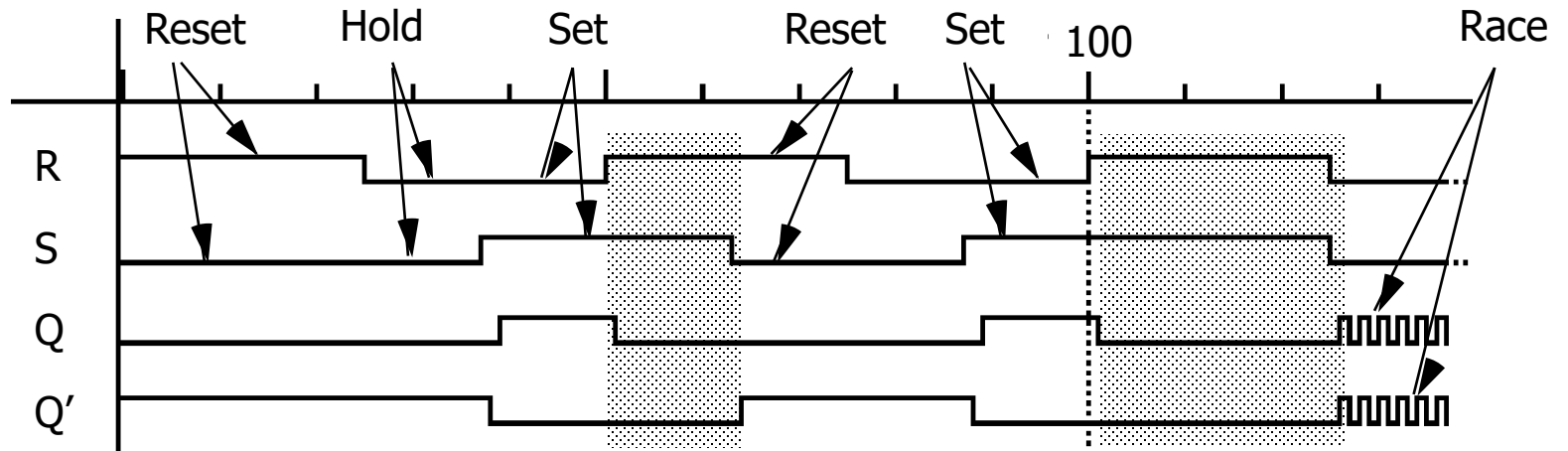
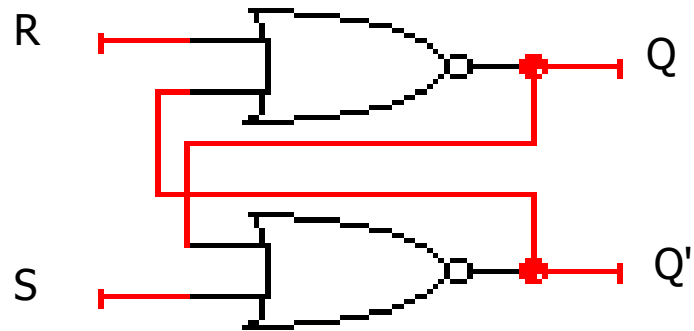
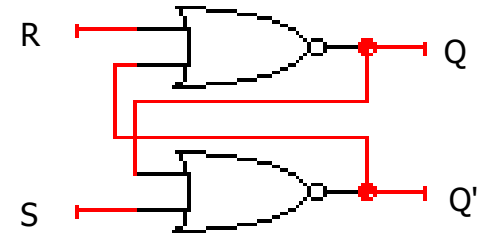


# Latch and Flip-flop - Latch

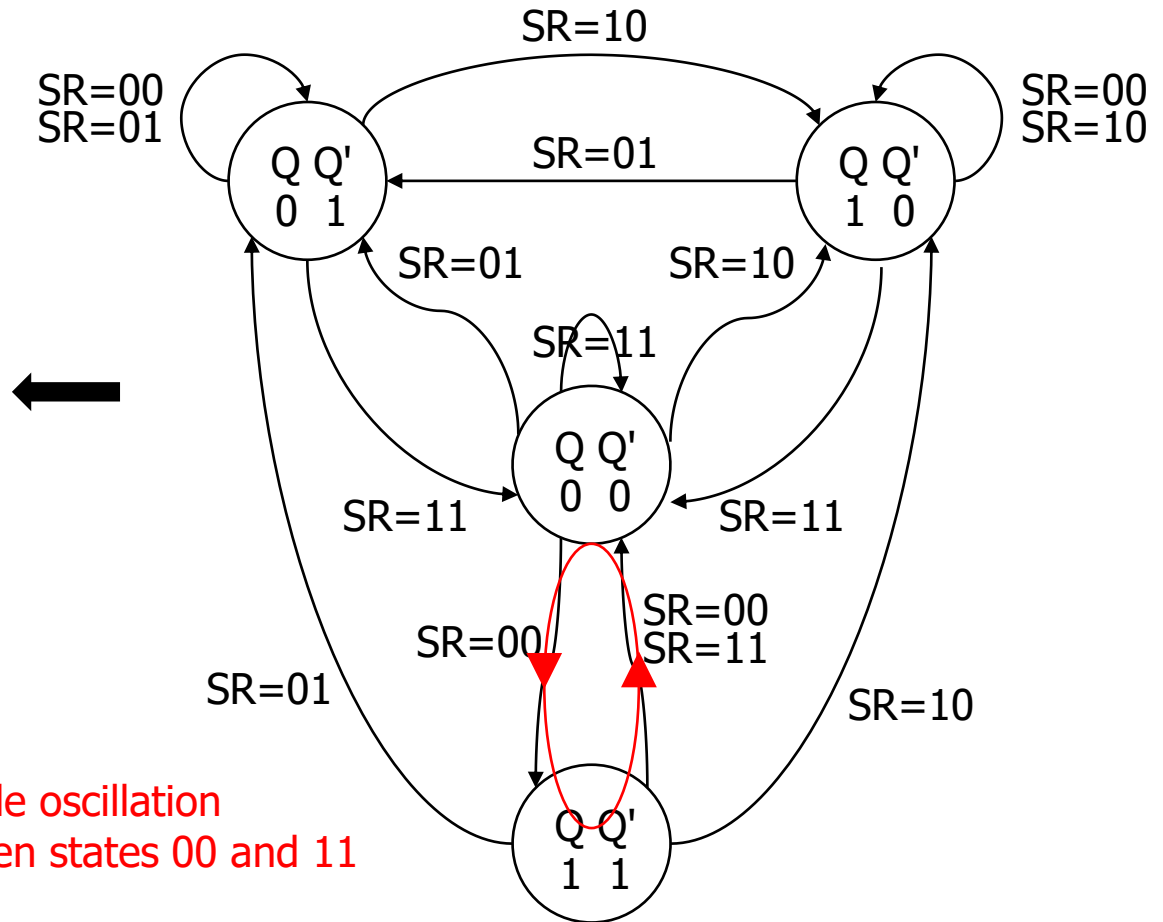
# R-S latch



- State transition diagram
  - states: possible values
  - transitions: changes based on inputs



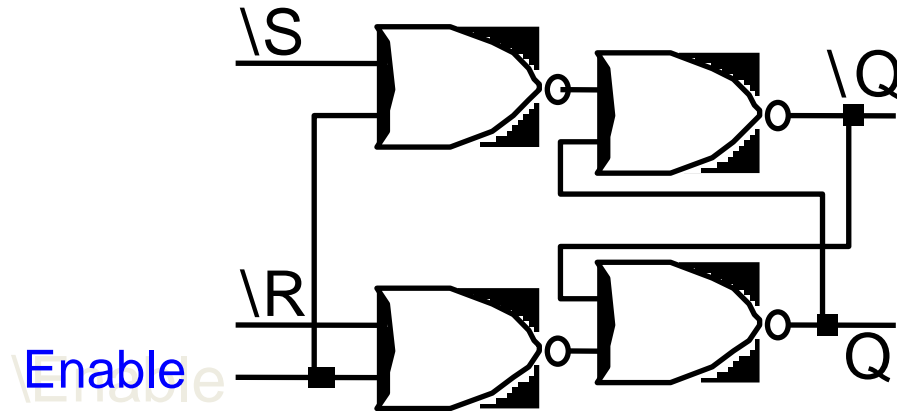
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable



possible oscillation  
between states 00 and 11

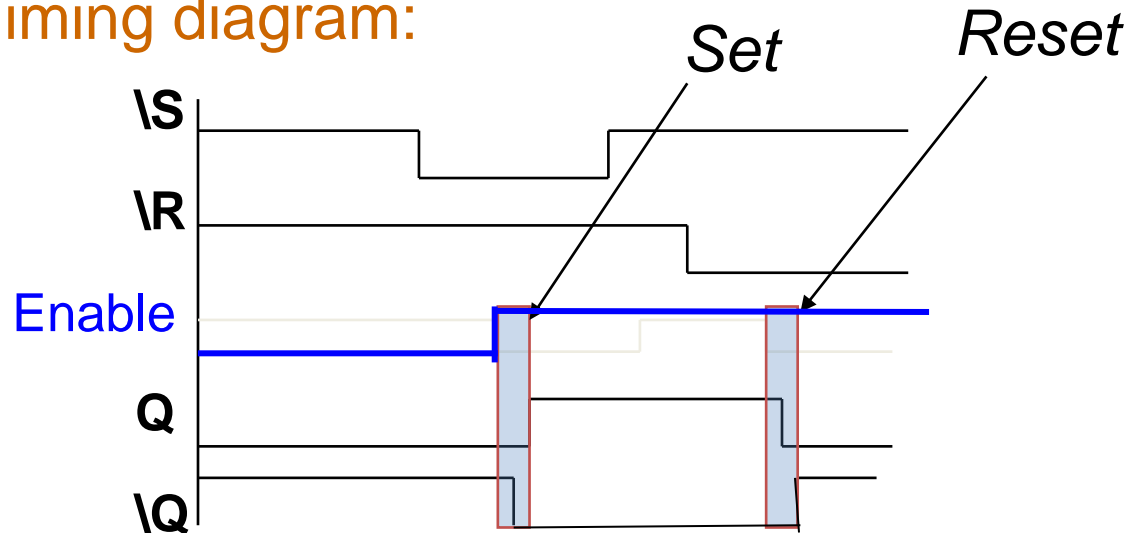
# Gated or Level-Sensitive Latch

- Latch is allowed to change state when the enable signal is asserted.



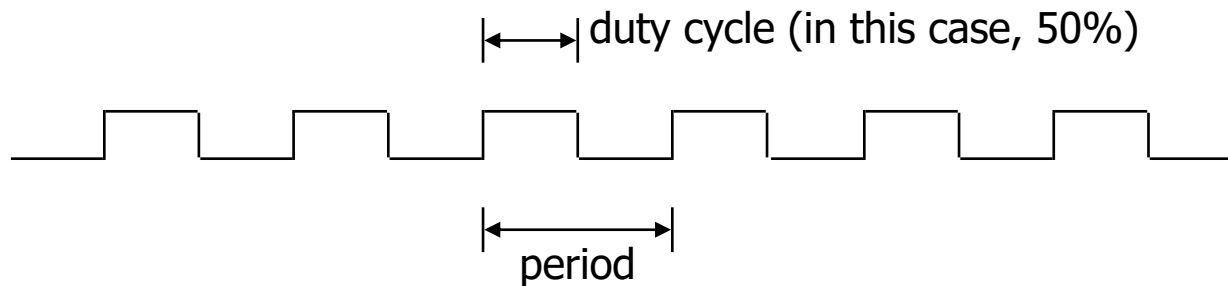
Timing diagram:

- Propagation delay from enable signal to output changes is shown in the diagram.



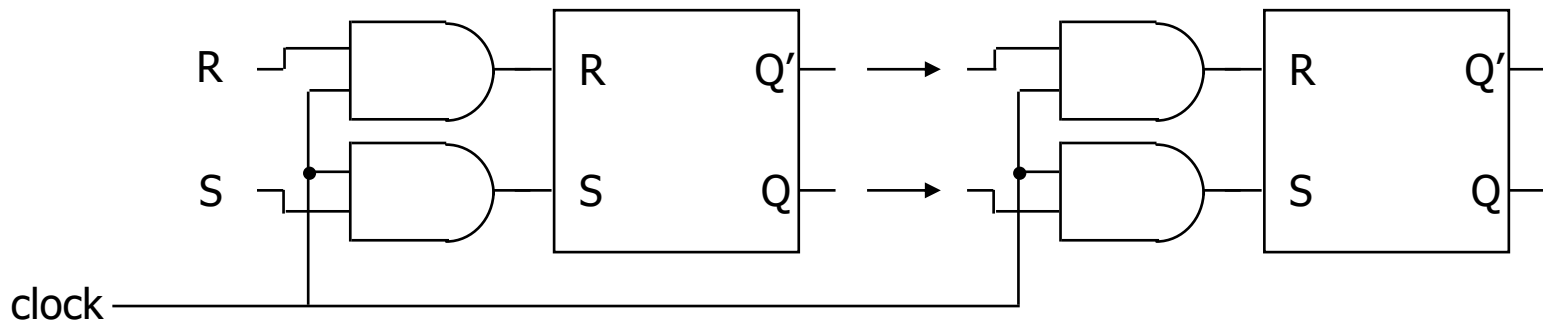
# Clocks

- Used to keep time
  - wait long enough for inputs (R' and S') to settle
  - then allow to have effect on value stored
- Clocks are regular periodic signals
  - period (time between ticks)
  - duty-cycle (time clock is high between ticks - expressed as % of period)



# Combining Latches

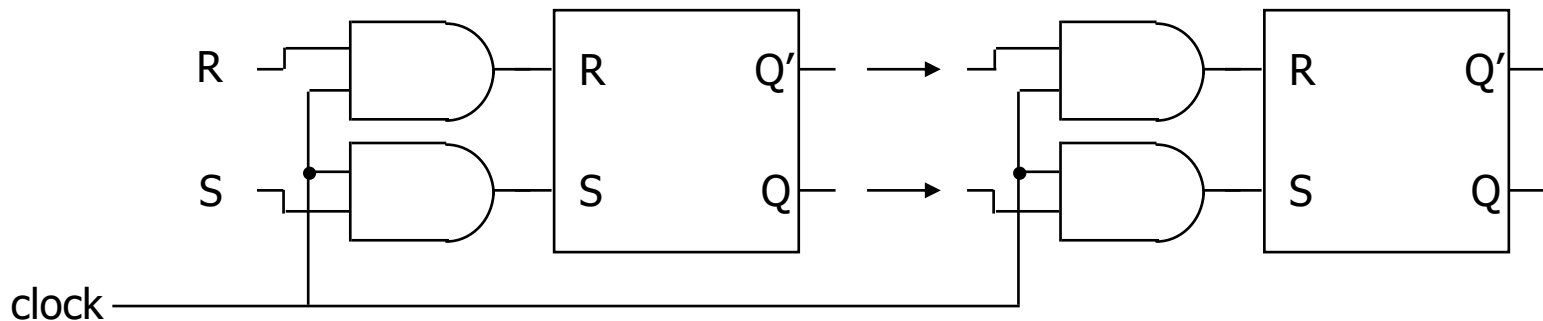
- Connect output of one latch to input of another
- How to stop changes from racing through chain?
  - need to be able to control flow of data from one latch to the next
  - move one latch per clock period
  - have to worry about logic between latches (arrows) that is too fast



# Latch and Flip-flop – Flop-flop

# Combining Latches

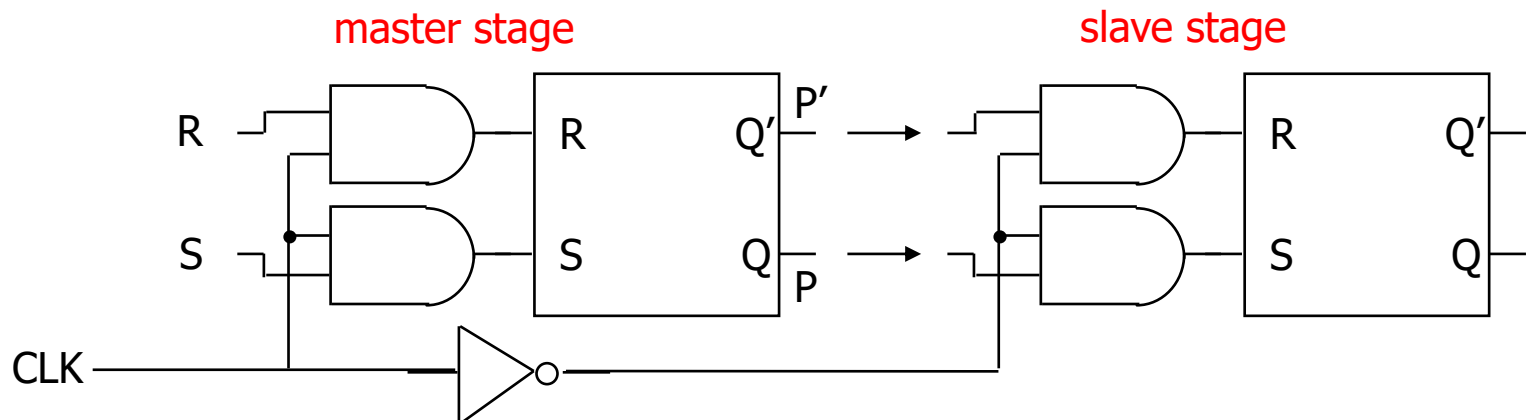
- Connect output of one latch to input of another
- How to stop changes from racing through chain?
  - need to be able to control flow of data from one latch to the next
  - move one latch per clock period
  - have to worry about logic between latches (arrows) that is too fast





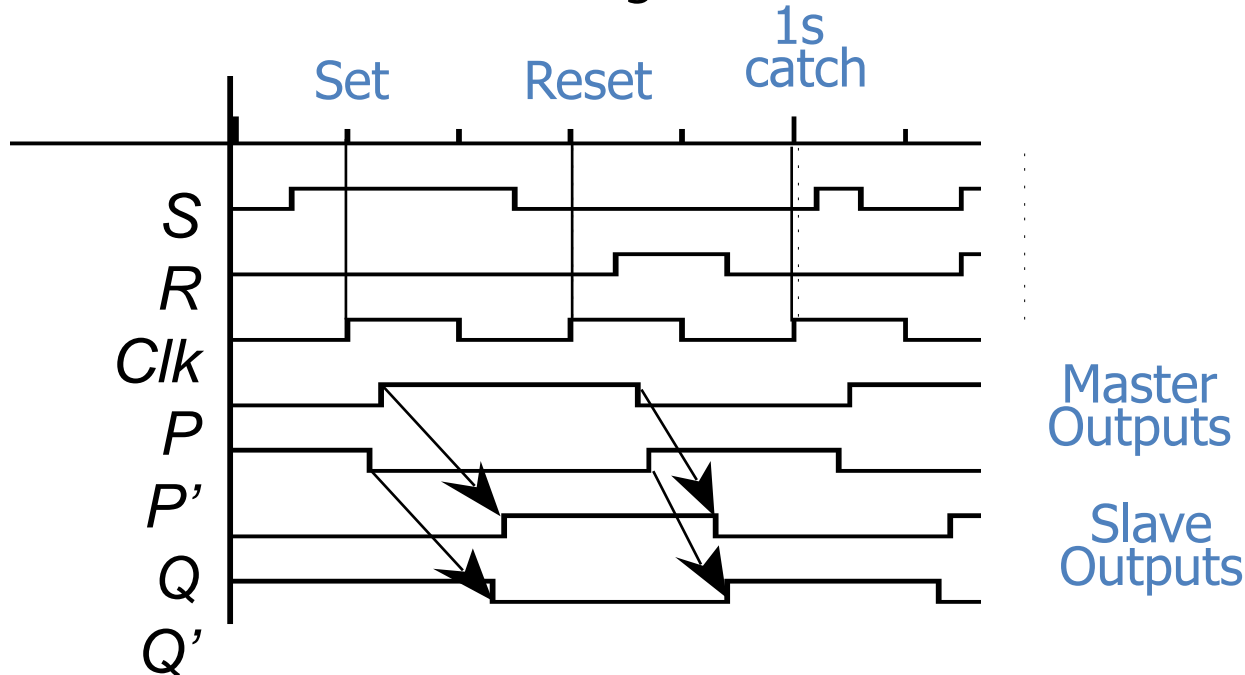
# Master-slave Flip-flop

- Break flow by alternating clocks
  - use positive clock to latch inputs into one R-S latch
  - use negative clock to change outputs with another R-S latch
- View pair as one basic unit
  - master-slave flip-flop
  - twice as much logic
  - output changes a few gate delays after the falling edge of clock but does not affect any cascaded flip-flops



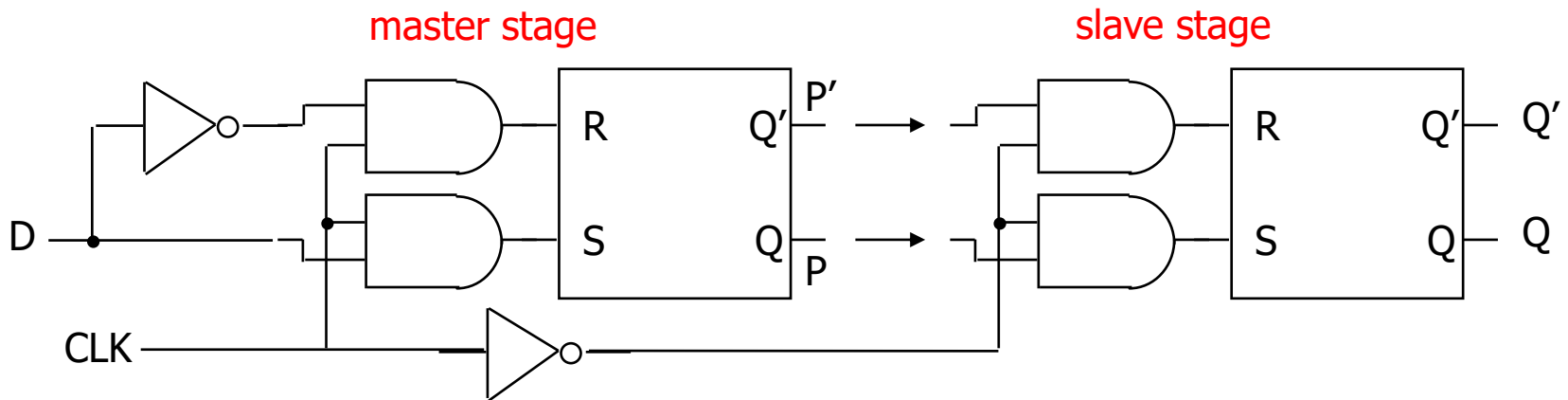
# 1s Catching Problem

- In first R-S stage of master-slave FF
  - 0-1-0 glitch on R or S while clock is high is "caught" by master stage
  - leads to constraints on logic to be hazard-free



# D Flip-flop

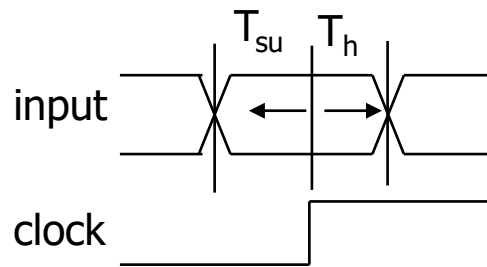
- Make S and R complements of each other
  - eliminates 1s catching problem
  - can't just hold previous value  
(must have new value ready every clock period)
  - value of D just before clock goes low is what is stored in flip-flop



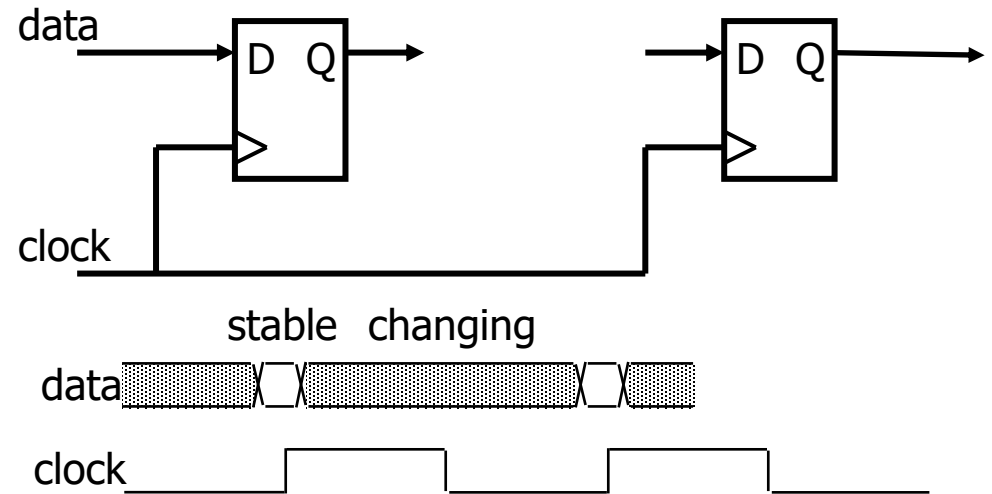
## Latch and Flip-flop – Timing definitions

# Timing definitions

- clock: periodic event, causes state of memory element to change
  - ✓ can be rising edge or falling edge or high level or low level
  - ✓ setup time: minimum time before the clocking event by which the input must be stable ( $T_{su}$ )
  - ✓ hold time: minimum time after the clocking event until which the input must remain stable ( $T_h$ )

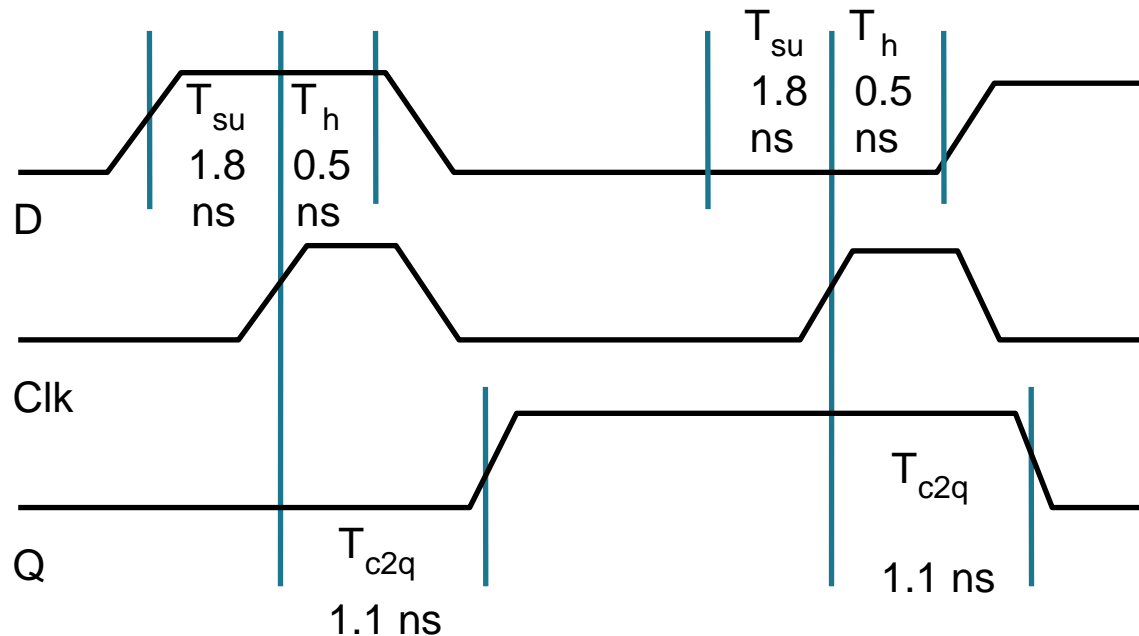


there is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized

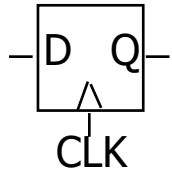


# Typical timing specifications

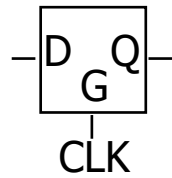
- Positive edge-triggered D flip-flop
  - setup and hold times
  - propagation delay



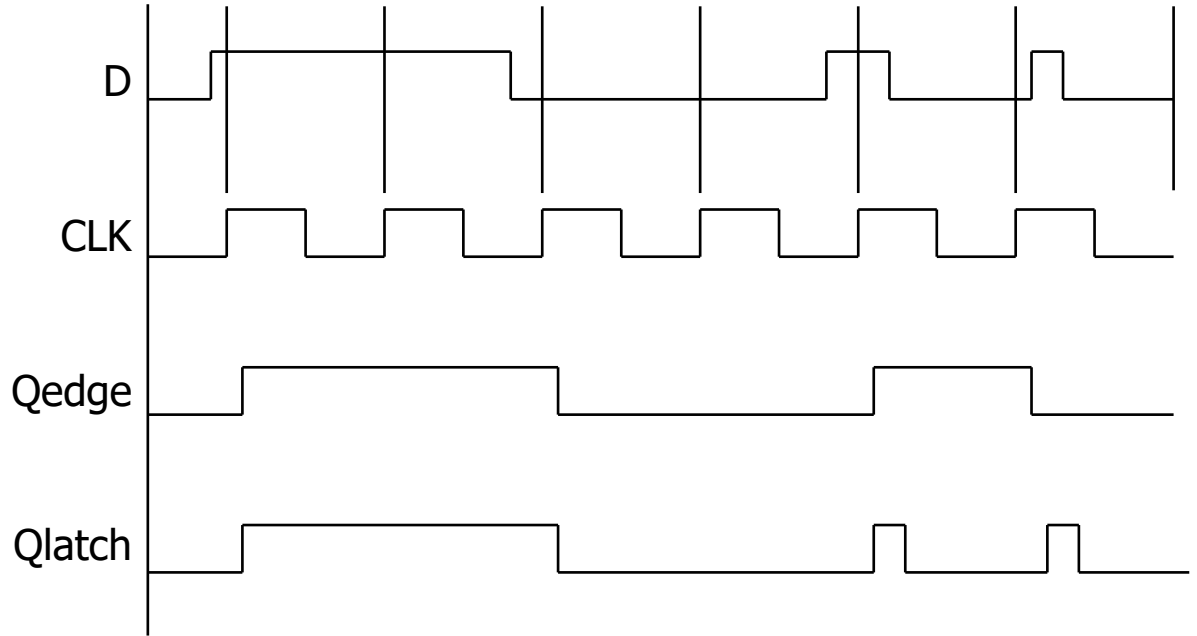
All measurements are made from the clocking event (the rising edge of the clock).



positive  
edge-triggered  
flip-flop



transparent  
(level-sensitive)  
latch



## Summary

- ✓ RS latch는 11의 입력은 사용하지 않는다.
- ✓ Latch 와 flip-flop의 근본적인 차이는 Latch 는 \_\_\_\_\_ 이고 Flip-flop은 \_\_\_\_\_ 이다.
- ✓ Master-slave flip-flop은 Gated latch의 Transparency 문제를 해결한다.
- ✓ D-flip-flop은 Master-slave flip-flop의 1's catching 문제를 해결한다.
- ✓ Flip-flop에 연관된 timing 요소 3가지는 \_\_\_\_\_, \_\_\_\_\_, \_\_\_\_\_ 이다.