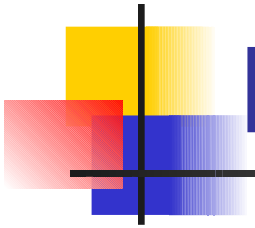
Logica sequenziale

I dispositivi logici si suddividono in due famiglie principali:

- **Logica combinatoriale**
  - L'uscita all'istante t_n dipende unicamente dallo stato degli ingressi sempre al tempo t_n, (trascuriamo i ritardi interni)
  - Porte logiche, decoders, multiplexers, ALU
- **Logica sequenziale**
  - L'uscita all'istante t_n dipende sia dallo stato degli ingressi al tempo t_n, sia dallo stato degli ingressi ad istanti precedenti ($t_{n-1,n-2,.}$,ecc.)
  - Ha quindi una memoria che viene chiamata STATO.
  - Latch, flip-flops, macchine a stati, contatori, shift registers



Flip-flop & Latches

- Logica combinatoriale con "feed-back"
  - Uscita riportata all'ingresso
  - Latch è una memoria ad 1 bit che cambia stato a seconda dell'ingresso
  - Flip-flop è un latch dotato di un ingresso di clock (segnale di controllo)
    - Le uscite commutano solo sui fronti (salita e/o discesa) del clock

Definizioni (1)

- Stato: è un insieme di variabili **(variabili di stato per l'appunto)** che rappresentano la memoria di quello che è successo fino ad ora
  - Conta-persone: lo stato è il numero di persone passate fino adesso, ogni volta che passa una persona il numero viene incrementato di uno
- Logica sequenziale digitale
  - Le variabili di stato sono binarie
  - Un circuito con n variabili di stato ha 2^n stati possibili
    - Macchina a stati finiti (Finite State Machine)
  - I cambiamenti di stato sono solitamente sincronizzati da un clock

CLOCK (1)

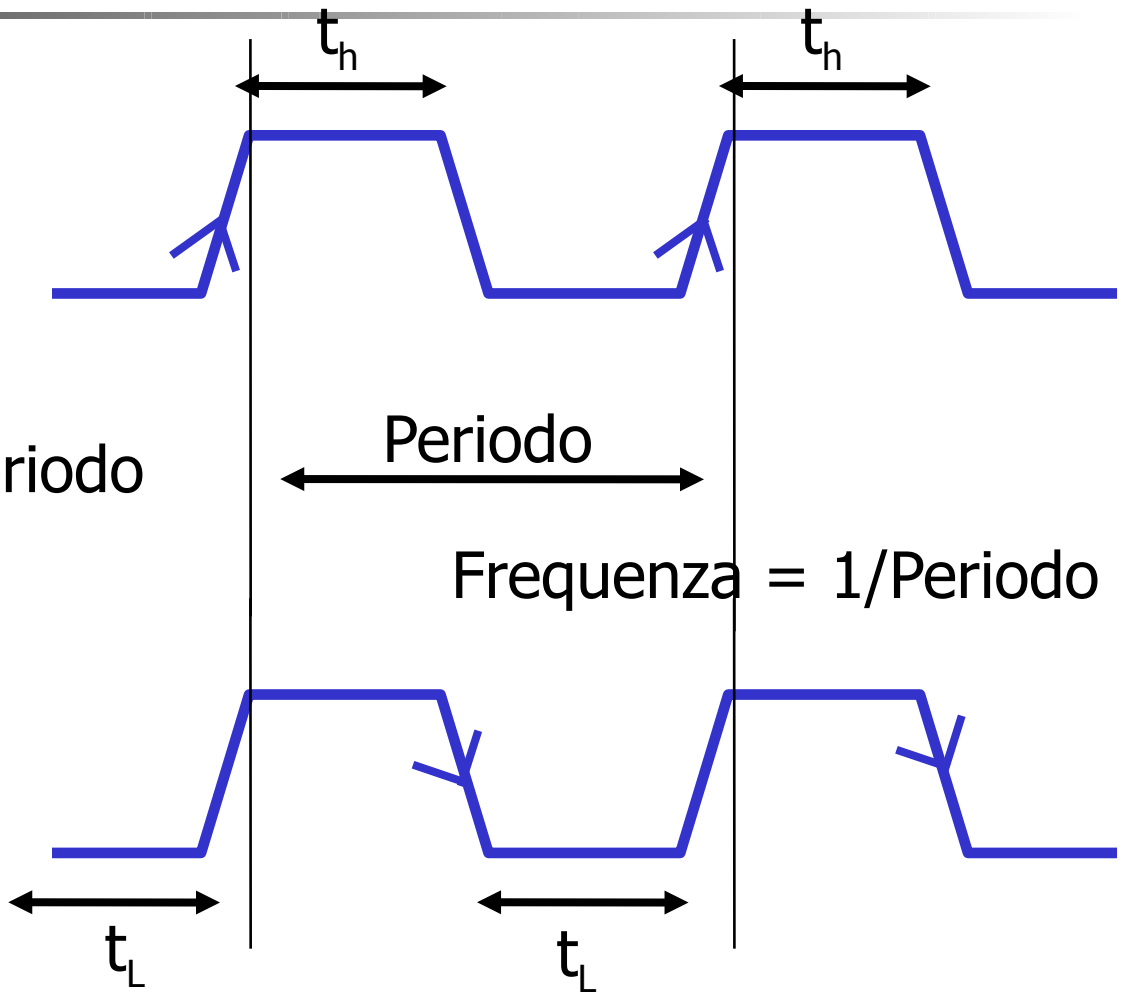
- Clock: fornisce la marca temporale che scandisce i cambiamenti di stato del circuito.
  - Il segnale di clock è active high se lo stato cambia sui fronti di salita
  - active low se lo stato cambia sui fronti di discesa.
- Periodo del Clock: è il tempo che trascorre tra due transizioni dello stesso tipo
- Frequenza del Clock: è il numero di volte che questo accade per secondo
- Duty cycle è la percentuale di tempo, durante un periodo, in cui il clock è attivo.

CLOCK (2)

- ACTIVE HIGH

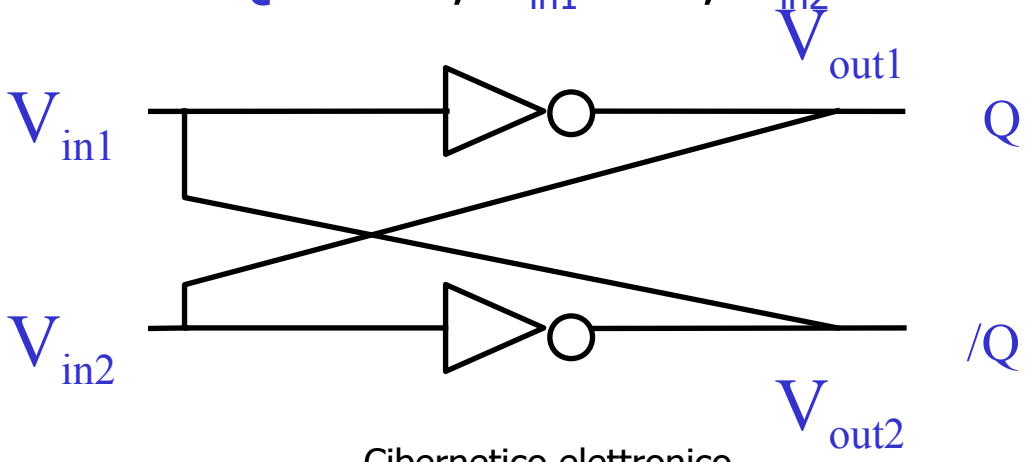
Duty cicle = t_h (t_L) / Periodo

- ACTIVE LOW



Elemento bistabile

- Memoria ad un bit
- L'elemento più semplice:
- E' **bistabile** in quanto ha **due** stati stabili:
  - Stato 1: Q alto, V_{in1} basso, V_{in2} alto
  - Stato 2: Q basso, V_{in1} alto, V_{in2} basso



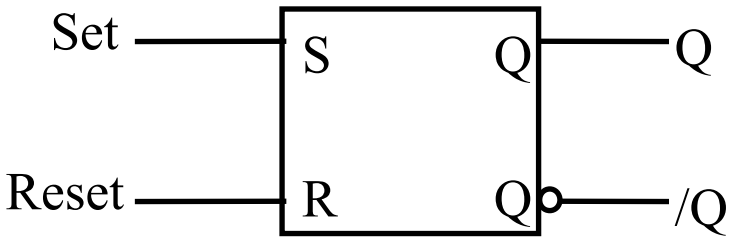
Bestiario dei latch & flip-flop

- Latch
  - S-R Latch
  - /S-/R Latch
  - S-R Latch con Enable
  - D Latch
- Flip-flops
  - Edge-Triggered D Flip-Flop
  - Master/Slave S-R Flip-Flop
  - Master/Slave J-K Flip-Flop
  - Edge-Triggered J-K Flip-Flop
  - T Flip-Flop

S-R Latch

- Caveat:
  - Propagazione (cfr. con porta singola)
  - Durata minima degli impulsi

Simbolo



Schema

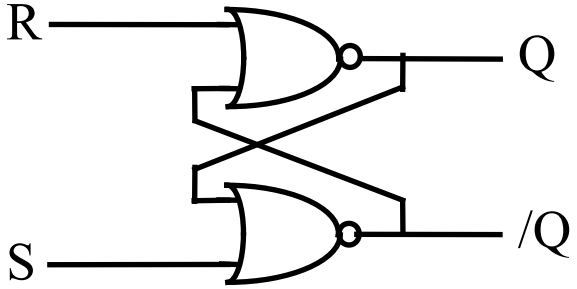


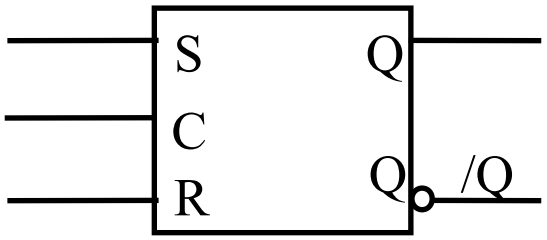
Tabella della verità

	S	R	Q_n	$/Q_n$
Hold	0	0	Q_{n-1}	$/Q_{n-1}$
Reset	0	1	0	1
Set	1	0	1	0
ILLEGALE	1	1	0	0

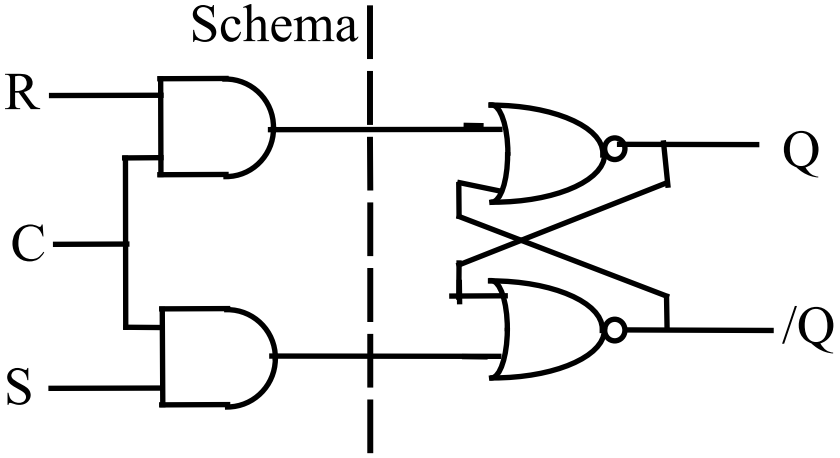
- Si attiva solo con enable alto
- S=R=1 illegale

S-R Latch con enable

Simbolo



Schema



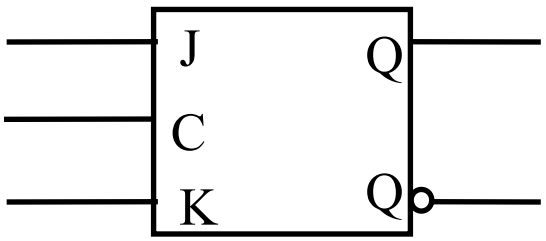
S	R	C	Q_n	$/Q_n$
0	0	1	Q_{n-1}	$\mathbf{/Q_{n-1}}$
0	1	1	0	1
1	0	1	1	0
1	1	1	0	0
X	X	0	$\mathbf{Q_{n-1}}$	$\mathbf{/Q_{n-1}}$

J-K Latch

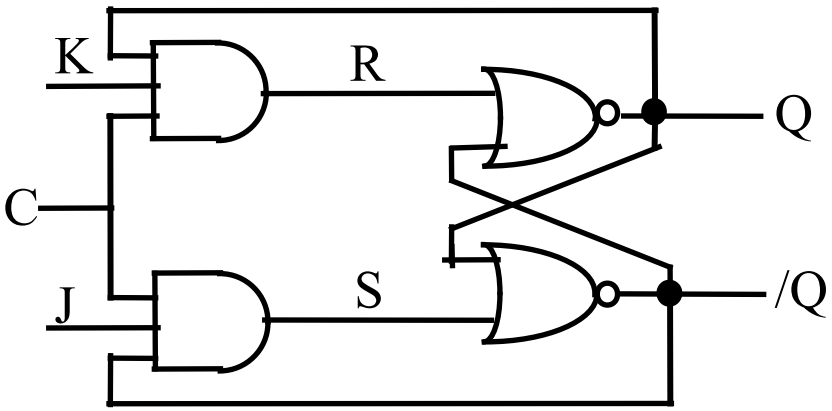
- Si attiva solo con enable alto
- Rimane il problema dell'instabilità
- J=K=1 lecito

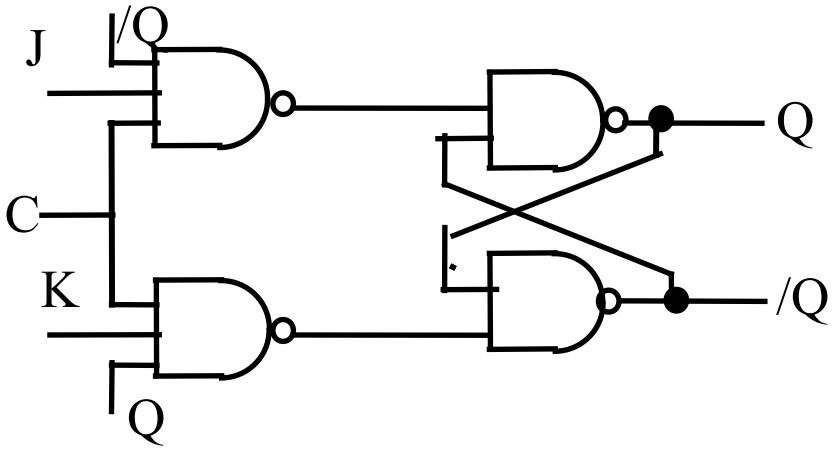
J	K	C	Q_n	$/Q_n$
0	0	1	Q_{n-1}	$/\mathbf{Q_{n-1}}$
0	1	1	0	1
1	0	1	1	0
1	1	1	$/\mathbf{Q_{n-1}}$	Q_{n-1}
X	X	0	$\mathbf{Q_{n-1}}$	$/\mathbf{Q_{n-1}}$

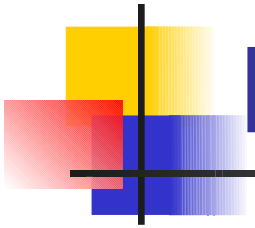
Simbolo



Schema (oscillatore)







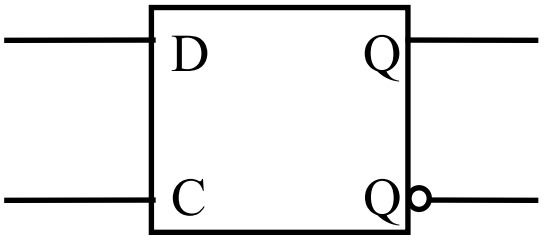
Preset, Clear

- I flip-flops hanno due ingressi asincroni.
  - Preset e Reset (Clear).
  - Servono ad impostare direttamente le uscite dei latch S-R.
  - Operano indipendentemente dall'enable (clock).
- Quando disegnate un circuito:
  - Usate questi ingressi SOLO per inizializzare la logica. Mai per realizzare funzioni logiche.

Latch D-type

- Immagazzina un dato di un bit (NON set e reset)
- Quando Enable è alto il latch è trasparente
- Risolve il problema di SET=RESET=1

Simbolo



Schema

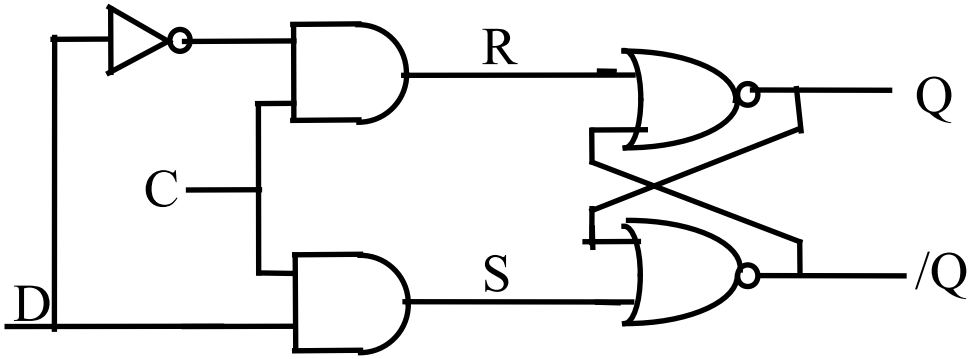


Tabella della verità

C	D	Q_n	$/Q_n$
1	0	0	
1	1	1	
0	X	Q_{n-1}	

Architettura Master-Slave

- Risolve i problemi di "race around" legati ai tempi di propagazione finiti delle porte logiche.
- Due latch in cascata:
  - il primo pilotato direttamente dal clock
  - il secondo dal clock invertito
- Quando è attivo il primo stadio, non è attivo il secondo (e viceversa).
- La commutazione delle uscite avviene sul fronte di salita (discesa) del clock.

D-type flip-flop

- Transizioni sul fronte di discesa del clock
- Se aggiungo un inverter al clock le transizioni avvengono sul fronte di salita

Simbolo

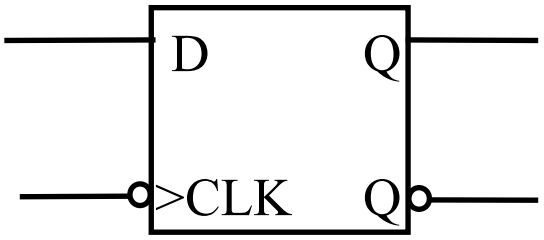
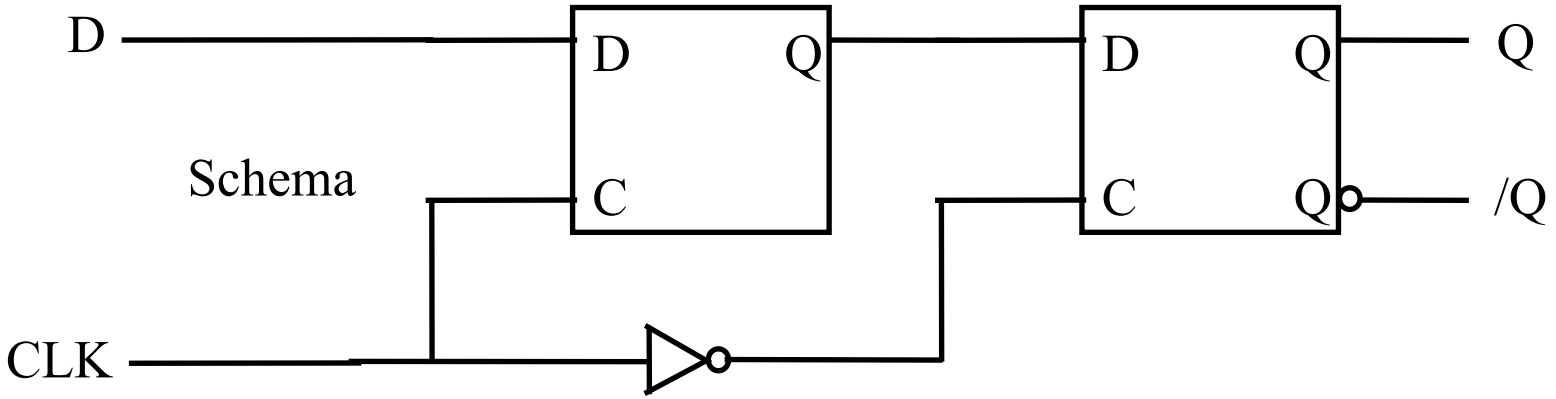


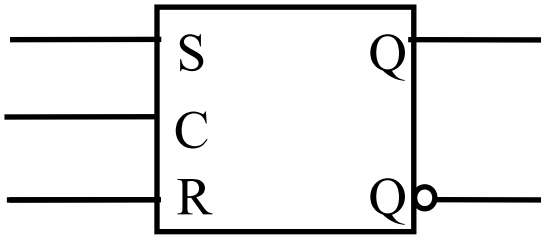
Tabella della verità

D	CLK	Q_n	$/Q_n$
0	↓	0	1
1	↓	1	0
X	0	Q_{n-1}	$/Q_{n-1}$
X	1	Q_{n-1}	$/Q_{n-1}$

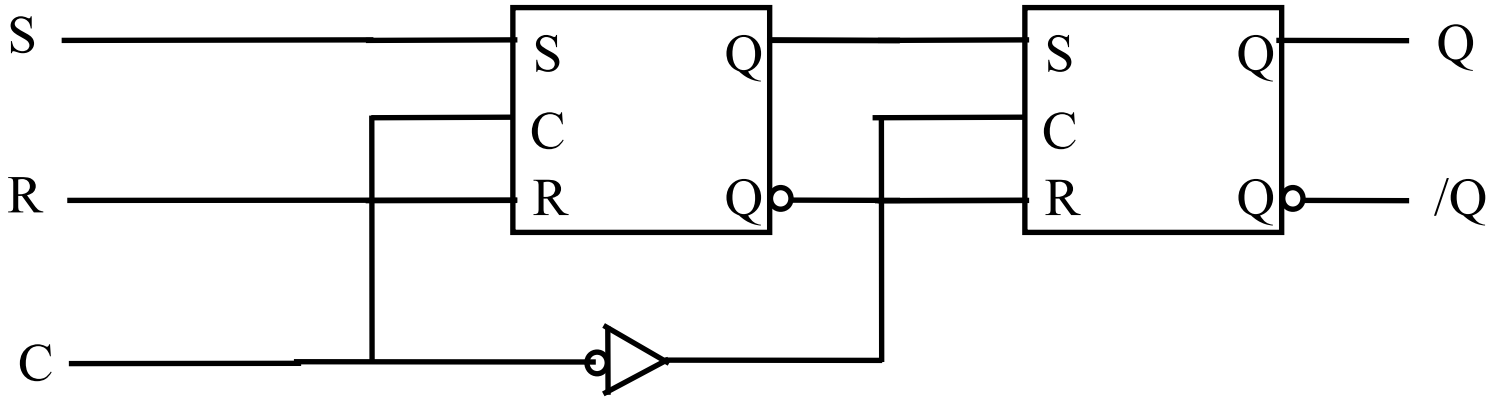
Schema



S-R flip-flop

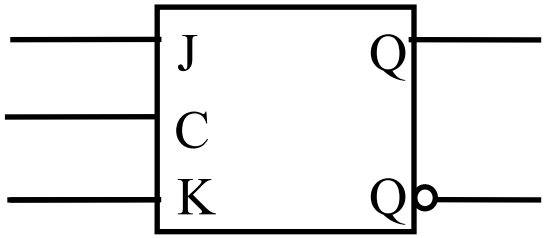


S	R	C	Q_n	$/Q_n$
X	X	0	Q_{n-1}	$/Q_{n-1}$
0	0	↓	Q_{n-1}	$/Q_{n-1}$
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	Illegale?	Illegale?

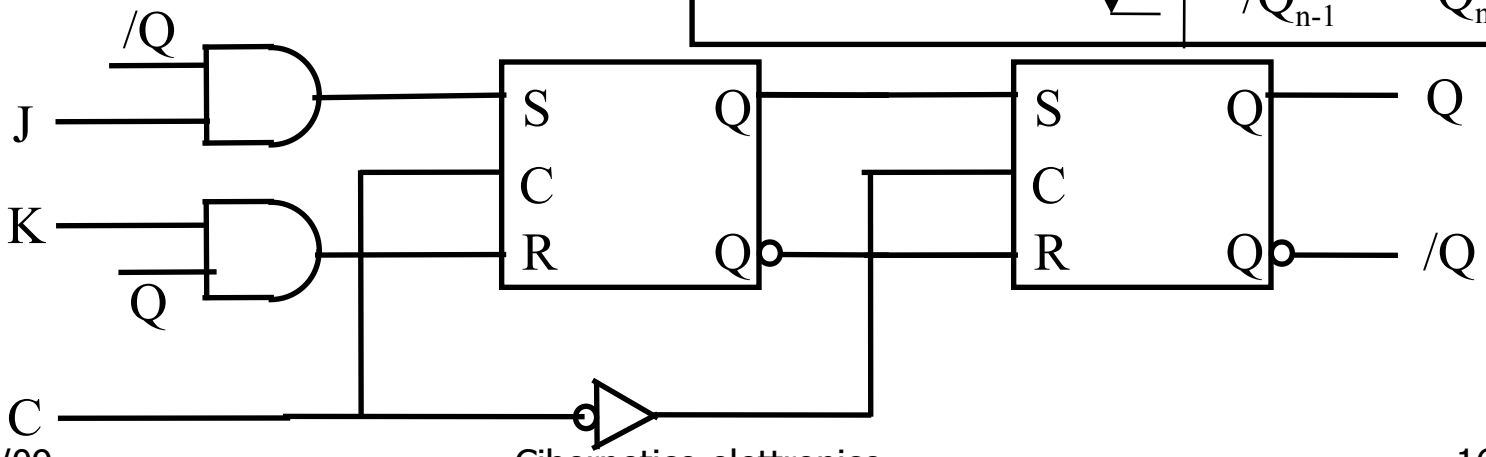


- Risolve il problema S=R=1
- Adesso funziona il toggle !

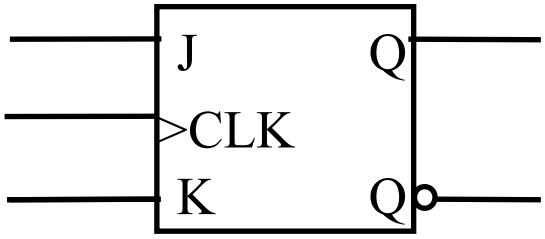
J-K flip-flop



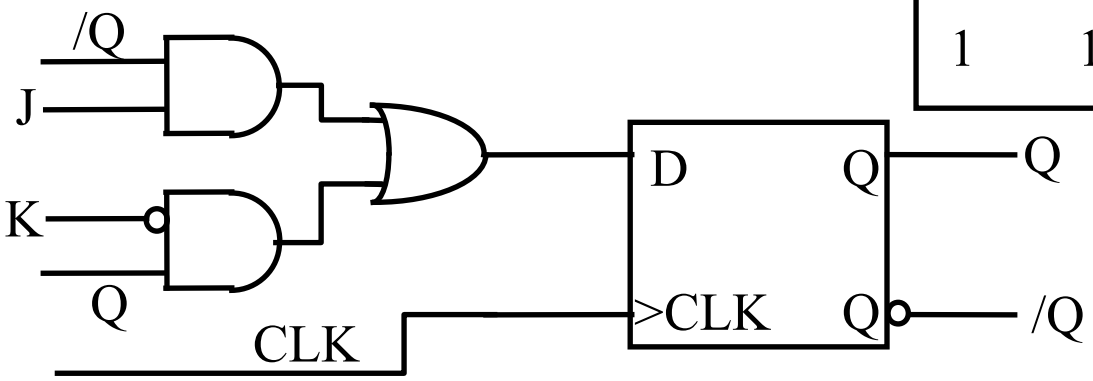
J	K	C	Q_n	$/Q_n$
X	X	0	Q_{n-1}	$/Q_{n-1}$
0	0	↓	Q_{n-1}	$/Q_{n-1}$
0	1	↓	0	1
1	0	↓	1	0
1	1	↓	$/Q_{n-1}$	Q_{n-1}



J-K con un D-type



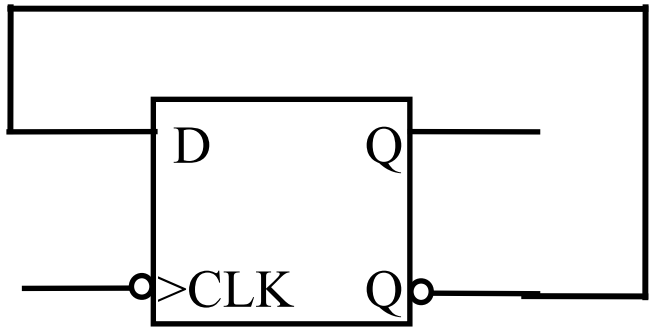
J	K	C	Q_n	$/Q_n$
X	X	0	Q_{n-1}	$/Q_{n-1}$
X	X	1	Q_{n-1}	$/Q_{n-1}$
0	0	↑	Q_{n-1}	$/Q_{n-1}$
0	1	↑	0	1
1	0	↑	1	0
1	1	↑	$/Q_{n-1}$	Q_{n-1}



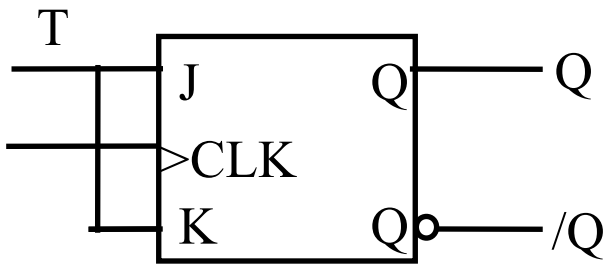
T (toggle) flip-flop

- T=1 → toggle → $Q_n = /Q_{n-1}$
- T=0 → $Q_n = Q_{n-1}$

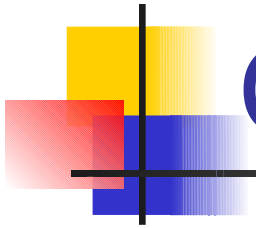
Senza T



Con T



Divisore di frequenza per due

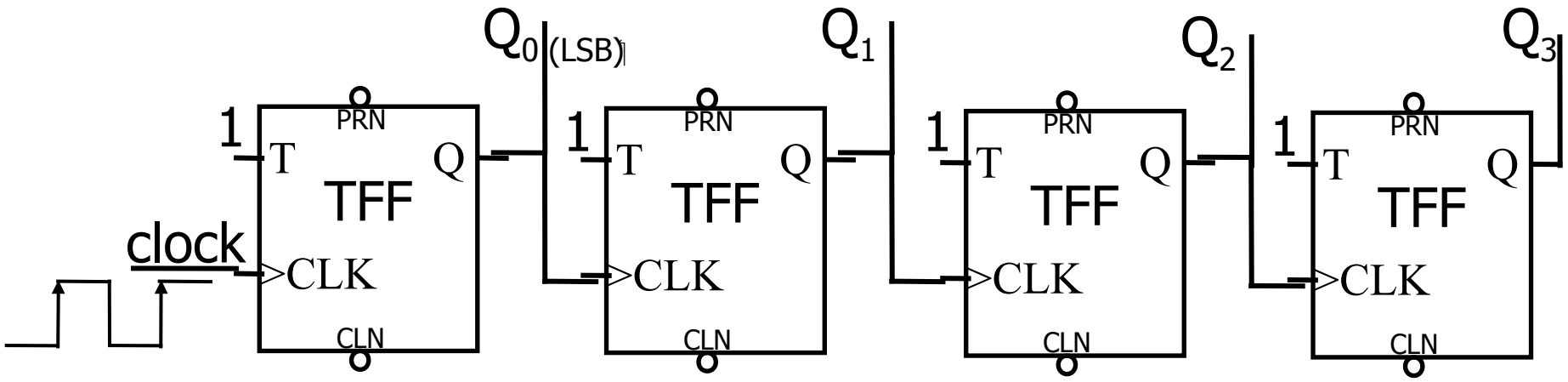


Contatori

- Un insieme di *n* flip-flop.
  - 2^n stati possibili
  - Possono contare in su o in giù
- Tipologia:
  - Ripple counter
  - Syncronous counter

Contatore ripple (1)

- Implementazione molto semplice
- Il clock viene dato in ingresso al primo TFF
- Le uscite Q dei flip-flop sono i bit di uscita
- LSB a sinistra (TFF con il clock)

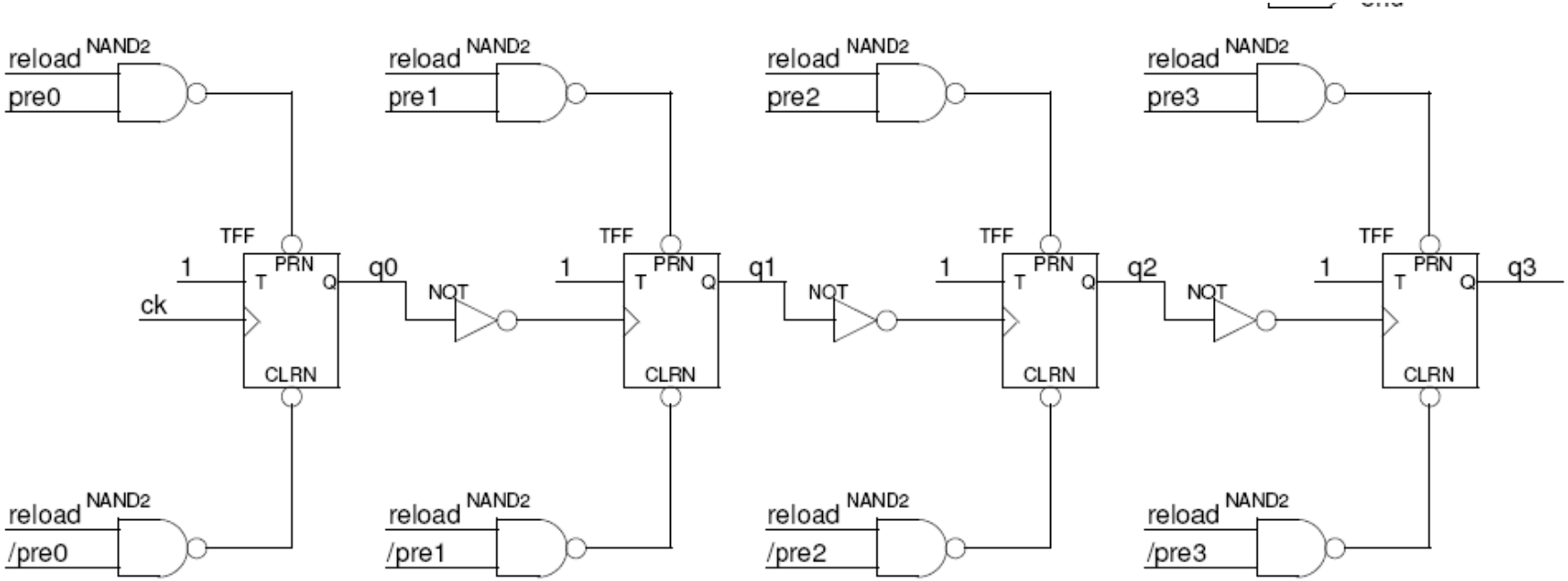


Contatore ripple (2)

- Il contatore visto conta in giù.
  - Per farlo contare in su basta usare dei TFF che commutino sui fronti di discesa.
- Il nostro contatore a 4 bit può contare fino a 15 ma:
  - Tramite i preset si può "caricare" in maniera asincrona qualsiasi numero di partenza.
  - Si può quindi farlo contare per qualsiasi numero da 1 a 15.
- Abbiamo bisogno dell'ausilio di un minimo di logica combinatoriale:
  - Un circuito che riconosca quando il contatore è arrivato a 16 e generi un segnale logico
  - Un circuito che imposti il complemento a 15 del numero da contare
    - es. devo contare fino a 9 → devo far partire il contatore dal numero 6
    - Il contatore si fermerà a 15 dopo aver contato quindi 9 colpi
- Realizzo così un contatore programmabile.
  - Attenzione anche agli ingressi di clear, sono necessari per caricare gli zeri
  - I preset caricano solo 1 NON 0.

Contatore programmabile

- Reload !
  - Il caricamento viene fatto sia alla partenza, sia ogni volta che il contatore raggiunge il conteggio di F (1111)

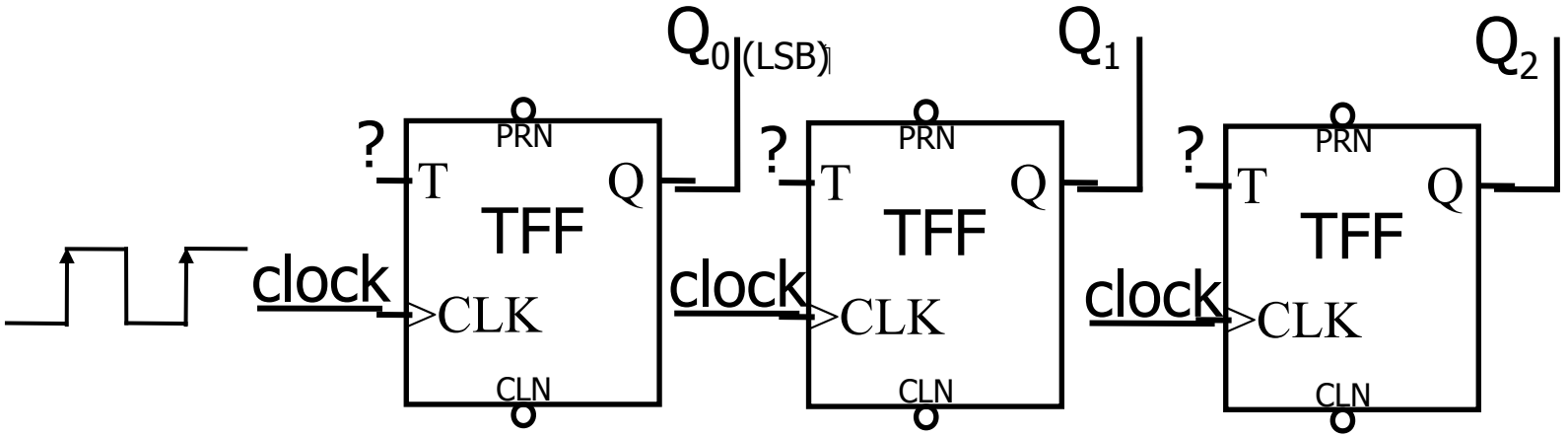


Contatore ripple (3)

- Svantaggio principale:
  - Ritardo di propagazione del segnale:
    - t_{pr} di circa 10 ns a flip-flop
    - n bit → ritardo totale = n t_{pr}
  - La criticità aumenta con il numero di bit e con la frequenza di clock.
    - Il conteggio finale rimane affidabile !
      - Cioè se interrompete il clock e aspettate un tempo sufficiente (> n t_{pr}), il valore letto sarà il numero effettivo di colpi di clock ricevuti.
    - Non sono più affidabili i conteggi intermedi !!!
      - Ad esempio nel nostro circuito precedente si possono avere dei "reload" spuri generati in mezzo alla sequenza di conteggio.

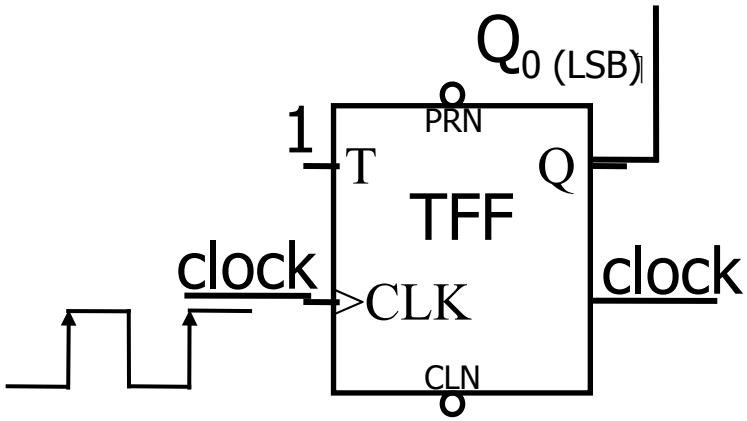
Contatore sincrono

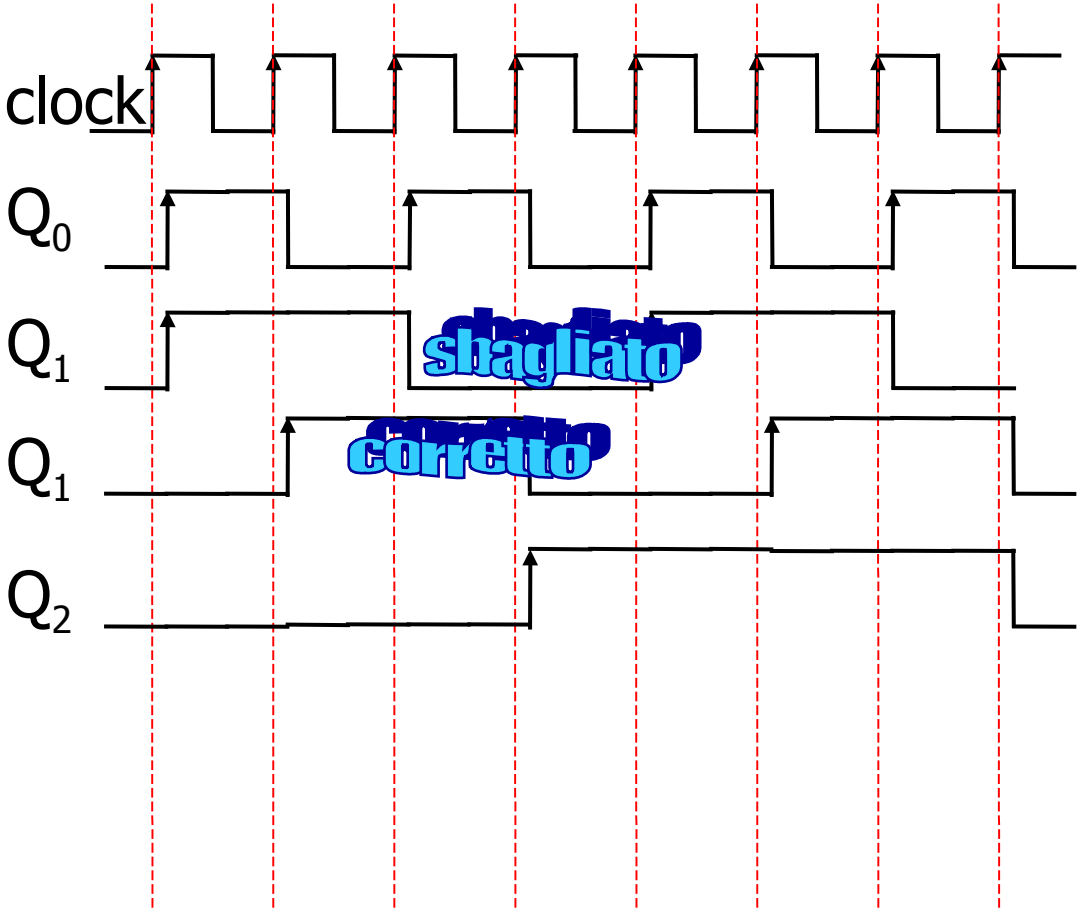
- Sfruttiamo a nostro vantaggio il ritardo di propagazione.
- I flip-flop sono tutti pilotati in parallelo dallo stesso clock.
- Uso di logica combinatoriale per stabilire se il flip-flop debba caricare 1 o 0.



Contatore sincrono

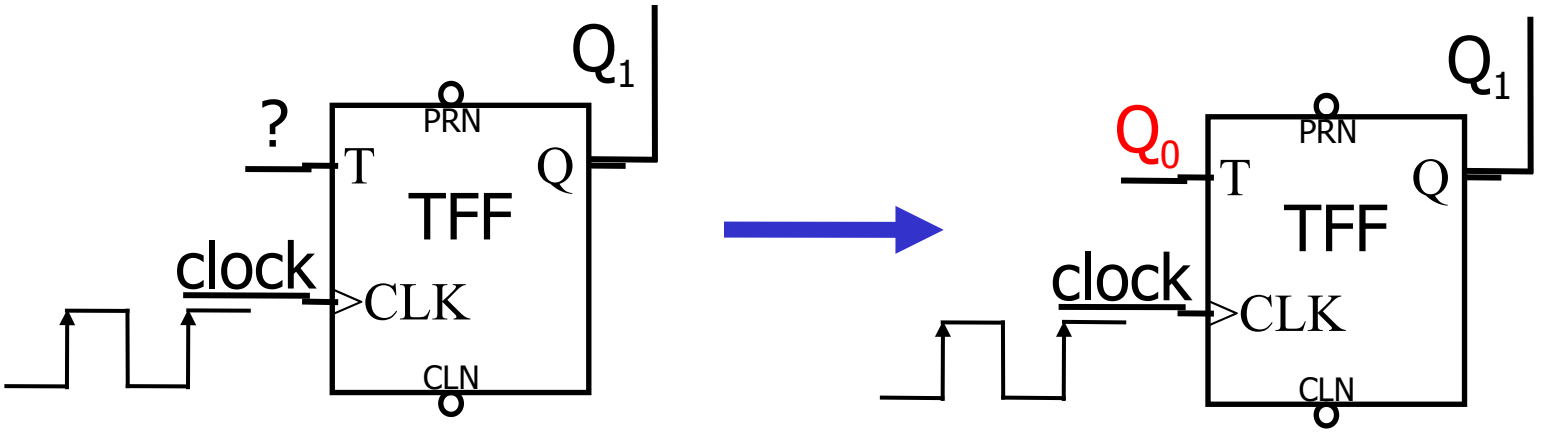
- Importante l'ingresso T
  - T=1 → $Q_n = /Q_{n-1}$
  - T=0 → $Q_n = Q_{n-1}$
- Per Q_0 connettere T=1, ma per Q_1 ?





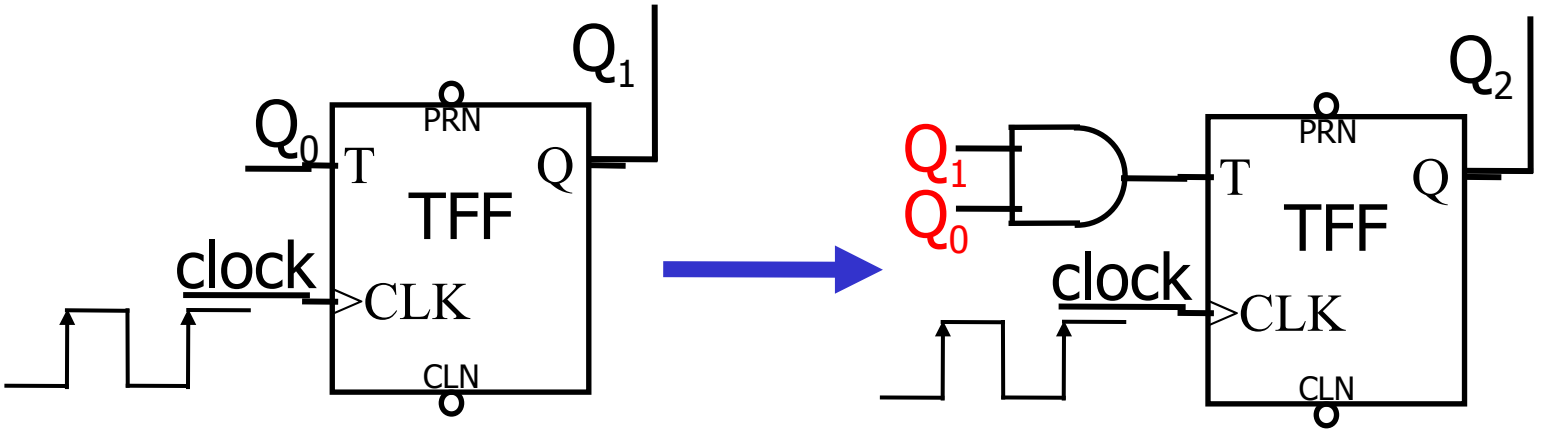
Quindi come connettiamo T_1 ?

- Connettiamo l'ingresso T a Q_0 !!!
- Sfruttiamo il ritardo di propagazione nei flip-flop
  - Sul primo fronte di salita del clock Q_0 diventa 1, MA solo dopo un tempo t_{pr} (tipicamente qualche ns)
  - Quindi il secondo flip-flop, sul primo fronte di salita del clock, in ingresso ha ancora zero e quindi la sua uscita Q_1 rimarrà nello stato precedente.
  - Sul secodo fronte di salita del clock Q_0 diventa 0, MA solo dopo un tempo t_{pr} (tipicamente qualche ns)
  - Quindi il secondo flip-flop, sul secondo fronte di salita del clock, in ingresso ha ancora uno e quindi la sua uscita Q_1 si ribalta rispetto allo stato precedente.



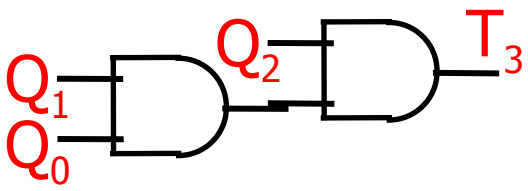
Come connettiamo T_2 ?

- Q_2 deve commutare nel momento in cui Q_0=Q_1 transiscono da 1 a 0
- Teniamo presente t_{pr} !

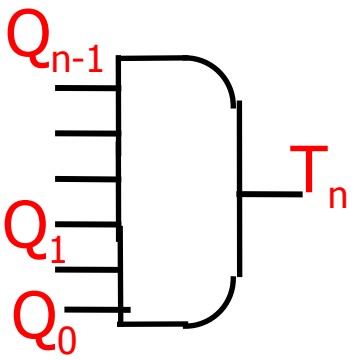


Come connettiamo T_n ?

- Vale il solito principio:
  - Q_n deve commutare nel momento in cui $Q_0=Q_1$ ..... $=Q_{n-1}$ transiscono da 1 a 0
- Riporto seriale:
  - AND2 connessi in cascata



- Riporto parallelo:
  - AND3.....n



Riporto seriale vs Riporto parallelo

- Riporto seriale:
  - semplicità costruttiva
  - Ritardo totale scala con i bit del contatore
  - t_{pr} di un FF + t_{pr} (di una porta AND !!!!) * n
- Riporto parallelo:
  - Diventa complicato quando i bit sono tanti
  - Ritardo totale fisso!!
  - t_{pr} di un FF + t_{pr} (di una porta AND !!!!) e BASTA !

Contatore Programmabile

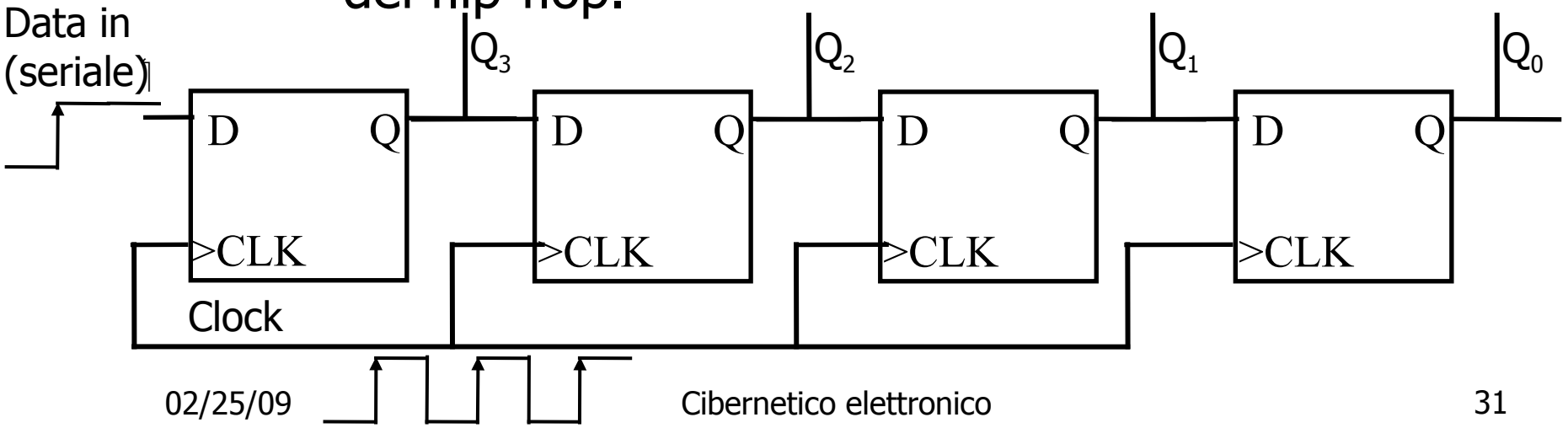
- Necessità di contare per un numero arbitrario di colpi di clock:
  - Combinazione sulle uscite → Counter Enable/Reset
  - Combinazione sugli ingressi → Data Load
  - Si usa sempre quest'ultima.

Tutti i contatori hanno un bus in ingresso per caricare un numero di "partenza" cioè da cui partire a contare:

- Linea di Load (sincrona o asincrona)
- Tecnica flessibile che permette di cambiare numero di conteggi senza modificare l'hardware

Shift registers

- Serializzatori e parallelizzatori (Ugh!)
- Vediamo per primo un Parallelizer:
  - Ingresso data (seriale), ingresso clock
  - Tipicamente usati i D-type flip-flop
  - La parola di uscita è formata dalle singole uscite Q dei flip-flop.



RS232 (protocollo seriale)

	9-Pin	25-Pin
Carrier Detect	1	8
Recieve Data	2	3
Transmit Data	3	2
Data Terminal Ready	4	20
System Ground	5	7
Data Set Ready	6	6
Request to Send	7	4
Clear to Send	8	5
Ring Indicator	9	22

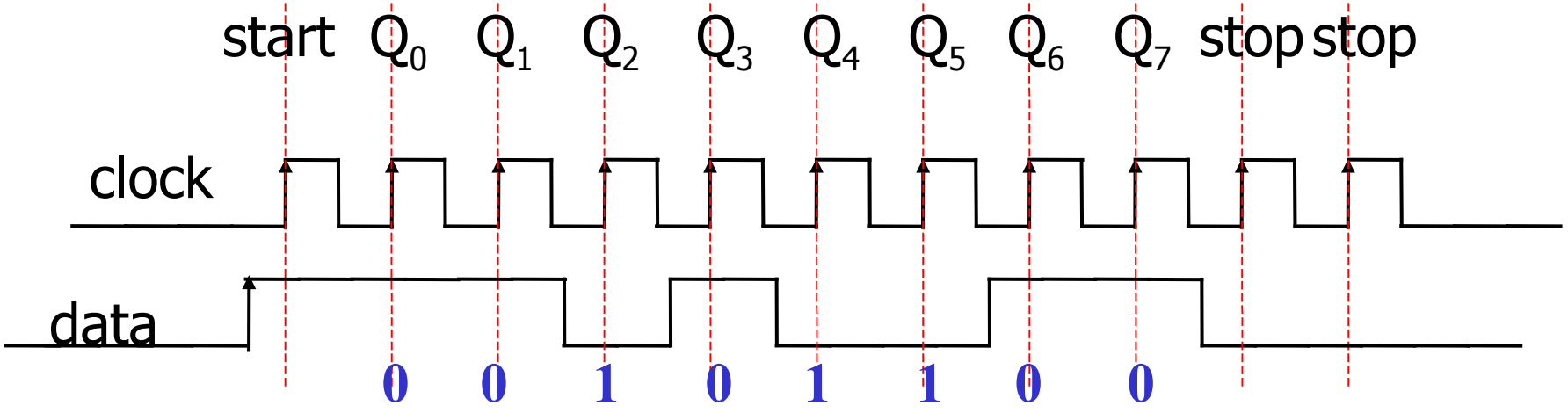
- Connessioni minime, 3 !!!
- Tx,Rx,GND.

RS232 (protocollo seriale)

- Tra i due apparecchi collegati bisogna stabilire prima:
  - frequenza dei dati (baud rate) (9600, 56K)
  - parità (normalmente off)
  - stop bit (normalmente uno o due)
- Parametri fissi e non negoziabili:
  - bit vengono trasmessi a gruppi di 8
  - in inizio trasmissione viene inviato uno start bit
  - a fine trasmissione vengono trasmessi il bit di parità e gli stop bit
- Il clock viene generato dal ricevente appena riceve lo start bit !

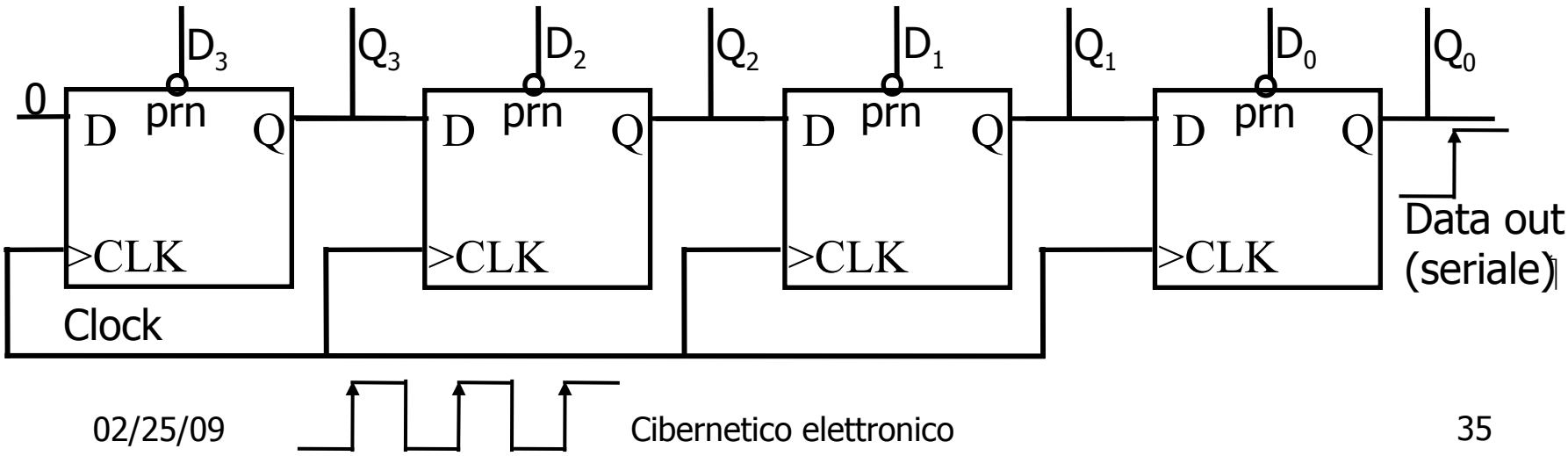
RS232 (protocollo seriale)

- Quando la linea di data va a zero (start bit)
  - Inizia la trasmissione dati
  - Il ricevitore genera un clock interno che campiona la linea dati con uno shift-register
  - Se il protocollo prevede 1 start, 2 stop e no parity, il clock interno dovrà fornire 11 colpi.



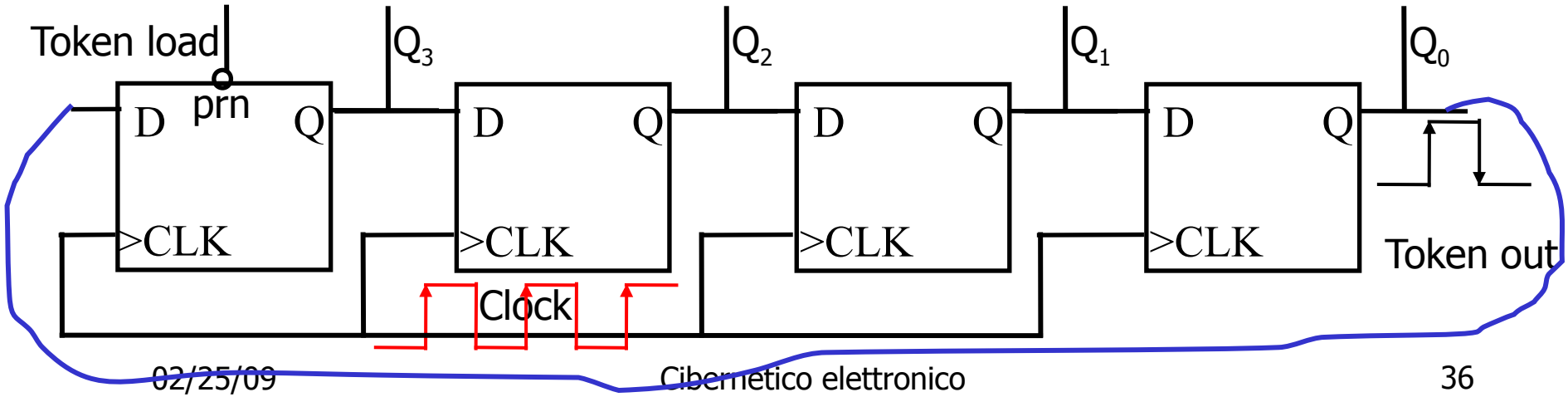
Shift registers

- Abbiamo visto un SIPO (Serial In – Parallel Out)
- Ora vediamo un PISO (Parallel In – Serial Out)
  - La parola di *n* bit viene caricata (load) su *n* flip-flop dello shift-register.
  - Sotto l'azione del clock, l'uscita dell'ultimo flip-flop restituisce la parola caricata in modo seriale.



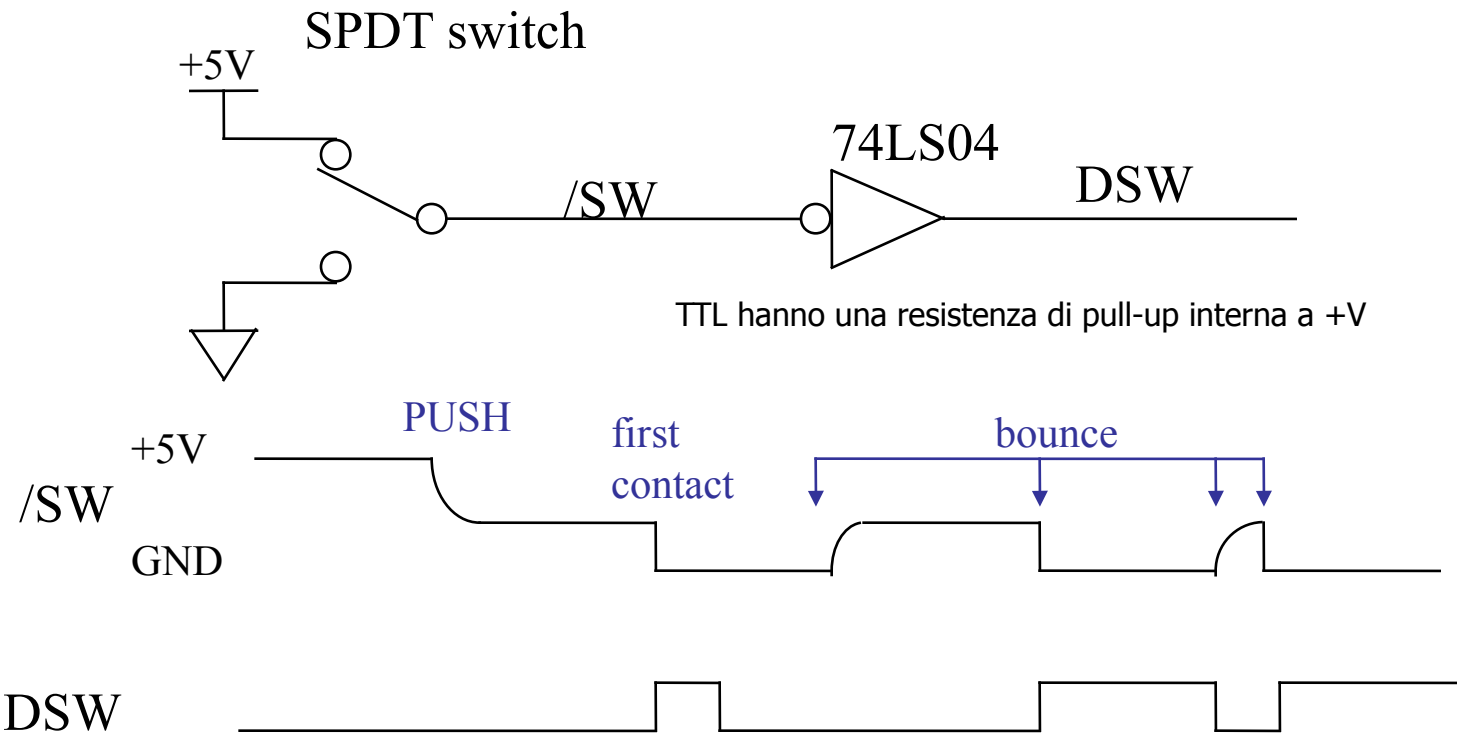
Ring counter

- Shift register con l'uscita connessa all'ingresso
  - Il "token" viene caricato sul primo flip-flop
- Ad ogni colpo di clock:
  - Il token passa da un flip-flop a quello successivo
  - Quando arriva all'ultimo flip-flop, viene ricaricato sul flip-flop iniziale
- Es. La sequenza di un ring counter a 4 bit:
  - 1000 – 0100 – 0010 – 0001 – 1000 ecc. ecc.



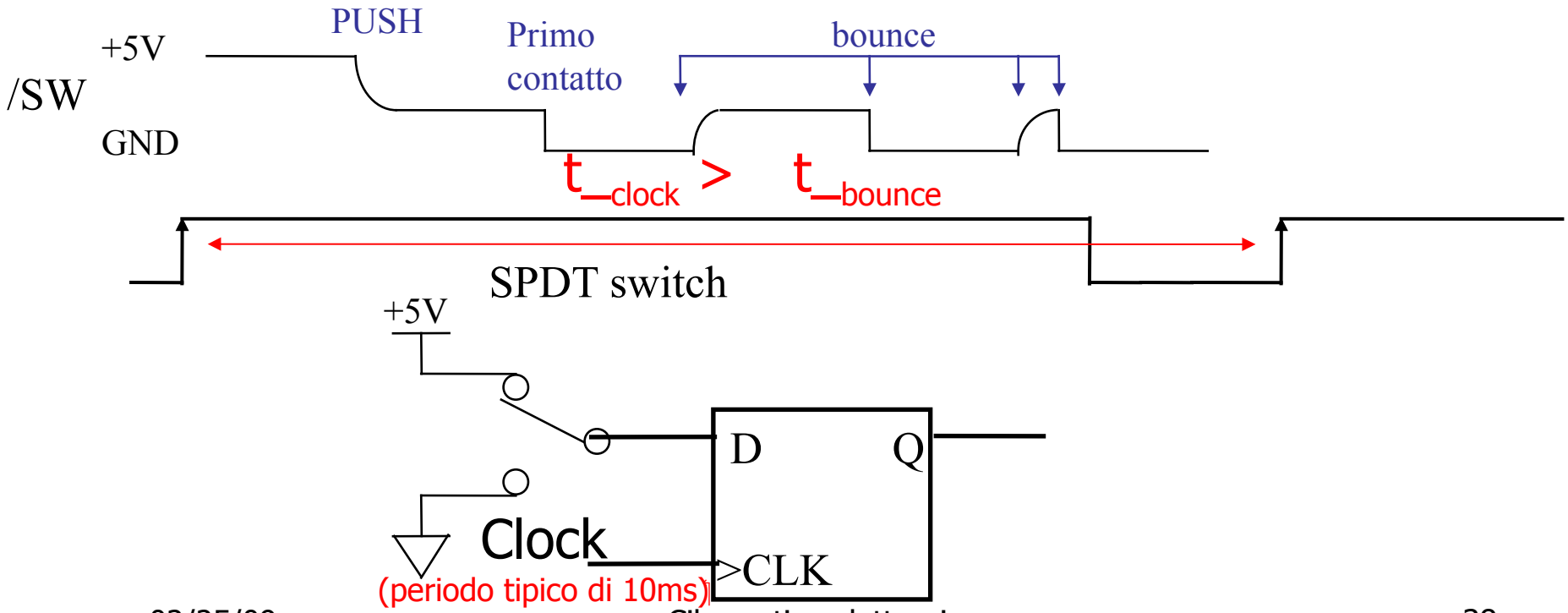
Circuito di debouncer !

- Al momento della chiusura dell'interruttore si hanno vari rimbalzi tra lel lamelle degli interruttori



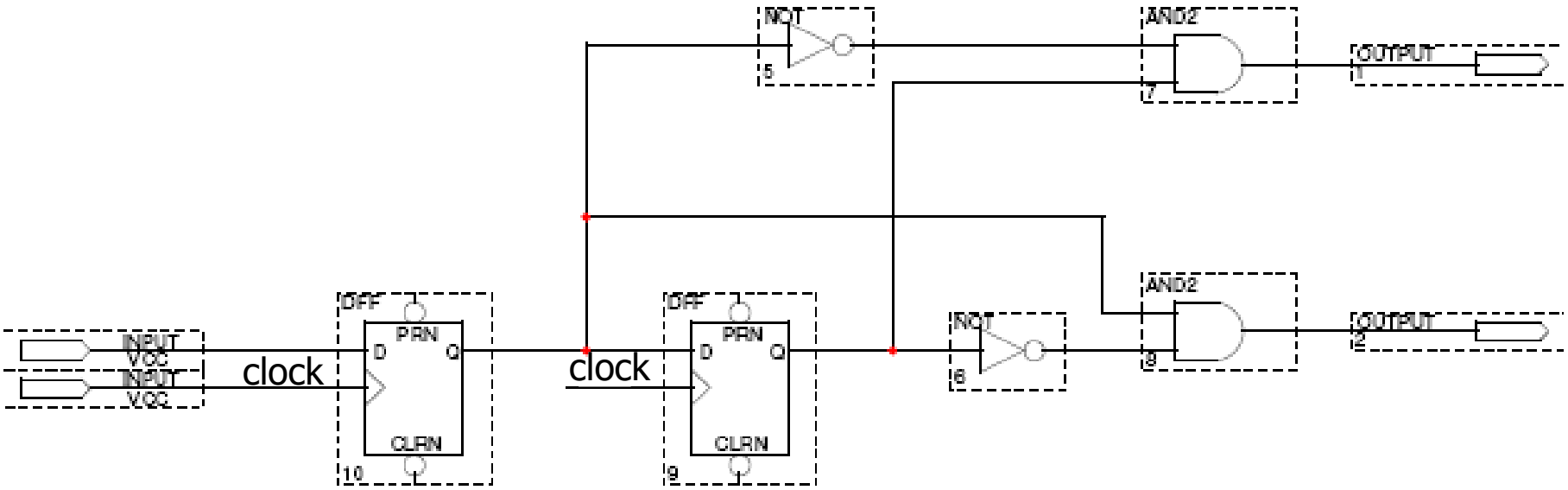
Campionare gli ingressi

- Usare un clock "lento"
  - Periodo più lento della durata dei "RIMBALZI"



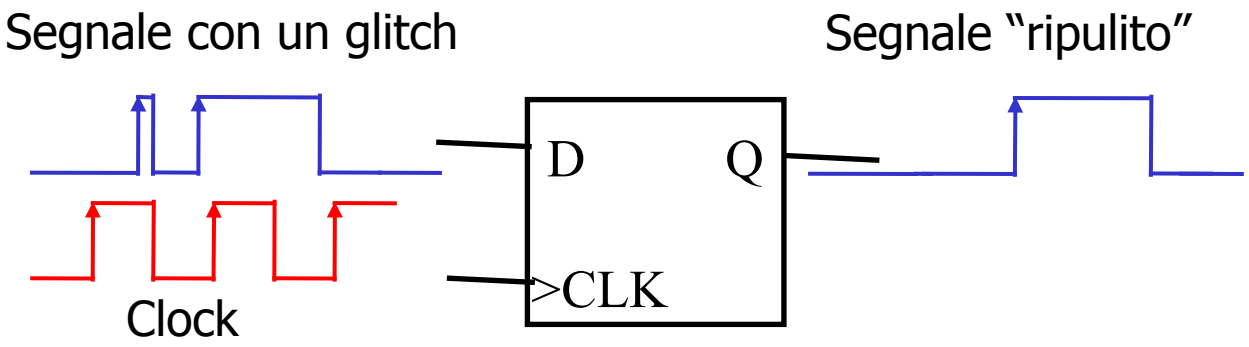
Differenziatore

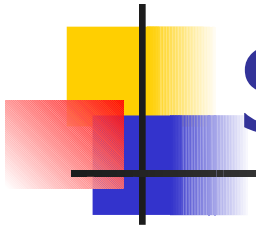
- Riconosce i fronti di salita e di discesa



De-glitcher

- A volte, nonostante i nostri sforzi, può capitare di avere dei "glitch" all'uscita del nostro circuito digitale.
  - I "glitch" sono delle transizioni di brevissima durata, non volute e non previste.
- Se la nostra logica è comunque tutta pilotata da un Master clock, il seguente circuito può servire per rimuovere i glitch:
  - Il D type FF usa come ingresso di clock il master clock



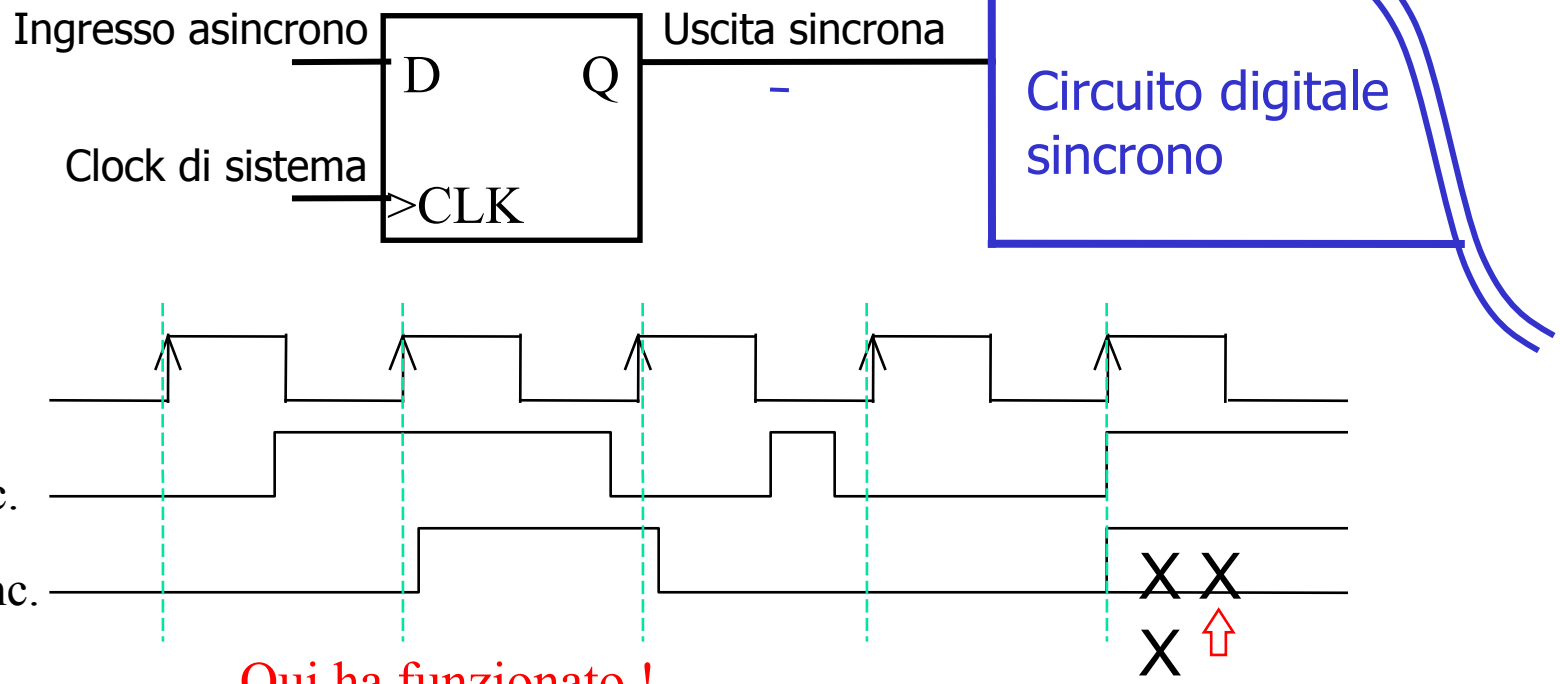


Sincronizzatore

- Input asincroni:
  - Sono sempre presenti in un sistema digitale:
    - Ingresso da tastiera, pulsanti, comparatori, ecc.
  - Normalmente sono molto più lenti del clock di sistema
  - Non è vitale sapere quando esattamente (al ns) è avvenuto l'evento
  - PER ESSERE USATI DEVONO ESSERE SINCRONIZZATI !

- Si usa sempre un D type flip-flop
  - Clock di sistema (Master clock)

Sincronizzatore singolo



Qui ha funzionato !

Qui NO ! Metastabile ..
Per sfortuna il fronte di salita dell'ingresso asincrono coincide con il fronte di salita del clock

Sincronizzatore doppio

- Pagando un piccolo prezzo:
  - Aumento della latenza
- Raddoppiamo la catena:
  - L'eventuale metastabilità sarà terminata quando ricampioneremo il segnale
- Prezzo pagato:
  - Sono necessari due colpi di clock prima di avere un segnale in uscita ! (Aumento della latenza)!