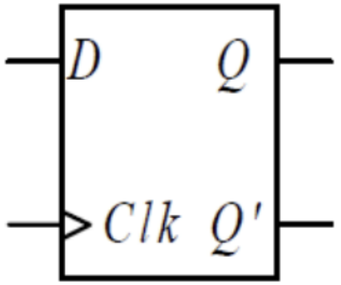
내부의 상태값에 따라 출력이 발생하는 논리회로

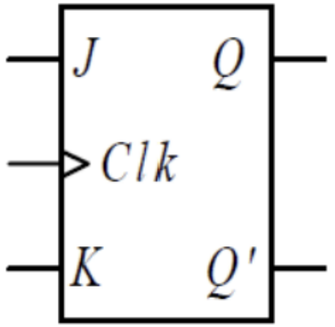
작동 방식에 따라 SR, D, JK, T 타입

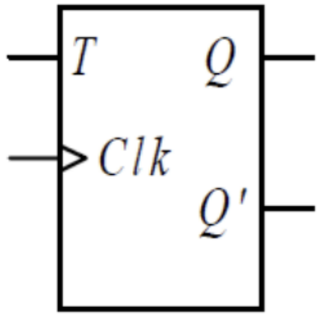
전기 신호가 지속적으로 공급되어야만
정보를 유지할 수 있는 휘발성 메모리

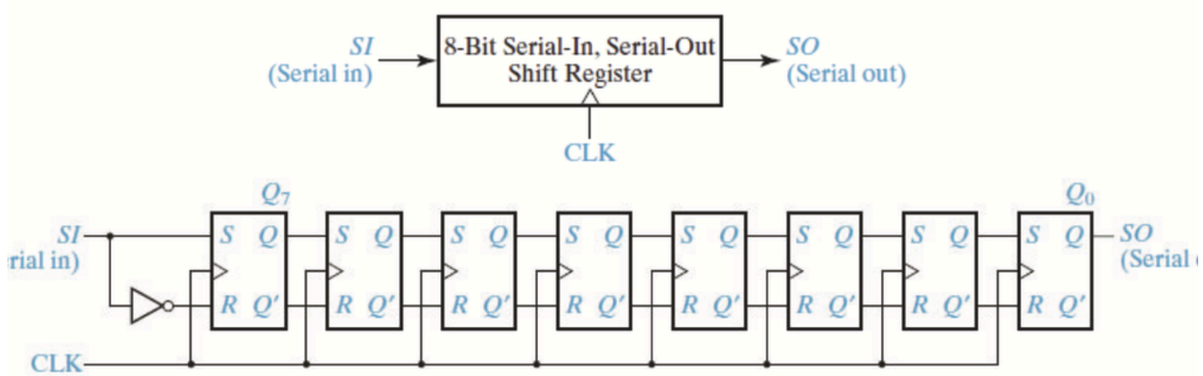
플립 플롭이 여러 개 모여있는 장치가 레지스터



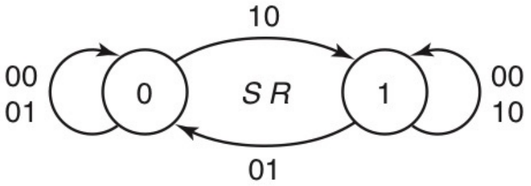



# RS Flip-Flop

**Figure 6.16** *SR* flip flop state diagram.




**Map 6.1** *SR* flip flop behavioral map.

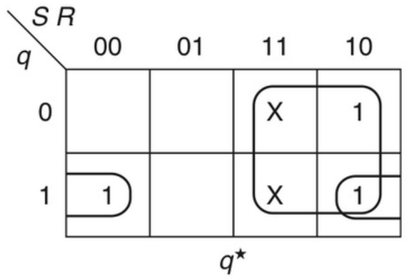



**Table 6.5** *SR* flip flop behavioral tables.

| $S$ | $R$ | $q$ | $q^\star$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | — | not |
| 1 | 1 | 1 | — | allowed |

| $S$ | $R$ | $q^\star$ | |
|---|---|---|---|
| 0 | 0 | $q$ | |
| 0 | 1 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | — | not allowed |

| R | S | Q(t+1) (다음 상태) |
|---|---|---|
| 0 | 0 | 현재 상태 유지 |
| 0 | 1 | Q(t+1)을 1로 Set |
| 1 | 0 | Q(t+1) 0로 Reset |
| 1 | 1 | 오류가 난다. |

$$q^\star = S + R'q$$

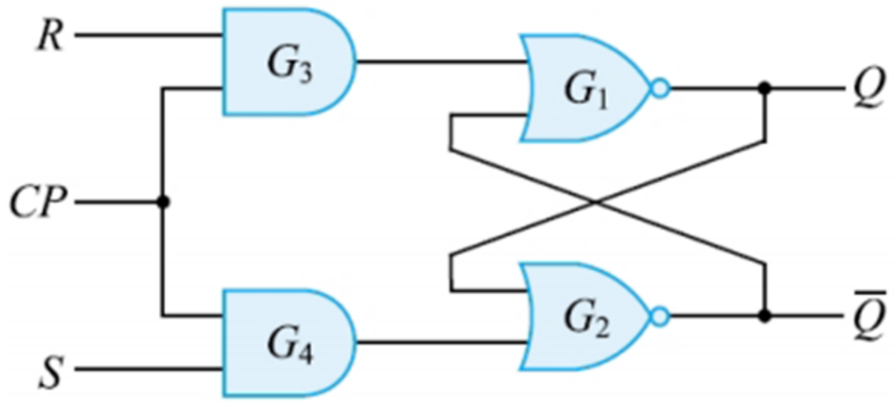
Set을 뜻하는 S가 활성화 상태이면 Q(t+1)을 1로 Set

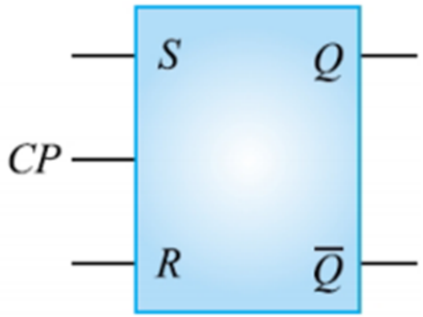
Reset을 뜻하는 R이 활성화 되면 Q(t+1)을 0으로 Reset

R, S, 둘 다 활성화되어 1의 값을 갖는 경우, 오류 발생

위의 오류를 보완하는 것이 JK Flip-Flop

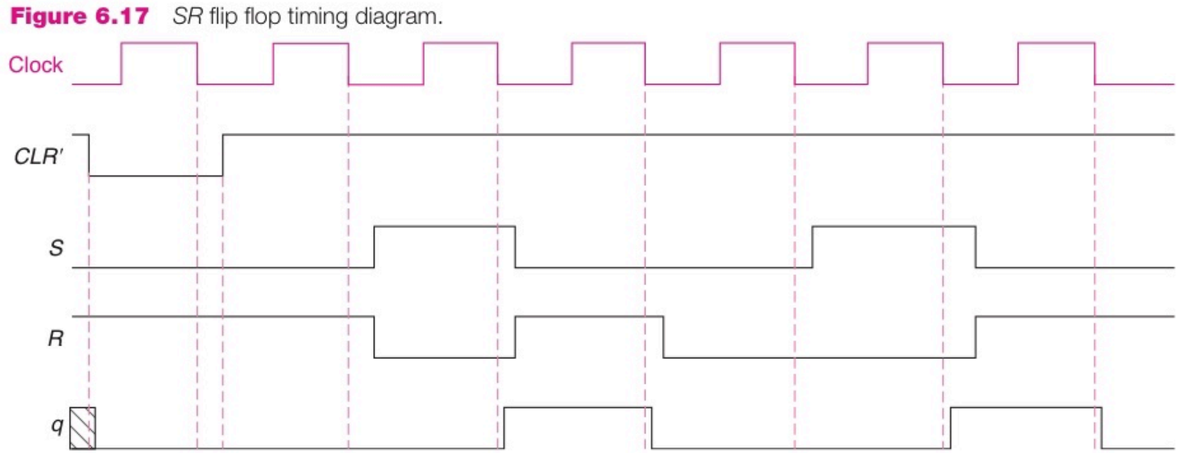
# RS Flip-Flop의 schematic



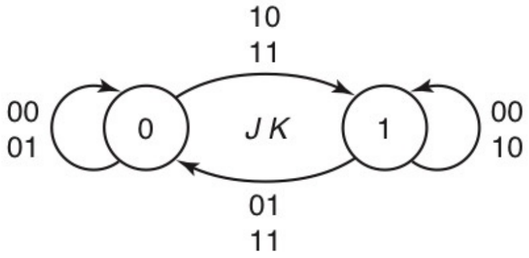

피드백이란?

게이트의 출력이 다시 다른 게이트의 입력으로 들어오는 것



# JK Flip-Flop

**Figure 6.20** *JK* flip flop state diagram.




**Map 6.2** *JK* flip flop behavioral map.

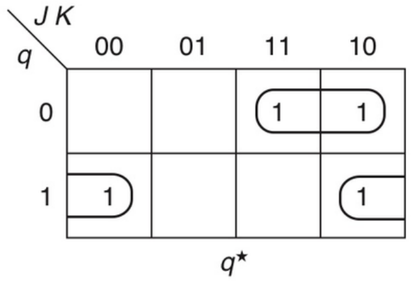



**Table 6.7** *JK* flip flop behavioral tables.

| $J$ | $K$ | $q$ | $q^\star$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

| $J$ | $K$ | $q^\star$ |
|---|---|---|
| 0 | 0 | $q$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $q'$ |

| J(S) | K(R) | Q(t+1) (다음 상태) | |
|---|---|---|---|
| 0 | 0 | Q(t) | 현재 상태 유지 |
| 0 | 1 | 0 | Q(t+1)을 0으로 Reset |
| 1 | 0 | 1 | Q(t+1)을 1로 Set |
| 1 | 1 | Q'(t) | 반전, 토글 |

$$q^\star = Jq' + K'q$$

RS Flip-Flop에서 S, R 값이 모두 1을 갖는 경우 오류 보완

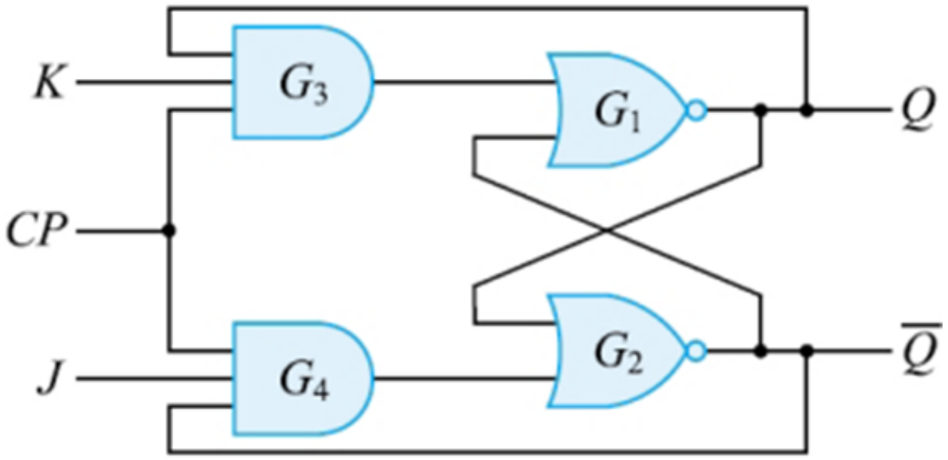
Reset을 뜻하는 K가 활성화 되면 Q(t+1)을 0으로 Reset

Set을 뜻하는 J가 활성화 되면 Q(t+1)을 1로 Set

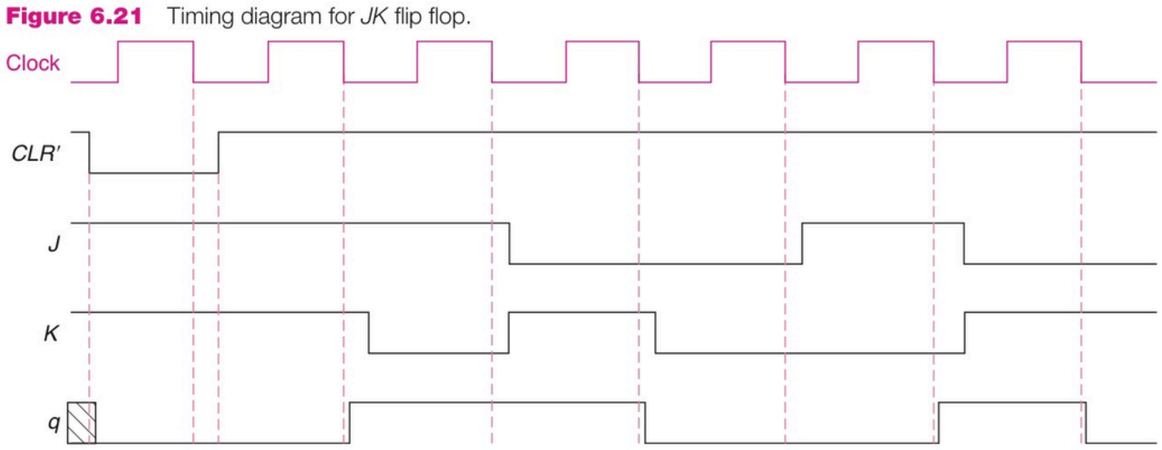
J와 K가 모두 1이면 Q(t+1)은 반전된 상태, 토글

토글(toggle) : 클럭 신호마다 현재 상태와 반대 상태

# JK Flip-Flop의 schematic




클리어(clear) 신호는 플립플롭이나 카운터 같은 디지털 회로를 초기 상태로 리셋하는 역할, 클럭의 의존 여부에 따라 동기 클리어 비동기 클리어로 나뉜다.



# D Flip-Flop

**Figure 6.9** *D* flip flop state diagram.

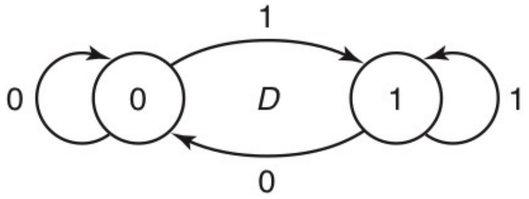



**Figure 6.8** *D* flip flop diagrams.

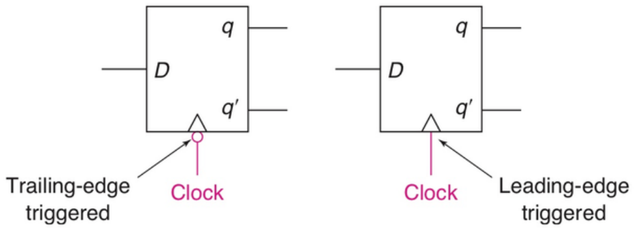



**Table 6.3** The *D* flip flop behavioral tables.

| $D$ | $q$ | $q^\star$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| $D$ | $q^\star$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

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

입력 값을 그대로 출력하는 Flip-Flop

이전 출력에 상관 없이 입력에 따라 출력값이 결정됨

D는 Data와 Delay의 약자로 신호를 지연시키는 역할

클럭이 활성화될 때만 새로운 D값이 Q에 반영

Reset, Set 값 대신 D와 D'을 연결함

# D Flip-Flop의 schematic

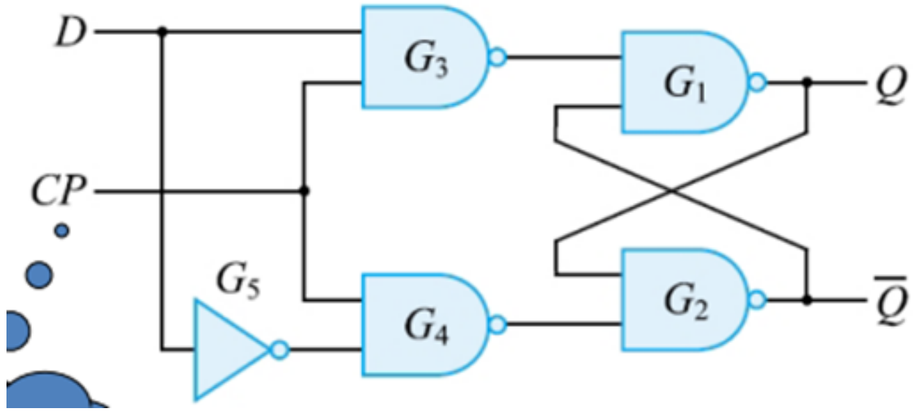
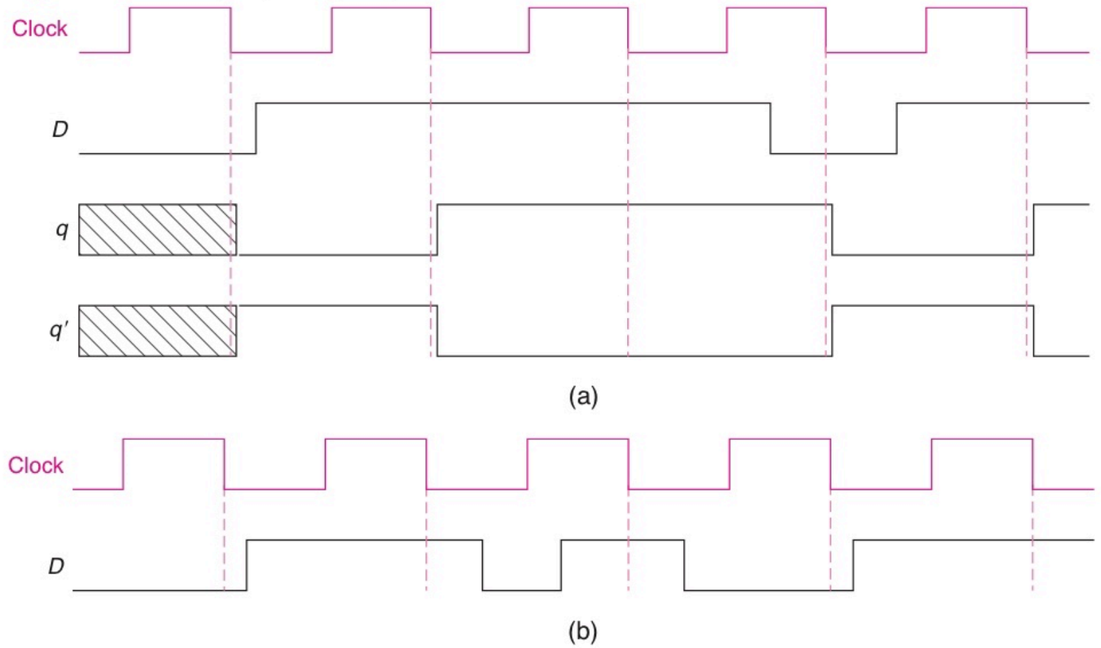
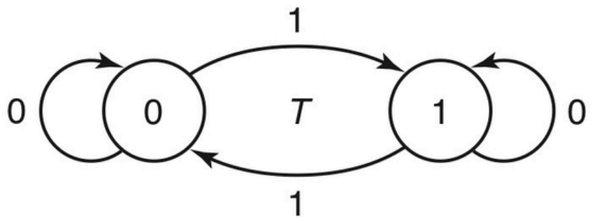



**Figure 6.10** *D* flip flop timing diagram.



# T Flip-Flop

**Figure 6.18** *T* flip flop state diagram.




The behavioral equation is

$$q^\star = T \oplus q$$

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

**Table 6.6** *T* flip flop behavioral tables.

| $T$ | $q$ | $q^\star$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

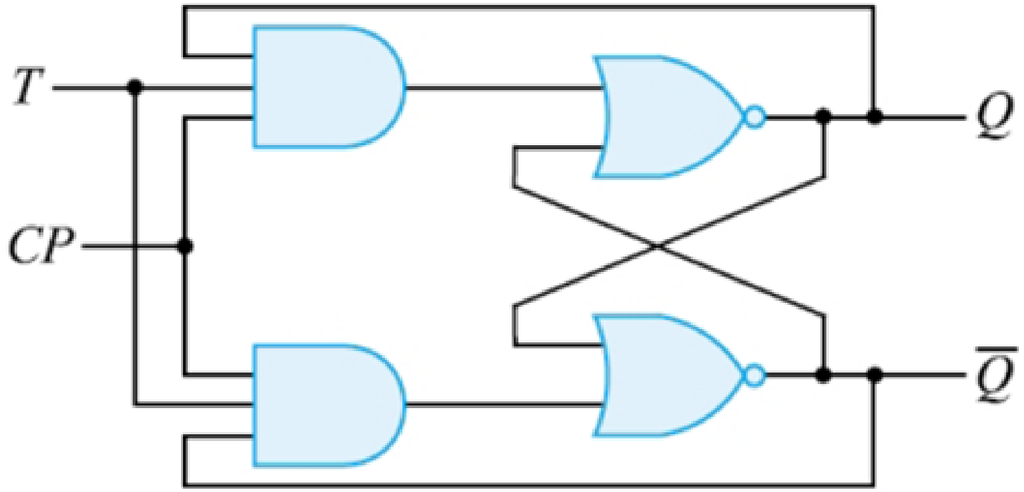
| $T$ | $q^\star$ |
|---|---|
| 0 | $q$ |
| 1 | $q'$ |

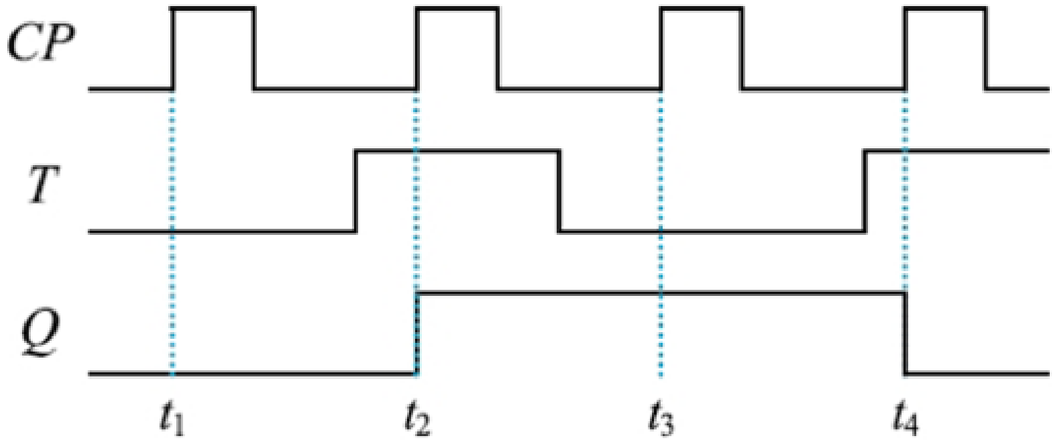
T Flip-Flop의 T는 Toggle(반전)의 뜻을 가짐

T가 0이면 이전 값을 유지하고, T가 1이면 토글

XOR게이트와 동일한 효과

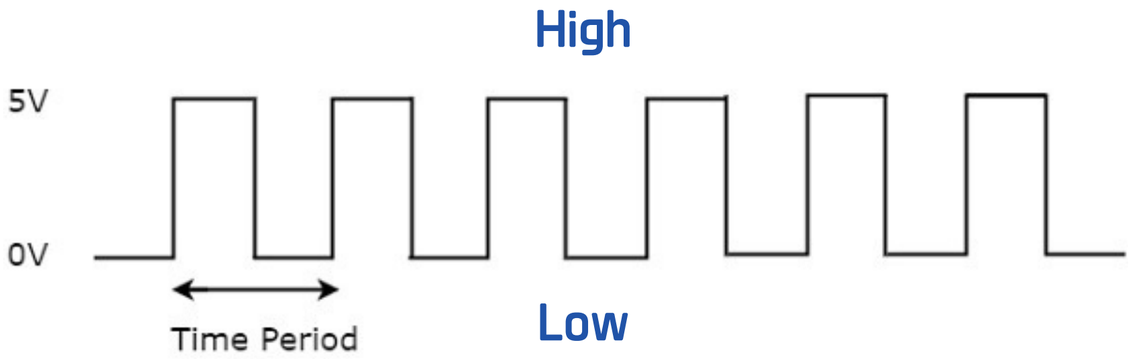
# T Flip-Flop의 schematic



# 클럭(Clock)

- High와 Low가 주기적으로 등장하는 파형 신호
- 순차 논리 회로의 연산 시간을 맞추는 기준
- CPU의 처리 속도를 담당
- High로 시작해서 Low로 끝나는 한 번의 주기

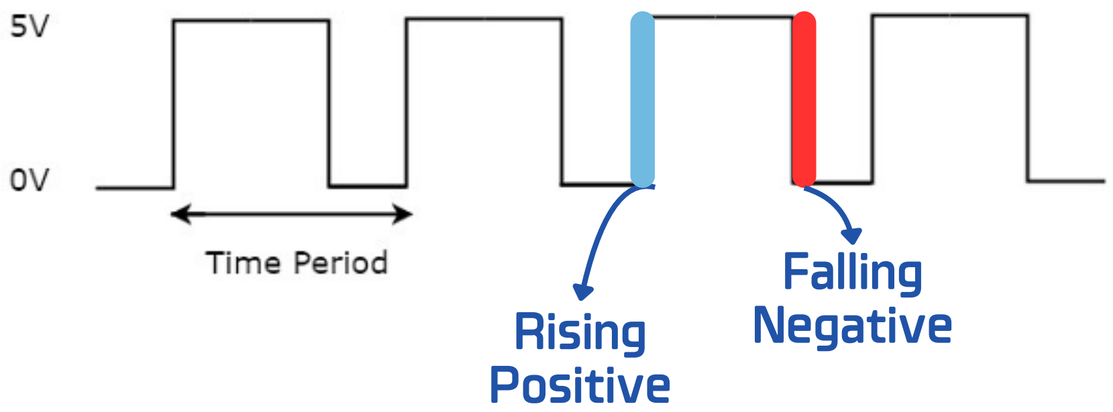


# 클럭(Clock)

- 클럭의 주기 = 1 / 주파수
- 주기와 실제 작동 시간 사이의 효율성
- Duty cycle = 주기에 따른 High의 시간의 비율
- High와 Low의 전환 지점 Edge

$$Duty\ Cycle(\%) = \frac{T_{on}}{T_{on} + T_{off}} \cdot 100$$

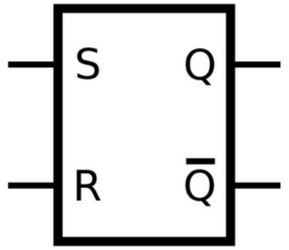
$$= \frac{T_{on}}{T_{total}} \cdot 100$$



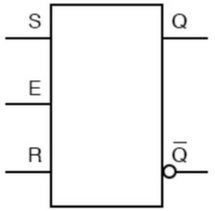
# 래치(Latch)

- 두 개의 출력 상태 중 하나의 상태를 가지며, 그 출력을 바꿀 수 있게 만드는 하나 이상의 입력을 가지는 기억 소자
- 이전 출력을 현재 입력으로 가짐
- 클럭을 사용하지 않음 (비동기식)
- 두 개의 출력이 서로 보수 관계

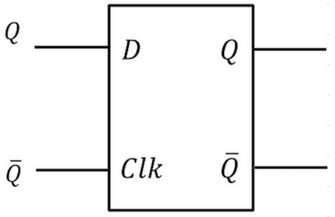
- SR Latch



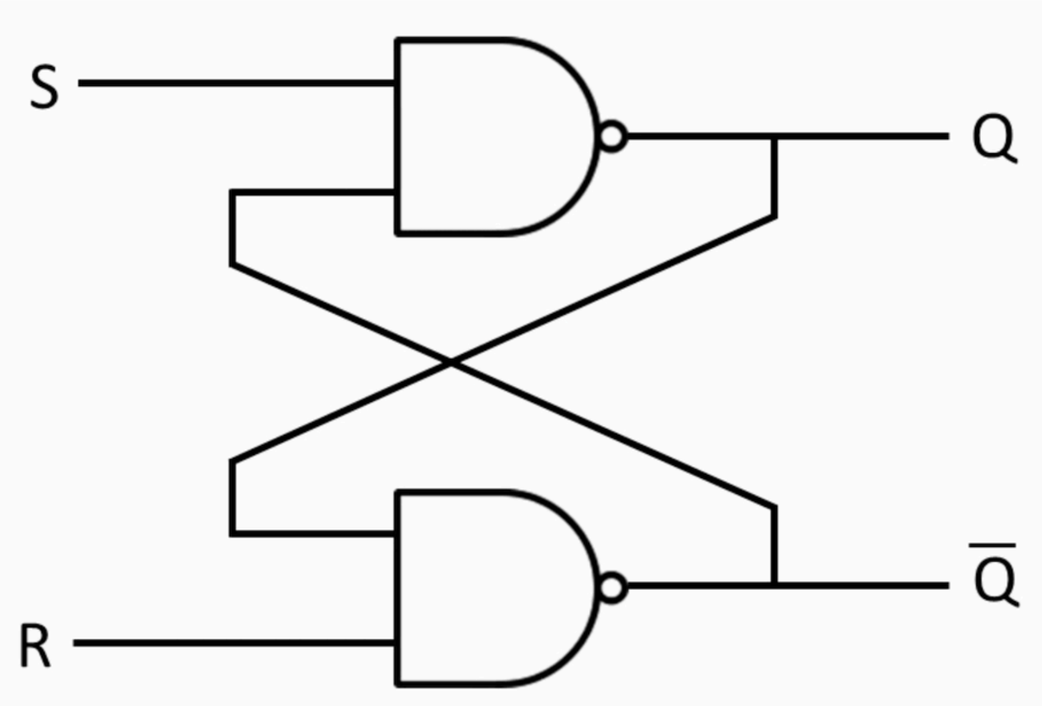

- gated SR Latch




- D Latch

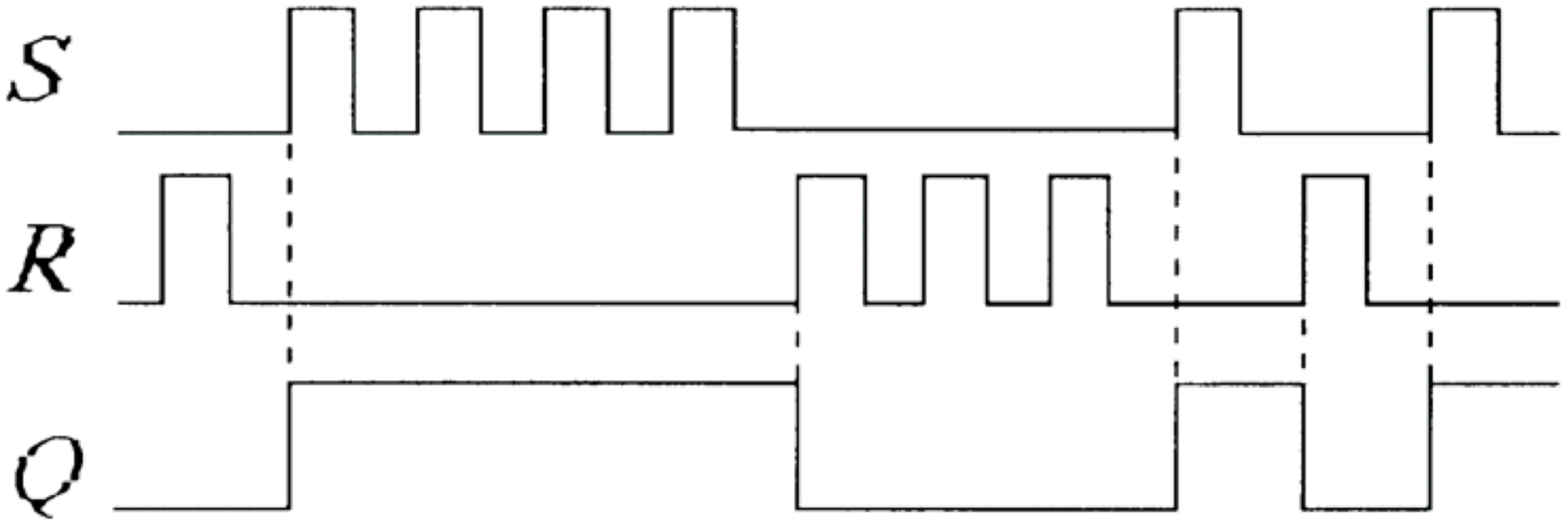


# SR Latch

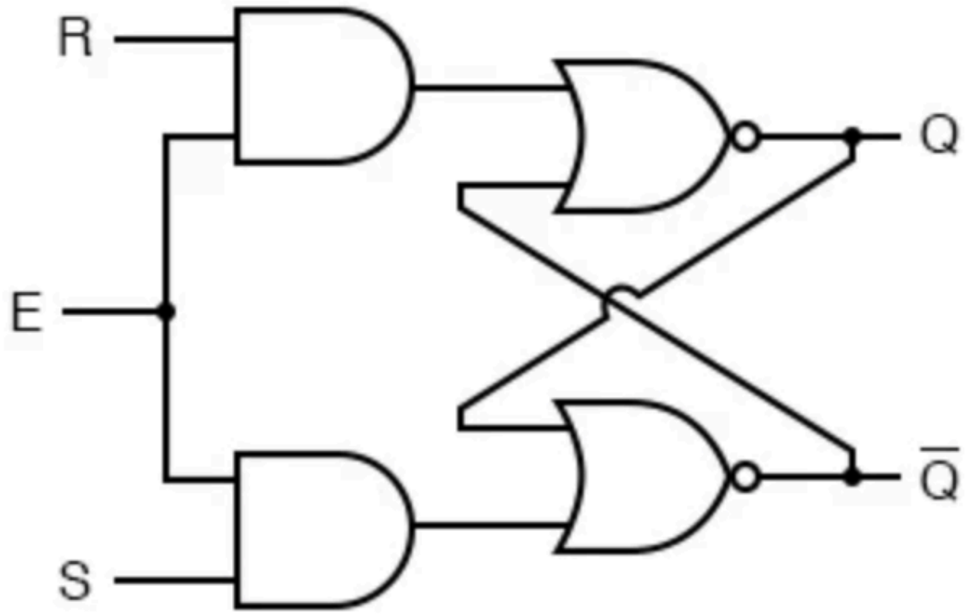



| S | R | Q | Q’ | 상태 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | set |
| 0 | 0 | 1 | 1 | set |
| 0 | 1 | 0 | 1 | reset |
| 1 | 0 | 1 | 0 | reset |
| 1 | 1 | NA | NA | NA |

# SR Latch

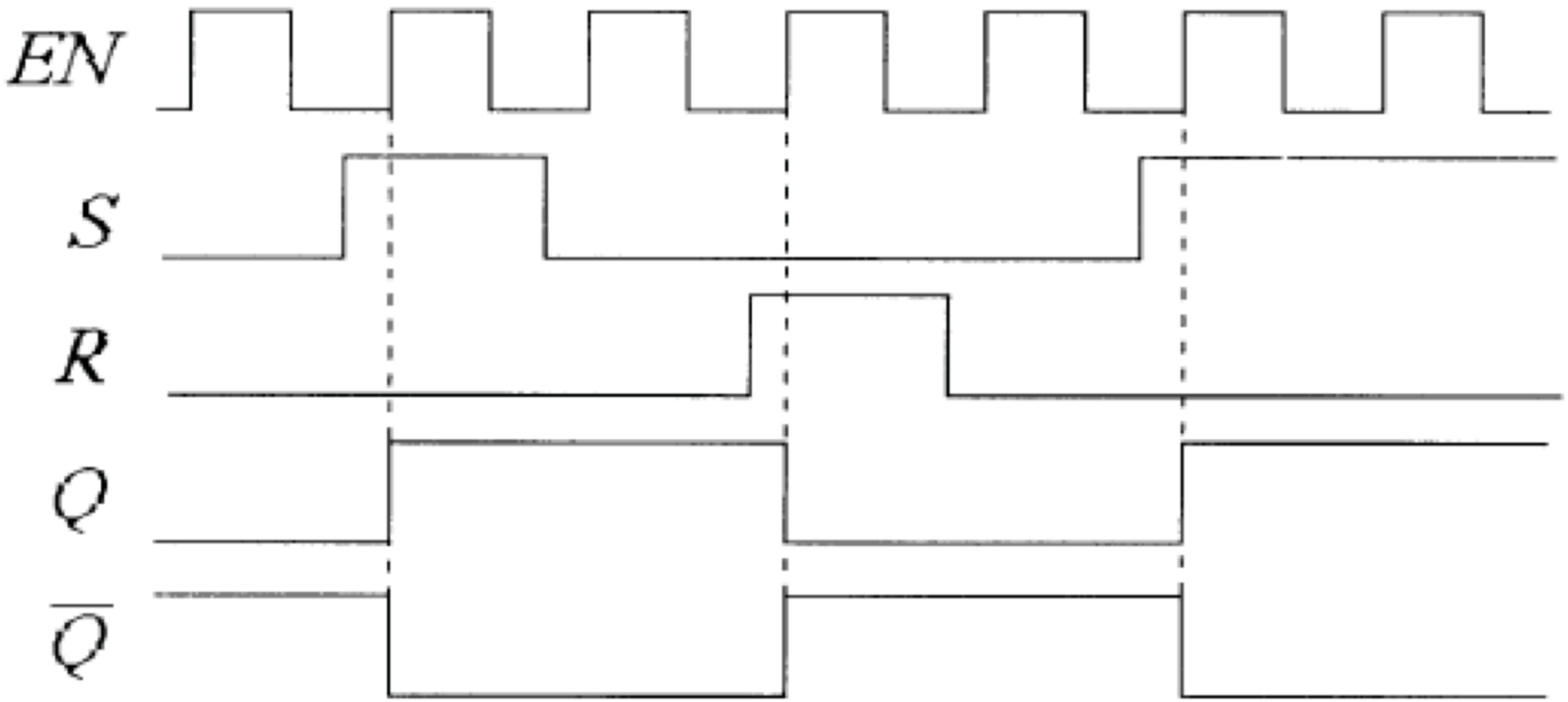


# gated SR Latch

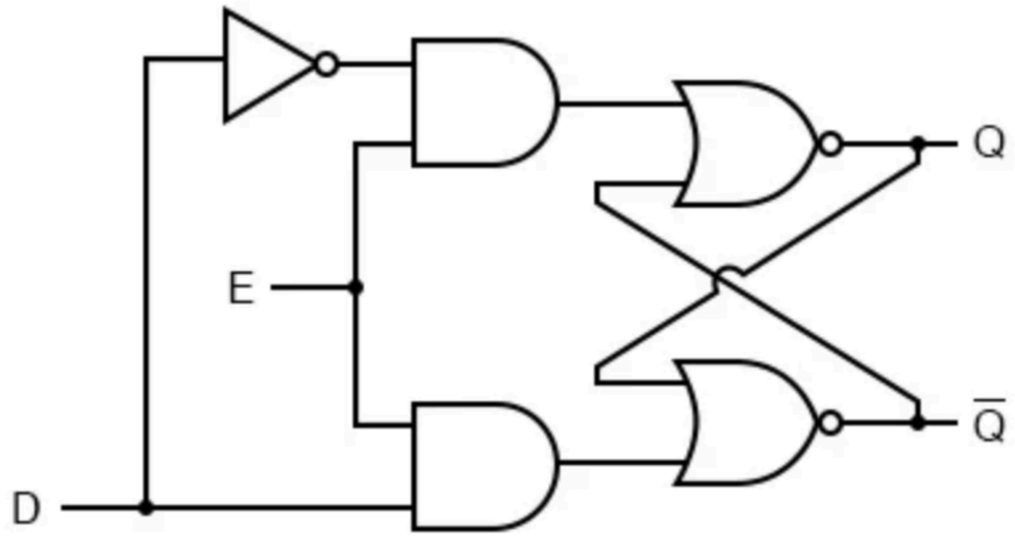



| E | S | R | Q | Q’ | 상태 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 이전 | 이전 | set |
| 1 | 0 | 0 | 이전 | 이전 | set |
| 1 | 0 | 1 | 0 | 1 | reset |
| 1 | 1 | 0 | 1 | 0 | reset |
| 1 | 1 | 1 | NA | NA | NA |

# gated SR Latch

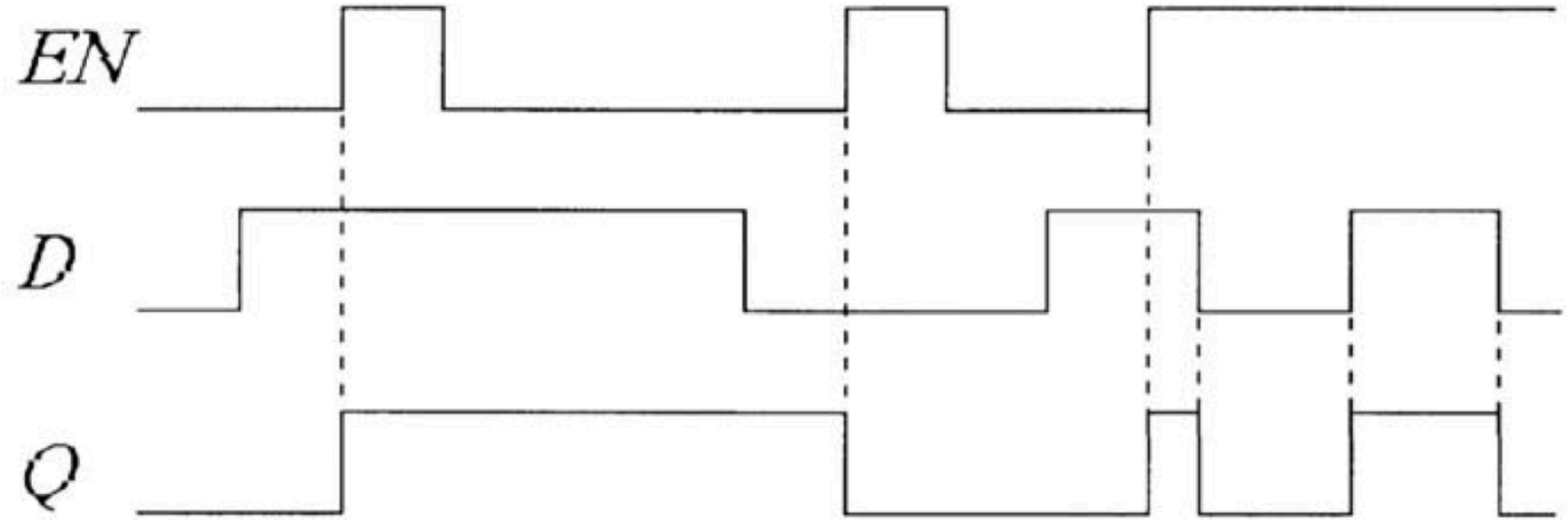


# D Latch




| E | D | Q | Q’ | 상태 |
|---|---|---|---|---|
| 0 | 0 | 이전 | 이전 | set |
| 0 | 1 | 이전 | 이전 | set |
| 1 | 0 | 0 | 1 | reset |
| 1 | 1 | 1 | 0 | reset |

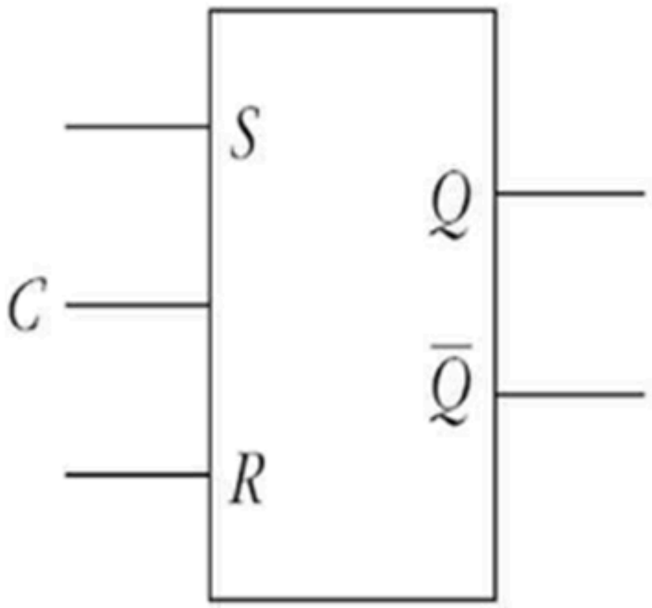
# D Latch



# 래치의 활용

- 레지스터의 기본 구성 요소
- RAM, 메모리 셀의 기본 구성 요소
- 데이터 버퍼
- 시스템 데이터 저장

# Level Trigger




- Trigger(트리거):
  상태 변화를 촉발하는 기동신호

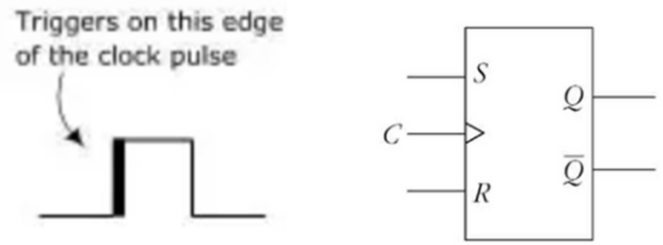
- Level Trigger:
  논리 상태가 High이거나 Low일 때만
  입력 데이터를 받아들임

- Level Trigger 중
  논리 상태가 High일 때를 트리거로 인식하는 것
  ->high level trigger

  논리 상태가 Low일 때를 트리거로 인식하는 것
  -> low level trigger

# Edge Trigger

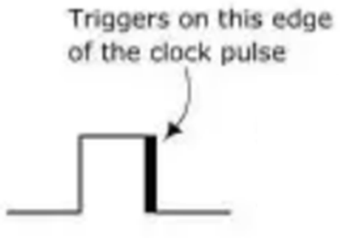
## Positive Edge Trigger

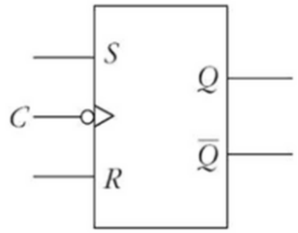






## Negative Edge Trigger



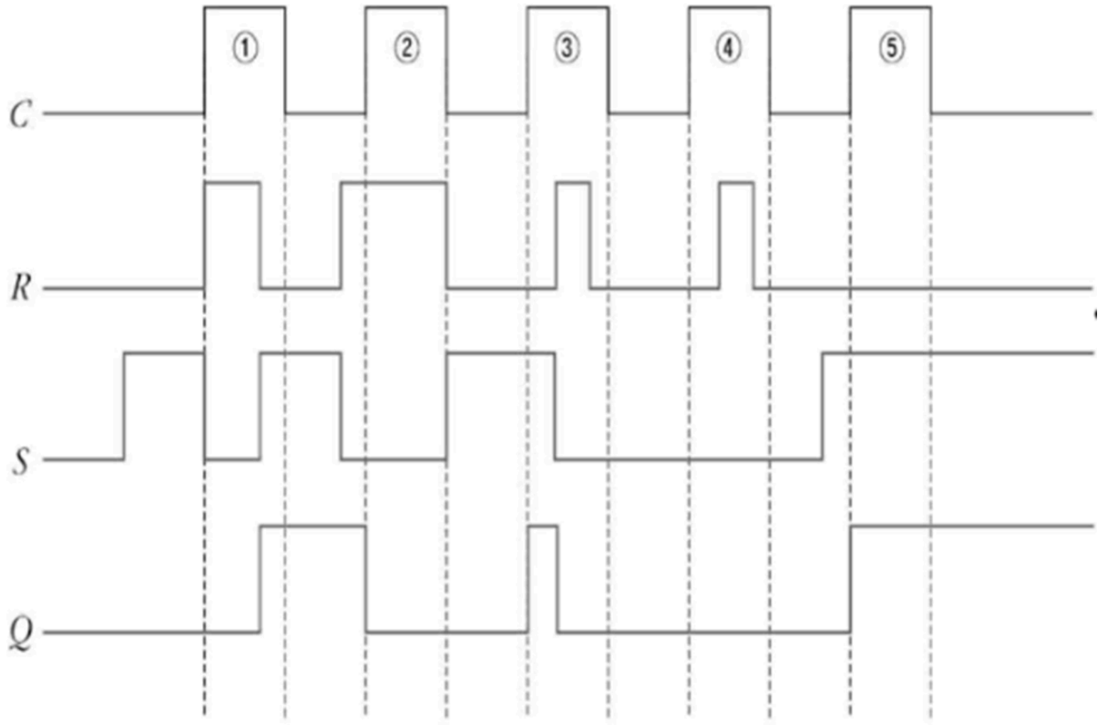




- Edge Trigger:
  클럭 천이(클럭 에지)에서만 입력 데이터를 받아들임

- Edge Trigger 중
  상승 에지를 트리거로 인식하는 것
  -> 상승(또는 positive) 에지 트리거

  하강 에지를 트리거로 인식하는 것
  -> 하강(또는 negative) 에지 트리거

# Comparison of triggers



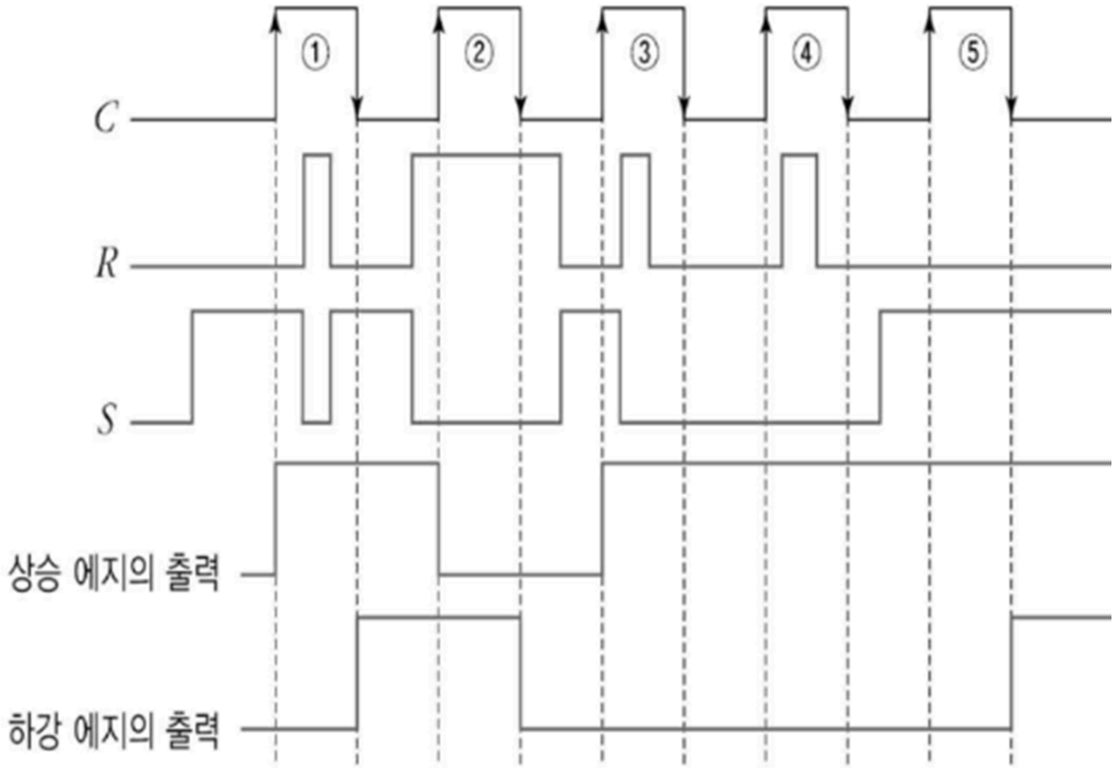

Level Trigger

High level trigger라고 가정

SR 래치

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

# Comparison of triggers

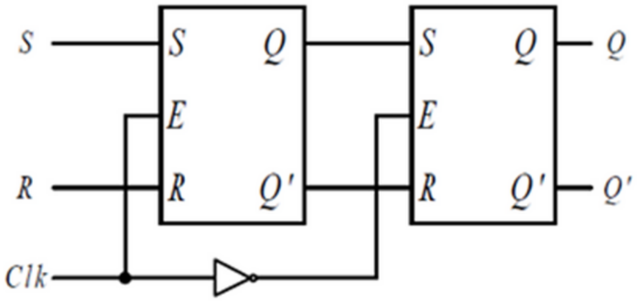



Edge Trigger

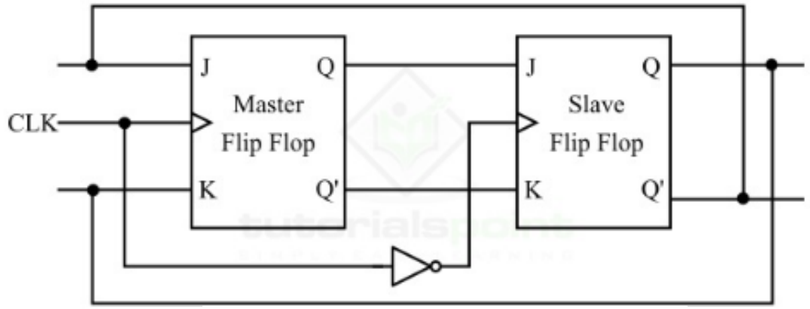
SR 플립플롭

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

# Master-Slave structure




SR flip flop




JK master slave flip flops

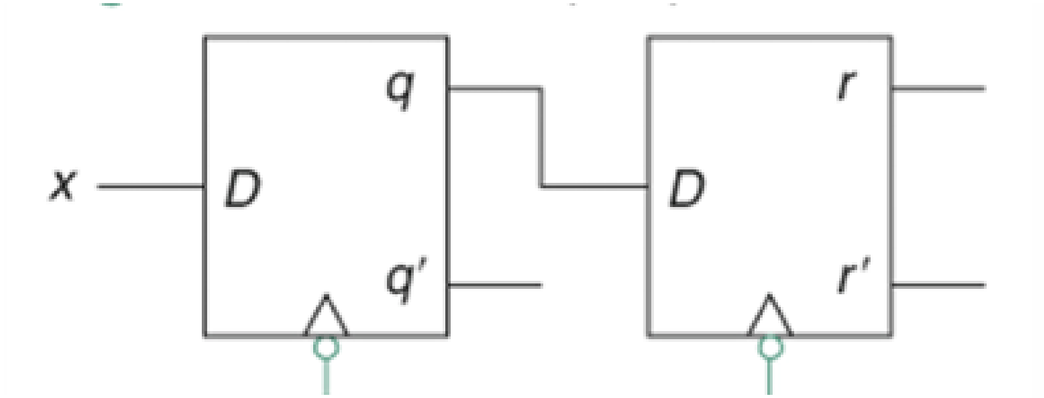
- 마스터-슬레이브:
  두 개의 플립플롭 혹은 래치를 직렬로 연결하여
  안정성과 정확성을 높이는 구조

- 첫 번째 플립플롭이 마스터, 두 번째 플립플롭이 슬레이브.

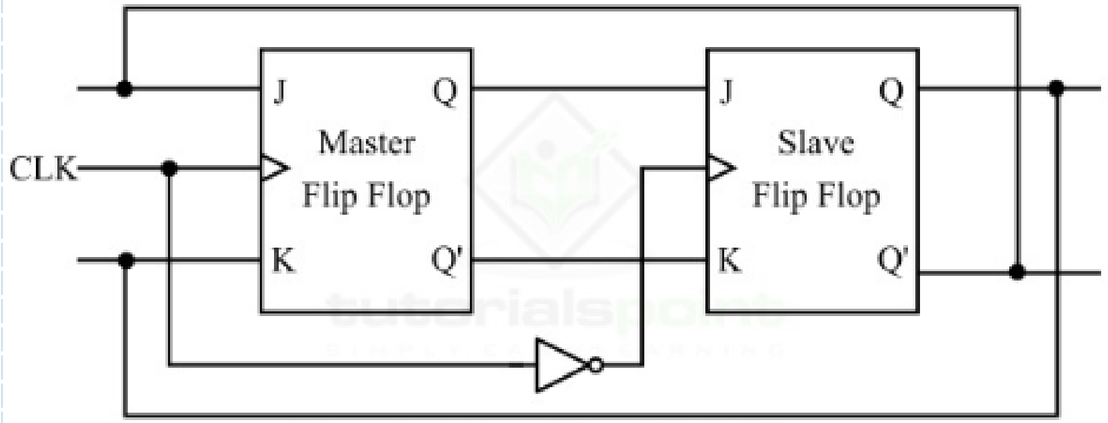
- 클럭 신호가 하이로 바뀌면,
  ->마스터 플립플롭은 입력을 받아들임.
  ->슬레이브 플립플롭은 이전의 값을 유지.

- 클럭 신호가 로우로 바뀌면
  ->마스터 플립플롭이 상태를 유지.
  ->슬레이브 플립플롭은 마스터의 출력을 받아들여
  자신의 출력을 변경.

# Examples of Master-Slave

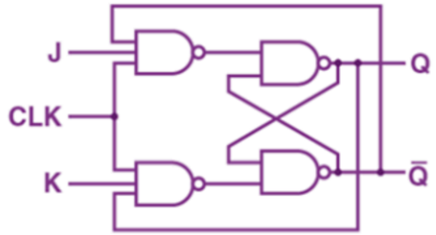



Using D flip flops



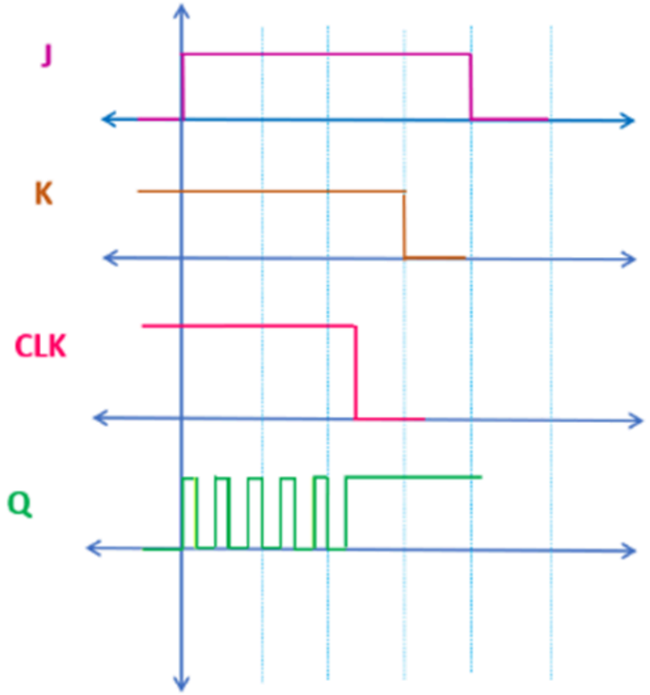

Using JK flip flops

# Racing in JK flip flop



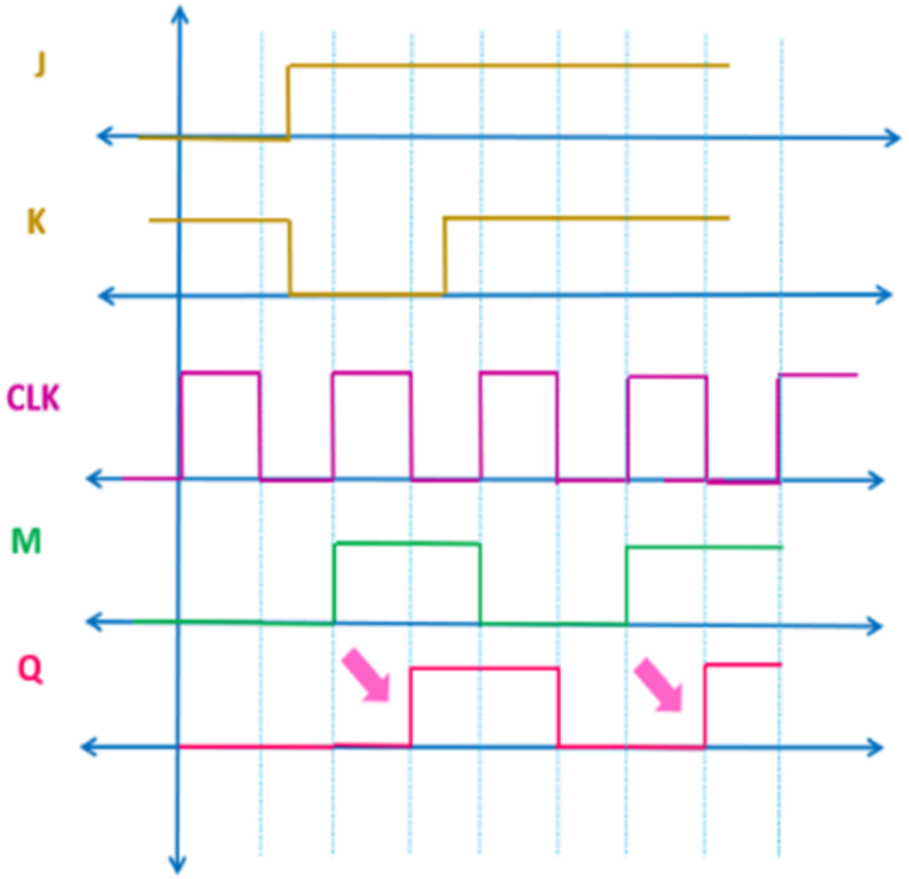

Truth Table

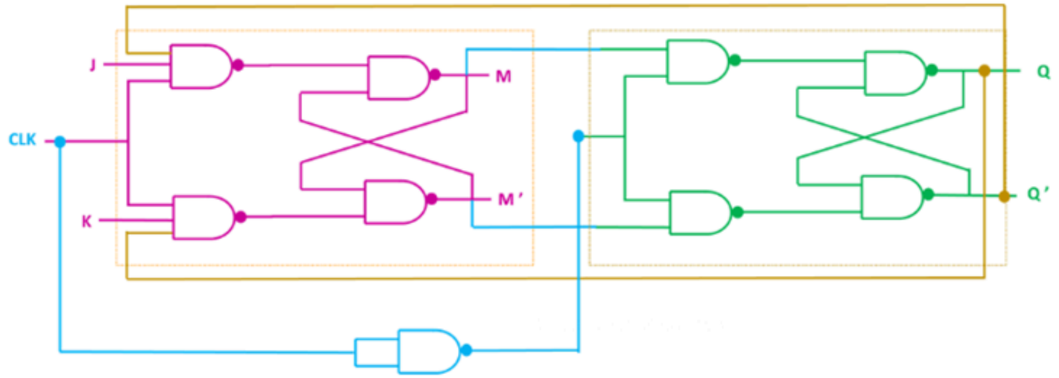
| J | K | $Q_N$ | $Q_{N+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |




JK플립플롭은 출력이 보수가 된 다음에도 Clock Pulse가 계속 남아있게 되면,
다시 보수가 되면서 출력이 불안정하게 요동치는 문제점이 발생한다.

# JK master slave flip flop

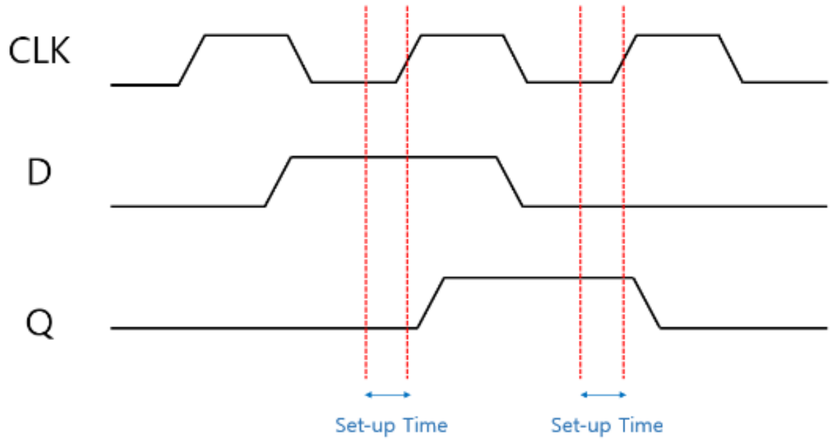


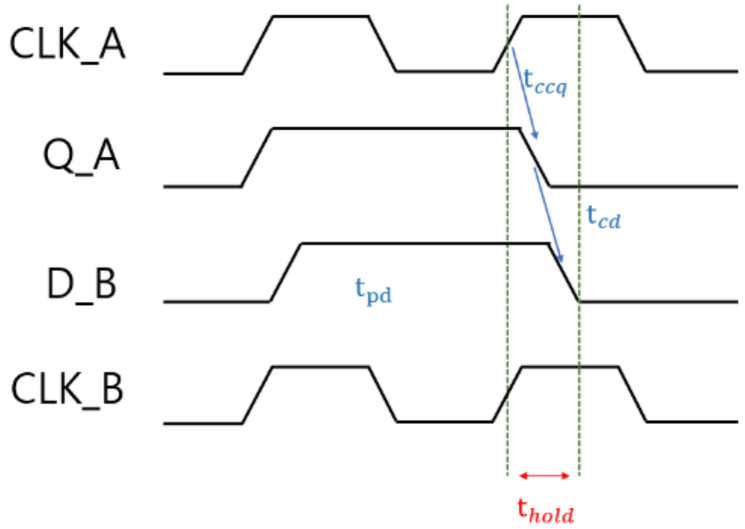



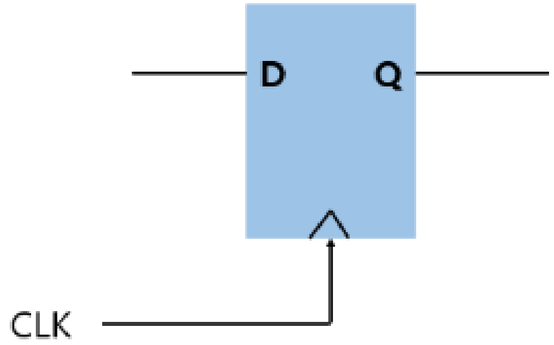

Truth Table

| J | K | $Q_N$ | $Q_{N+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

# Setup time & Hold time



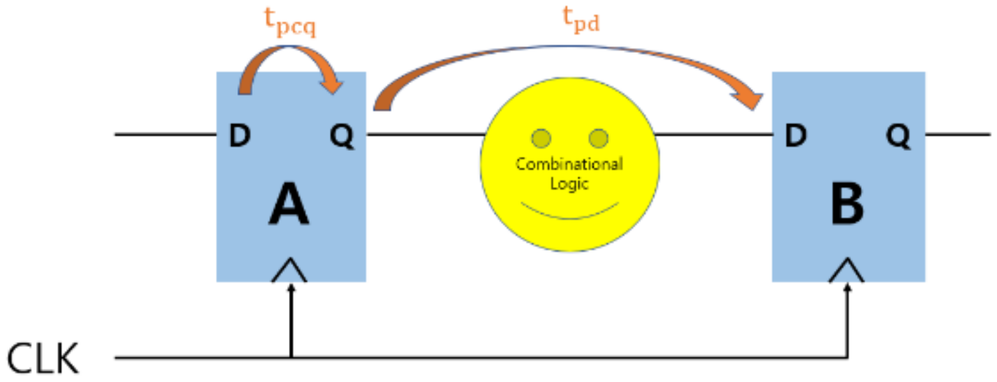



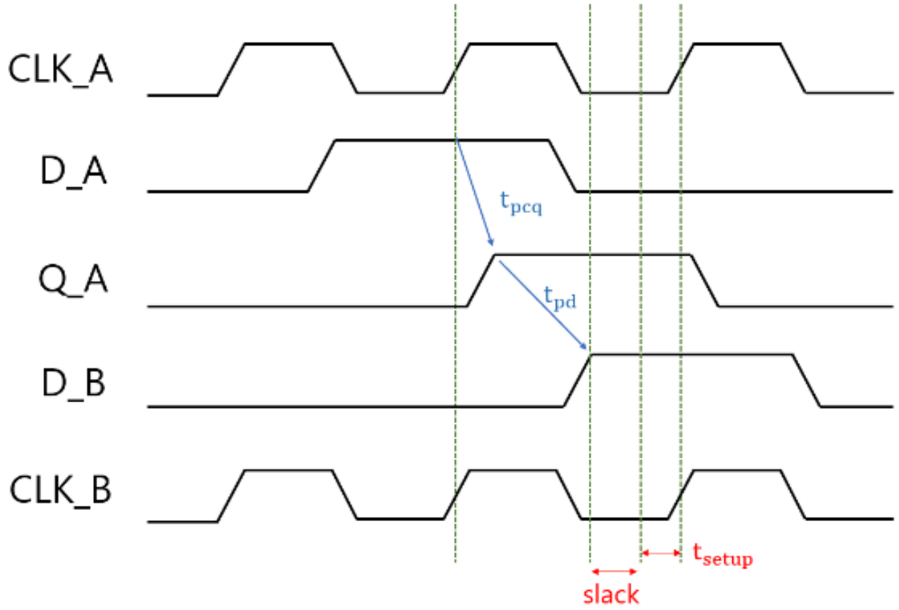

- **Setup time:**
  상승(혹은 하강) 에지 전 출력을 유지하기 위한 최소 시간

- **Hold time:**
  상승(혹은 하강) 에지 후 출력을 유지하기 위한 최소 시간

- 구성 요소에 의해 결정되며, 데이터와 클럭이 오류를 출력하는 최소 시간을 측정하여 결정한다.

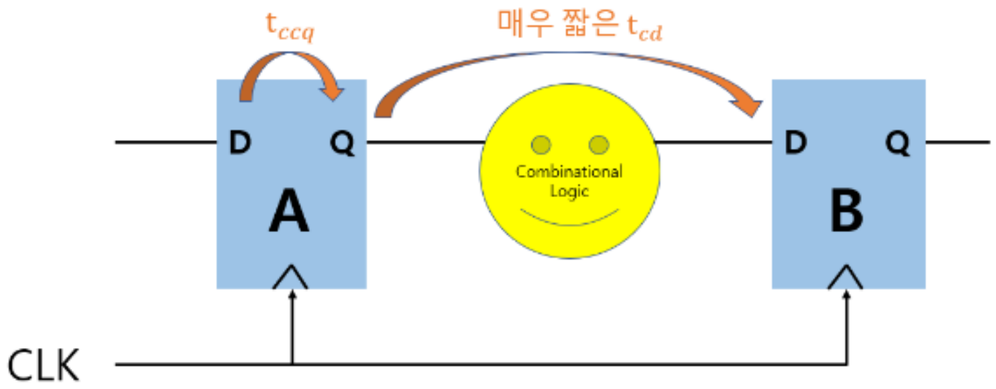
# Setup time & Hold time

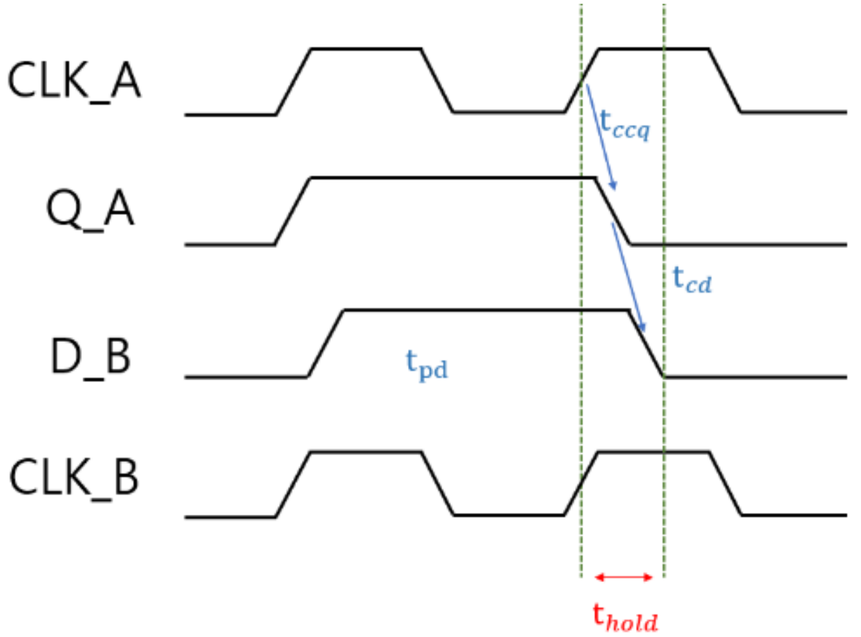



**Tpcd + Tpd + Tsetup <= Tck**

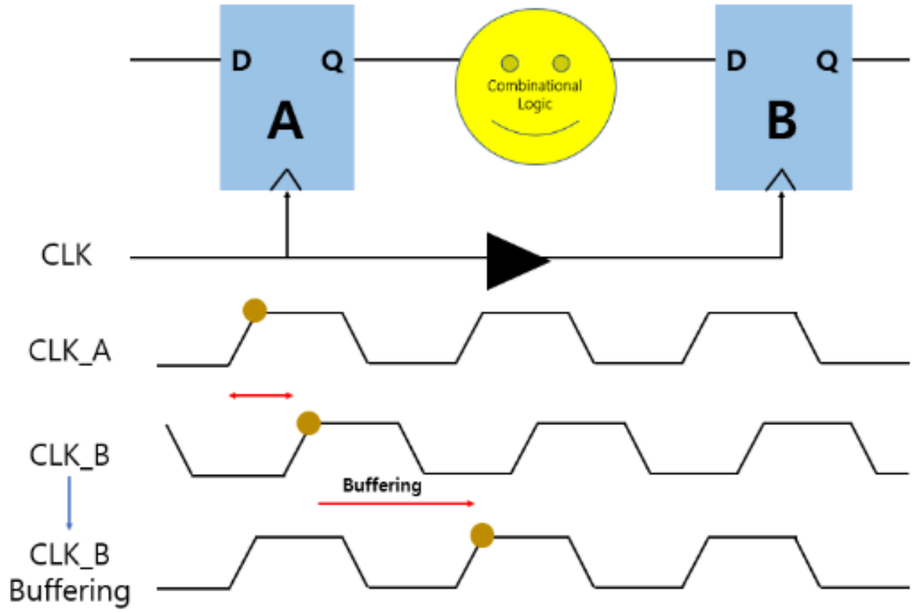
# Setup time & Hold time




Tccq + Tcd >= Thold

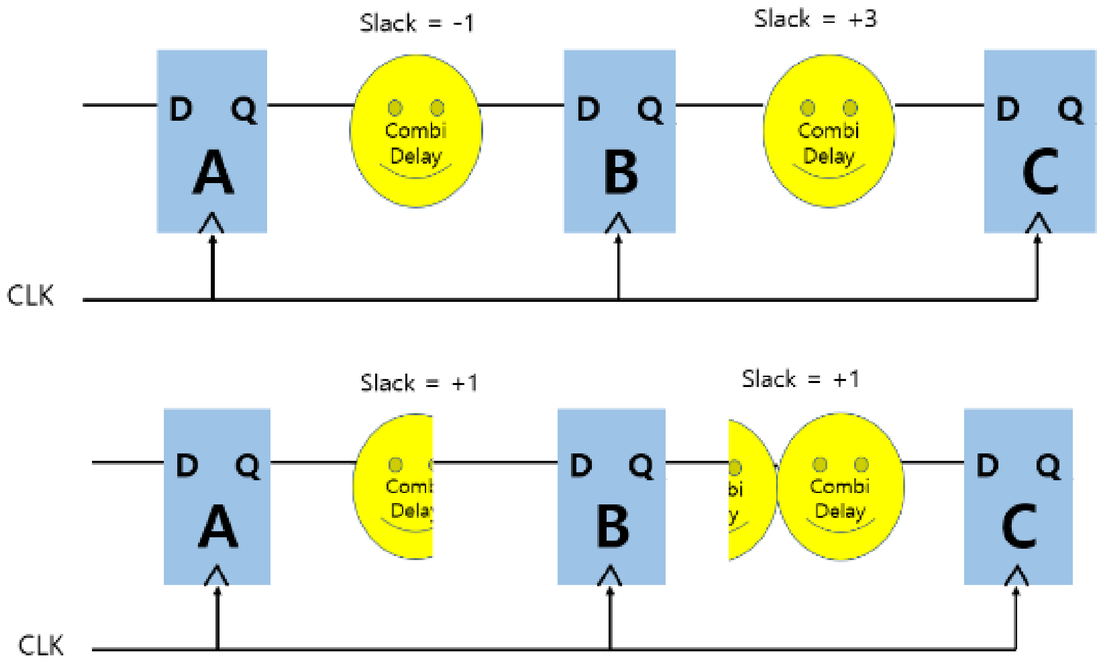
# Solution to time violation



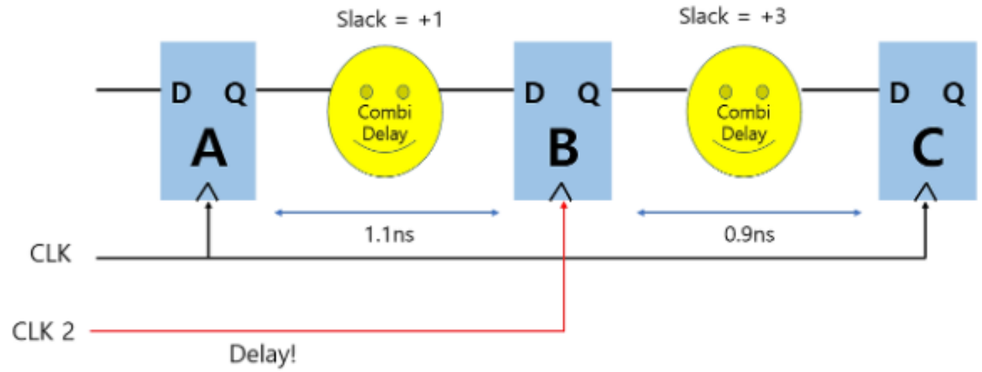

- 클럭 주파수 감소
  ->성능 저하

- clock buffer:
  버퍼를 이용해 클락에
  delay를 주어 타이밍을 맞춘다

# Solution to time violation




- register retiming
  게이트를 다음 시스템으로 이동




- time stealing
  delay가 조금 있는 클럭을
  두 번째 플립플롭에 연결

# 참고문헌

Marcovitz, Alan B. Introduction to Logic Design. 3rd ed., Pearson, 2010.

Wikipedia contributors. 플립플롭. Wikipedia, The Free Encyclopedia. https://ko.wikipedia.org/wiki/%ED%94%8C%EB%A6%BD%ED%94%8C%EB%A1%AD