Corso di Calcolatori Elettronici I
A.A. 2012-2013

Elementi di memoria

ing. Alessandro Cilardo

Accademia Aeronautica di Pozzuoli
Corso Pegaso V “GArn Elettronici”

Reti logiche “con memoria”

- In molte situazioni è necessario progettare reti logiche **sequenziali**, ovvero *dotate di memoria*:
  - l’uscita in un dato istante non è funzione soltanto del valore degli ingressi applicati in quell’istante, ma anche di tutti i valori applicati negli istanti precendenti

- Per costruire reti sequenziali, abbiamo bisogno di circuiti elementari che permettano di introdurre “memoria” all’interno delle reti

Reti logiche “con memoria”

- Reti in grado di **conservare uno stato** possono essere realizzate a partire da reti combinatorie tramite una “retroazione”

(*il triangolo denota un circuito “buffer”*)

- La retroazione mantiene un valore ‘0’ o ‘1’ fisso sull’uscita
  - se ad esempio nel punto **A** è presente il valore 0, questo ritorna in ingresso e, negato due volte, si “autosostiene” in uscita
  - In questo semplice schema, non è però possibile decidere dall’esterno il valore dello stato, ovvero il valore memorizzato

Latch RS con NOR

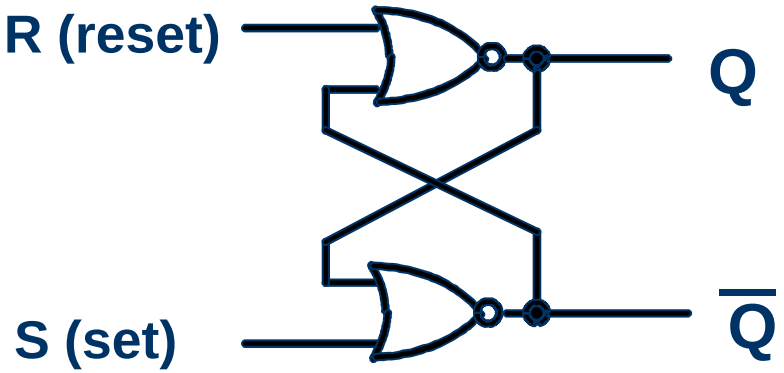
- Sostituendo le NOT con delle NOR è possibile cambiare lo stato agendo sugli ingressi esterni **R** (reset) ed **S** (set)

Quando **R**=**0** ed **S**=**0**, le due NOR si comportano infatti come delle NOT, ed il circuito equivale alla coppia di NOT in retroazione vista prima

Quando invece uno dei due segnali diventa **1**, forza le uscite ad un determinato stato (ad esempio **S**=**1** forza **Q**=**1** e $\overline{\mathbf{Q}}$ =**0**)

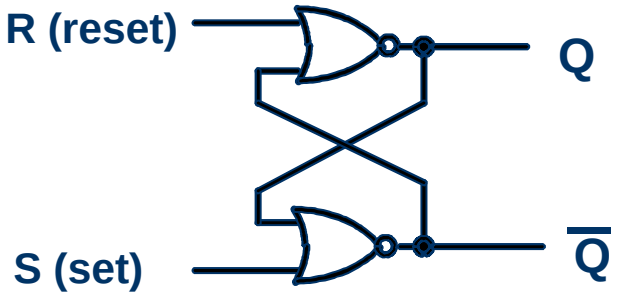
Appena i due ingressi **R** ed **S** ridiventano **0**, la rete permane nello stato imposto dall'ultimo ingresso applicato, ovvero lo ***memorizza***

Latch RS



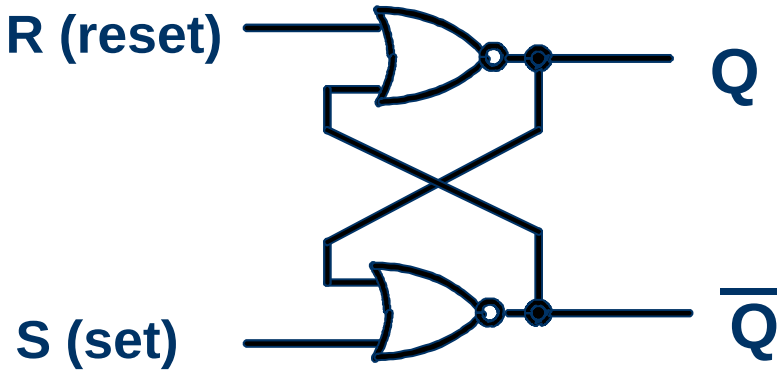
Latch RS con NOR

- Se gli ingressi esterni **R** (reset) ed **S** (set) diventano entrambi **1** contemporaneamente, le uscite sono entrambe forzate a **0**, rendendo quindi non coerenti le uscite **Q** e $\overline{\mathbf{Q}}$ (non sono più di valore opposto)
- Inoltre, se gli ingressi passano insieme a **0**, il circuito ridiventa equivalente ad una coppia di NOT in retroazione, in cui però, per un istante, **entrambe le NOT** hanno sia in uscita sia in ingresso il valore **0**
- Le NOT portano quindi ad 1 il valore delle loro uscite, che ritornando in ingresso causa un nuova transizione a **0**, etc etc…..
- Il circuito oscilla per un certo tempo non predicibile (situazione di **metastabilità**), fino a portarsi in una configurazione stabile, ma aleatoria, per i valori di **Q** e $\overline{\mathbf{Q}}$



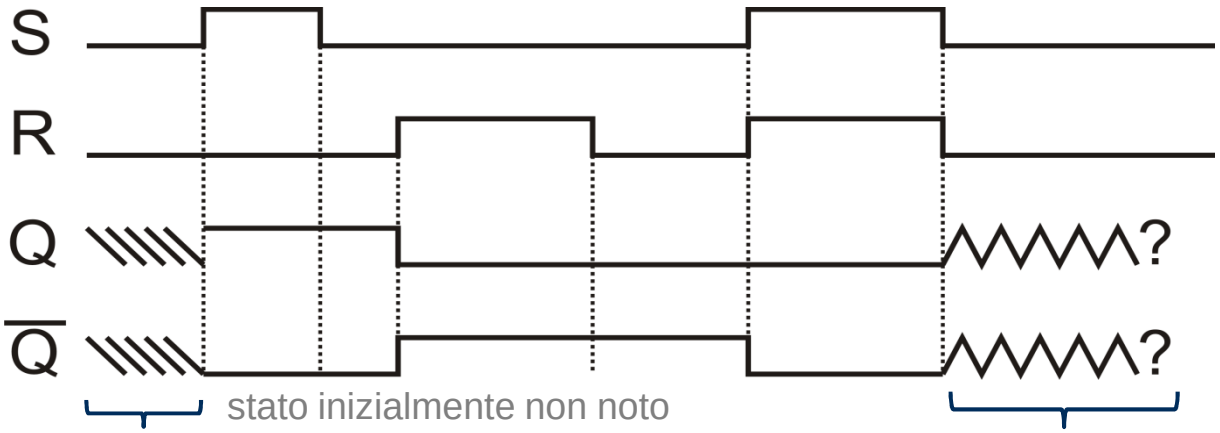
Latch RS

Latch RS con NOR



tempo

R	S	Q	$\overline{Q}$	*…cosa succede…*
0	0	?	?	stato Q precedente (non noto)
0	1	1	0	“setta” Q a 1
0	0	1	0	Q “ricorda” 1
1	0	0	1	“resetta” Q a 0
0	0	0	1	Q “ricorda” 0
1	1	0	0	uscite entrambe basse, non coerenti
0	0	?	?	metastabilità



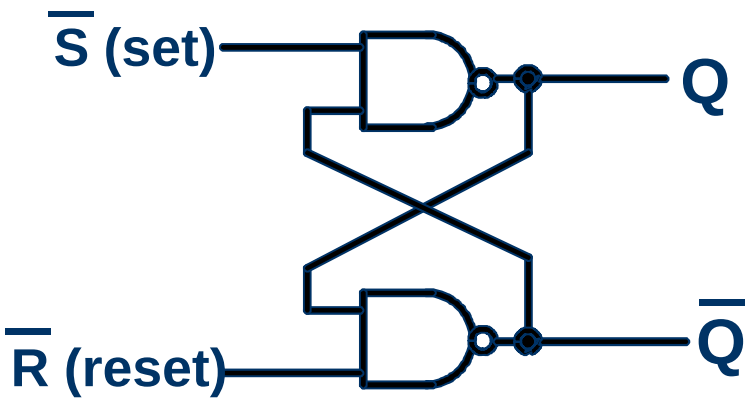
metastabilità (**RS** passano insieme da **11** a **00**)

Latch RS con NAND

Usando delle NAND è possibile ottenere lo stesso comportamento, usando però i segnali **R** ed **S** in forma negata

In questo caso, il valore neutro di **S** ed **R** è **1**

La modifica dello stato avviene quando uno dei due diventa **0**

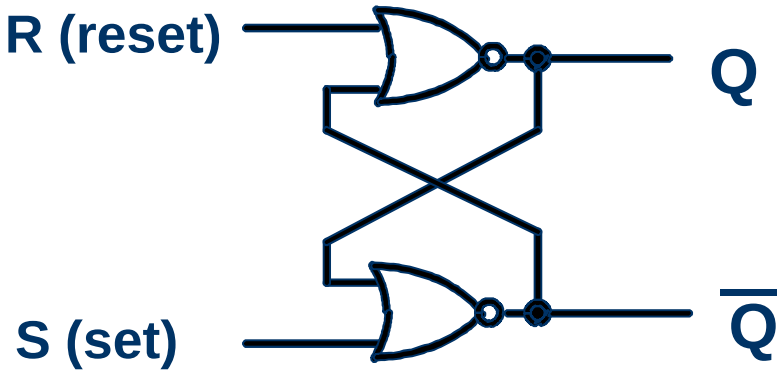


tempo

R	S	Q	$\overline{Q}$	*…cosa succede…*
1	1	?	?	stato Q non noto
1	0	1	0	“setta” Q a 1
1	1	1	0	Q “ricorda” 1
0	1	0	1	“resetta” Q a 0
1	1	0	1	Q “ricorda” 0
0	0	1	1	uscite entrambe alte, non coerenti
1	1	?	?	metastabilità

Latch RS: tabella funzionale

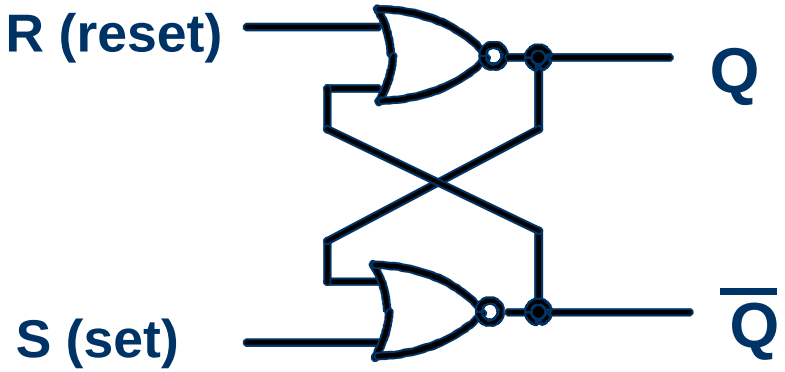
tabella funzionale: indica quale sarà il prossimo stato **Q(t+1)**, per ogni combinazione di stato corrente **Q(t)** ed ingressi **R** ed **S**



Q(t)	S	R	Q(t+1)	Commento
0	0	0	0	inalterato
0	0	1	0	resetta Q
0	1	0	1	setta Q
0	1	1	???	*non ammessa*
1	0	0	1	inalterato
1	0	1	0	resetta Q
1	1	0	1	Setta Q
1	1	1	???	*non ammessa*

Latch RS: tabella di eccitazione

tabella di eccitazione: per ciascuna transizione da un qualsiasi valore dello stato corrente **Q(t)** ad un qualsiasi valore dello stato prossimo **Q(t+1)**, indica la combinazione degli ingressi **R** ed **S** da applicare per determinare la transizione desiderata

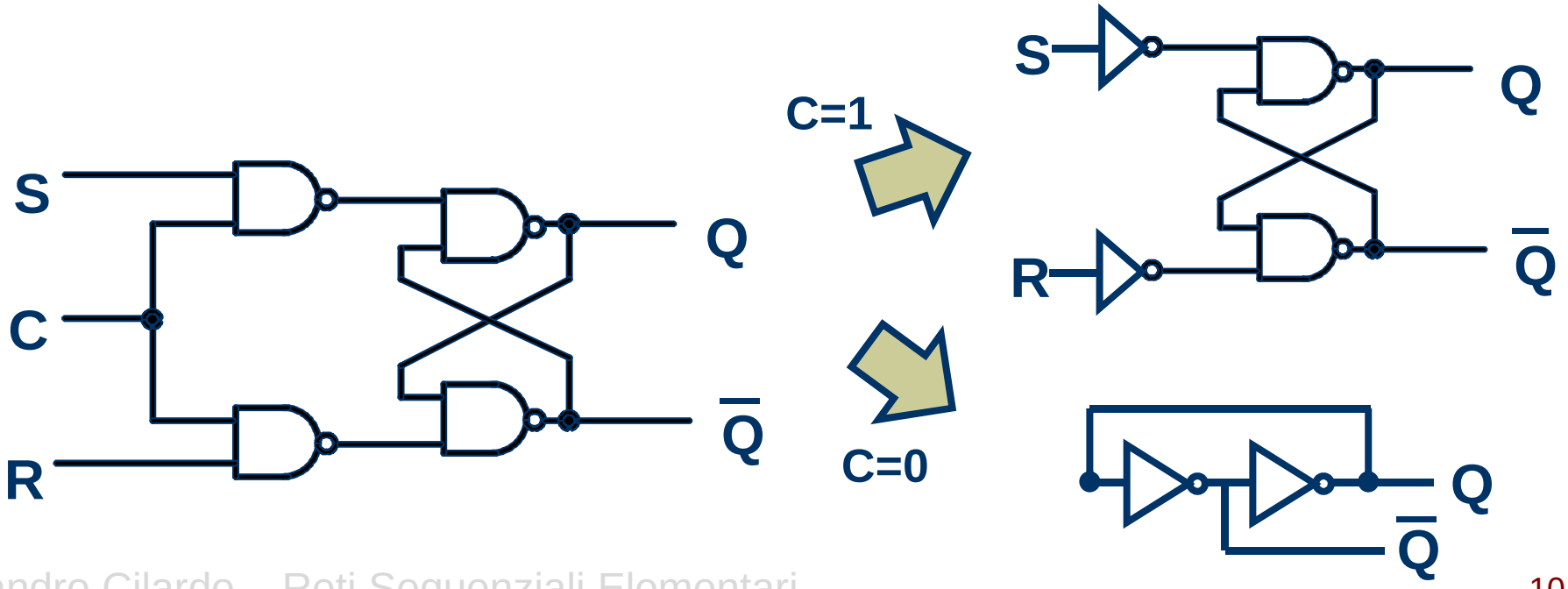


Q(t)	Q(t+1)	R	S
0	0	-	0
0	1	0	1
1	0	1	0
1	1	0	-

- ad esempio, se lo stato corrente **Q(t)** è **0** e si vuole che il prossimo stato **Q(t+1)** sia **1**, occorre applicare gli ingressi **R=0**, **S=1**.
- Se invece si vuole una transizione **0→0**, basta imporre **S=0**

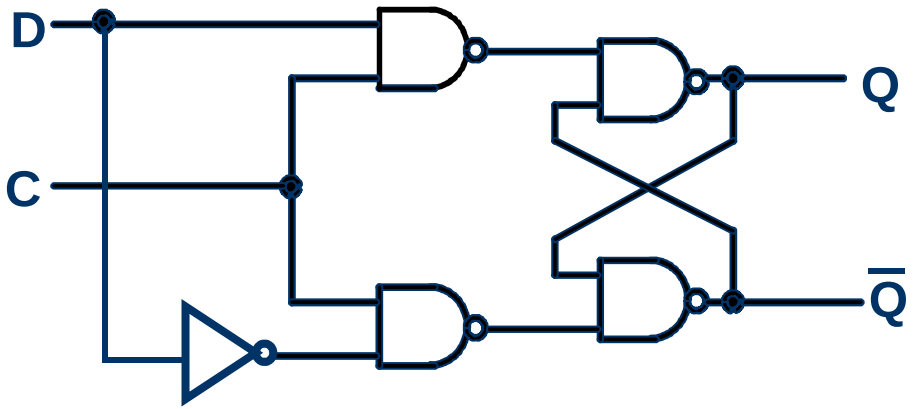
Latch RS con abilitazione

- L'ingresso **C** (*C*ontrollo, o ***C*lock**) agisce come <u>abilitazione</u> per gli altri due ingressi
- Quando **C=1** il latch si comporta come un normale RS
- Quanto **C=0** il latch memorizza lo stato indipendentemente dai valori di **S** e **R**, che *possono anche assumere configurazioni non ammesse*

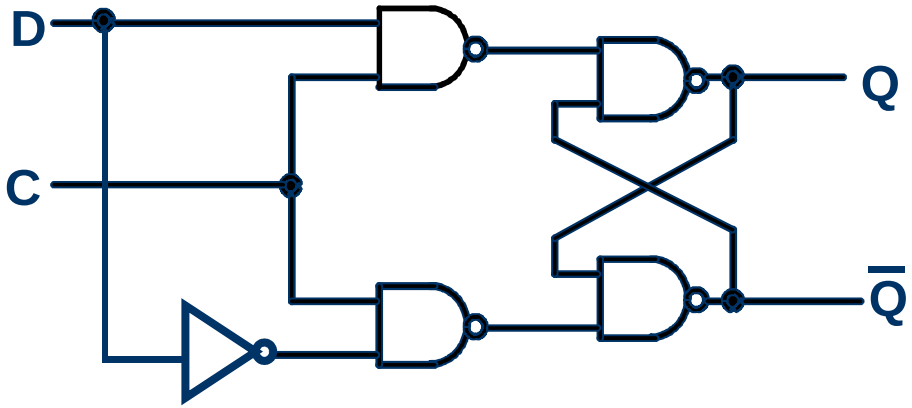


Latch di tipo D

- Ha un solo ingresso (**D**ato), memorizzato nello stato **Q**
- Elimina l'indeterminazione causata dal caso **RS=11**
- Lo stato ***segue*** l'ingresso (ovvero, ne assume lo stesso valore) quando **C=1**
  - si dice in questo caso che il latch è **trasparente** all'ingresso
- Lo stato **Q** "congela" l'ingresso nel momento in cui **C** diventa basso: **1→0**



Latch di tipo D



Q	D	Q(t+1)	Commento
0	0	0	inalterato
0	1	1	setta Q
1	0	0	resetta Q
1	1	1	inalterato

tabella funzionale

Le tabelle a destra sono riferite al caso di latch abilitato (**C=1**).

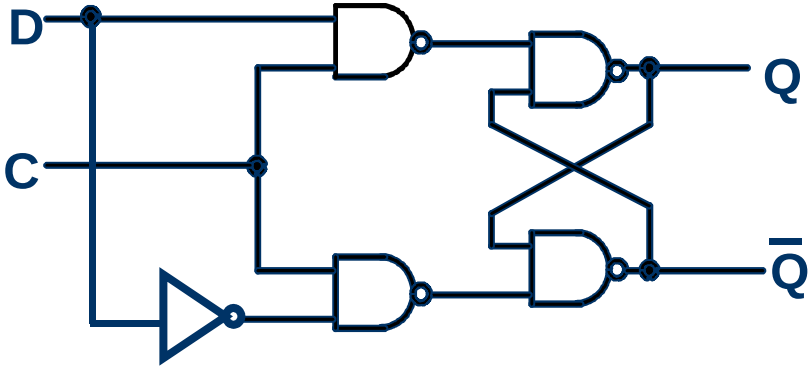
Nel caso di latch disabilitato (**C=0**) lo stato rimane in ogni caso inalterato

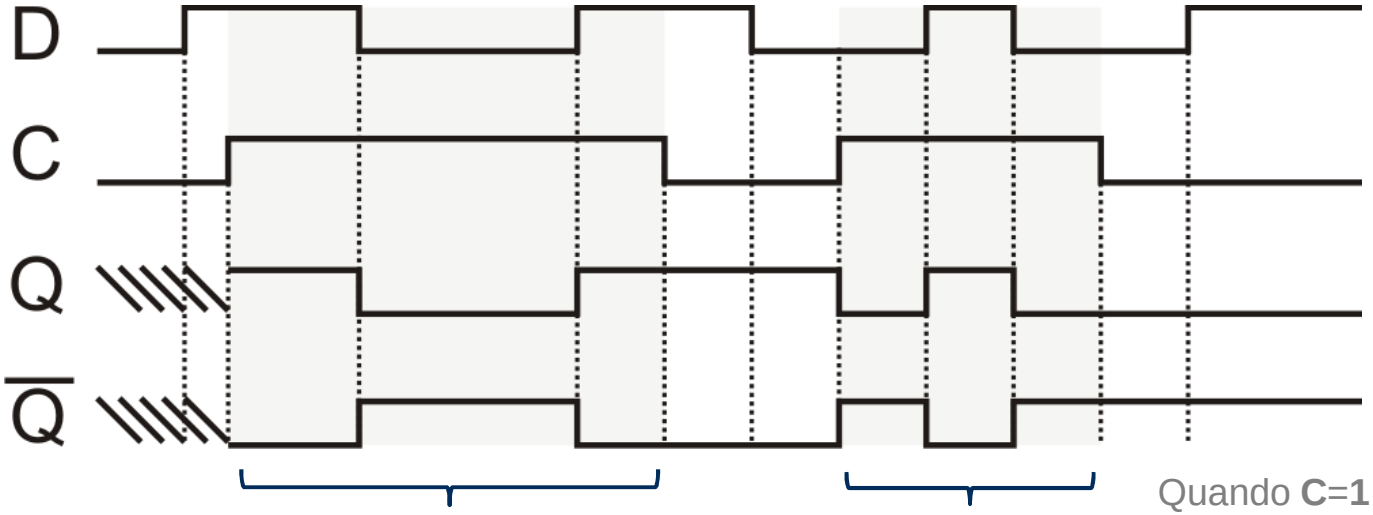
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

tabella di eccitazione

Latch di tipo D

- Esempio di tempificazione



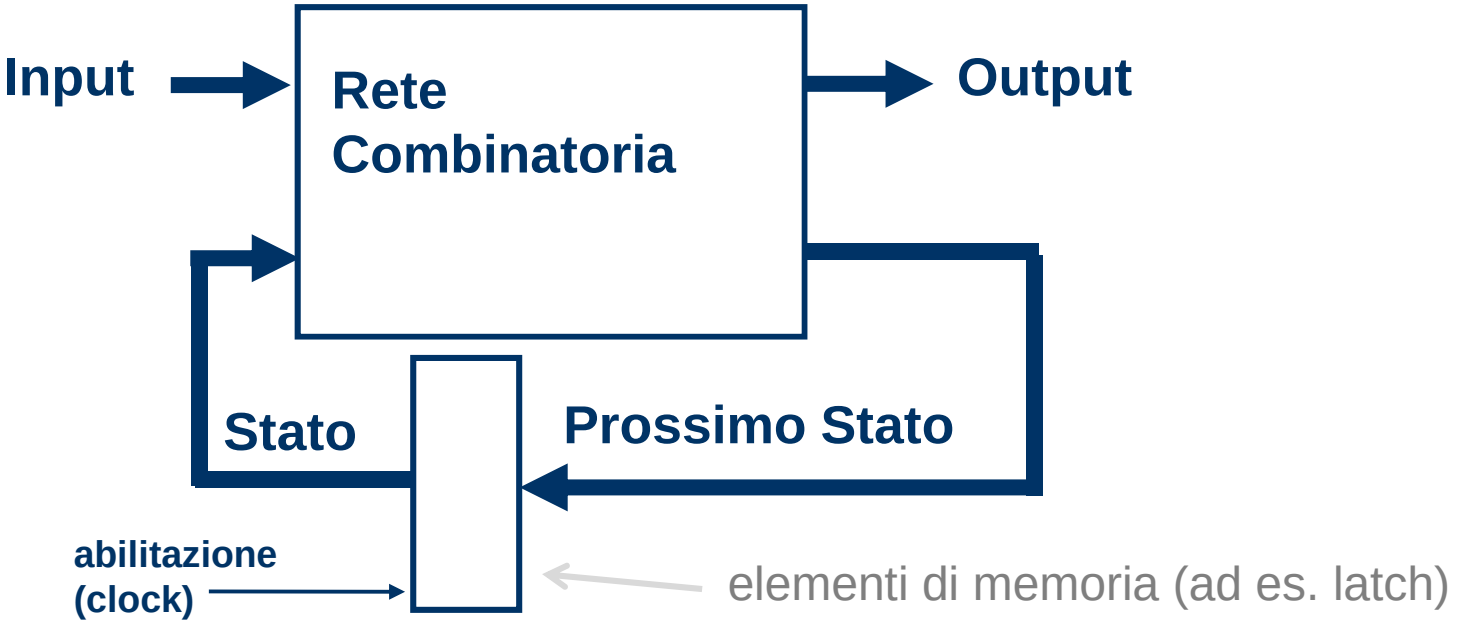


Quando **C=1** il latch è **trasparente**, ovvero lo stato è identico all'ingresso **D**

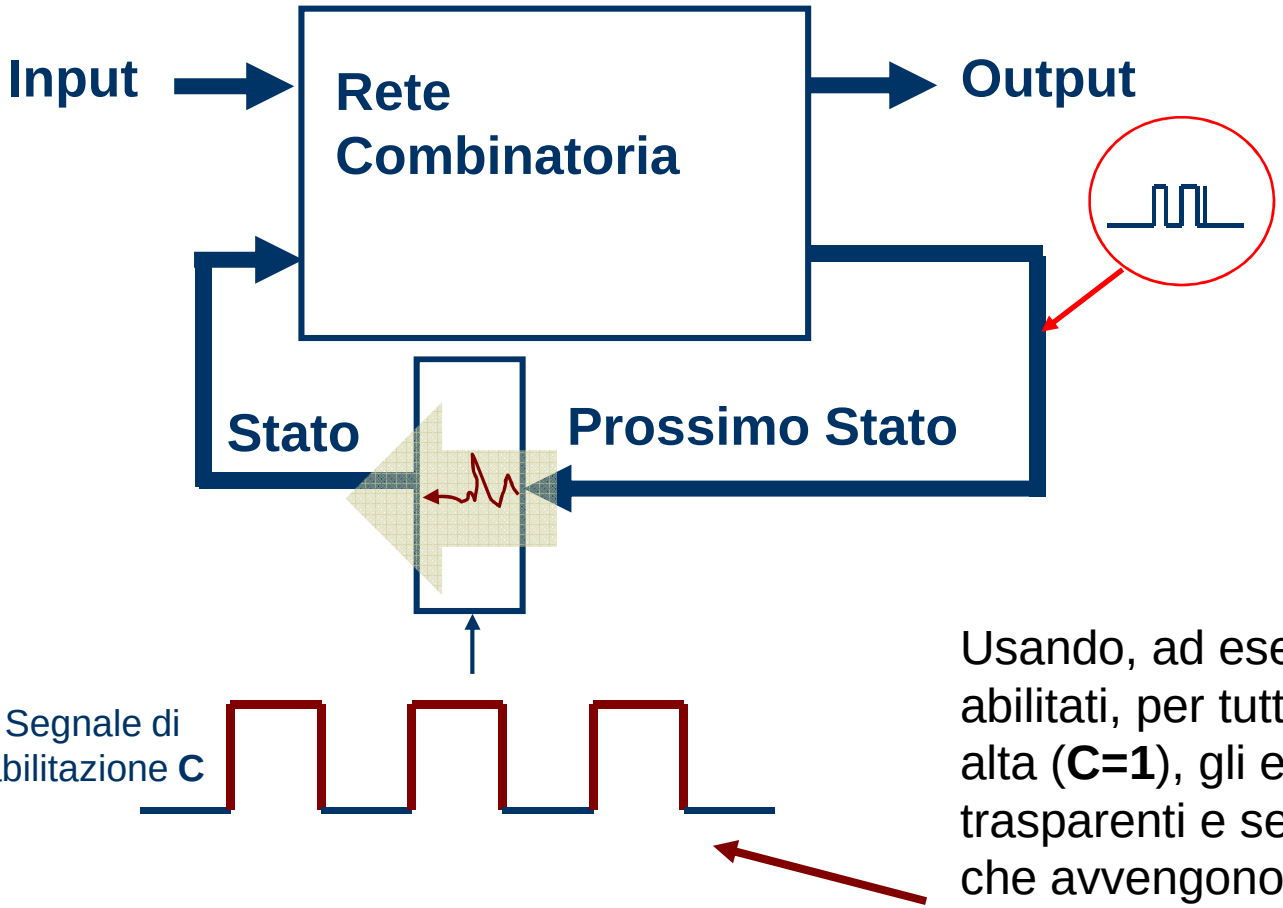
Elementi di memoria nelle macchine sequenziali

Useremo spesso gli elementi di memoria come “registri” per memorizzare lo stato corrente di una macchina sequenziale

Dallo stato corrente, attraverso un’opportuna rete combinatoria, dipende lo stato prossimo, a sua volta ingresso per gli elementi di memoria:



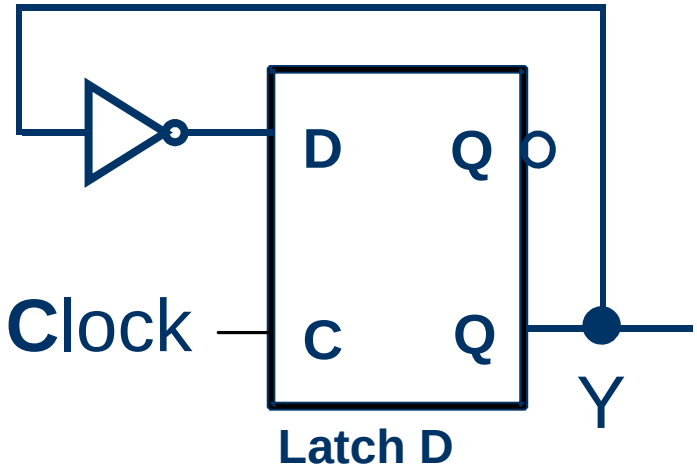
Problemi di tempificazione con i Latch



A causa dei diversi ritardi, le uscite della rete combinatoria possono presentare delle transizioni “spurie” prima di assestarsi sui loro valori definitivi

Usando, ad esempio, dei latch **RS** o **D** abilitati, per tutto il tempo in cui l’abilitazione è alta (**C=1**), gli elementi di memoria sono trasparenti e sensibili alle transizioni “spurie” che avvengono prima che l’uscita della rete combinatoria si assesti

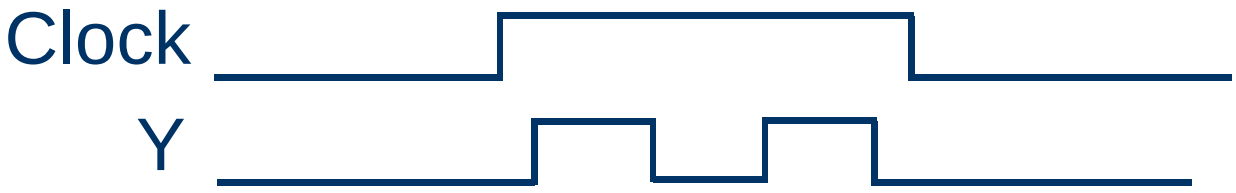
Problemi di tempificazione con i Latch



Semplice esempio di macchina sequenziale:

Stato = *Uscita* = **Y** *Stato Prossimo* = **NOT**(**Y**)

Ad ogni impulso del Clock, lo *Stato* **Y** viene aggiornato con lo *Stato Prossimo* **NOT**(**Y**). Tuttavia, poiché il latch è trasparente per tutto l'intervallo di tempo in cui **Clock=1**, l'aggiornamento di stato si ripete più volte durante questo intervallo, determinando un numero di transizioni aleatorio. Tale numero dipende sia dalla durata del **Clock=1**, sia dalla velocità di risposta del latch



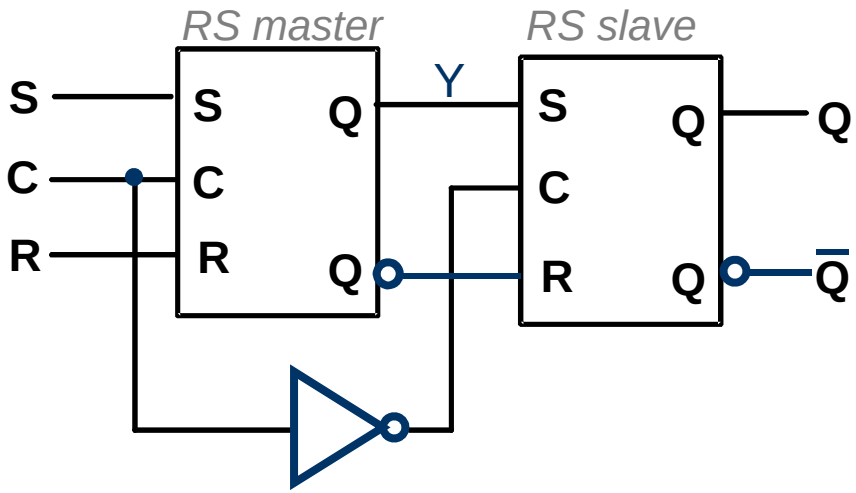
diverse transizioni di stato (4 nell'esempio) avvenute mentre Clock=1

Flip-Flop

- Una soluzione al problema visto prima è bloccare la propagazione del nuovo valore dall'ingresso all'uscita dell'elemento con memoria (la "*trasparenza*" del latch)
  - non deve *mai essere possibile* che l'uscita dell'elemento di memoria dipenda dall'ingresso applicato nello stesso istante
- I ***flip-flop*** indicano reti sequenziali sincrone, che cambiano l'uscita solo in particolari istanti in cui varia il segnale di abilitazione (o clock)
  - l'uscita è sensibile all'ingresso solo in un intervallo di tempo estremamente limitato, di durata pressocchè nulla
- Due tipi
  - Master-Slave (o pulse-triggered)
  - Edge-triggered

Flip-Flop master-slave

- Formato a partire da due Latch **RS** (*master* + *slave*)
- Quando **C=1** l'ingresso è "osservato" dal primo latch, il master, che in questa fase è trasparente
  - la sua uscita **Y** può variare, ma lo slave è disabilitato e non la legge
- Quando **C=0** il valore è "memorizzato" dal secondo latch, lo slave, mentre il master è disabilitato e mantiene quindi la sua uscita **Y** ad un valore costante



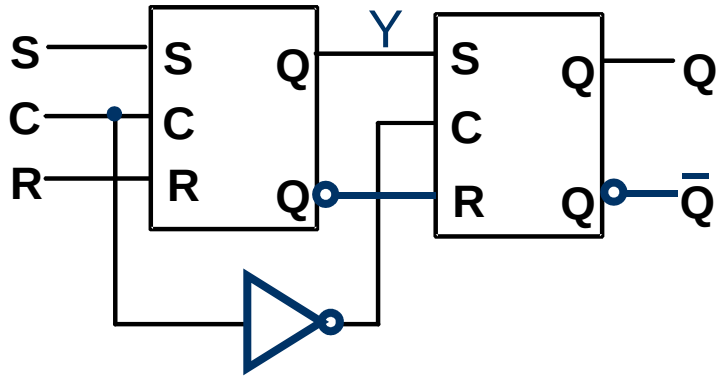
Lo slave cambia stato solo in corrispondenza dell'istante in cui il segnale di abilitazione **C** passa dal valore **1** al valore **0** (*fronte di discesa*)

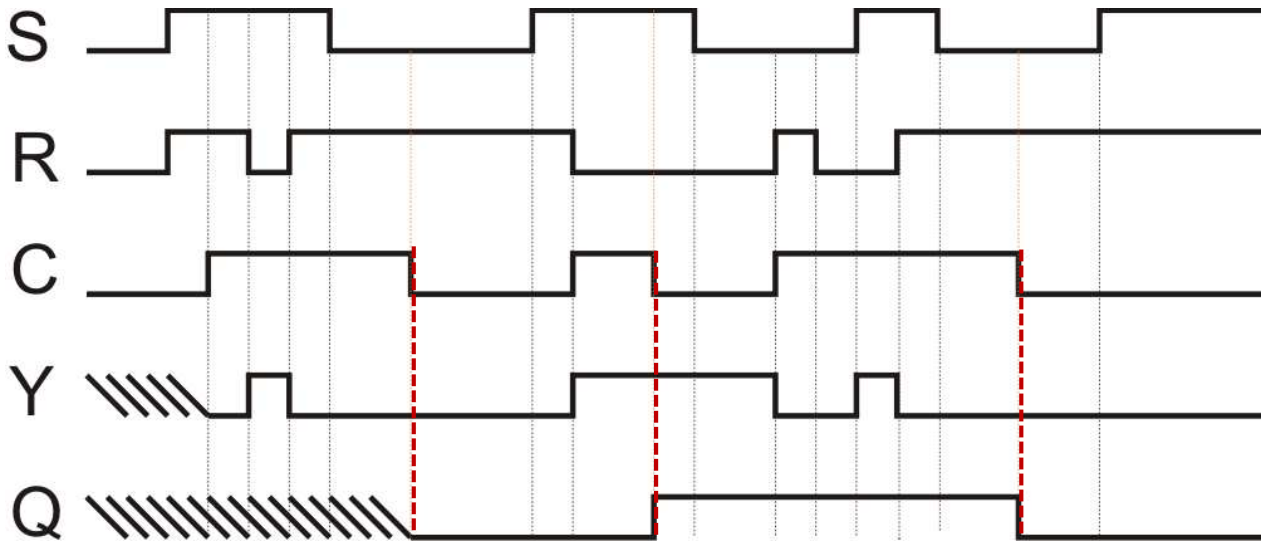
(Volendo, è invece possibile ottenere cambiamenti sui *fronti di salita* negando con una **NOT** il segnale **C** proveniente dall'esterno)

Flip-Flop master-slave

- Esempio di tempificazione

  (Flip-Flop master-slave su fronti di discesa)





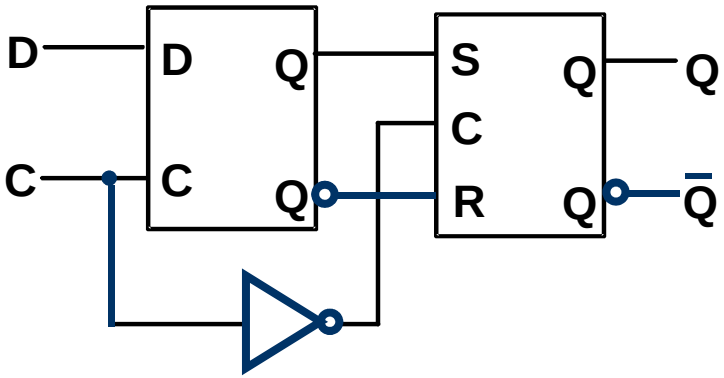
Lo slave cambia stato **solo in corrispondenza dell'istante** in cui il segnale di abilitazione **C** passa dal valore **1** al valore **0** (*fronte di discesa*), indicato dalle linee rosse in figura

Flip-Flop master-slave

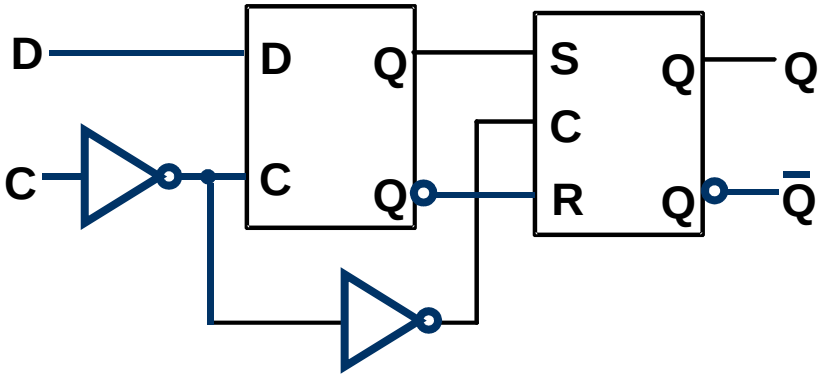
- Risolve in parte il problema legato all'uso dei latch
  - non-trasparente agli ingressi
  - la transizione in uscita avviene sul fronte di **C 1→0** (o **0→1** se si nega dall'esterno **C** con una **NOT**)
- E' tuttavia ancora sensibile alle transizioni in ingresso
  - il latch **RS** master, quello a sinistra negli schemi precedenti, è direttamente esposto agli ingressi
  - eventuali transizioni contemporanee degli ingressi **RS** da **11** a **00** causeranno ancora metastabilità

Flip-Flop edge-triggered

- “Campionano”, ovvero leggono, il valore dell’ingesso soltanto ***sui fronti*** del clock, e non durante tutta la fase attiva (ad es. nell’istante del fronte di discesa di **C**: **1→0** e non per tutto il tempo in cui **C**=**1**)
- Formato da un latch **D** ed uno **RS**
- “congela” l’ultimo valore seguito da **D**



flip-flop edge-triggered attivo su fronte di discesa del clock **C**

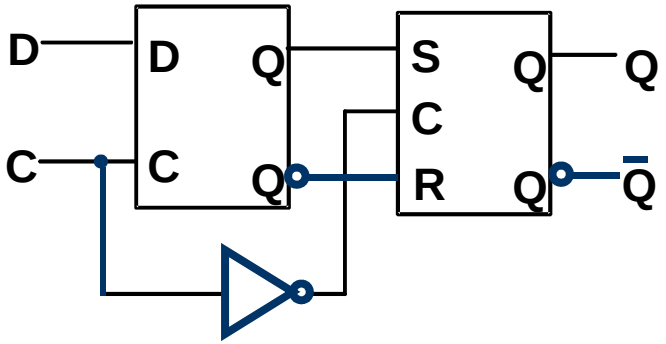


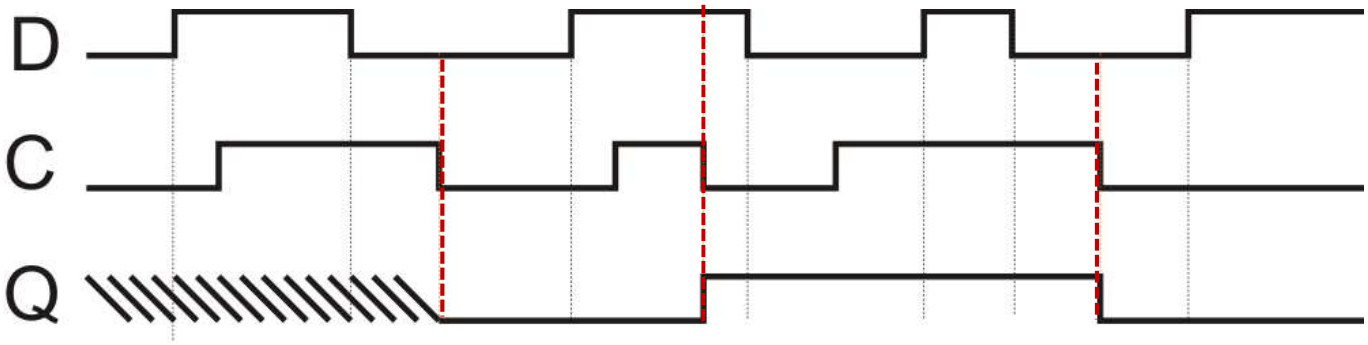
flip-flop edge-triggered attivo su fronte di salita del clock **C**

Flip-Flop edge-triggered

- Esempio di tempificazione

  (Flip-Flop edge-triggered su fronti di discesa)

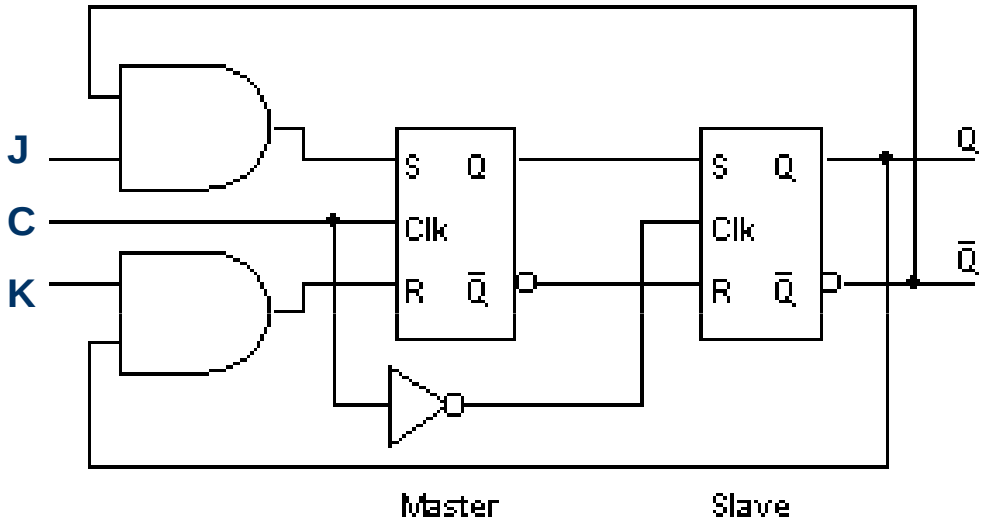




Il primo latch legge l'ingresso ed il secondo lo riporta in uscita solo in corrispondenza dell'istante in cui il segnale di abilitazione **C** passa dal valore **1** al valore **0** (*fronte di discesa*)

Flip-Flop JK master slave

- Il clock **C** tempifica questo flip-flop come nel caso master-slave
- Gli ingressi **J** e **K** operano come gli ingressi **S** ed **R** del latch **RS:** **J**→*set*, **K**→*reset*
- Il caso **JK=11**, è però ammissibile e agisce in modalità *toggle:*
  - se lo stato **Q** è **1**, sul successivo fronte esso diventa **0**, e viceversa

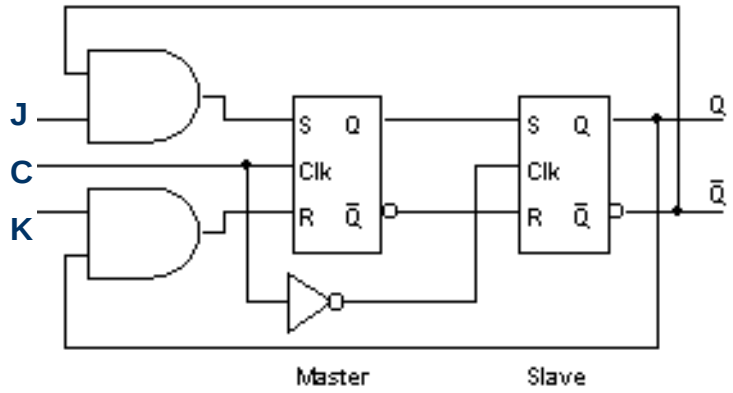


Flip-Flop JK master slave

Flip-Flop JK master slave

Q(t)	J	K	Q(t+1)	Commento
0	0	0	0	inalterato
0	0	1	0	resetta Q
0	1	0	1	setta Q
0	1	1	1	*toggle* (inverte Q)
1	0	0	1	inalterato
1	0	1	0	resetta Q
1	1	0	1	Setta Q
1	1	1	0	*toggle* (inverte Q)

tabella funzionale

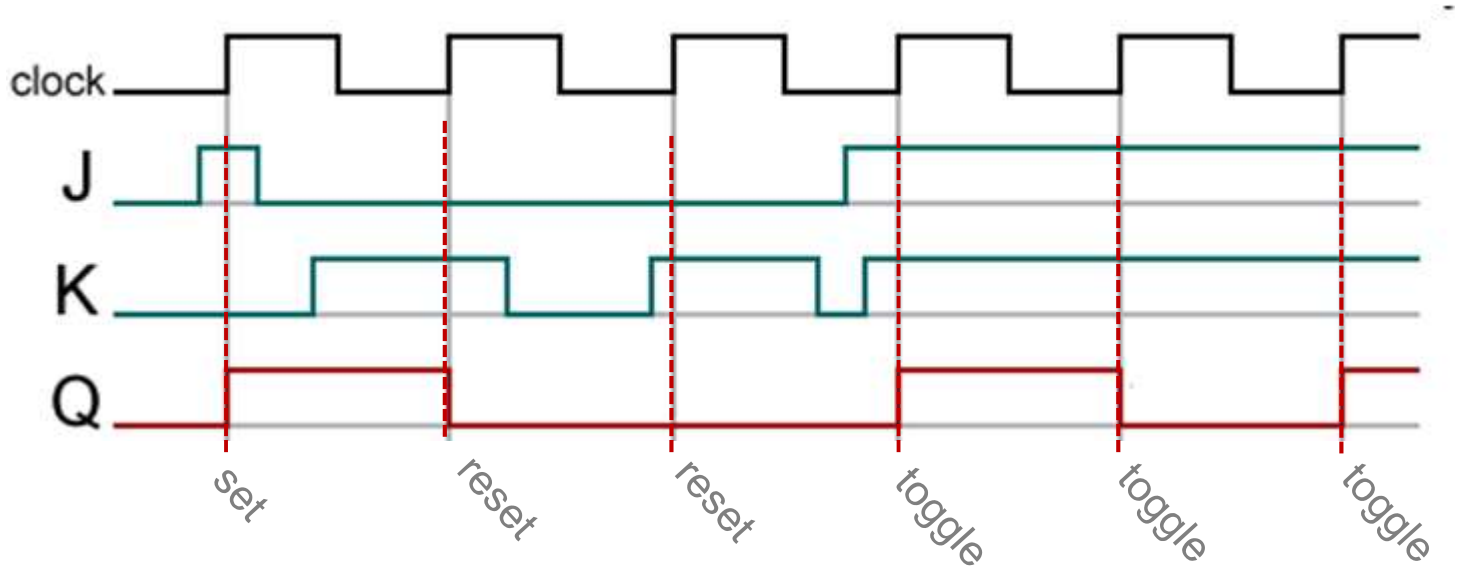


Q(t)	Q(t+1)	K	J
0	→ 0	-	0
0	→ 1	-	1
1	→ 0	1	-
1	→ 1	0	-

tabella di eccitazione

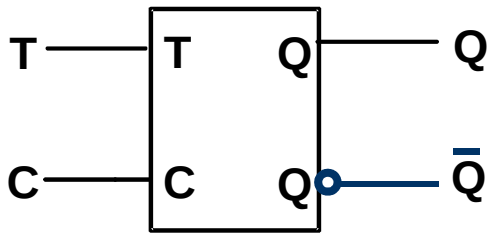
Flip-Flop JK edge-triggered

- Il JK può essere realizzato anche con tempificazione edge-triggered
- Esempio di flip-flop JK edge-triggered attivo su fronti di salita:

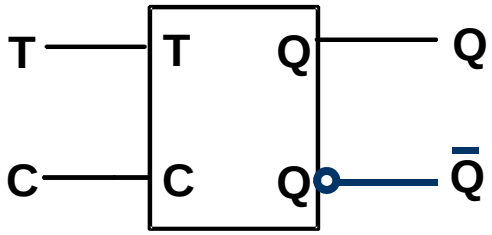


Flip-Flop di tipo T

- Ha un solo ingresso **T**
- opera solo in modalità toggle:
  - ingresso **T=1**: se lo stato **Q** è **1**, sul successivo fronte del clock **C** esso diventa **0**, e viceversa
  - ingresso **T=0**: lo stato **Q** resta inalterato



Flip-Flop di tipo T



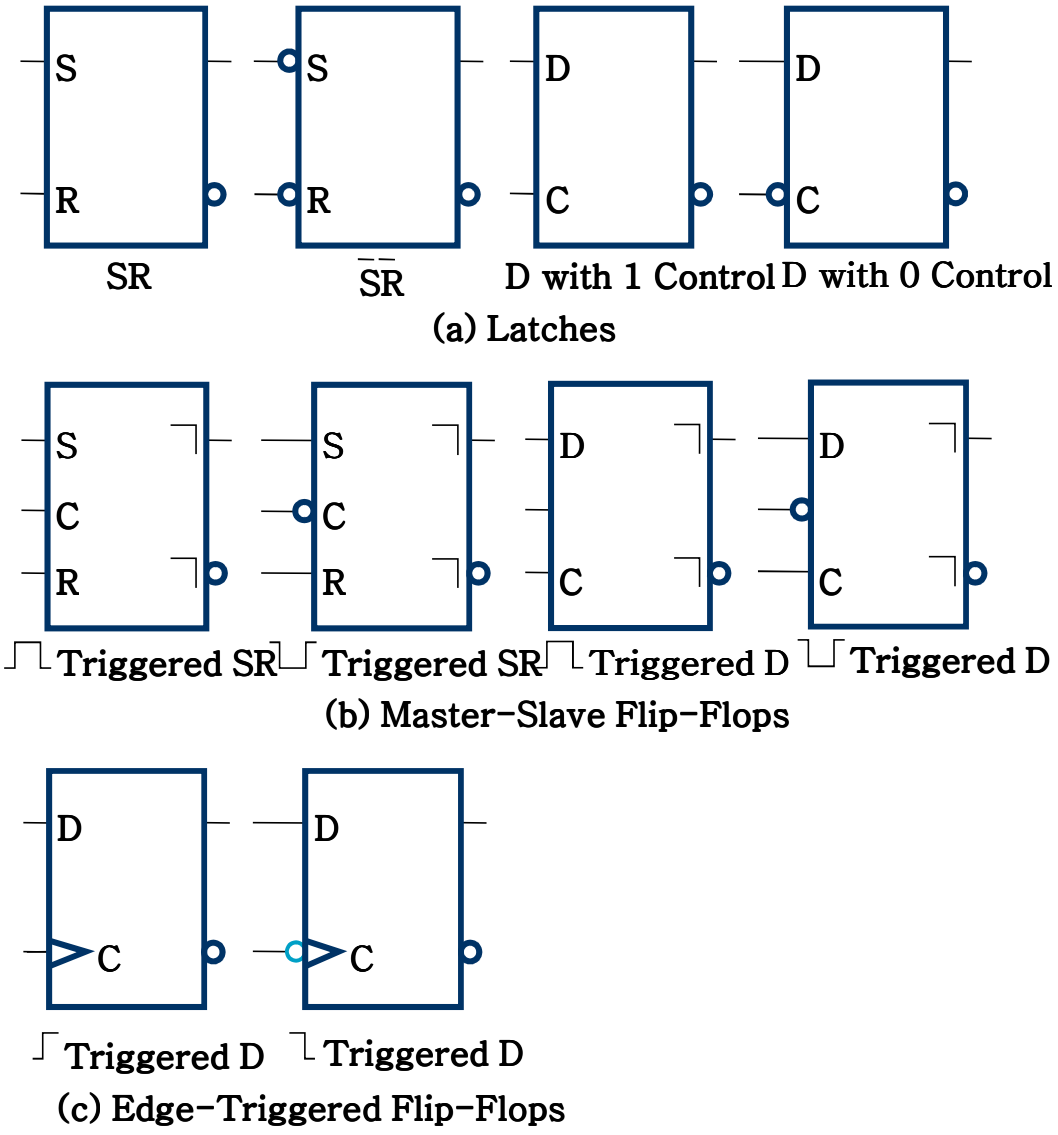
Q	T	Q(t+1)	Commento
0	0	0	inalterato
0	1	1	toggle (inverte Q)
1	0	1	inalterato
1	1	0	toggle (inverte Q)

tabella funzionale

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

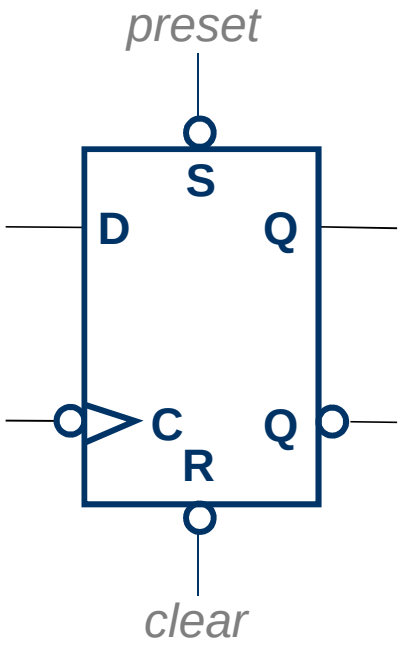
tabella di eccitazione

La figura a destra mostra alcuni simboli grafici tipicamente usati per rappresentare i diversi tipi di elementi di memoria



Ingressi diretti (opzionali)

- *direct set* (o *preset*)
  - impone stato alto **Q=1**
- *direct reset* (o *clear*)
  - impone stato basso **Q=0**
- possono essere usati per impostare lo stato (altrimenti indeterminato) in cui gli elementi di memoria si trovano all'accensione
- possono essere "**asincroni**"
  - agiscono in qualsiasi momento, indipendentemente da **C**
- o "**sincroni**"
  - agiscono solo quando l'elemento è abilitato tramite il segnale **C**



Elementi di memoria: Riepilogo

- Classificazione per tipo di tempificazione:
  - **latch**
  - **flip-flop master-slave** (o pulse-triggered)
  - **flip-flop attivo su fronti** (o edge-triggered)
- Classificazione per funzionamento degli ingressi:
  - tipo **RS**
  - tipo **D**
  - tipo **JK**
  - tipo **T**

  (ognuno ha la propria tabella funzionale/di eccitazione)