DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

# Sistemi Digitali

## Introduzione alla Logica Sequenziale

Marta Bagatin, marta.bagatin@unipd.it

Corso di Laurea in Ingegneria dell'Informazione

Anno accademico 2022-2023

# Scopo della lezione

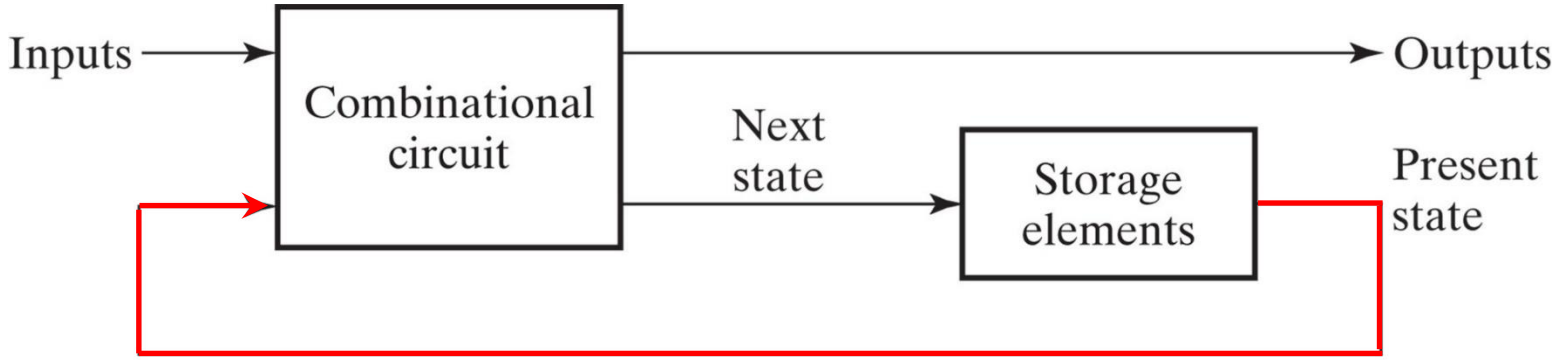
- Introdurre i circuiti sequenziali e il concetto di temporizzazione (sistemi sincroni vs. asincroni)
- Studiare i circuiti sequenziali elementari usati per immagazzinare informazioni
  - Latch di tipo SR
  - Latch di tipo D
  - Flip-flop di tipo D
- Capire i parametri per la temporizzazione dei flip-flop

# Circuito sequenziale: definizione

- E’ un circuito le cui **uscite dipendono non solo dagli ingressi correnti, ma anche dalla sua storia passata**
- A differenza di un circuito combinatorio, un circuito sequenziale **possiede memoria**
- Usando circuiti sequenziali possiamo realizzare sistemi capaci di immagazzinare informazioni tra un’operazione e la successiva

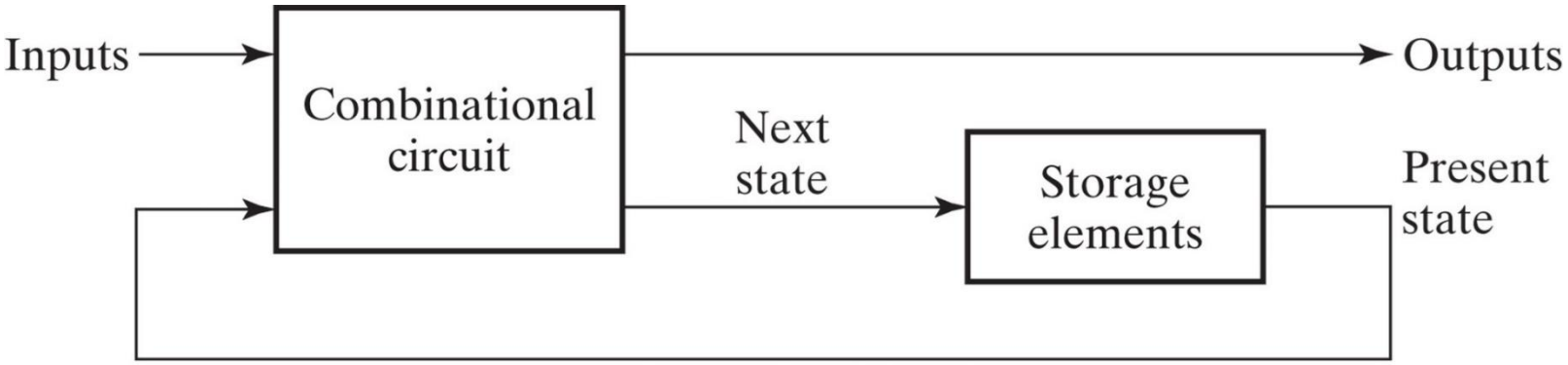
# Schema di un circuito sequenziale

- Un circuito sequenziale si realizza **connettendo un circuito combinatorio con uno o più elementi di memoria**




- Esiste un **percorso di retroazione (feedback)**: l'uscita dell'elemento di memoria è riportata in ingresso al sistema
  - Nota: i circuiti combinatori sono invece aciclici

# Schema di un circuito sequenziale



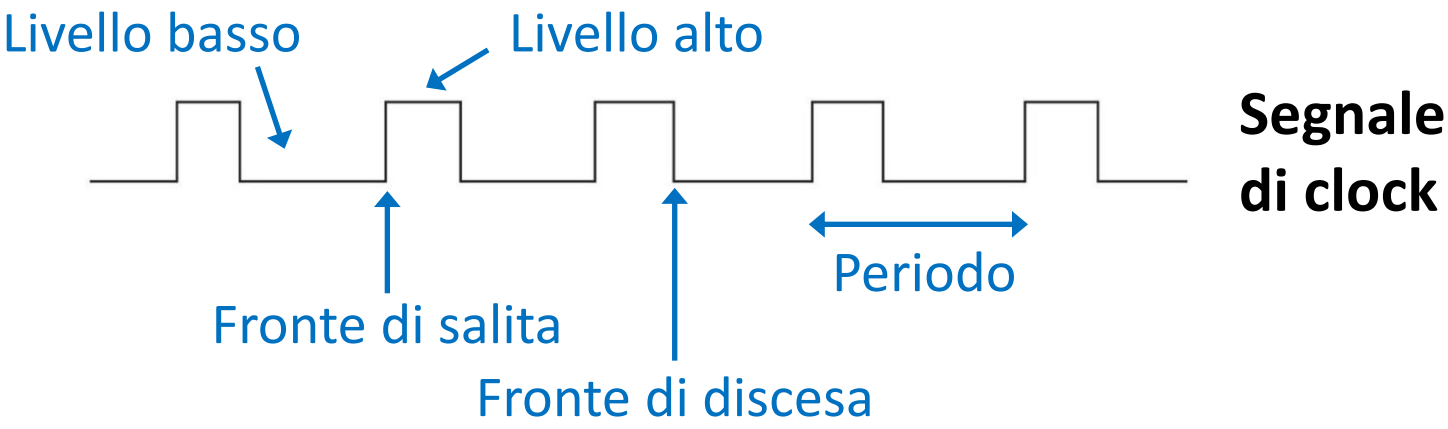

- **Uscite e stato futuro sono calcolati dalla parte combinatoria, a partire da ingressi e stato presente**
- Lo **stato presente** è dato dall'insieme dei valori in uscita degli elementi di memoria
- Lo **stato futuro** è dato dall'insieme dei valori in ingresso agli elementi di memoria

# Circuiti sincroni e asincroni

- Un circuito sequenziale si definisce sincrono o asincrono, a seconda degli **istanti di tempo in cui vengono valutati i suoi ingressi e aggiornato il suo stato**
- **Circuito sequenziale sincrono (macchina a stati finiti, finite state machine FSM)**: i cambiamenti nello stato del sistema sono sincronizzati con un segnale detto **clock**
  - Il comportamento di un circuito sincrono è determinato dalla conoscenza dei suoi segnali a istanti di tempo discreti
- **Circuito sequenziale asincrono**: non è regolato da clock, lo stato può cambiare in qualsiasi istante di tempo
- Gran parte dei dispositivi digitali moderni usano circuiti sincroni. I circuiti asincroni possono avere dei vantaggi (velocità, consumo), ma sono generalmente più complessi e meno robusti

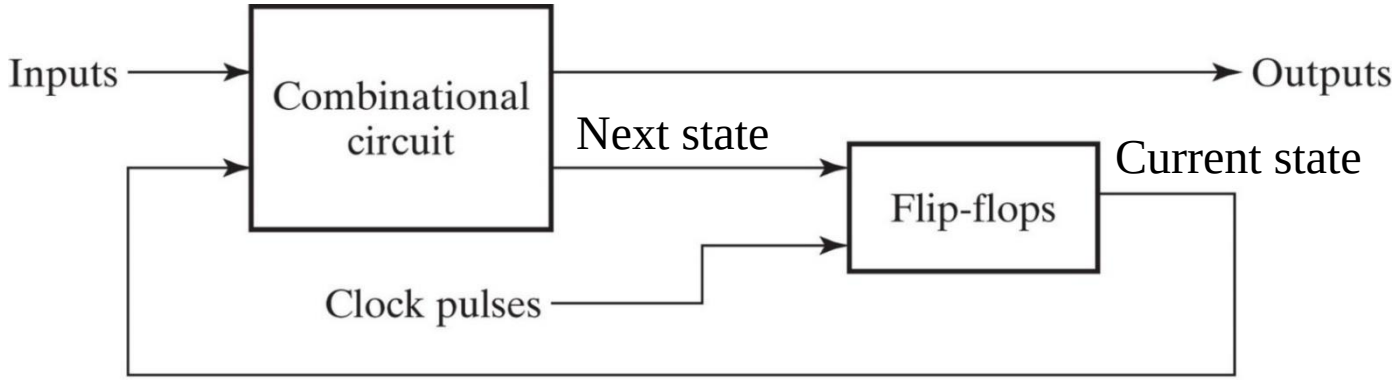
# Circuiti sequenziali sincroni

- La **temporizzazione è data da un segnale periodico detto clock**, costituito da un'onda quadra

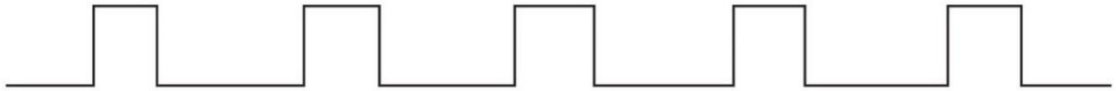



- Il segnale di clock è distribuito all'interno del sistema in modo che tutti gli elementi di memoria vengano aggiornati solo in presenza di determinati eventi sul clock (e.g. fronte di salita)
- Sono i circuiti sequenziali più usati, più semplici da progettare e più affidabili

# Circuiti sequenziali sincroni



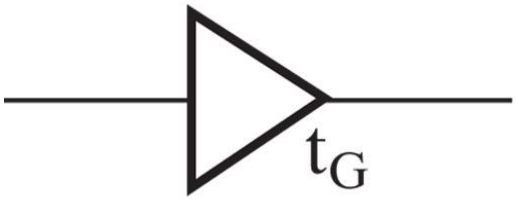

(a) Block diagram



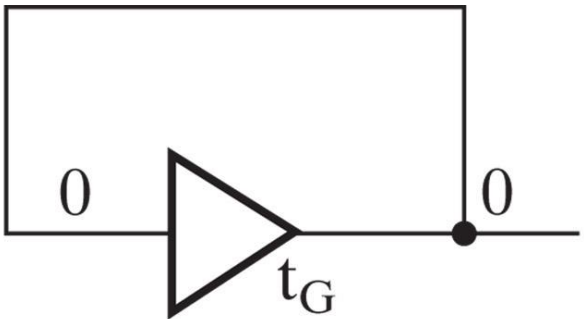
(b) Timing diagram of clock pulses

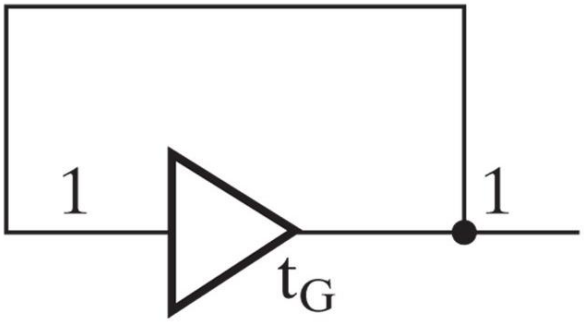
- Il segnale di **clock costituisce uno degli ingressi degli elementi di memoria**, che nei circuiti sequenziali sincroni sono chiamati flip-flop
- L'**uscita dei flip-flop può cambiare solo in corrispondenza a determinate variazioni** del clock (per es. al fronte di salita)
- Ad ogni ciclo di clock, lo stato corrente avanza allo stato futuro

# Elementi di memoria



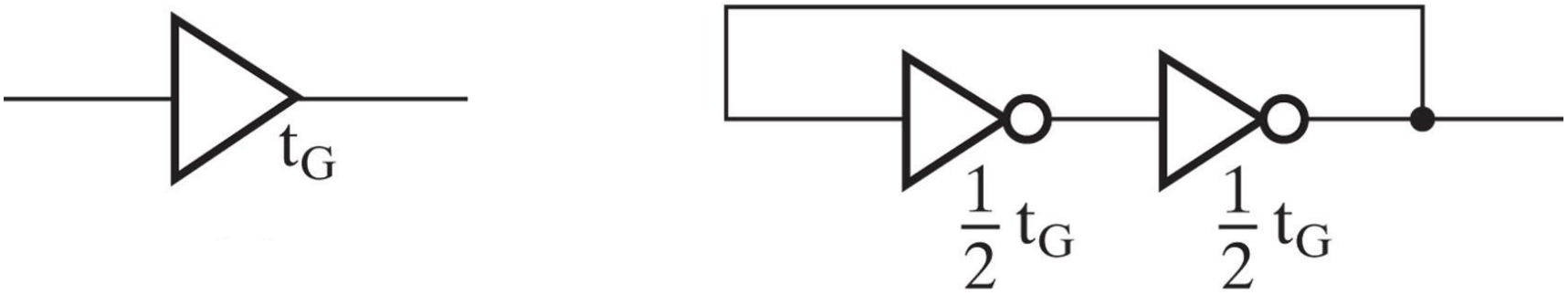

- Buffer con ritardo $t_G$: un dato presentato all'ingresso al tempo t appare in uscita al tempo $t + t_G$




- Aggiungiamo un feedback: colleghiamo l'uscita del buffer al suo ingresso
- Se l'ingresso è costante ('0' o '1') per un tempo pari ad almeno $t_G$, al tempo $t + t_G$ l'uscita del buffer sarà ancora stabile a quel valore
- Abbiamo realizzato un elemento di memoria!

# Elementi di memoria




- Il modo più comune di realizzare un elemento di memoria è connettere **due inverter in cascata** (ciascuno con tempo di ritardo pari a 1/2 $t_G$), in modo da negare due volte il segnale e ottenere in uscita il segnale originario:

$$\overline{\overline{X}} = X$$

- Nelle slide successive vedremo altri modi per realizzare elementi di memoria con ingressi addizionali che permettono di controllare il valore dei dati immagazzinati

# Latch e flip-flop

# Flip-flop

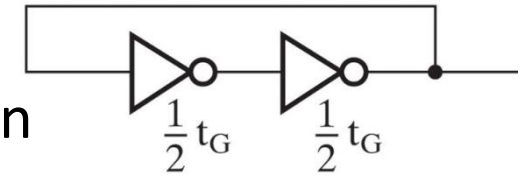
- Gli elementi di memoria usati nei circuiti sequenziali sincroni sono detti **flip-flop**
- Il flip-flop è un **elemento di memoria binario capace di immagazzinare un singolo bit e temporizzato da un clock**
  - Input: segnale di ingresso e clock
  - Output: segnale di uscita e (opzionale) sua negazione
- I flip-flop sono **costituiti da elementi chiamati latch**

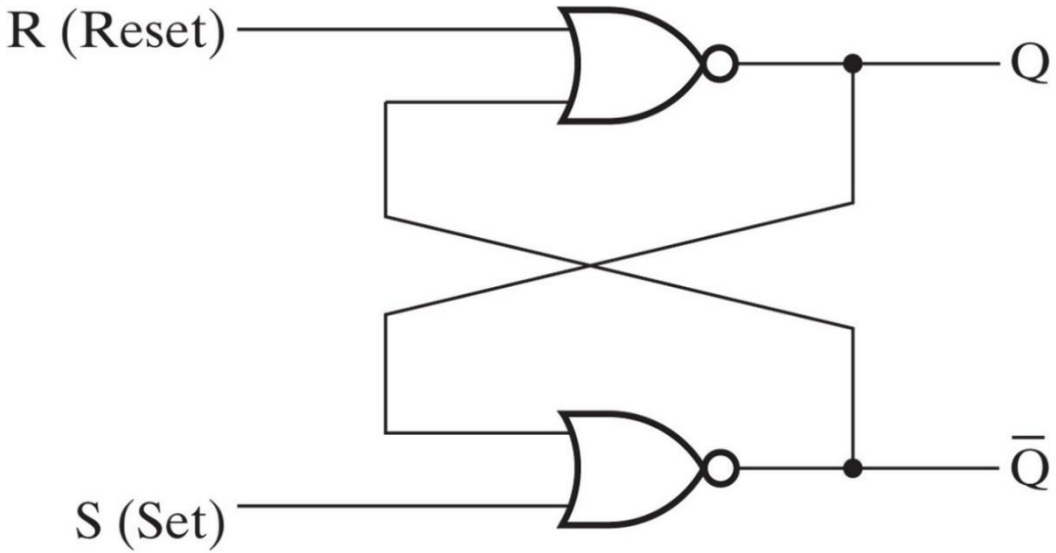
# Latch

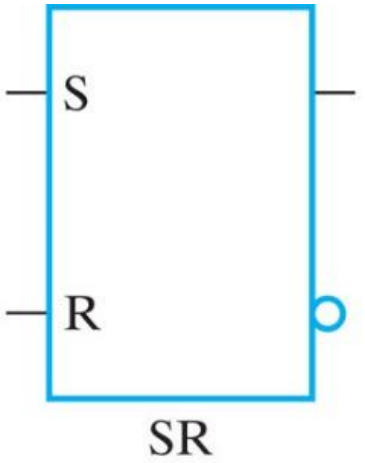
- Il latch è l'**elemento primitivo** con cui vengono realizzati i circuiti sequenziali
- E' un **circuito bistabile** (ha due stati stabili) che memorizza un bit di informazione, non temporizzato cioè non regolato da clock

# Latch SR

Il **latch Set Reset (SR)** è derivato dalla più semplice struttura di memoria con due inverter in cascata, sostituendo agli inverter **due porte NOR**

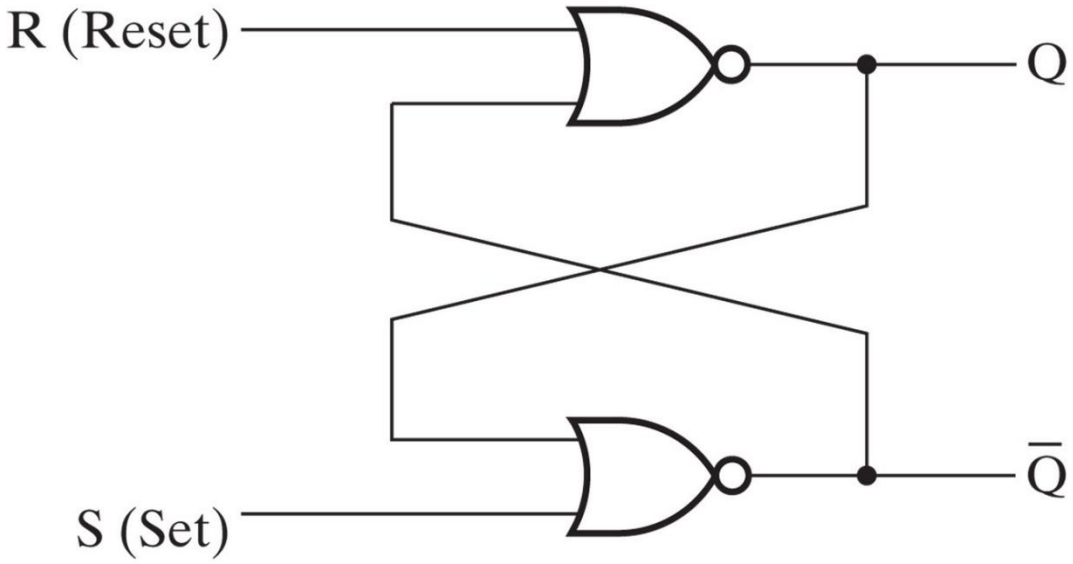


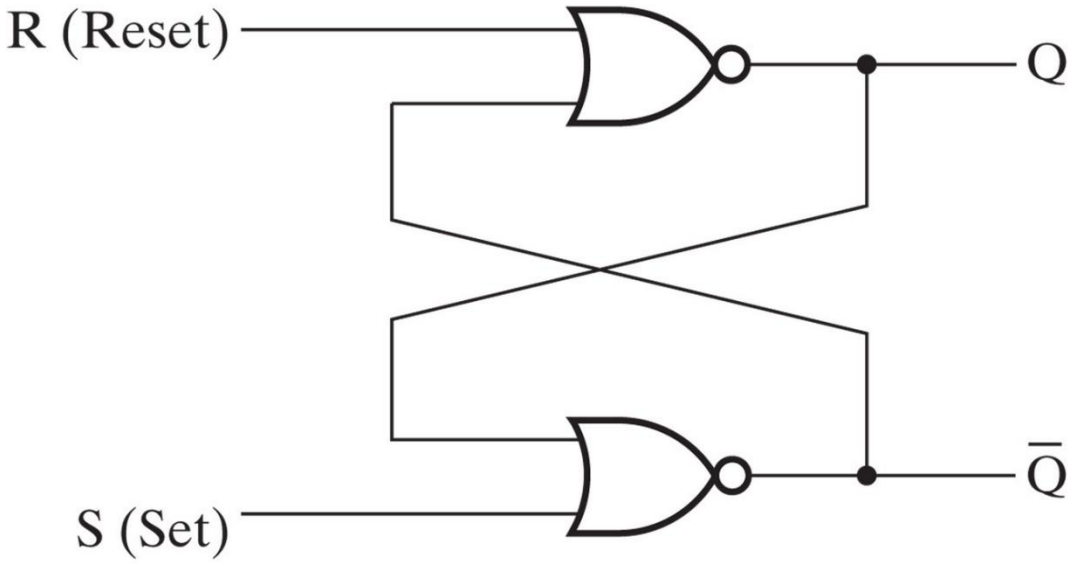

Cambiando il valore degli ingressi R e S possiamo forzare dall'esterno il valore del bit da immagazzinare nel latch

# Latch SR: Funzionamento




Porta NOR: basta che un ingresso sia '1' perché l'uscita sia '0'

# Latch SR: funzionamento e tabella caratteristica

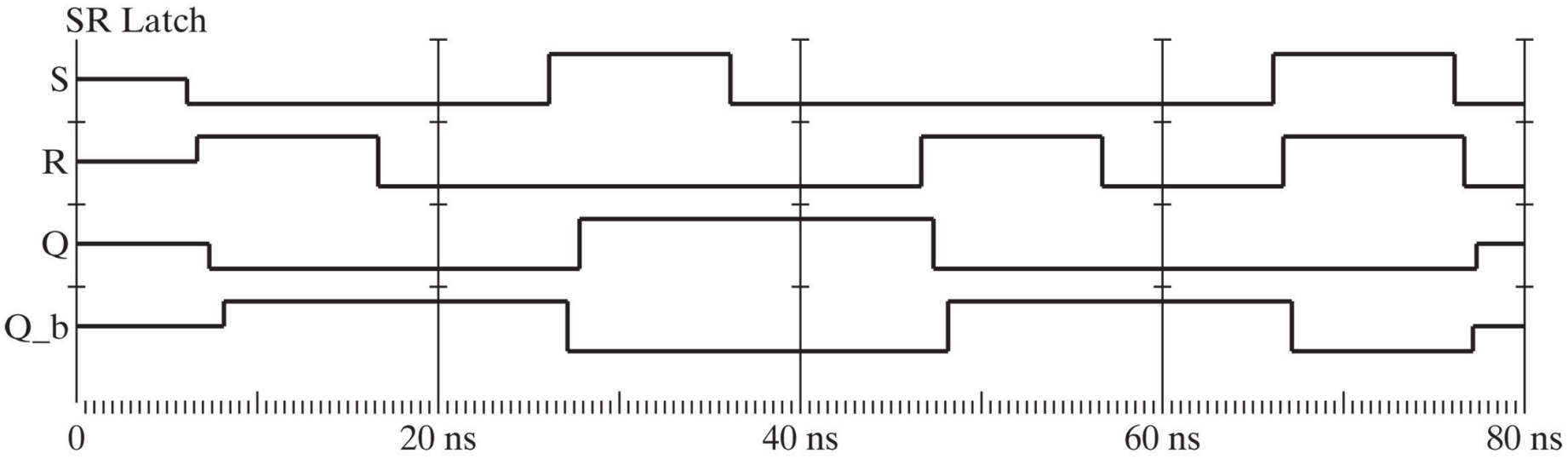



| S | R | Q | $\overline{Q}$ | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | Set state |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | Reset state |
| 0 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 0 | Undefined |

Porta NOR: basta che un ingresso sia '1' perché l'uscita sia '0'

- **S = '1', R = '0'**: $\overline{Q} = 0,\ Q = 1$ => stato di **set**
- **R = '1', S = '0'**: $Q = 0,\ \overline{Q} = 1$ => stato di **reset**
- **S = R = '0'**: stato di **memoria**, il circuito mantiene lo stato precedente
- Durante il normale funzionamento, $Q$ e $\overline{Q}$ sono uno il complemento dell'altro
- **S = R = '1'**: stato **proibito** (entrambi $Q$ e $\overline{Q}$ vanno a '0', violando il requisito che un'uscita sia complementare dell'altra) che porta ad oscillazioni e metastabilità

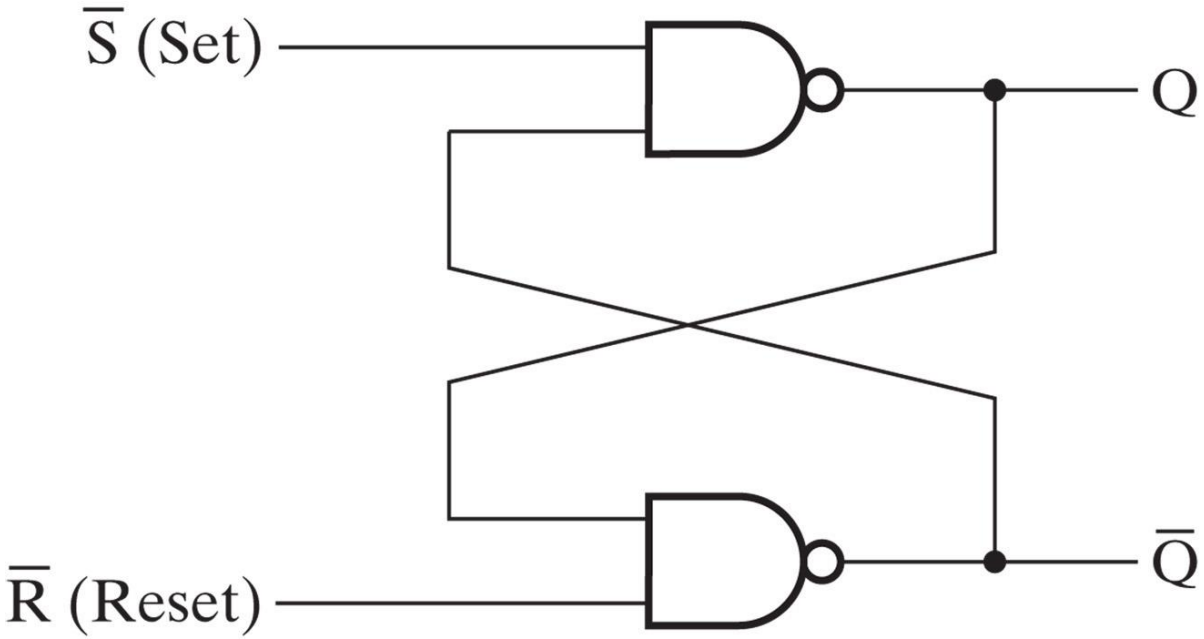
# Latch SR: Diagramma temporale

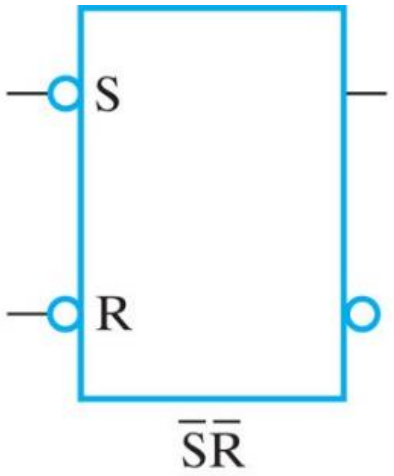



- Simulazione logica dell'andamento temporale delle uscite Q e $\overline{Q}$ (Q_b) in risposta a variazioni degli ingressi S e R
- Quando S = R = 1, entrambi Q e $\overline{Q}$ vanno a 0. Alla successiva transizione S = R = 0, l'uscita si porta ad uno stato intermedio tra '0' e '1'
  - E' la **simulazione** di una situazione ideale, in cui le commutazioni di S, R avvengono simultaneamente e le due porte NOR hanno lo stesso tempo di ritardo
  - Nella **realtà** il valore delle uscite dopo la sequenza di ingressi (S,R) da (1,1) a (0,0) dipenderà dall'ordine con cui commutano S e R e dai ritardi delle porte NOR

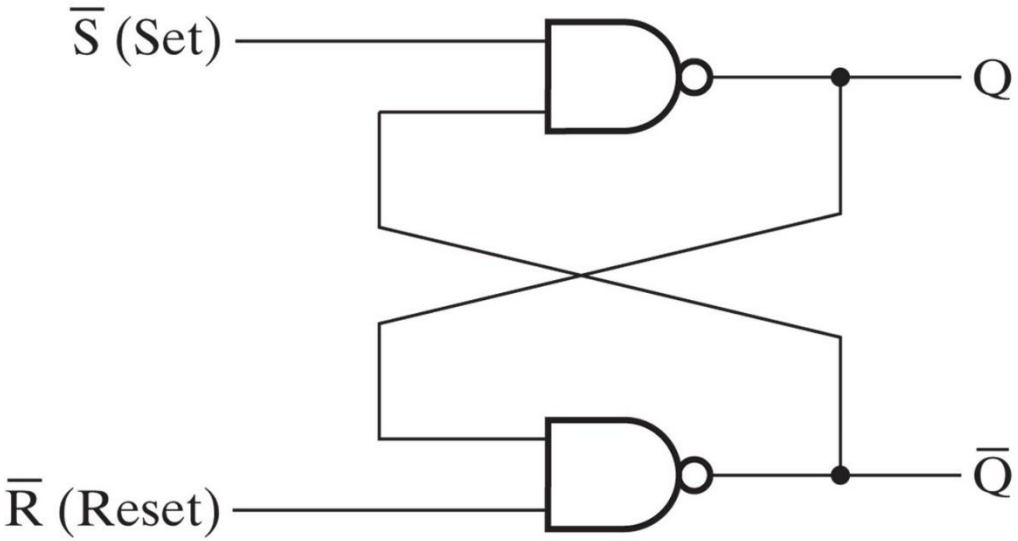
# Latch $\overline{S}$ $\overline{R}$

- Si ottiene dal latch Set Reset (SR), sostituendo le due porte NOR con **due porte NAND**



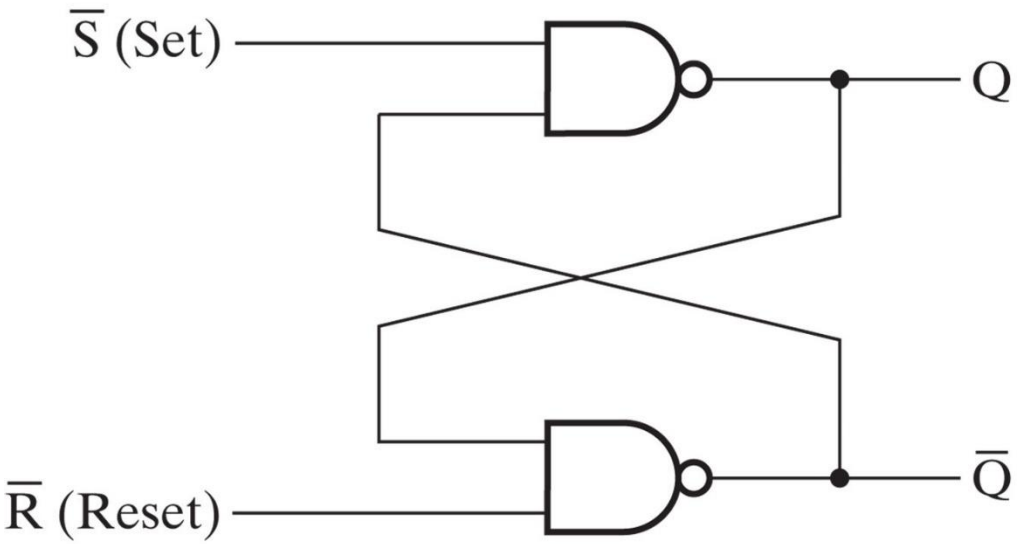



# Latch $\overline{S}$ $\overline{R}$: Funzionamento




Porta NAND: basta che un ingresso sia '0' perché l'uscita sia '1'

# Latch $\overline{S}$ $\overline{R}$: funzionamento e tabella caratteristica



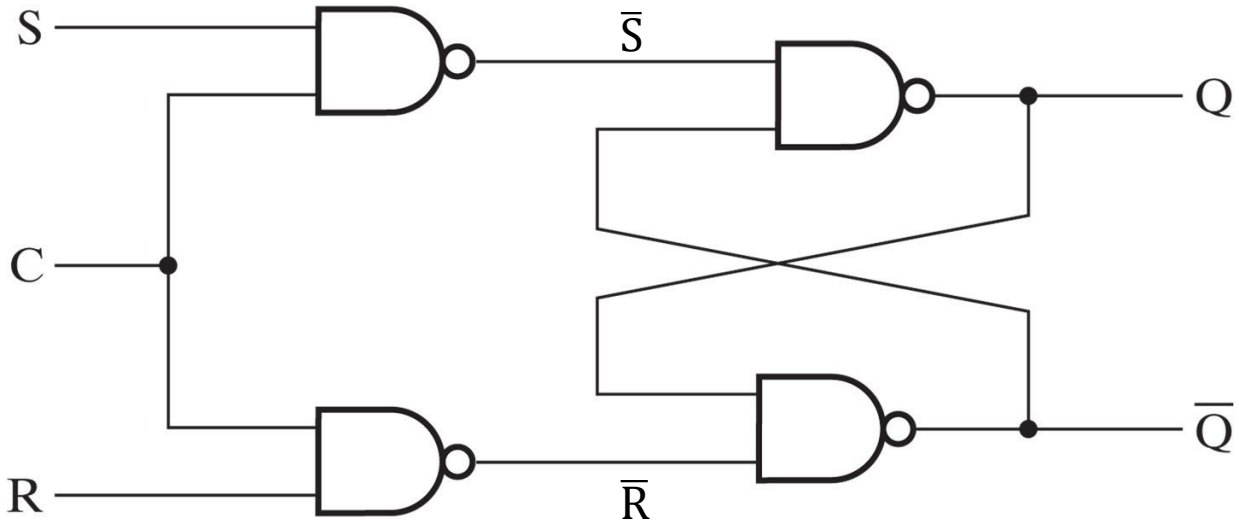

| $\overline{S}$ | $\overline{R}$ | Q | $\overline{Q}$ | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | Set state |
| 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | Reset state |
| 1 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 1 | Undefined |

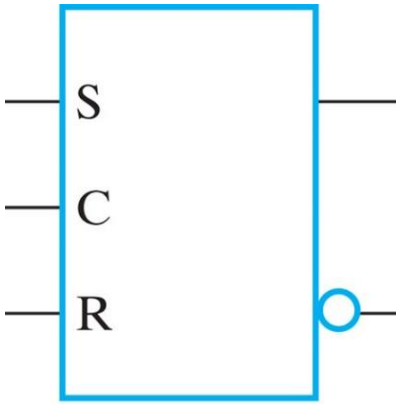
Porta NAND: basta che un ingresso sia '0' perché l'uscita sia '1'

- $\bar{\mathbf{S}} = '\mathbf{0}', \bar{\mathbf{R}} = '\mathbf{1}'$: $Q = 1,\ \overline{Q} = 0$, => stato di **set**
- $\bar{\mathbf{R}} = '\mathbf{0}', \bar{\mathbf{S}} = '\mathbf{1}'$: $\overline{Q} = 1,\ Q = 0$, => stato di **reset**
- $\bar{\mathbf{S}} = \bar{\mathbf{R}} = '\mathbf{1}'$: stato di **memoria**, il circuito mantiene lo stato precedente
- $\bar{\mathbf{S}} = \bar{\mathbf{R}} = '\mathbf{0}'$: stato **proibito** (entrambi Q e $\overline{Q}$ vanno a '1', nella successiva transizione di $\bar{\mathbf{S}}$ e $\bar{\mathbf{R}}$ ad $'\mathbf{1}'$, Q e $\overline{Q}$ assumeranno un valore indefinito)

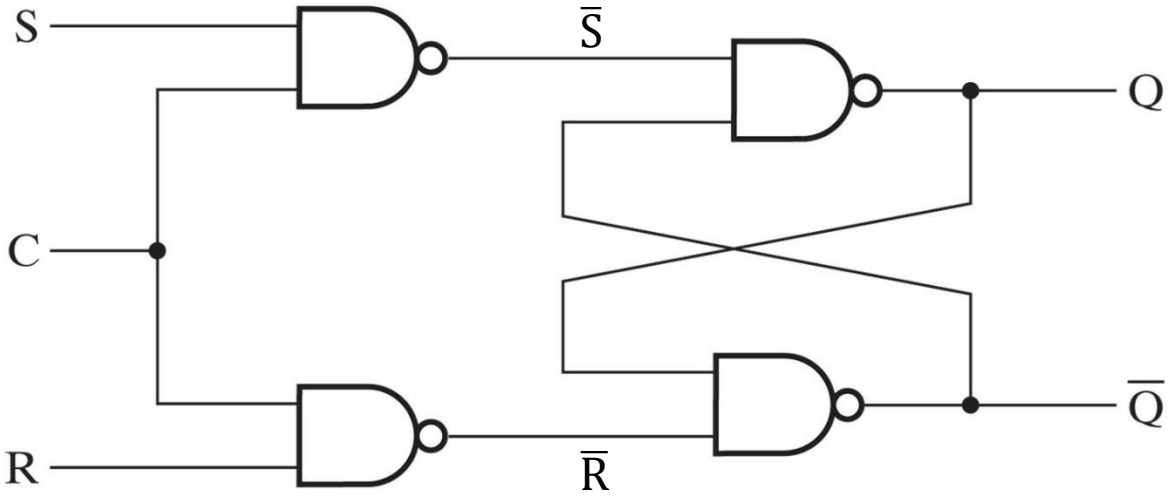
# Latch SR con ingresso di controllo

- Rispetto al latch SR con porte NAND, aggiunge un ulteriore **ingresso di controllo C**, che funge da **segnale di enable**, abilitando o meno il cambio dello stato del latch



# Latch SR con ingresso di controllo

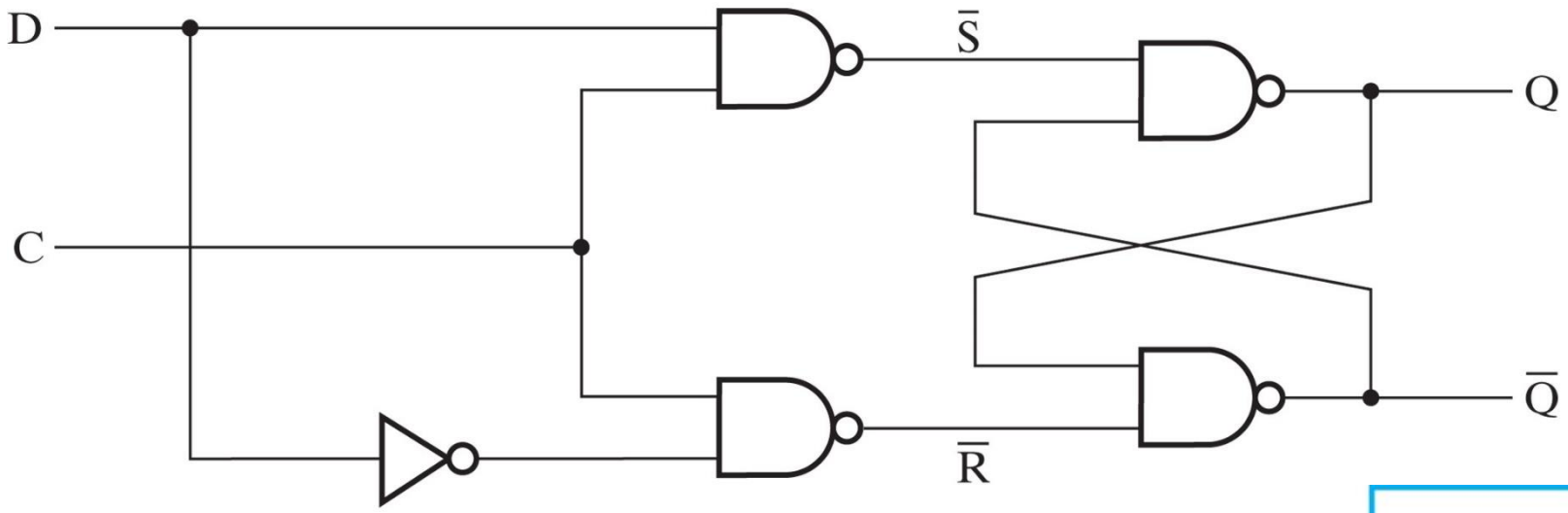


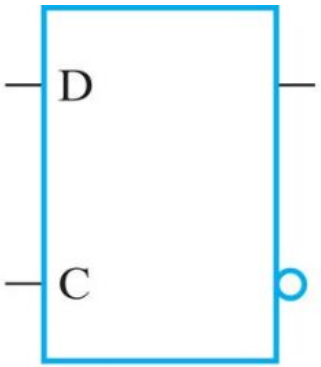

| C | S | R | Next state of Q |
|---|---|---|---|
| 0 | X | X | No change |
| 1 | 0 | 0 | No change |
| 1 | 0 | 1 | Q = 0; Reset state |
| 1 | 1 | 0 | Q = 1; Set state |
| 1 | 1 | 1 | Undefined |

- **C = ‘0’: latch non abilitato** (quiescent state), mantiene lo stato precedente, indipendentemente dai valori degli ingressi S e R
- **C = ‘1’: latch funziona come un latch SR**
  - **C = ‘1’, S = R = ‘0’**: stato di **memoria**
  - **C = ‘1’, S = ‘0’, R = ‘1’**: stato di **reset** => Q = ‘0’
  - **C = ‘1’, S = ‘1’, R = ‘0’**: stato di **set** => Q = ‘1’
  - **C = S = R = ‘1’**: stato proibito (entrambe le uscite vanno a ‘1’)

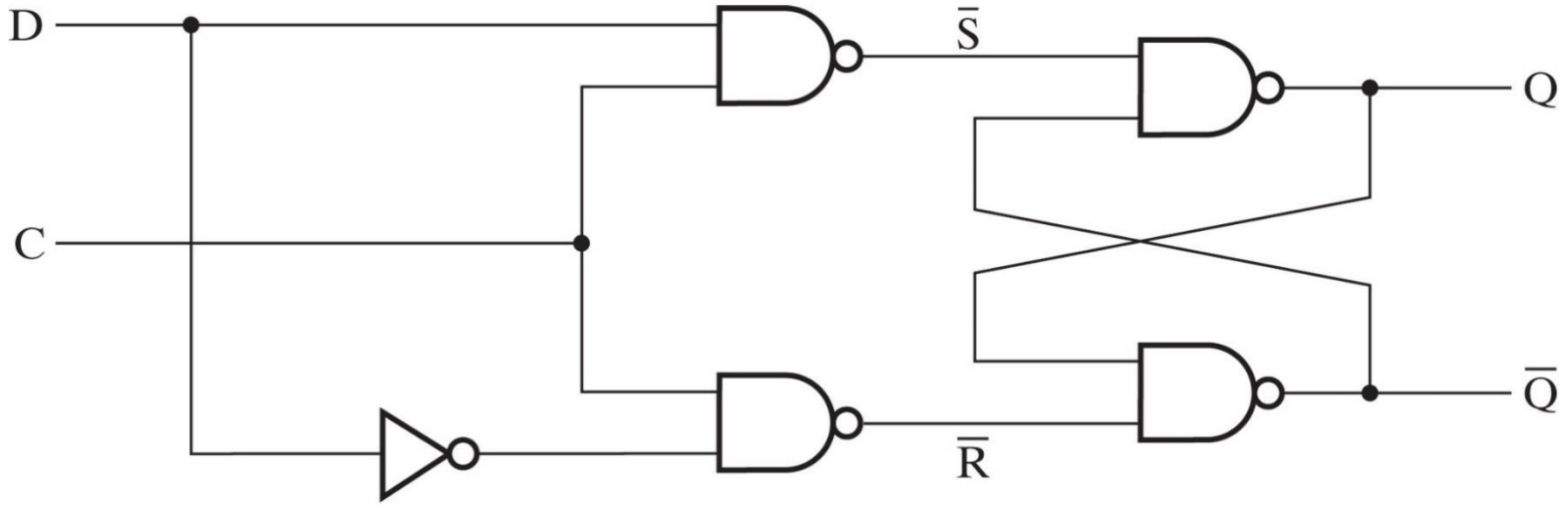
# Latch D

- Il latch D **evita il problema dello stato proibito**, evitando la condizione che $\overline{S}$ e $\overline{R}$ siano pari ad ‘1’ simultaneamente
  - 2 ingressi: Dato (D), Controllo (C)



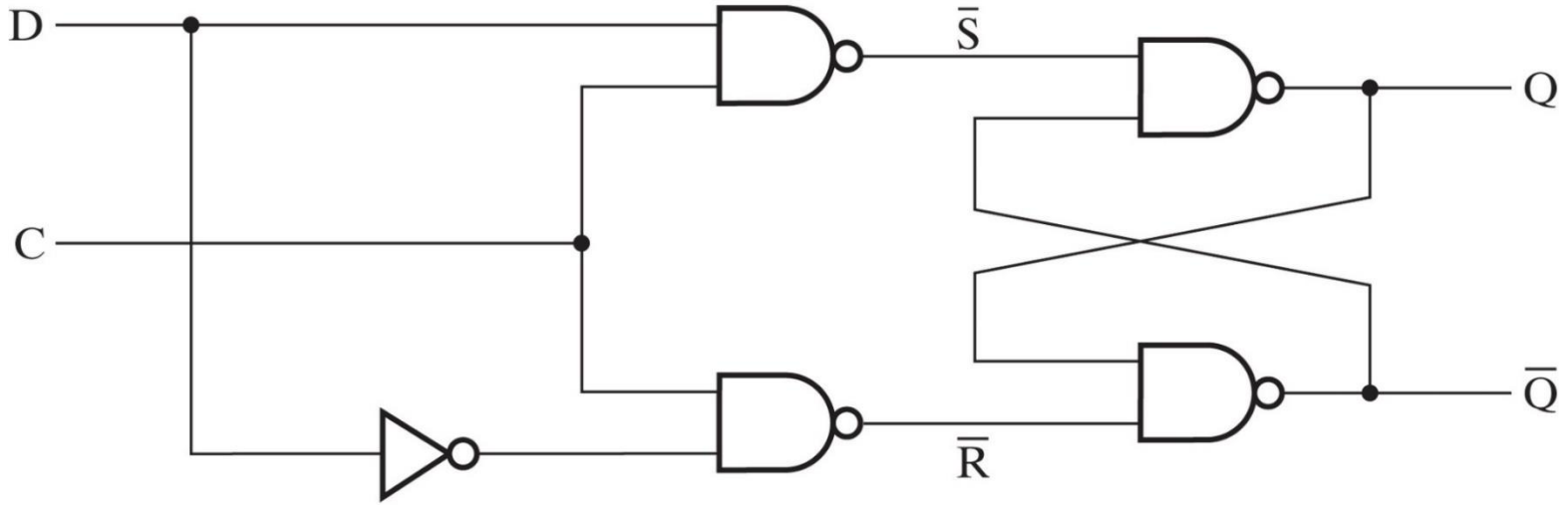
# Latch D: funzionamento e tabella caratteristica




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

- C = ‘0’: il circuito mantiene lo stato precedente, qualunque sia il valore dell’ingresso D
- C = ‘1’: D viene trasferito all’uscita Q
  - C = ‘1’, D = ‘0’: stato di reset => Q = ‘0’
  - C = ‘1’, D = ‘1’: stato di set => Q = ‘1’

# Trasparenza

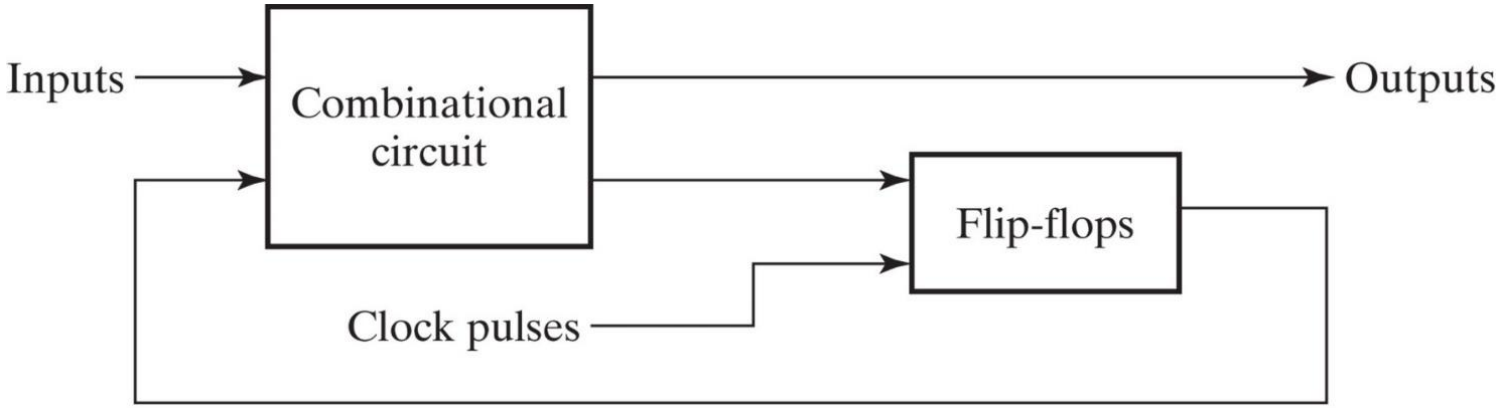



- Il cambiamento dell'ingresso di controllo abilita il cambiamento nell'uscita del latch, cioè agisce da trigger
- Quando C = 1, l'uscita del latch riflette ogni cambiamento del dato D in ingresso al latch: il latch è **trasparente**
- Il latch **è sensibile al livello logico dell'ingresso di controllo**

# Trigger

- Chiamiamo «**trigger**» (=innesco) un **cambiamento del valore di un ingresso che abilita i cambiamenti del valore dell'uscita** in un elemento di memoria

# Requisiti di un flip-flop

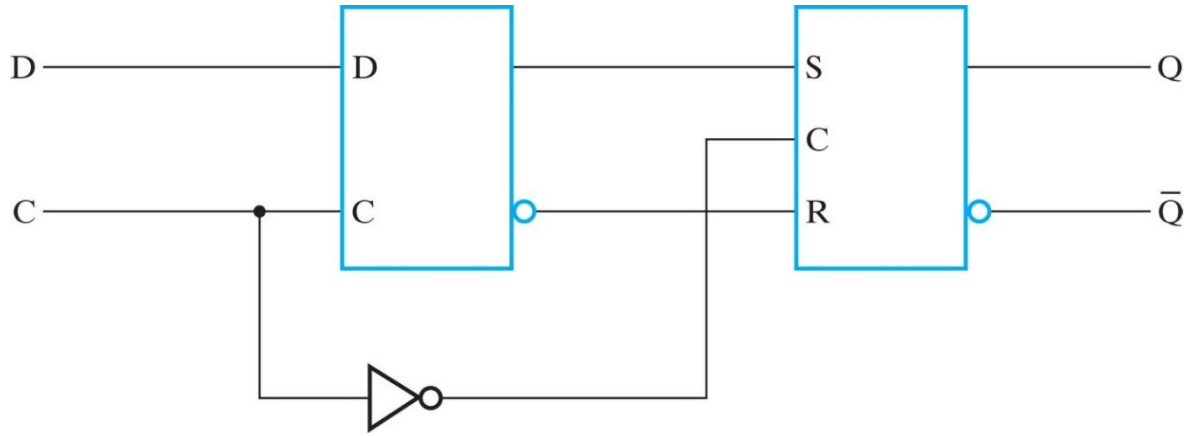



- Il feedback comporta che l'ingresso del flip-flop dipende in parte dalla sua uscita. Per operare correttamente in un circuito sincrono regolato da clock, un flip-flop **NON deve essere trasparente**: dalla sua uscita non si devono vedere i cambiamenti del suo ingresso all'interno dello stesso periodo di clock (ma solo tra due periodi di clock, al fronte di salita)
  - Se così non fosse, avremmo cambiamenti multipli indesiderati in uscita del flip-flop. In altre parole vogliamo che il nuovo stato dipenda SOLO dallo stato immediatamente precedente

  => Tra due fronti di clock consecutivi lo stato di un flip-flop è quindi insensibile al livello logico dei suoi ingressi

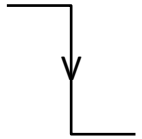
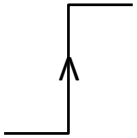
# Come costruire un flip-flop con i latch

- Un modo per realizzare un flip-flop è connettere in cascata due latch, con il latch di sinistra (**Master**) triggerato dal clock e il latch di destra (**Slave**) triggerato dal complementare del clock



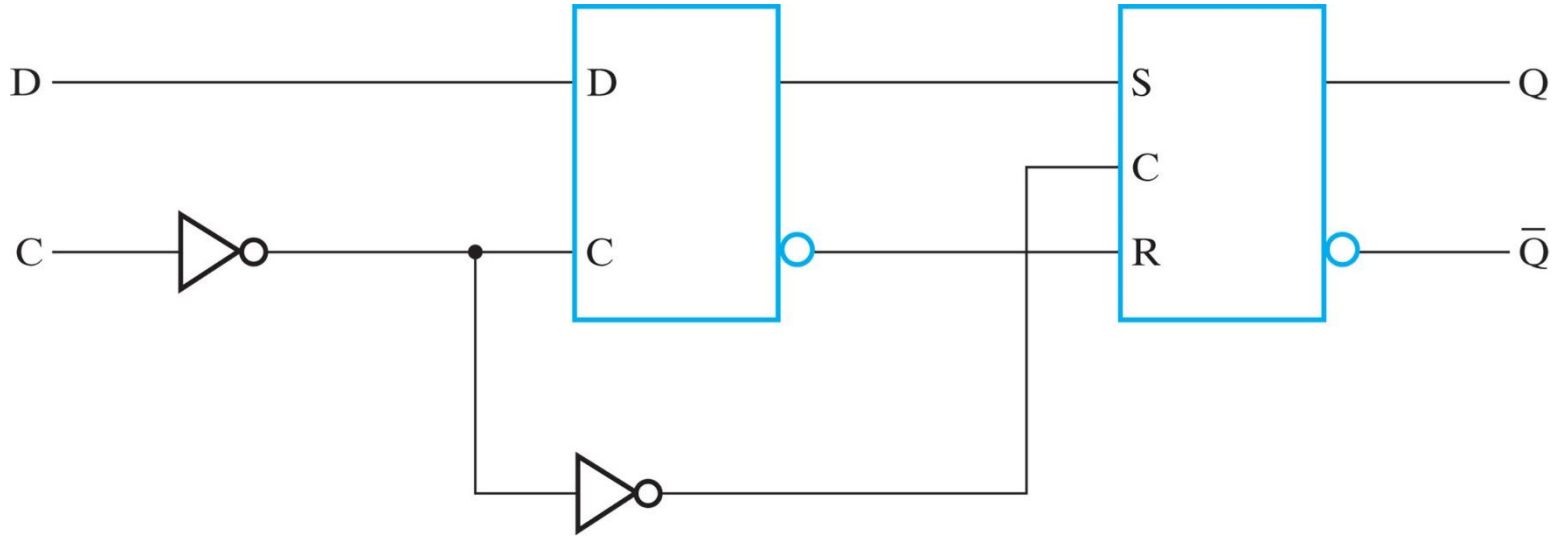

- **Non è una struttura trasparente**: Q può cambiare solo al successivo ciclo di clock rispetto a quando è avvenuto un cambiamento di D
- A seconda dei tipi di latch che usiamo, possiamo ottenere
  - Flip-flop sensibili al livello del clock
  - Flip-flop sensibili alle transizioni del clock (noi vedremo solo questi!)

# Flip-flop sensibili ai fronti (edge-triggered)

- Possono **cambiare il loro stato** solo in corrispondenza ad una **variazione (=fronte) del clock da '0' a '1' o viceversa**, mentre sono disabilitati in tutti gli altri istanti di tempo (es. quando il clock è alto)
- Sono più usati rispetto ai flip-flop sensibili al livello di clock, perché più veloci e più semplici da progettare
- Si dividono in
  - **Negative-Edge-Triggered**: sensibili ai fronti di discesa del clock 
  - **Positive-Edge-Triggered**: sensibili ai fronti di salita del clock 

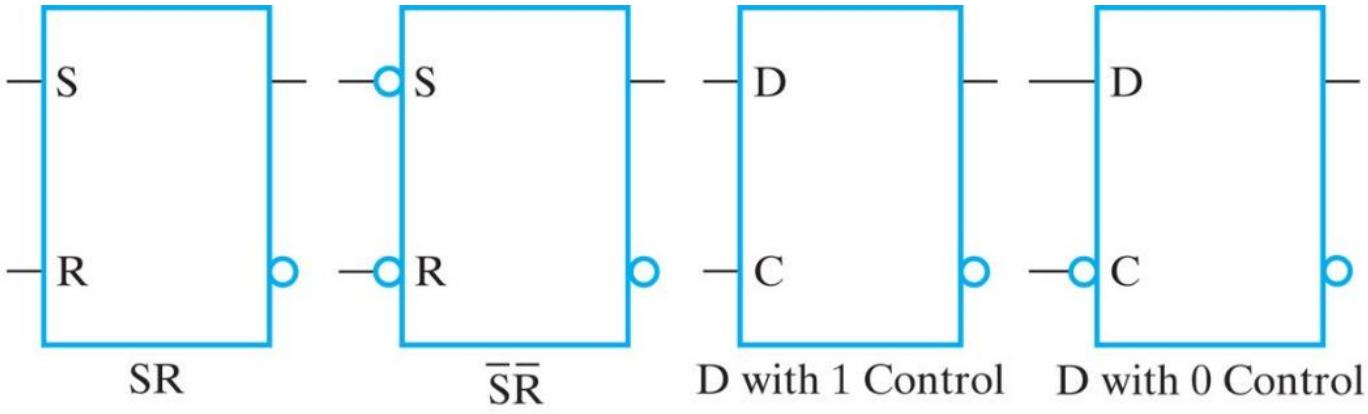
# Positive edge-triggered D Flip-flop

- Un flip-flop sensibile ai fronti di salita si ottiene connettendo **due latch in cascata**, con il clock negato che pilota l'ingresso di controllo del **Master** e il clock che pilota l'ingresso di controllo dello **Slave**

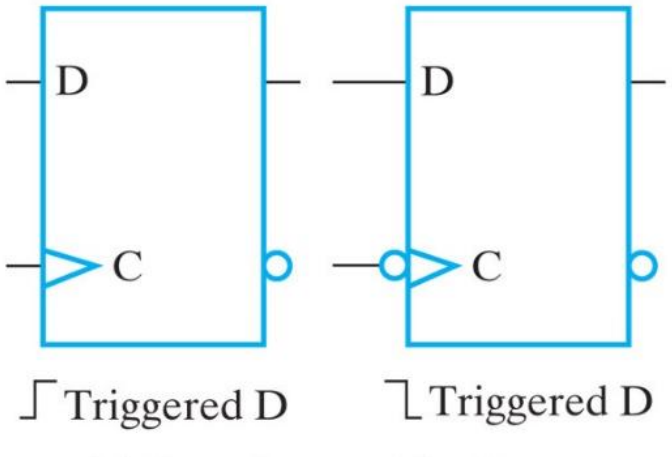



- Il master è abilitato quando C = 0, lo slave è abilitato quando C = 1
- Fintanto che C = 0: il master trasferisce il valore di D all'ingresso Set dello slave, mentre lo slave è disabilitato e mantiene lo stato precedente
- Al fronte di salita del clock, lo slave viene abilitato e l'ultimo valore di D campionato dal master viene trasferito all'uscita Q

# Simbologia: Latch e Flip-flop edge triggered



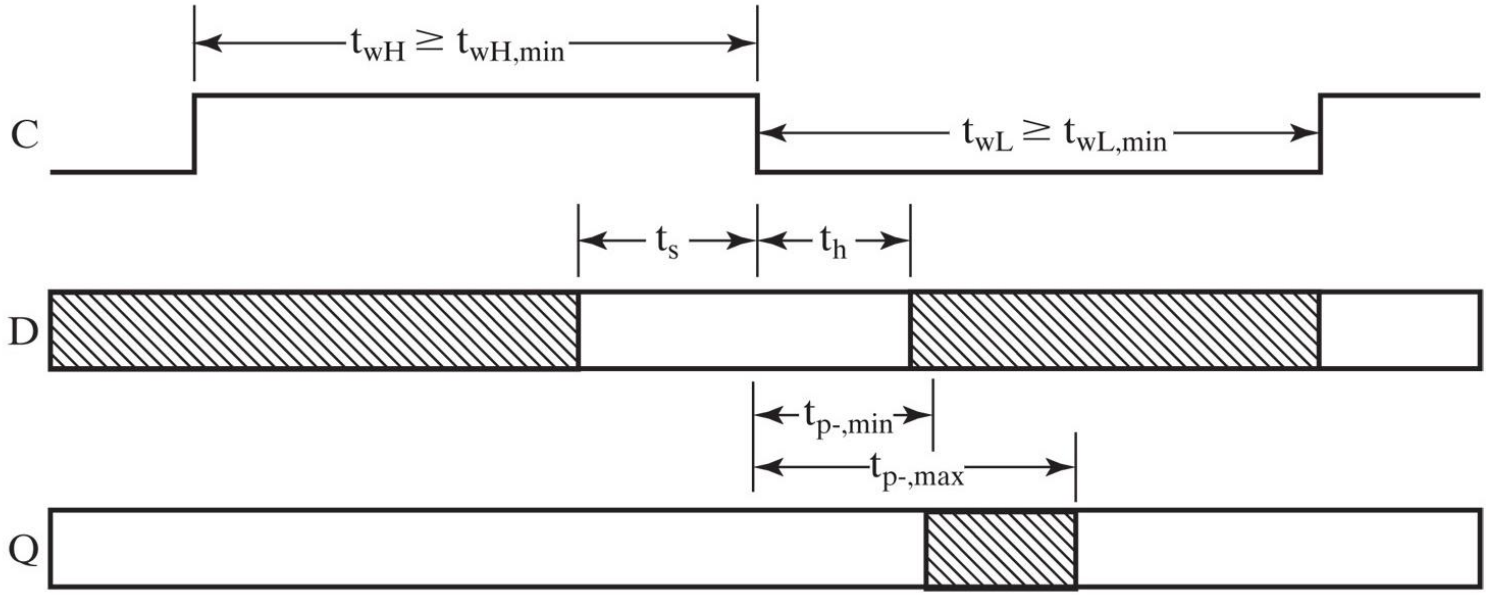

Latch



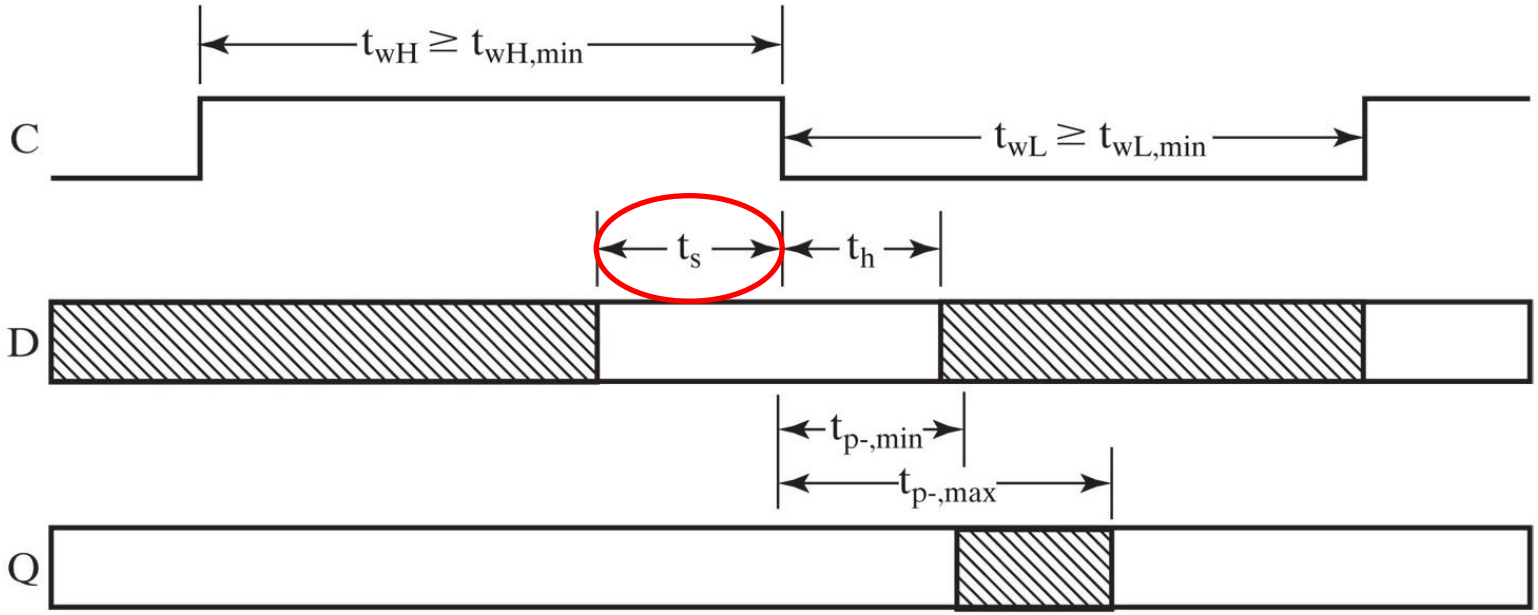

Flip-flop edge triggered

# Temporizzazione dei Flip-flop

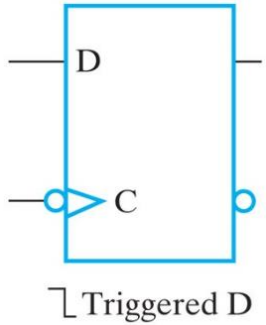
- Per il corretto funzionamento di un flip-flop, devono essere rispettati dei **vincoli di temporizzazione**. In caso contrario non è garantito che vengano memorizzati i valori corretti



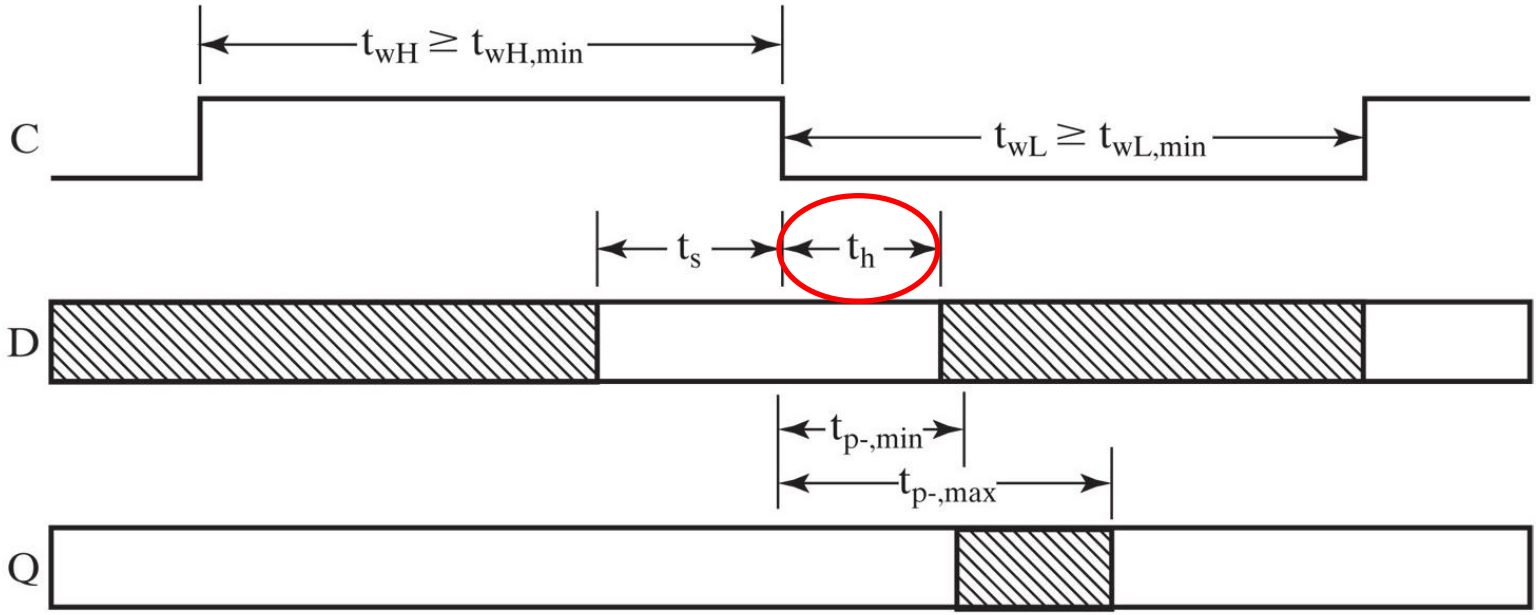
# Flip-flop: Tempo di setup



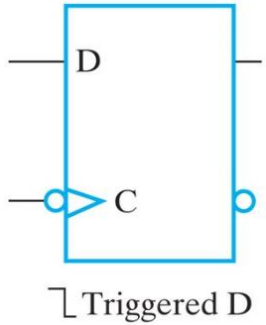

- **Tempo di setup ($t_s$)**: tempo minimo per cui gli **ingressi devono essere stabili al valore desiderato, prima del fronte di clock** che causa un cambiamento nell'uscita
  - Altrimenti il master potrebbe trovarsi ad un livello intermedio quando il suo valore viene trasferito allo slave



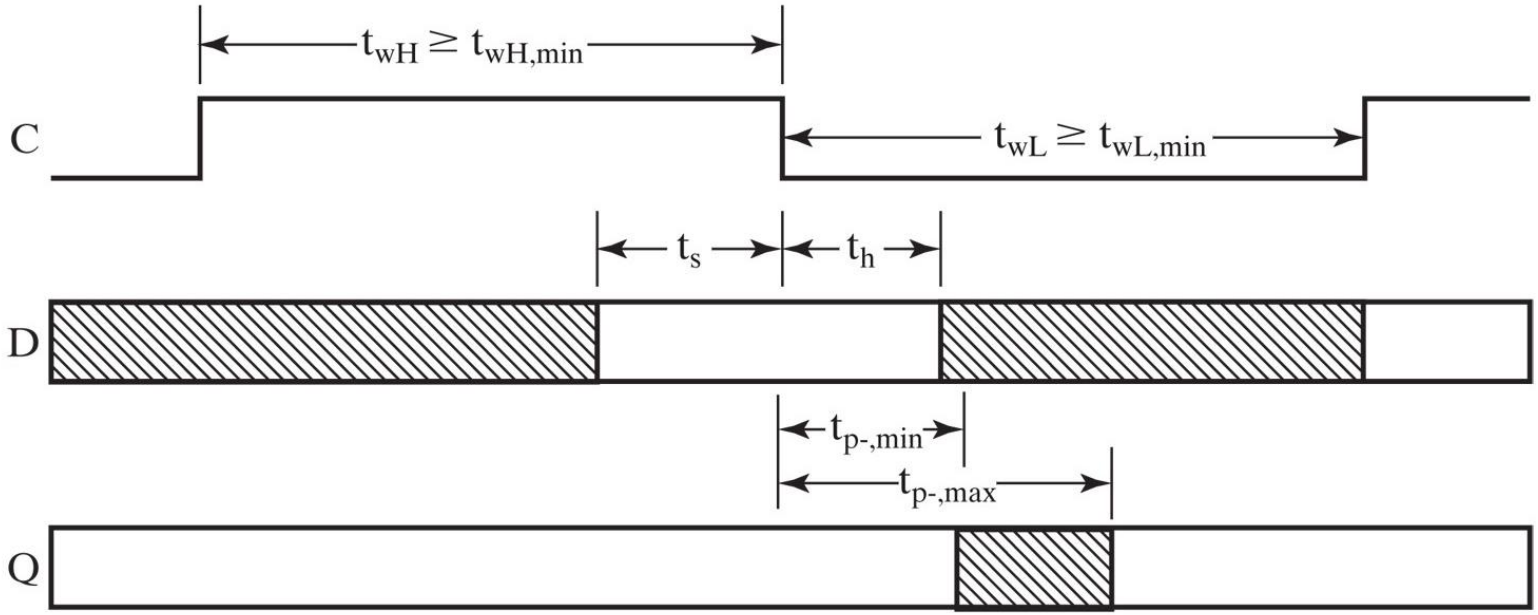
# Flip-flop: Tempo di hold



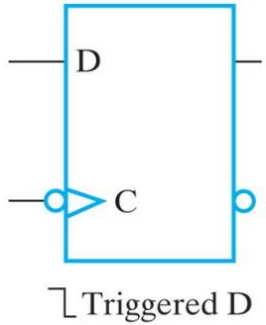

- **Tempo di hold ($t_h$)**: tempo minimo per cui gli **ingressi devono essere stabili dopo il fronte del clock che causa un cambiamento nell'uscita**
  - Altrimenti il master potrebbe reagire al cambiamento negli ingressi e cambiare il suo valore proprio mentre questo viene trasferito allo slave



# Flip-flop: Tempo di propagazione

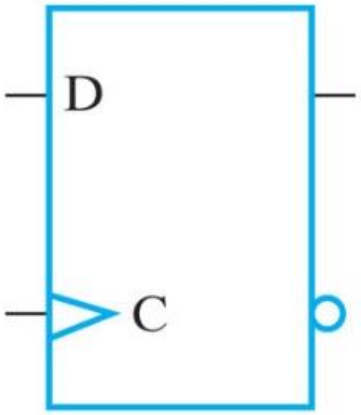



- **Tempo di propagazione** ($t_p$): intervallo di tempo **tra la transizione di clock che funge da trigger e l'istante in cui l'uscita si stabilizza**



# Flip-flop positive-edge-triggered

- E’ preferibile che tutti i flip-flop all’interno di un circuito siano dello stesso tipo, in modo da reagire allo stesso modo in presenza delle medesime variazioni del clock
- Da qui in avanti, se non specificato diversamente, assumeremo che tutti i flip-flop siano di **tipo D positive-edge-triggered**




Triggered D

# Una nota sulla terminologia

- Nel 2020, in seguito all'ondata internazionale di proteste **antirazziste** per la morte di George Floyd, colossi della tecnologia IT, tra cui **Google, Microsoft, Apple, Twitter, hanno dichiarato di voler sostituire la terminologia «Master Slave»**
- Linus Torvalds (Linux) ha indicato delle linee guida con alcune **proposte per sostituire i termini master/slave**:
  - primary/secondary
  - main/replica o subordinate
  - initiator/target
  - requester/responder
  - controller/device
  - host/worker o proxy
  - leader/follower
  - director/performer

# Riepilogo

- In un **sistema sequenziale** le uscite dipendono non solo dagli ingressi correnti, ma anche dalla storia passata (= stato del sistema). Un sistema sequenziale si ottiene aggiungendo degli **elementi di memoria** ad un sistema combinatorio
- Un circuito sequenziale **sincrono** è temporizzato da un **segnale di clock**: le variabili di stato possono cambiare solo in corrispondenza a determinati eventi nel clock
- Il **latch** è l'elemento sequenziale di base non regolato da clock (è sensibile al livello logico dei segnali d'ingresso)
- Il **flip-flop** è costituito da latch ed è regolato da clock (è un circuito sensibile ad eventi sul segnale di clock, e.g. abbiamo visto flip-flop sensibili ai fronti del clock)
- Per funzionare in modo corretto, un flip-flop deve rispettare alcuni vincoli di temporizzazione (tempo di setup e tempo di hold)

# Disclaimer

Figures from *Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano | Kime | Martin