Calcolatori Elettronici A
a.a. 2008/2009

RETI LOGICHE:
RETI SEQUENZIALI

Massimiliano Giacomin

LIMITI DELLE RETI COMBINATORIE e RETI SEQUENZIALI

- Le reti combinatorie sono senza retroazione: il segnale di uscita da una porta non può mai riapparire come segnale di ingresso della stessa porta

  $\Rightarrow$ impossibile “riutilizzare” la stessa porta in tempi diversi

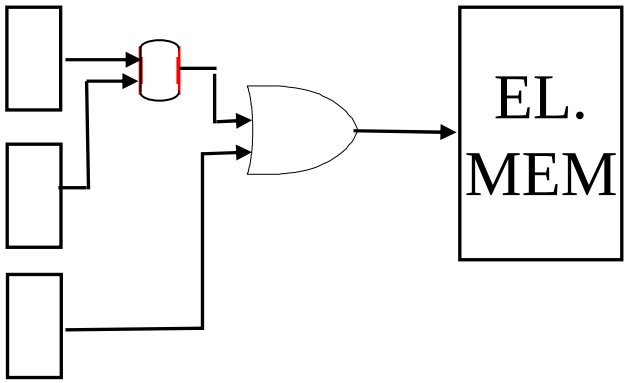
  $\Rightarrow$ per realizzare funzioni complesse sarebbe necessario usare reti con un numero molto elevato di porte

- Per riutilizzare le stesse porte in fasi successive del processo di calcolo: introduciamo ***elementi di memoria*** (le cui uscite dipendono sia dagli ingressi sia dal “valore memorizzato” al loro interno)
- Gli elementi di memoria memorizzano lo ***stato*** del sistema
- Reti con elementi di memoria (o di stato): dette ***sequenziali***

Il ruolo del segnale di clock

- Con elementi di memoria: diventa cruciale controllare gli istanti in cui i valori in ingresso agli elementi di memoria vengono memorizzati

Esempio

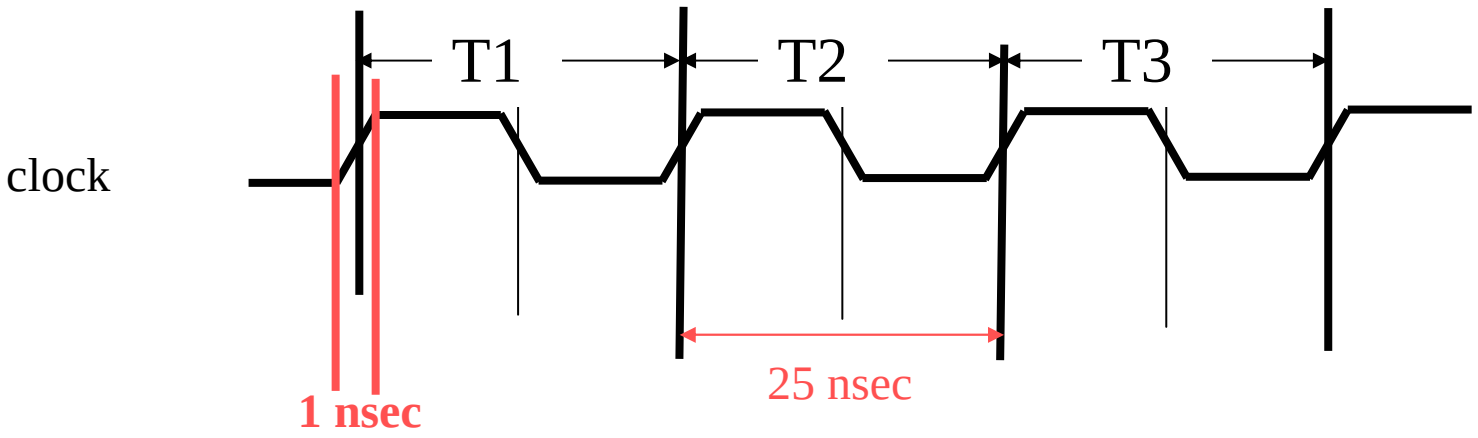


Quando memorizza l'ingresso?

Si utilizza un circuito in grado di generare un ***segnale di clock***
(clock pulse generator)

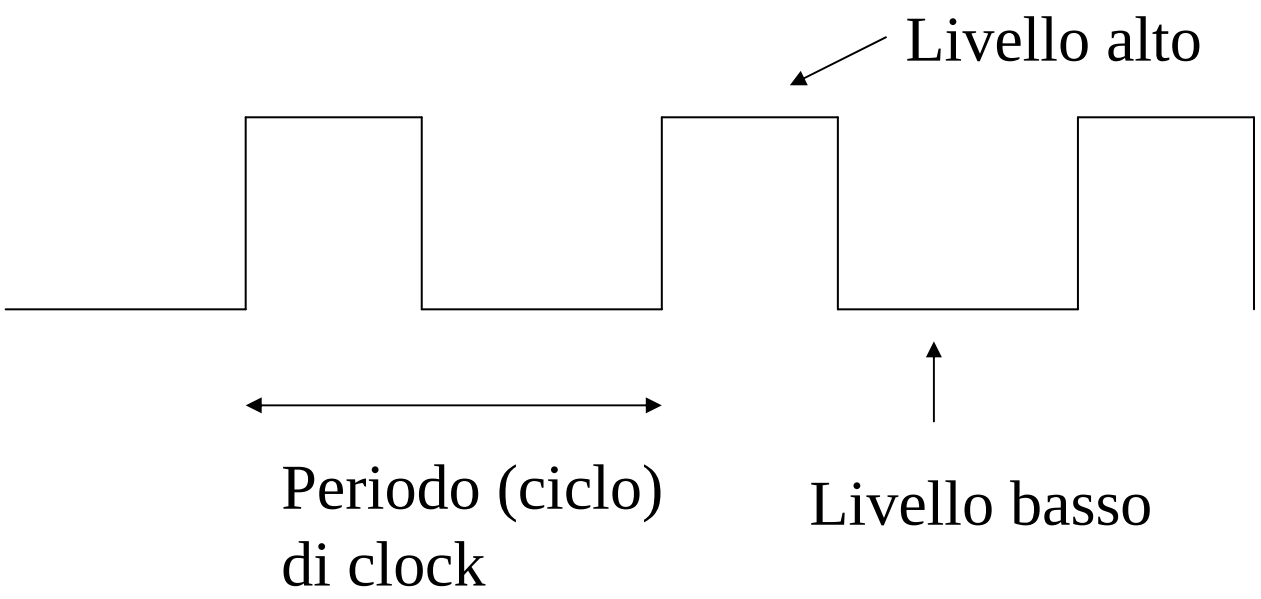
Il ruolo del segnale di clock è sincronizzare la rete: le reti che lo utilizzano sono dette **sincrone**

Il segnale di clock



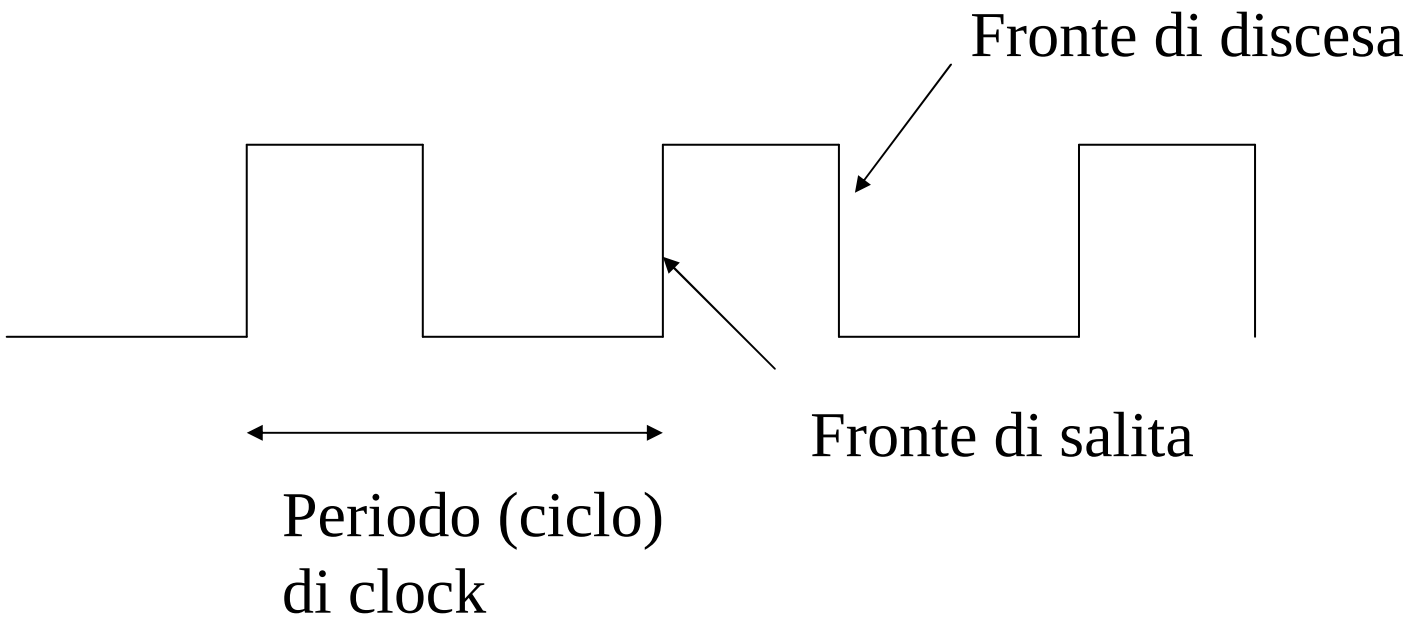
- Segnale che evolve con un *periodo* (tempo di ciclo) predeterminato e costante (intervallo di tempo fra 2 segnali di clock consecutivi)
- La *frequenza di clock* è l'inverso del periodo (numero di cicli di clock al secondo)
- Esempio: frequenza periodo 25 ns, frequenza 40 Mhz
- Esistono due ***metodologie di temporizzazione*** (approccio che determina quando gli elementi di memoria cambiano stato)

Metodologia di temporizzazione sensibile ai livelli



In questa metodologia si utilizzano elementi di memoria le cui **variazioni** (memorizzazione di un valore) avvengono in corrispondenza dei ***livelli alti*** o di quelli ***bassi***

Metodologia di temporizzazione sensibile ai fronti



In questa metodologia uno dei due fronti è detto ***attivo*** e causa le ***variazioni di stato***: il contenuto degli elementi di stato può cambiare **solo** in corrispondenza del fronte attivo del clock

GLI ELEMENTI DI MEMORIA: LATCH e FLIP-FLOP

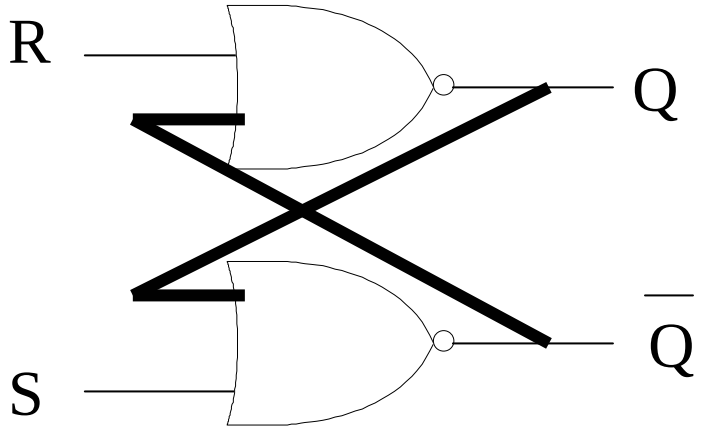
- ***Latch***: elemento di memoria che memorizza il valore presente in un dato ingresso in corrispondenza di determinati livelli dei segnali presenti in ingresso

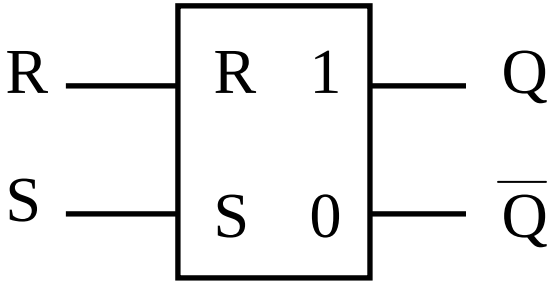
  (ad esempio: quando il clock è alto)

- ***Flip-flop***: elemento di memoria che può cambiare il valore memorizzato solo in corrispondenza di un fronte (di salita o di discesa) del segnale di clock

Latch SR (set-reset)

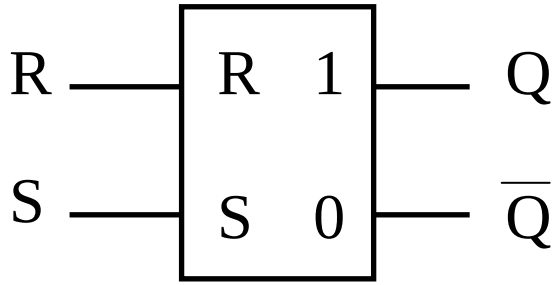
- Elemento di memoria per la memorizzazione di 1 bit
  (deve ricordare due situazioni)





- Ha 2 ingressi, R ed S, e si comporta secondo le regole:
  - R=1, S=0 memorizza 0
  - R=0, S=1 memorizza 1
  - R=S=0 ricorda la situazione corrente
  - R=S=1 non è permesso

Tabella di transizione



R	S	Q(t)	Q($t+\Delta t$)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	-
1	1	1	-

Per descriverlo…

Devo specificare:

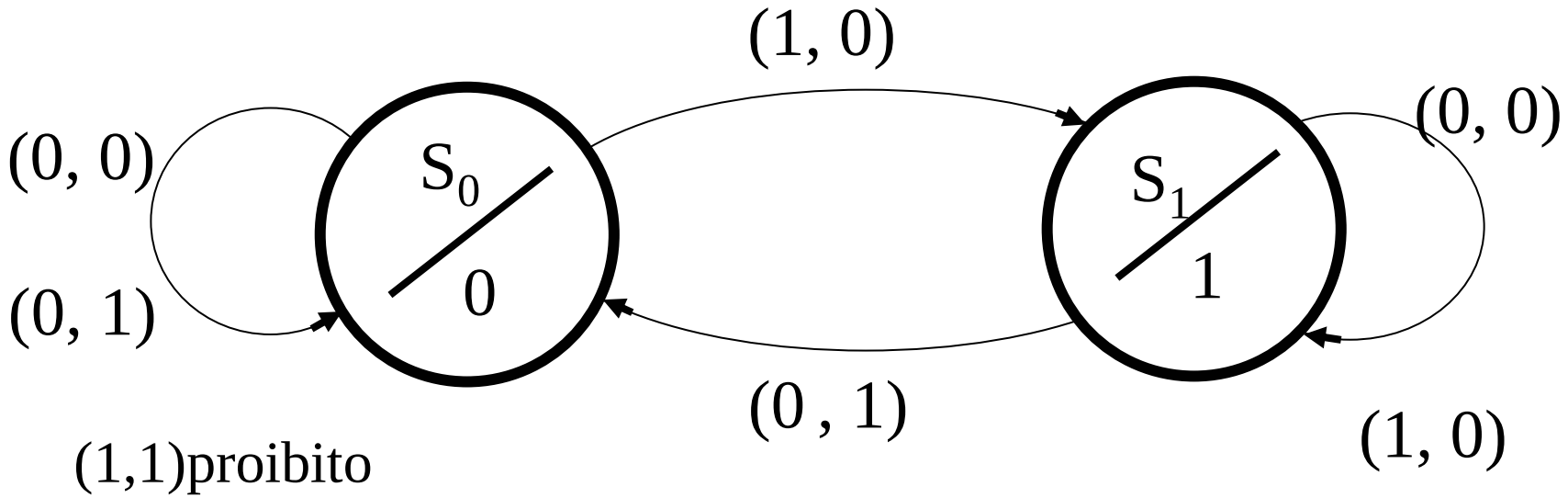
- **I** = {0,1} insieme dei simboli di ingresso
- **U** = {0,1} insieme dei simboli di uscita
- **S** = $\{S_0, S_1\}$ insieme degli stati (situazioni da ricordare)
- δ funzione di transizione che specifica il nuovo stato in base allo stato corrente e all'ingresso (la tabella)
- ω funzione di uscita che specifica il valore di uscita in base allo stato corrente e all'ingresso (in questo caso l'uscita dipende solo dallo stato)

$\Rightarrow$ Automa $A = \langle I, U, S, \delta, \omega \rangle$

Questi 5 insiemi possono essere descritti **anche** graficamente

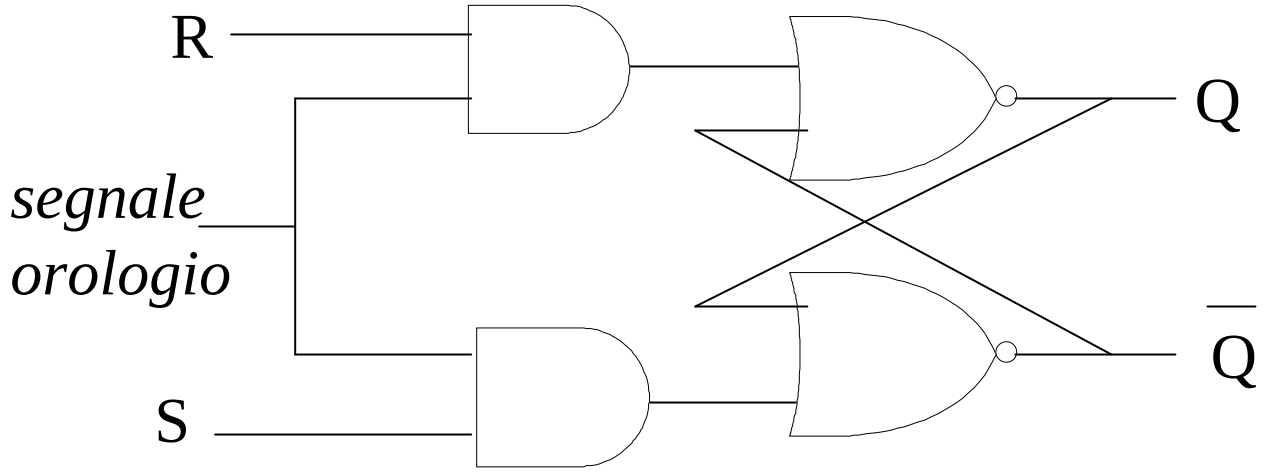
Descrizione grafica

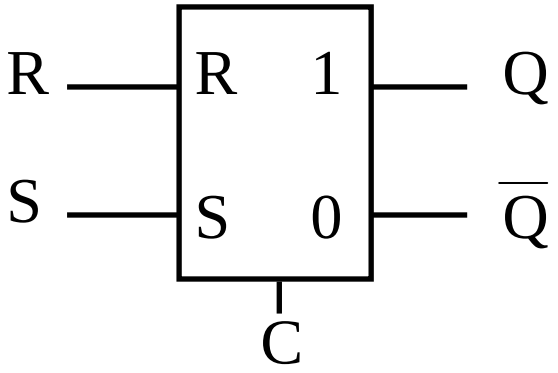
- Ingressi: (s,r)
- Uscita Q indicata negli stati (S_0 e S_1)



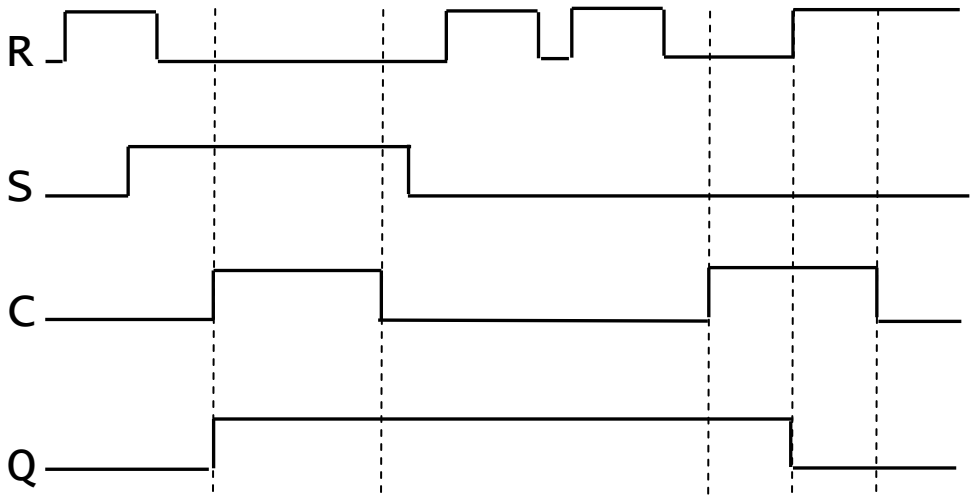
Latch SR sincrono

• Ha in ingresso un segnale di clock e la variazione di stato viene attivata dal clock





Esempio



NB: si vede che le variazioni possono avvenire solo nel livello attivo del clock

Latch di tipo D(elay) sincrono

- Memorizza il valore del dato D presente in ingresso
- Permette di risolvere l'ambiguità del latch RS (R=S=1 non si può verificare)

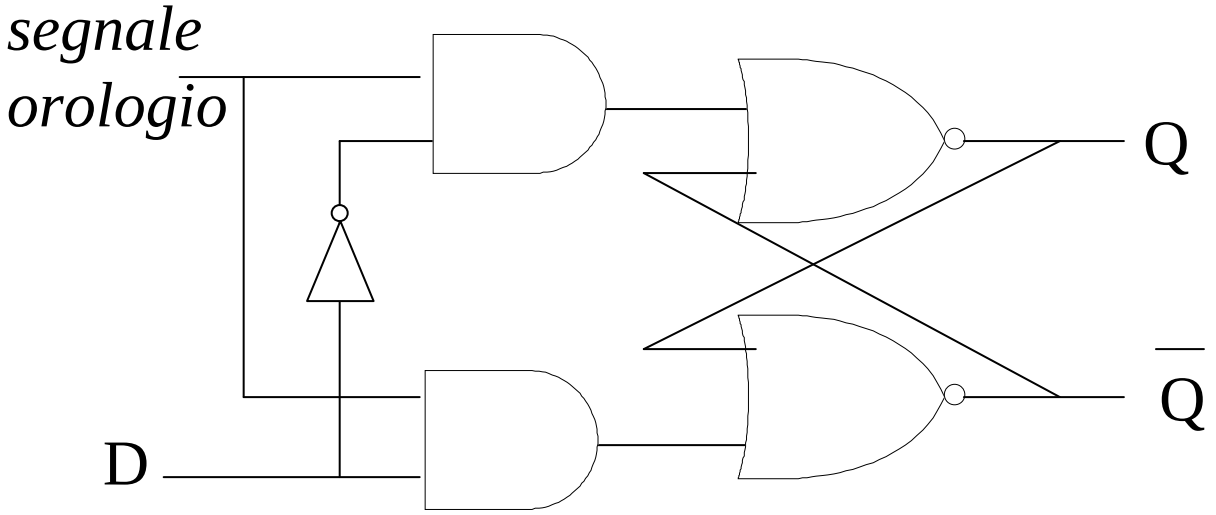


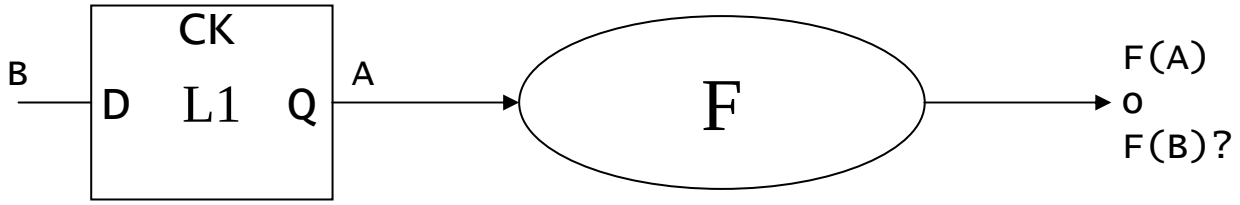
Tabella di transizione

CK	D	Q(t)	Q($t+\Delta t$)
1	0	x	0
1	1	x	1
0	x	y	y

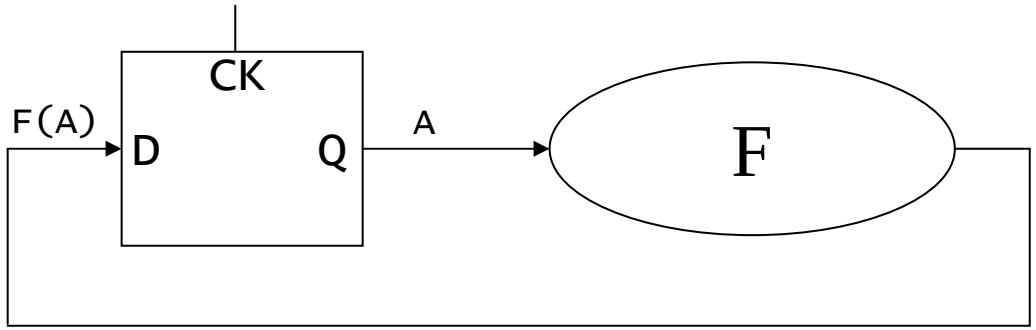
Limiti dei latch e della temporizzazione sensibile ai livelli

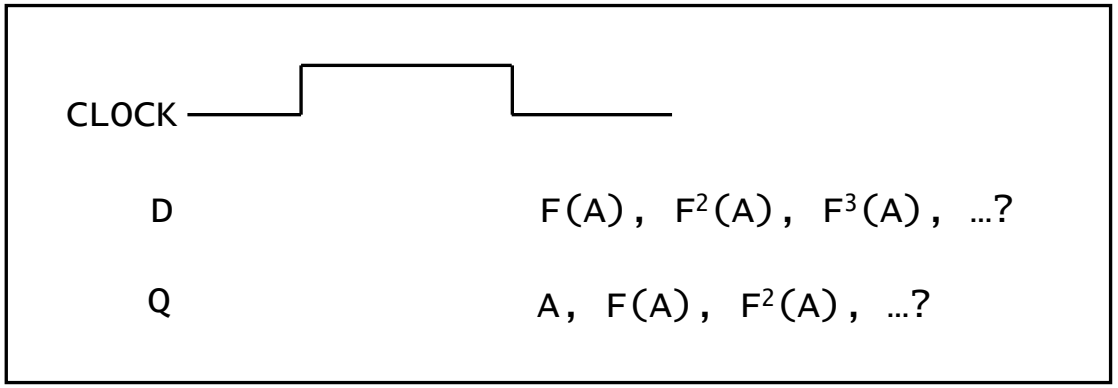
- Il latch non permette, *durante lo stesso ciclo di clock*, di leggere il segnale in uscita e di modificare quello in ingresso:
  - se il segnale d'ingresso è applicato *prima* che sia stato letto quello di uscita, il latch potrebbe commutare ed il segnale letto potrebbe non essere quello desiderato

Esempio 1



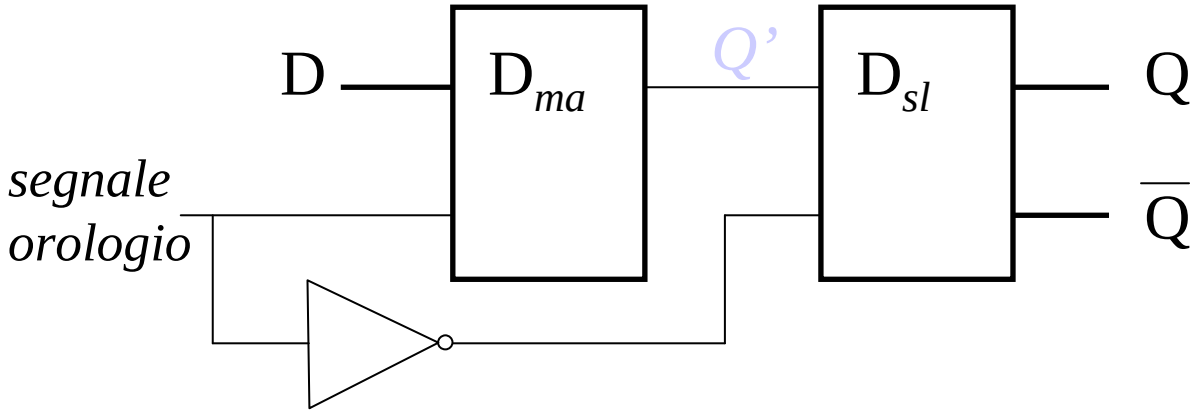
Esempio 2



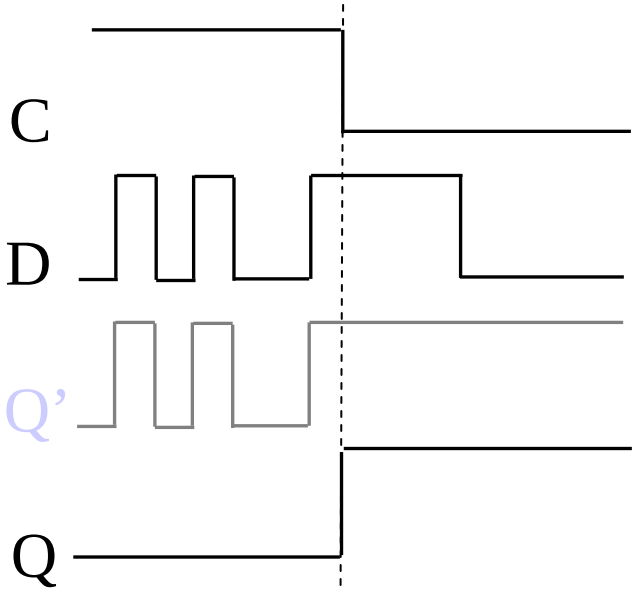


- Il **flip-flop** permette la scrittura e la lettura simultanea:
  nel seguito useremo sempre flip-flop come elementi di stato
- Flip-flop possono essere realizzati con circuiti **master-slave**

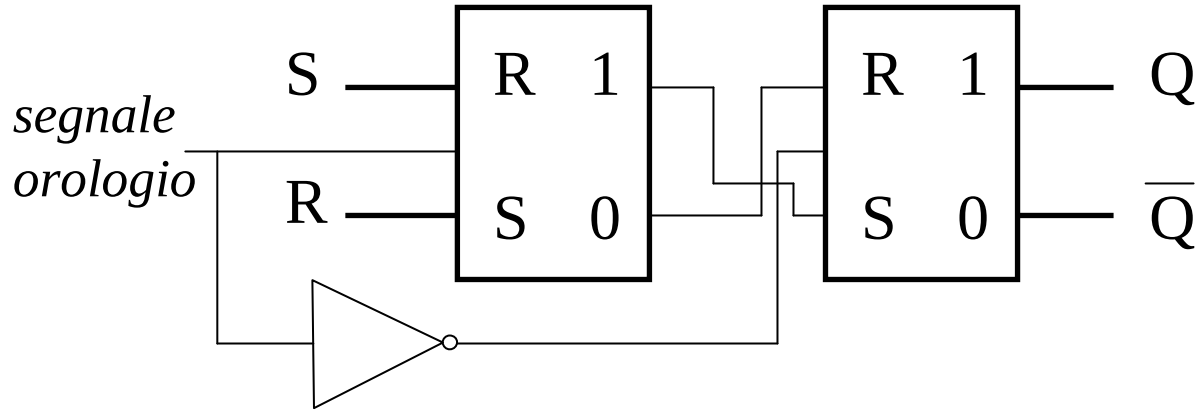
Flip-flop master-slave



- 2 latch in serie: il latch master riceve gli ingressi e il latch slave produce le uscite
- Alla "fine" del livello alto il master memorizza il valore disponibile in uscita a partire dal livello basso (FF attivo sul fronte di discesa)

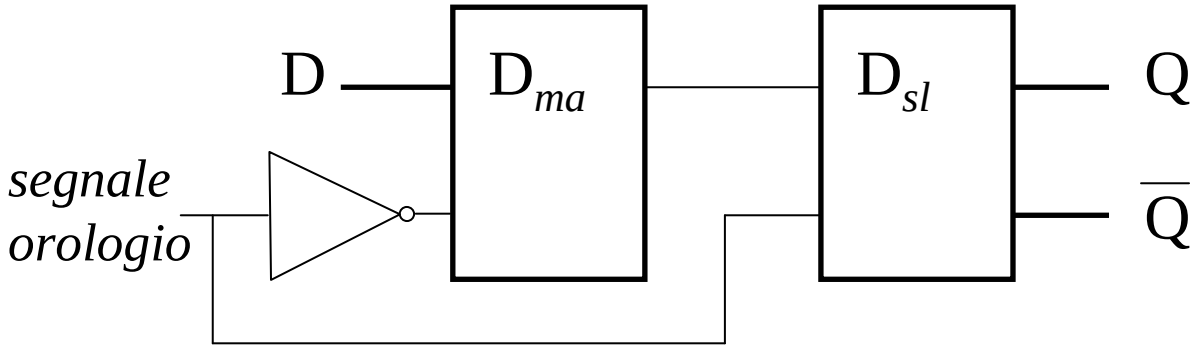


NB: usando latch S-R al posto di latch D

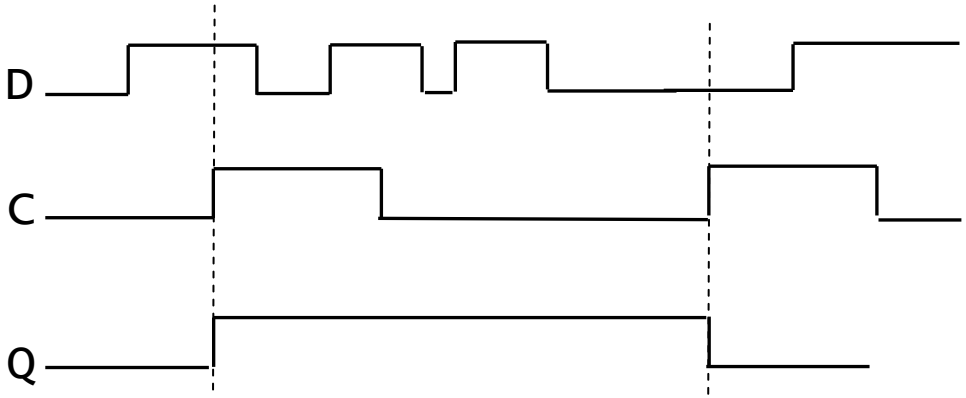


(FF di tipo S-R attivo sul fronte di discesa)

NB: per realizzare un FF sensibile al fronte di salita

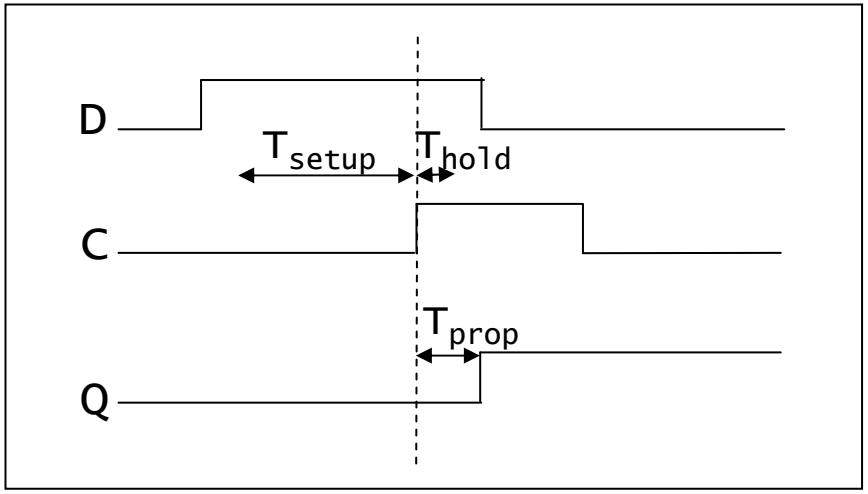


Esempio



TEMPO DI SET-UP, DI HOLD, DI PROPAGAZIONE

- Dato che l'ingresso viene "campionato" in corrispondenza del fronte attivo del clock, è facile capire che esso deve essere:
  - stabile almeno da un intervallo di tempo precedente al fronte (T_{setup})
  - stabile per almeno un intervallo di tempo successivo al fronte (T_{hold})
- Inoltre, l'uscita sarà stabile al più dopo un certo tempo di propagazione

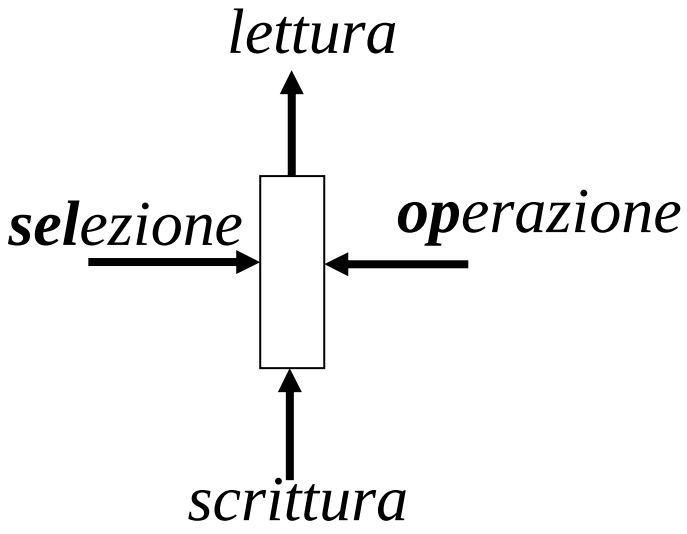


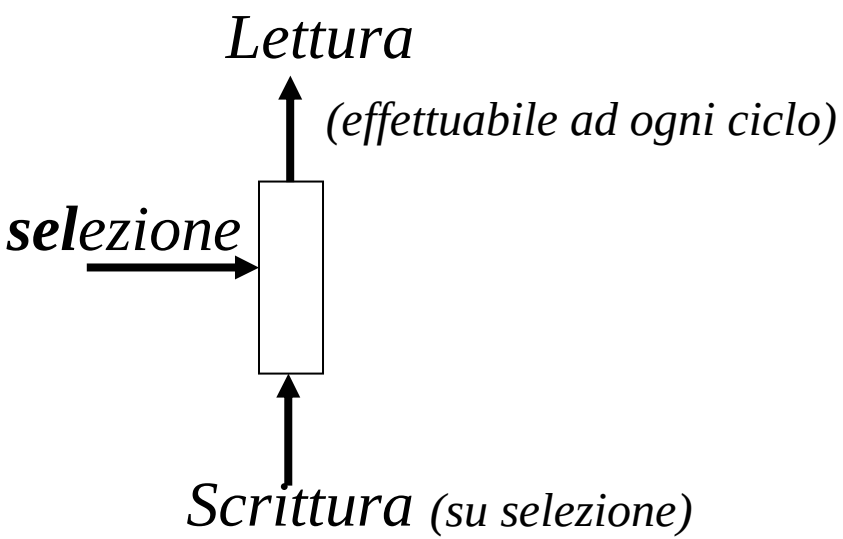
- Tutti i tempi riferiti al fronte del clock
- In generale $T_{hold} < T_{prop}$

- T_{setup}, T_{hold}, T_{prop} caratterizzano il FF (livello circuitale-fisico)

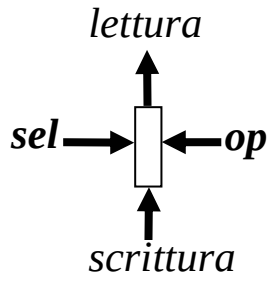
Cella elementare di memoria

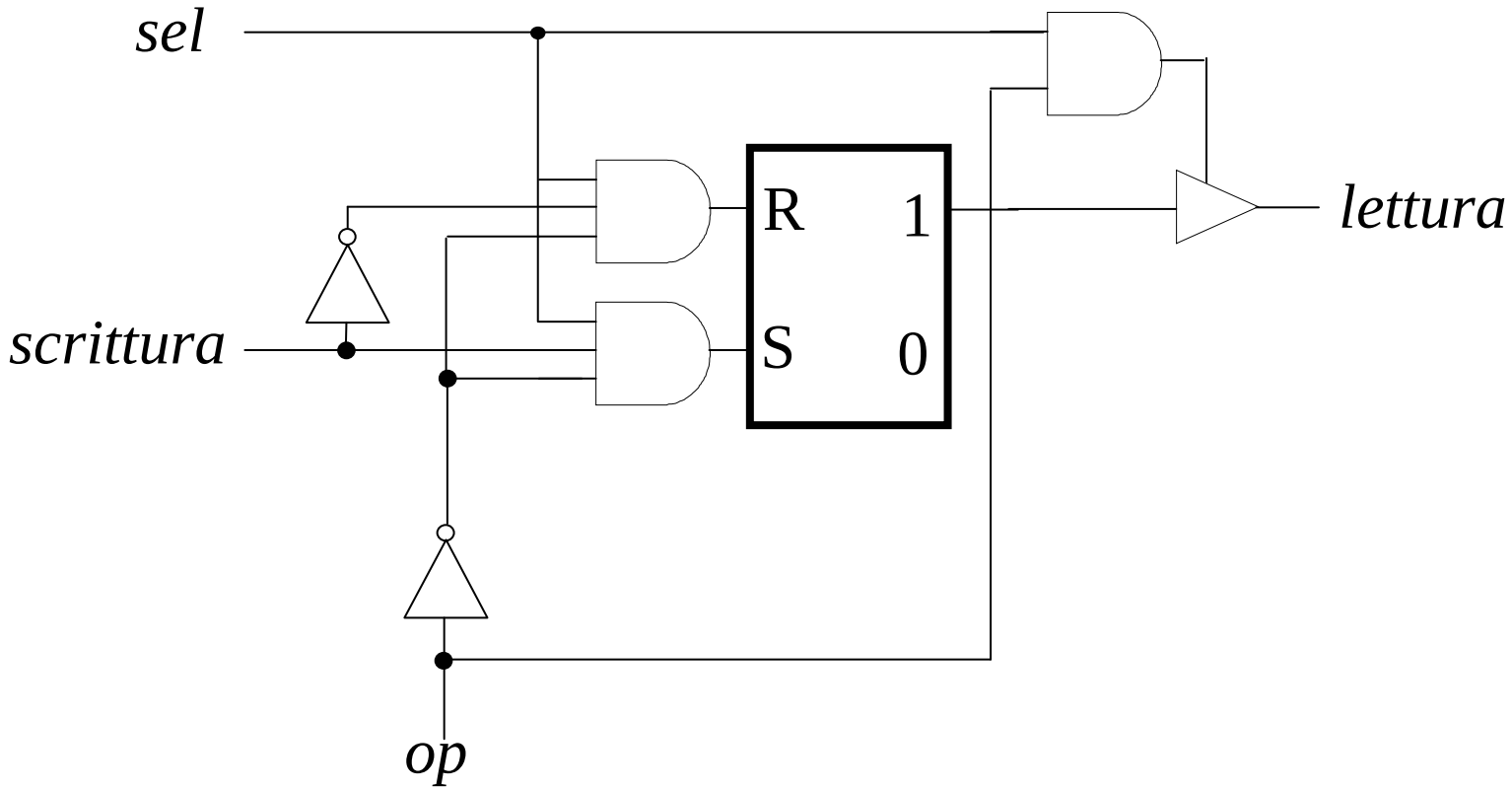
- Utilizza un flip-flop per consentire di:
  - conservare nel tempo l'informazione memorizzata
  - leggere il contenuto su un'opportuna linea di uscita
  - scrivere nella cella un valore arbitrario presente su una linea di scrittura
- Due schemi funzionali tipici di una cella elementare





Schema funzionale 1

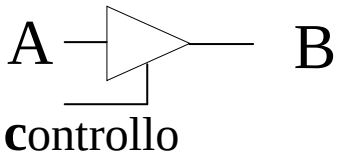


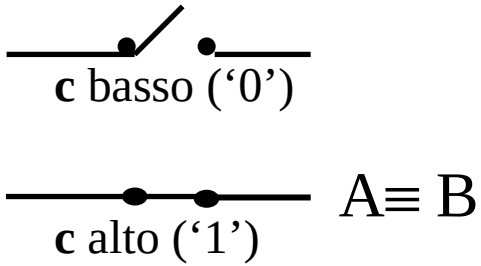


NB: non indicato il clock. Lettura asincrona, scrittura sul fronte attivo

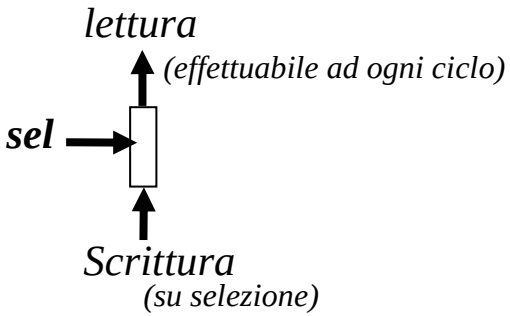
NB: nel precedente lucido è stato utilizzato un buffer tri-state

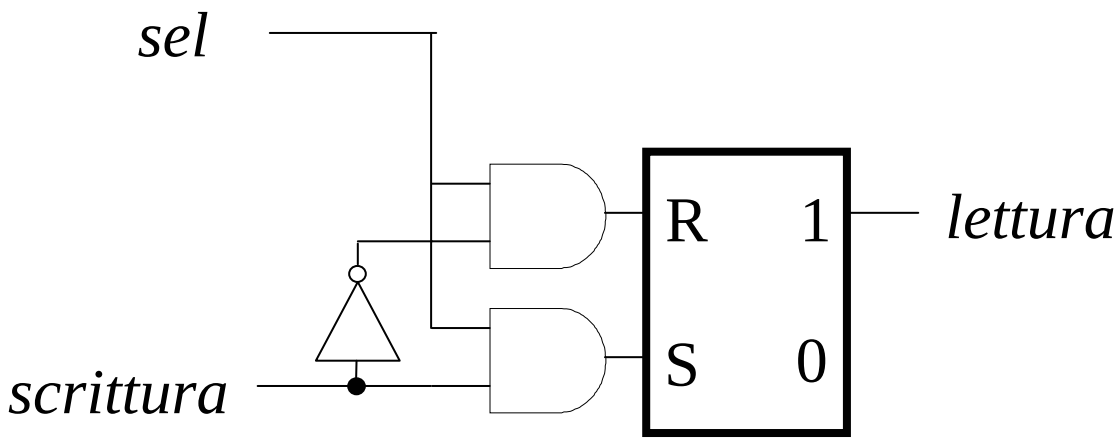
Buffer tri-state





Schema funzionale 2

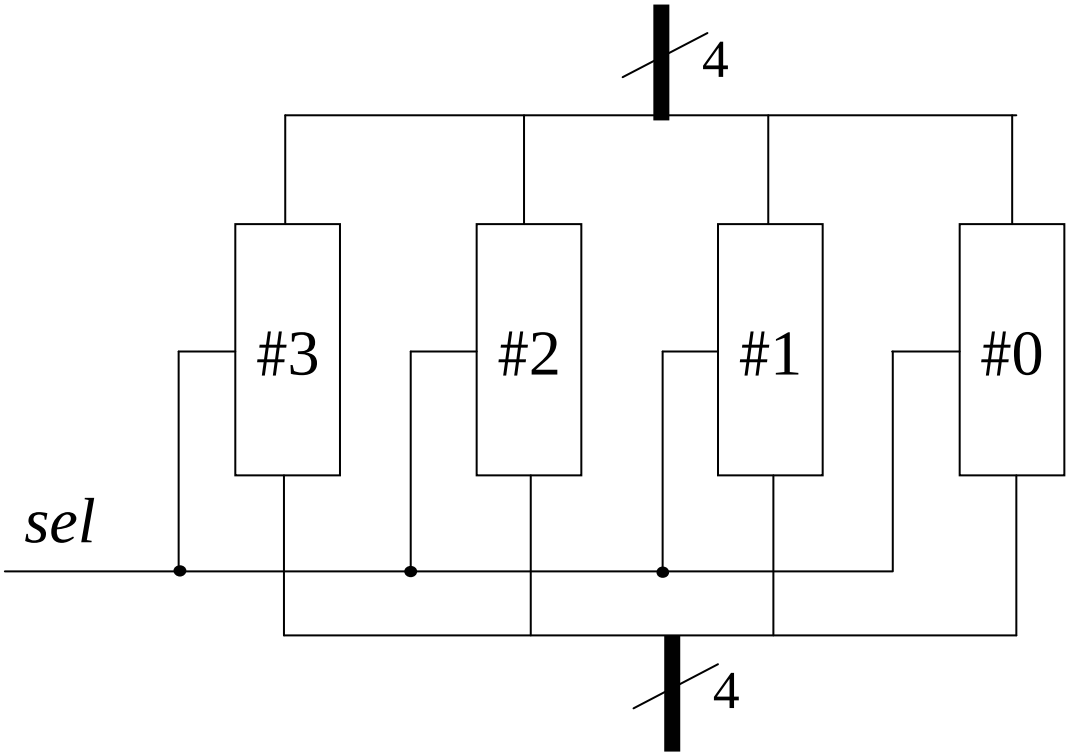


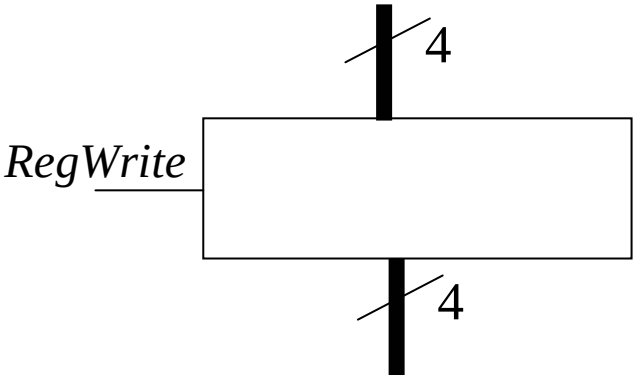


NB: non indicato il clock. Lettura asincrona, scrittura sul fronte attivo

Registro

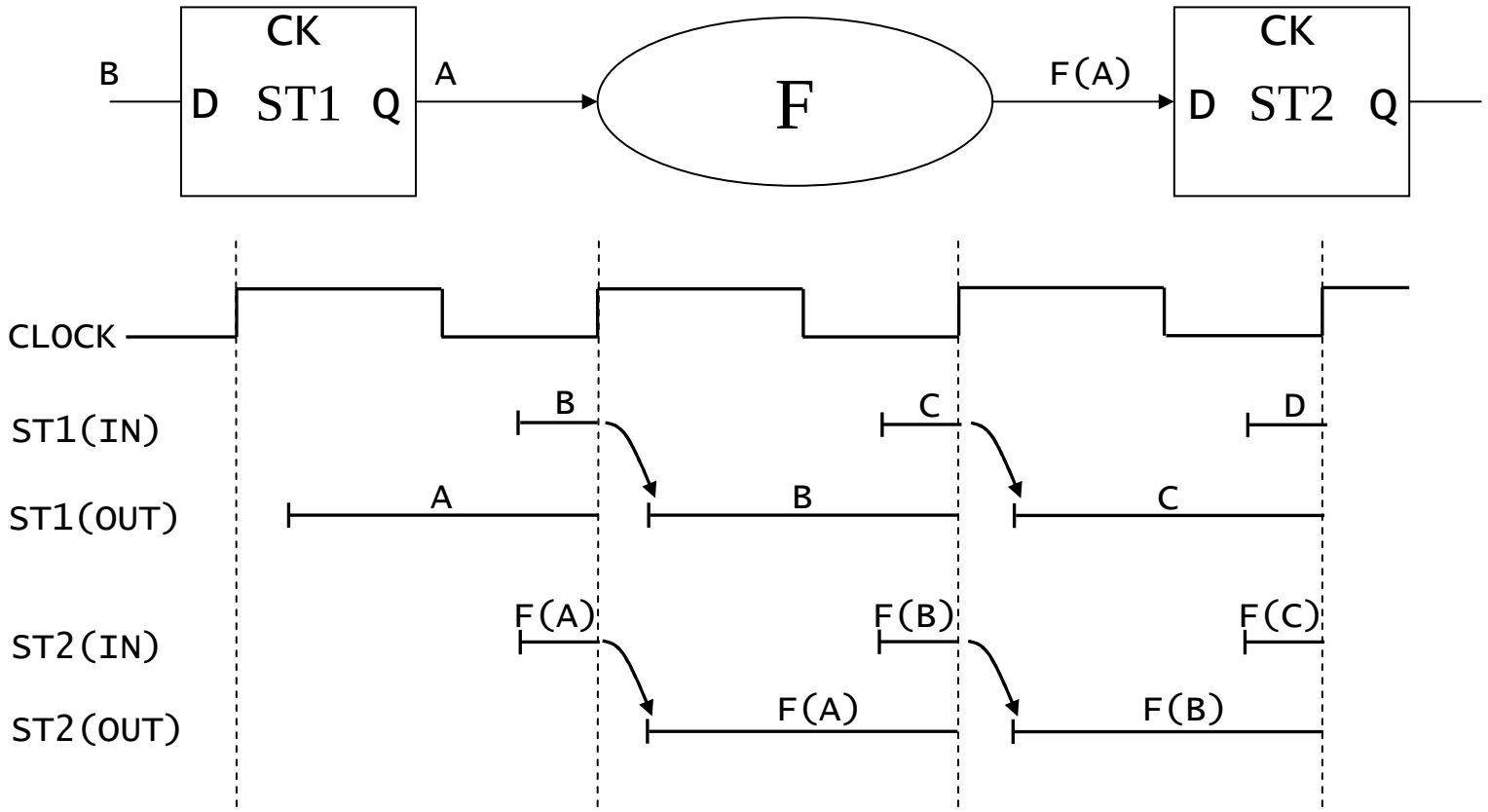
- Una cella complessa in grado di memorizzare *n* bit
- Unità indivisibile di informazione: selezione unica
- *n* linee di ingresso e *n* linee di uscita
- Consente di scrivere nella cella un valore arbitrario a *n* bit presente su una linea di scrittura



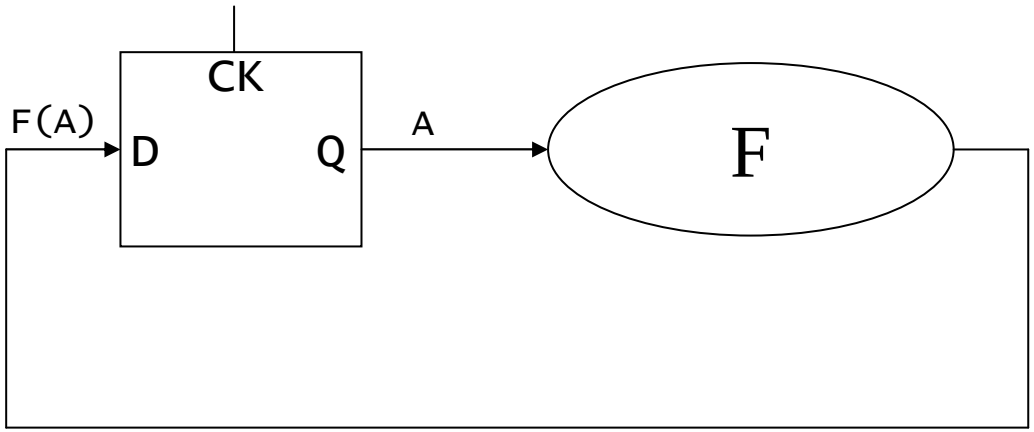


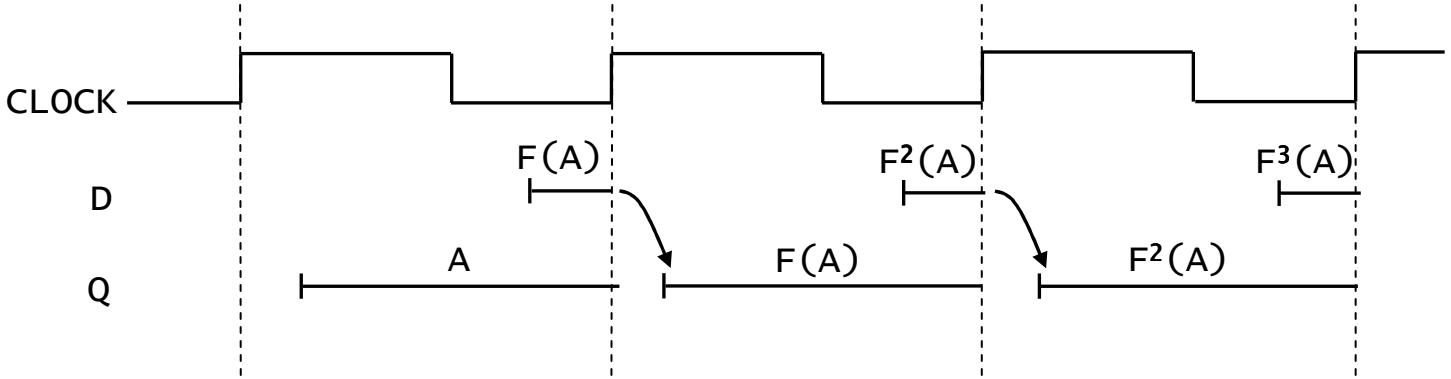
TEMPORIZZAZIONE SENSIBILE AI FRONTI

• Sistemi sincroni: segnale di clock comune determina aggiornamento elementi di stato

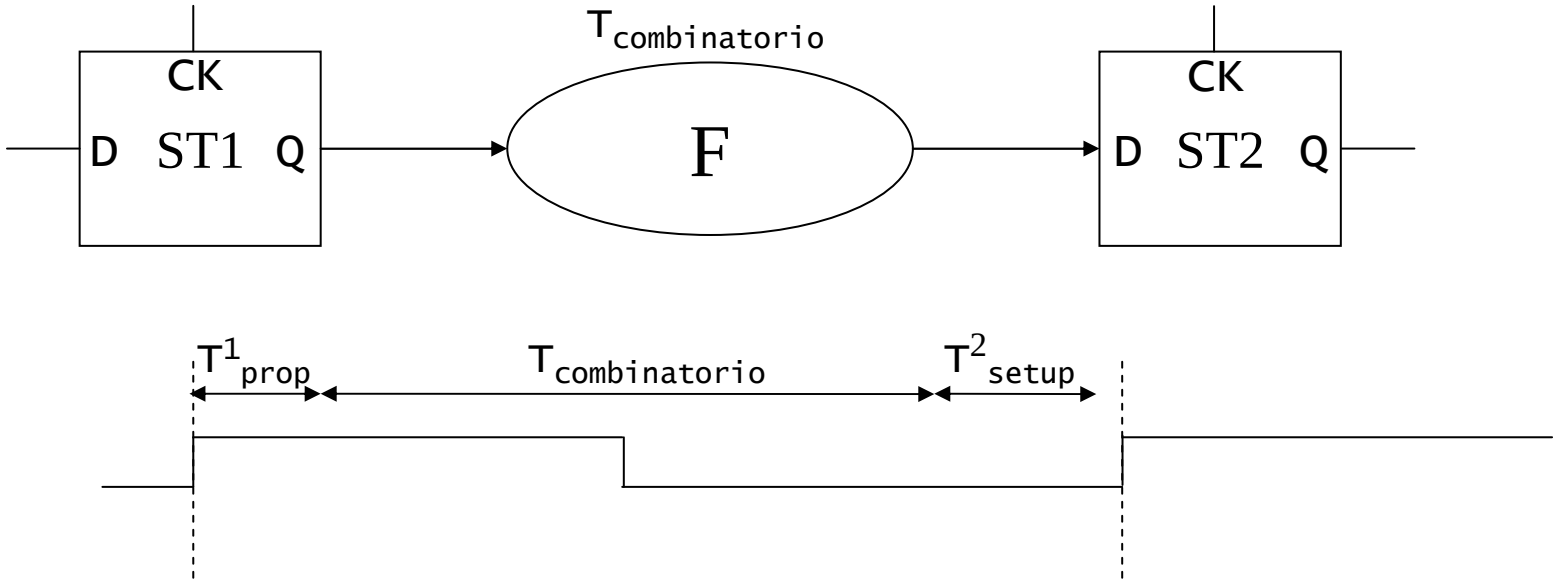


• Al fronte di clock, un elemento di stato memorizza il valore di ingresso
• Nel periodo di clock, un nuovo valore di ingresso viene propagato dalla parte combinatoria e sarà disponibile al successivo fronte


CK
F(A)
D
Q
A
F


CLOCK
D
Q
F(A)
F²(A)
F³(A)
A
F(A)
F²(A)

TEMPORIZZAZIONE E VINCOLI TEMPORALI



- Dopo $T^1_{prop} + T_{combinatorio}$, ingresso a ST2 è stabile: anticipo di almeno T^2_{setup}

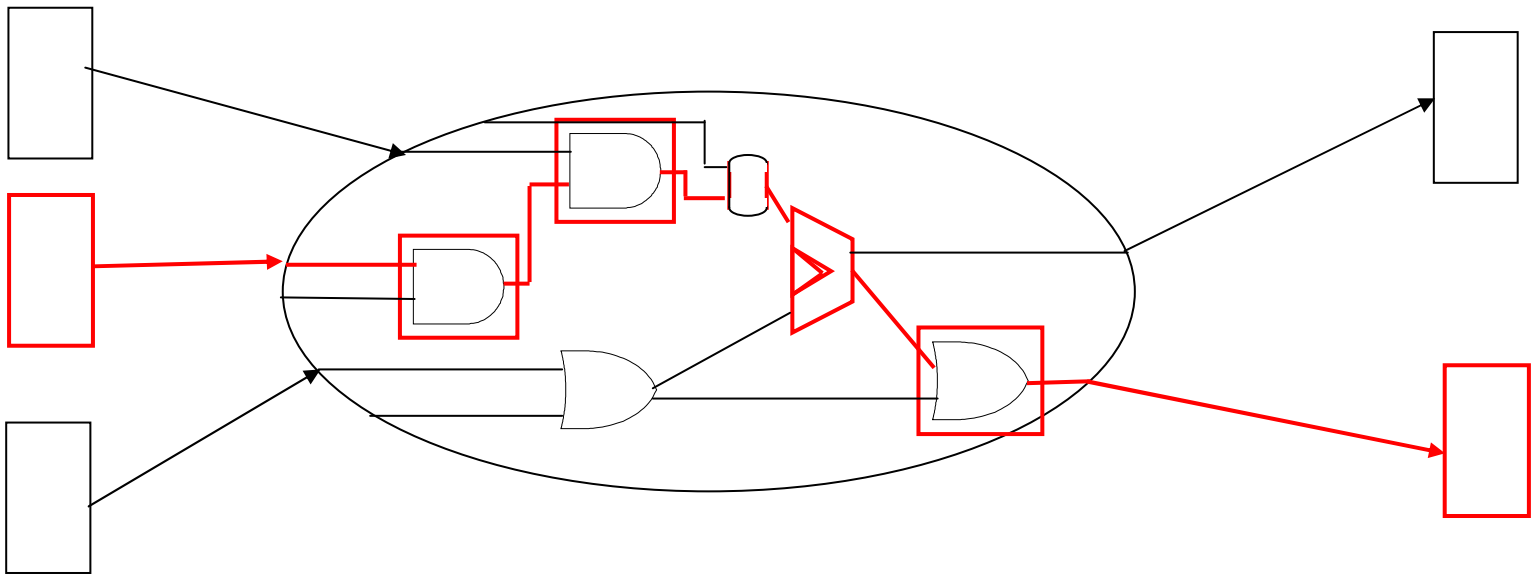
$$\Rightarrow \quad T_{clock} \geq T^1_{prop} + T_{combinatorio} + T^2_{setup}$$

- Vincolo per rispetto di T^2_{hold} : ingresso ST2 permane per almeno T^2_{hold} dopo il fronte

$$\Rightarrow \quad T^1_{prop} + T_{combinatorio} \geq T^2_{hold}$$

[verificato automaticamente perché $T_{hold} < T_{prop}$]

ESTENDENDO QUESTE CONSIDERAZIONI AD UNA RETE COMPLESSA…



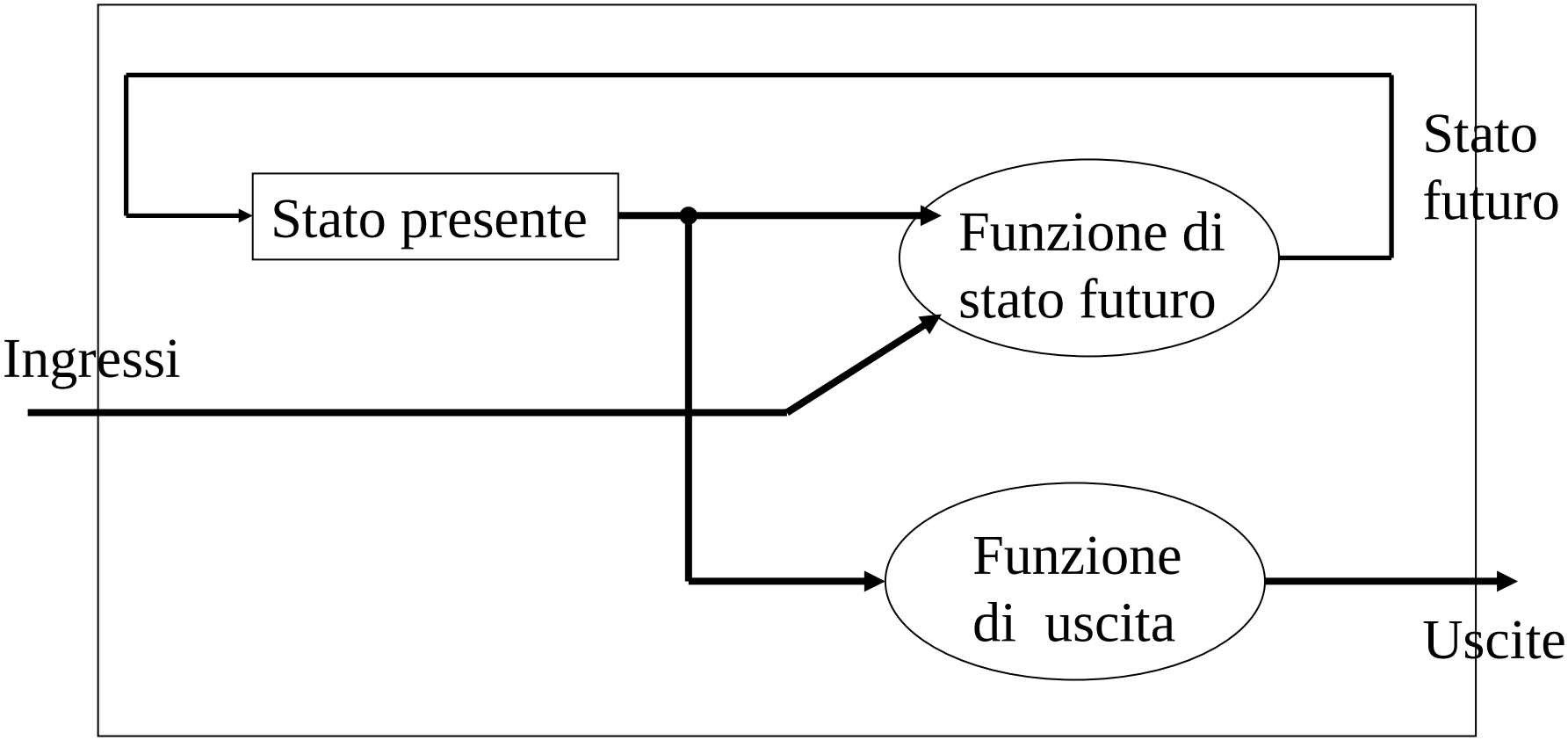
Occorre considerare il caso peggiore; in particolare il **“cammino critico”** vincola la lunghezza del periodo di clock e quindi limita la frequenza ottenibile!

$$\Rightarrow \quad T_{clock} \geq T_{prop} + T_{\text{cammino critico}} + T_{setup}$$

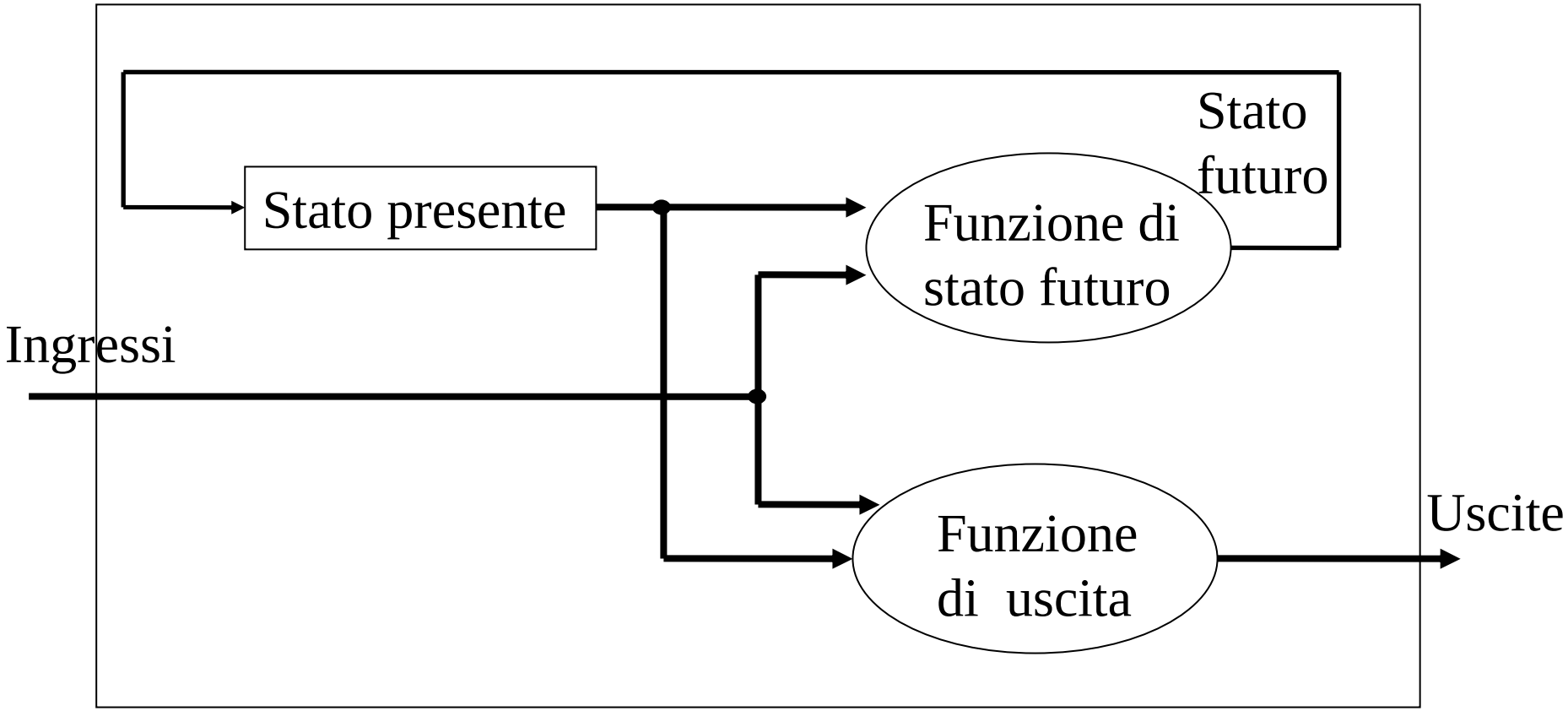
SPECIFICA DI UNA RETE SEQUENZIALE

- Consideriamo sistemi il cui numero di stati interni distinti è **finito** e corrisponde a tutti i possibili valori degli elementi di memoria interna (con *n* bit di memoria: 2^n stati)
- Consideriamo sistemi sincroni in cui lo stato è aggiornato sul fronte attivo del segnale di clock:
  - lo ***stato futuro*** dipende dagli ingressi correnti e dallo stato futuro: la **funzione di stato futuro** è una funzione combinatoria che identifica il valore dello prossimo stato assunto dal sistema in base agli ingressi e allo stato presente
  - le uscite dipendono dallo stato corrente e [eventualmente] dagli ingressi: la **funzione di uscita** (combinatoria) calcola le uscite

Macchina sequenziale di Moore



Macchina sequenziale di Mealy



Due modelli: Moore vs. Mealy

- Le macchine (automi) a stati finiti sono descrivibili mediante:

  I = $\{i_1, i_2, \ldots, i_p\}$ insieme dei simboli di ingresso

  U = $\{u_1, u_2, \ldots, u_r\}$ insieme dei simboli di uscita

  S = $\{s_1, s_2, \ldots, s_n\}$ insieme degli stati

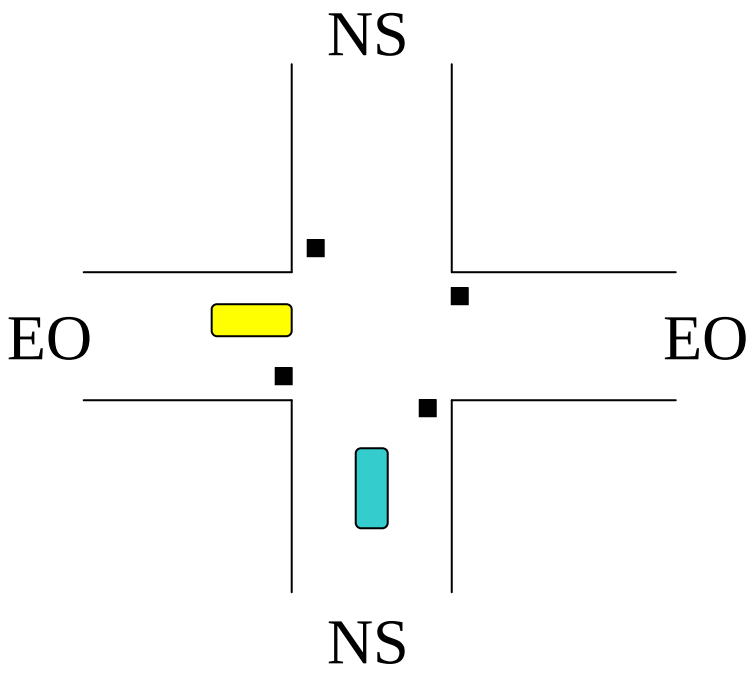
  δ funzione di transizione che specifica il nuovo stato in base allo stato corrente e all'ingresso

  ω funzione di uscita che specifica il valore di uscita in base allo stato corrente e all'ingresso

  $\Rightarrow$ Automa A = <I, U, S, δ, ω>

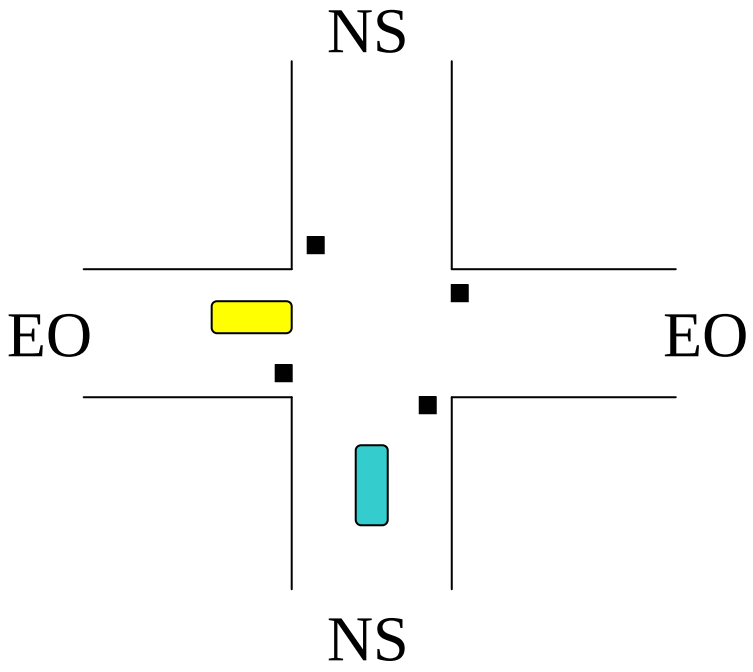
- Nel ***modello di Moore*** le uscite dipendono solo dallo stato corrente, nel ***modello di Mealy*** le uscite dipendono anche dagli ingressi

Un esempio: macchina per il controllo di un semaforo



- Consideriamo solo le luci verde e rossa
- Ingressi = {AutoNS, AutoEO}
- Uscite = {LuceNS, LuceEO}

Un esempio: macchina per il controllo di un semaforo

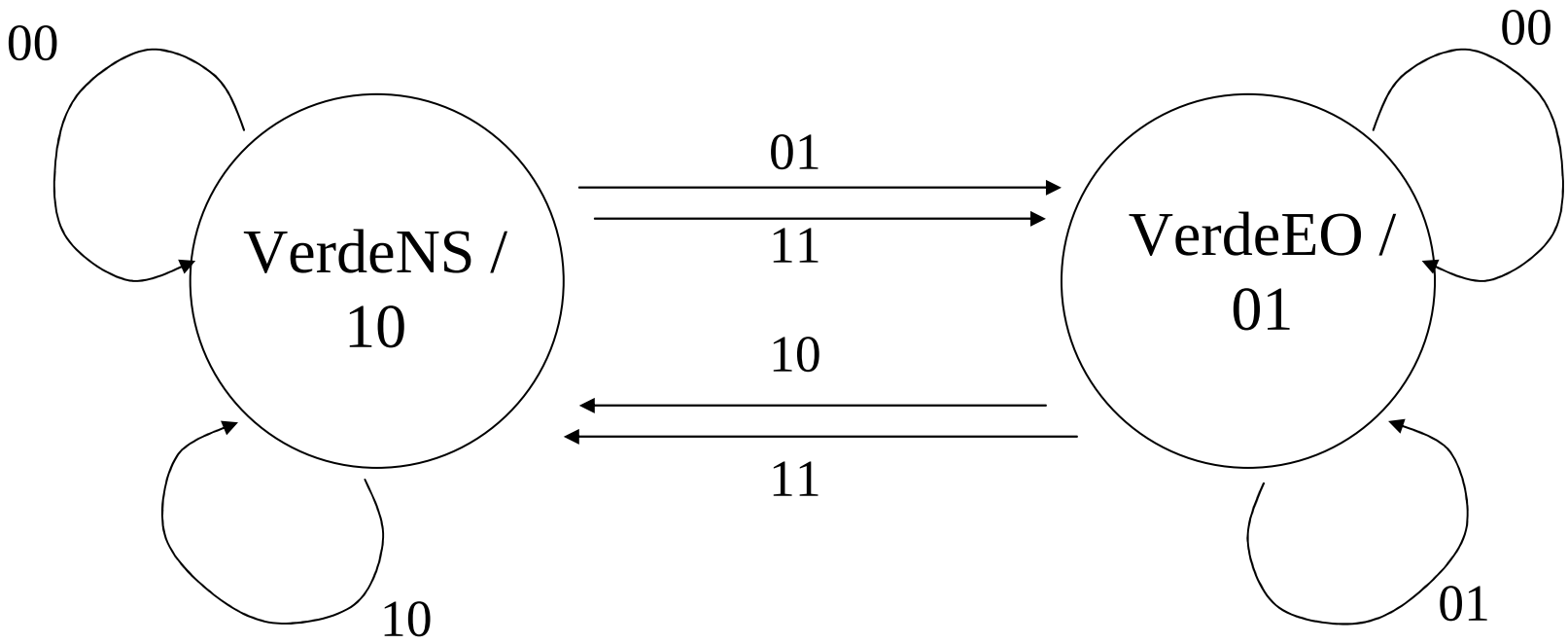


- Consideriamo solo le luci verde e rossa
- Ingressi = {AutoNS, AutoEO}
- Uscite = {LuceNS, LuceEO}

Abbiamo bisogno di due stati:

- Stati = {VerdeNS, VerdeEO}

MODELLO DI MOORE

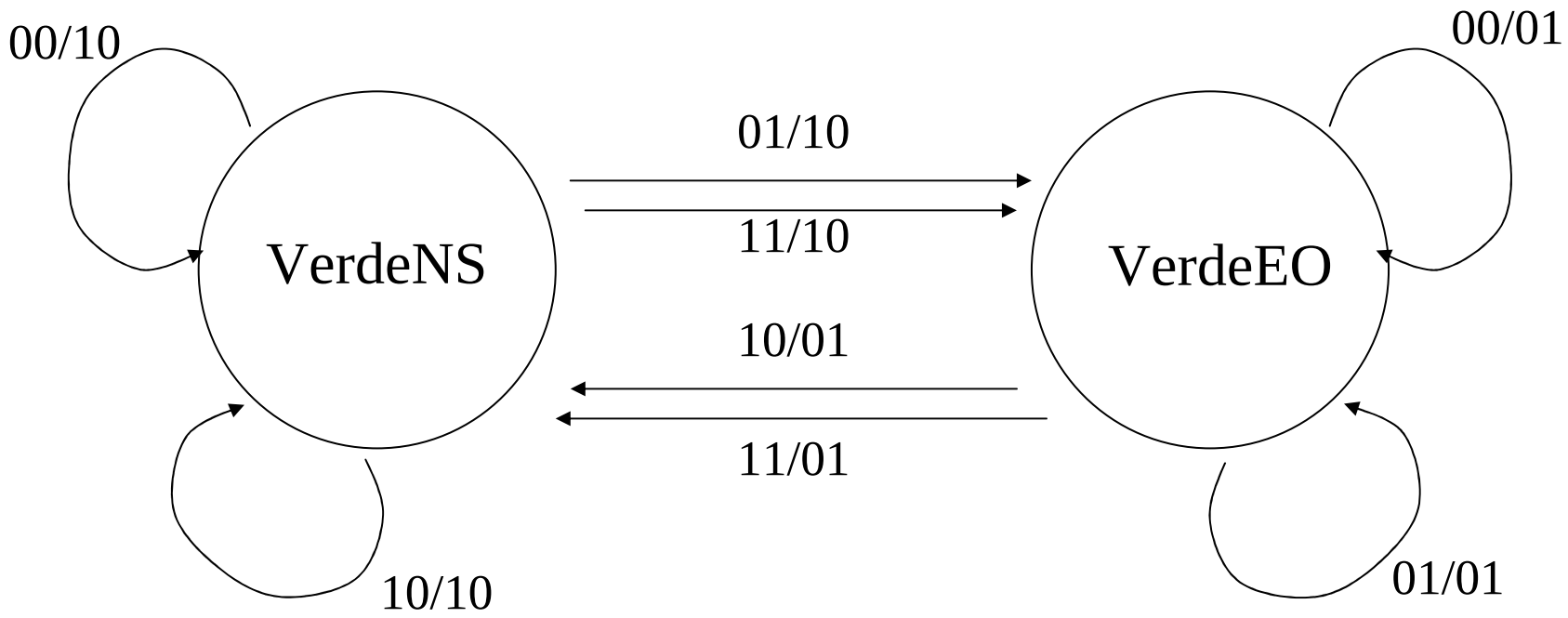


Ingressi: (AutoNS, AutoEO): 1 presente, 0 assente

Uscite: (LuceNS, LuceEO): 1 accesa, 0 spenta

NB: il “cambio” nel semaforo si ha al successivo ciclo di clock

MODELLO DI MEALY EQUIVALENTE

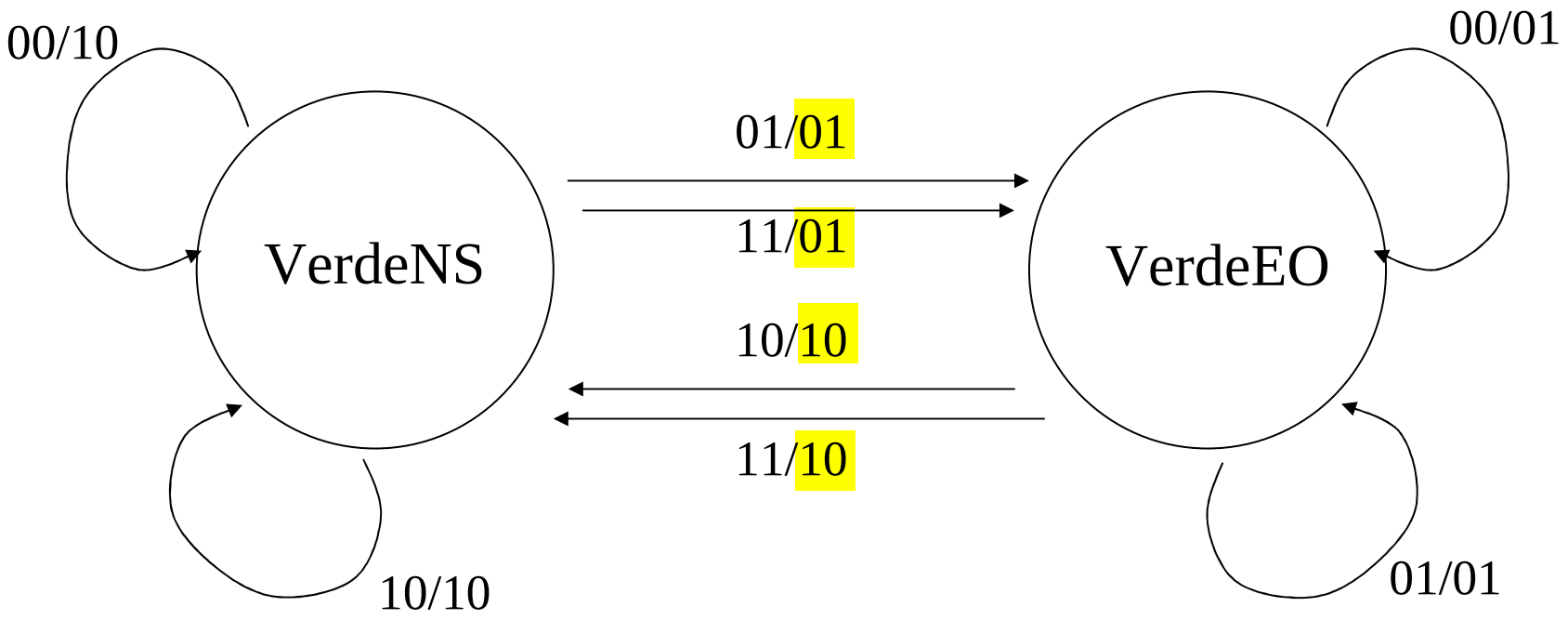


Ingressi: (AutoNS, AutoEO): 1 presente, 0 assente

Uscite: (LuceNS, LuceEO): 1 accesa, 0 spenta

NB: il “cambio” nel semaforo si ha al successivo ciclo di clock

MODELLO DI MEALY



Ingressi: (AutoNS, AutoEO): 1 presente, 0 assente

Uscite: (LuceNS, LuceEO): 1 accesa, 0 spenta

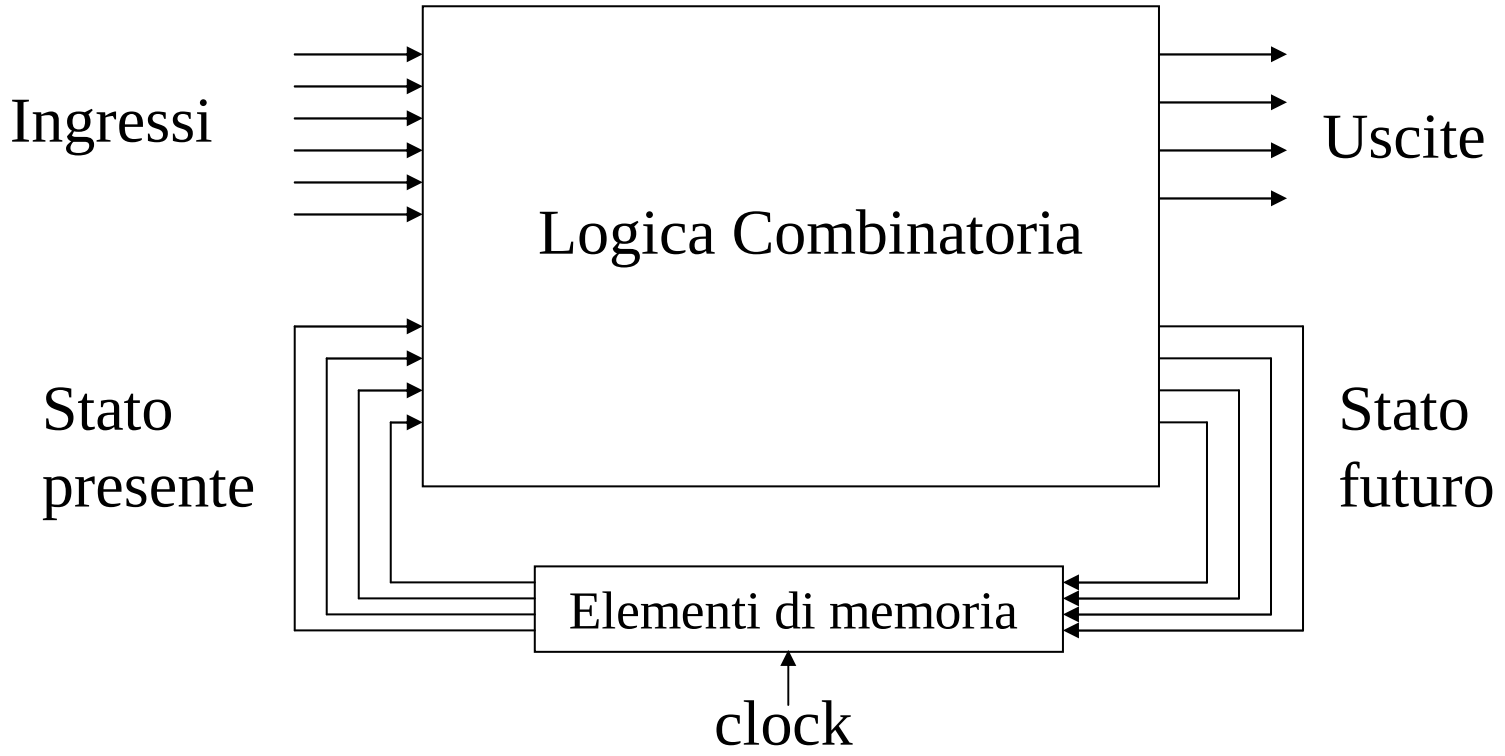
NB: il “cambio” si ha appena arriva un’auto nell’altra direzione

MEALY VS MOORE

- Le **macchine di Mealy** hanno, per ogni arco, un simbolo di entrata e uno di uscita
- Nelle **macchine di Moore** l'uscita è invece già codificata nel valore dello **stato** in cui si trova la macchina, ovvero la funzione di uscita dipende solo da S anziché da $I \times S$
- E' possibile trasformare una macchina di Mealy in una di Moore e viceversa, solitamente quelle di Moore hanno più stati (ma sono più veloci – vedremo quando parleremo del controllo CPU)
- In una macchina di Moore l'uscita nello stato iniziale è predeterminata (non dipende dagli ingressi)

SINTESI DI UNA RETE SEQUENZIALE

- E’ sufficiente usare:
  - un registro per rappresentare lo stato corrente
  - una rete combinatoria per calcolare lo stato futuro
  - una rete combinatoria per calcolare l’uscita



PROCEDURA DI SINTESI

1. Specificare la macchina a stati finiti o la tabella degli stati

---

2. Definire una codifica per i simboli di entrata e per i simboli di uscita
3. Determinare il numero dei flip-flop necessari (in base al numero degli stati) e definire una codifica per gli stati (valori da memorizzare nei flip-flop)
4. Derivare la tabella delle transizioni e delle uscite
5. Ricavare le formule per le variabili di stato (a seconda del tipo di flip-flop scelto)
6. Ricavare le formule per le variabili di uscita
7. Trovare la realizzazione circuitale per il blocco combinatorio

ESEMPIO DEL SEMAFORO (CON MOORE)

- Ingressi (AutoNS, AutoEO): (00, 01, 10, 11)
- Uscite (LuceNS, LuceEO): (10, 01)

NB: solo una delle due accesa!

Assegnamento degli stati

	y
VerdeNS	0
VerdeEO	1

⇨ 2 stati: sufficiente 1 FF

NB: se uso FF D, l'ingresso D coincide con lo stato futuro y

Tabelle di transizione e delle uscite e formule

Funzione di stato prossimo

Stato presente	***Ingressi***		***Stato prossimo***
y	***AutoNS***	***AutoEO***	***y'***
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

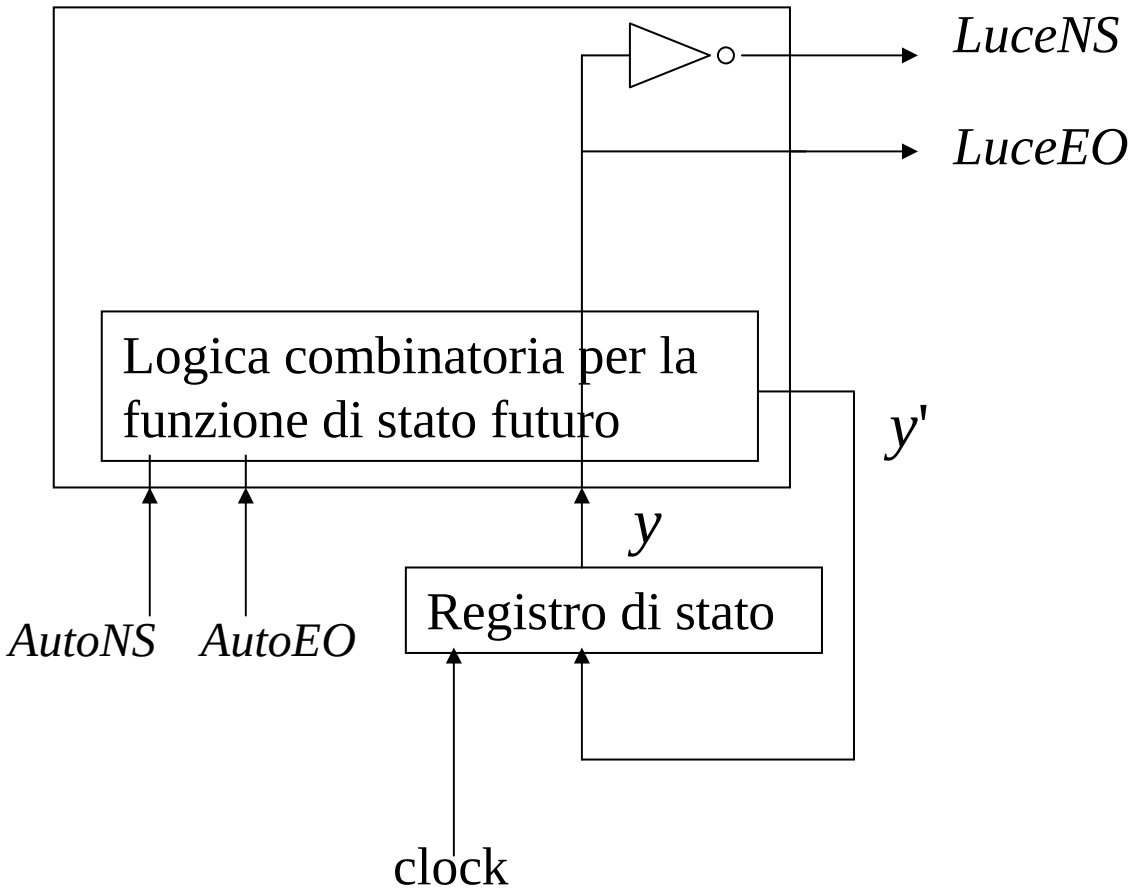
Funzione di uscita

Stato presente	***Uscite***	
y	***LuceNS***	***LuceEO***
0	1	0
1	0	1

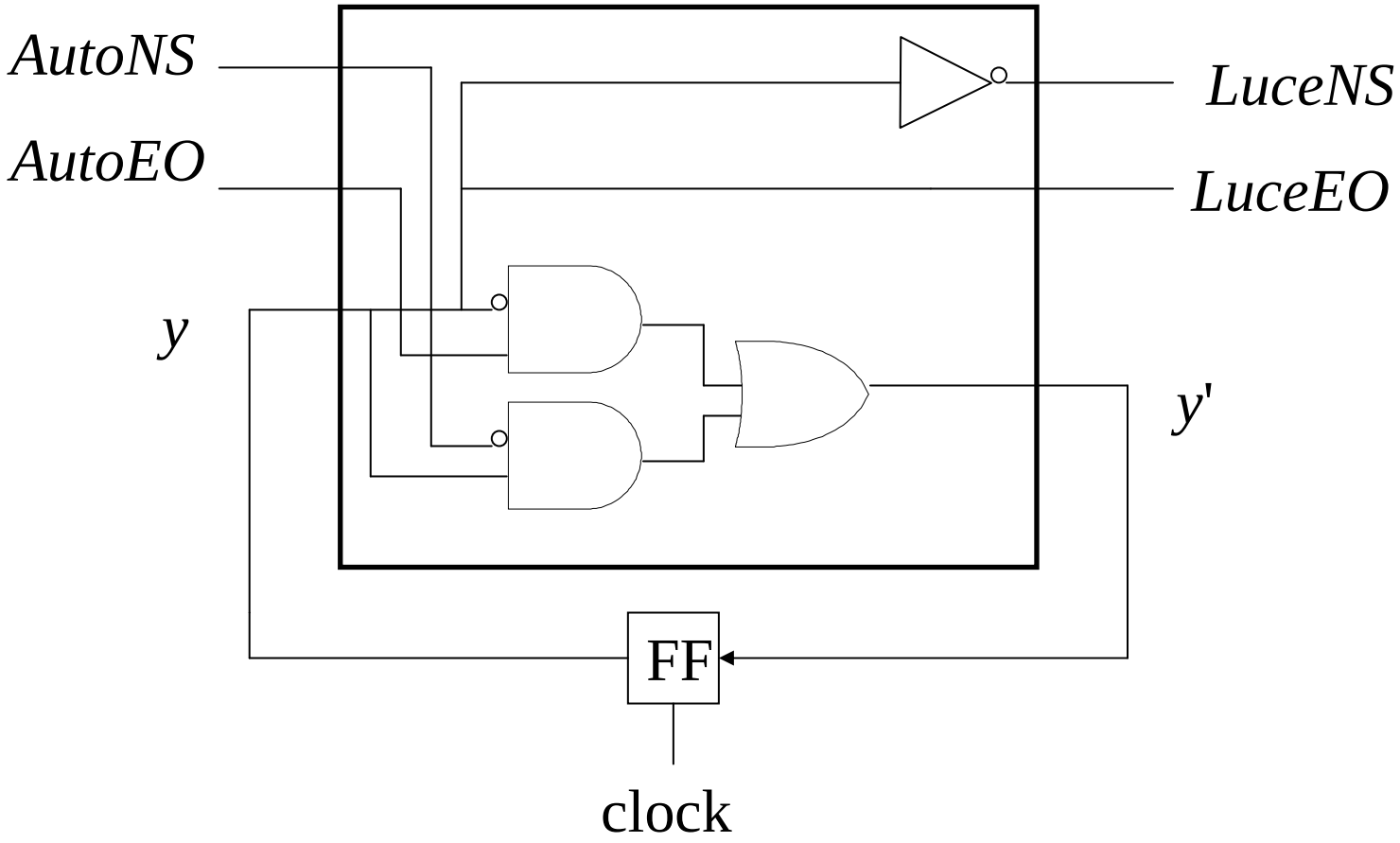
$$luceNS = \overline{y}$$
$$luceEO = y$$

$$\boldsymbol{y'} = \overline{y}\,\overline{AutoNS}\,AutoEO + \overline{y}\,AutoNS\,AutoEO +$$
$$y\,\overline{AutoNS}\,\overline{AutoEO} + y\,\overline{AutoNS}\,AutoEO$$
$$= \overline{y}\,AutoEO + y\,\overline{AutoNS}$$

Realizzazione circuitale (1)



Realizzazione circuitale (2)



ESEMPIO DEL SEMAFORO (CON MEALY 2)

- Tabella degli stati (alternativa alla forma grafica dell'automa)

Stato \ ***Ingresso***	00	01	10	11
VerdeNS	VerdeNS,10	VerdeEO,01	VerdeNS,10	VerdeEO,01
VerdeEO	VerdeEO,01	VerdeEO,01	VerdeNS,10	VerdeNS,10

- Ingressi (AutoNS, AutoEO): (00, 01, 10, 11)
- Uscite (LuceNS, LuceEO): (10, 01) NB: solo una delle due accesa!

Assegnamento degli stati

	y
VerdeNS	0
VerdeEO	1

⇨ 2 stati: sufficiente 1 FF

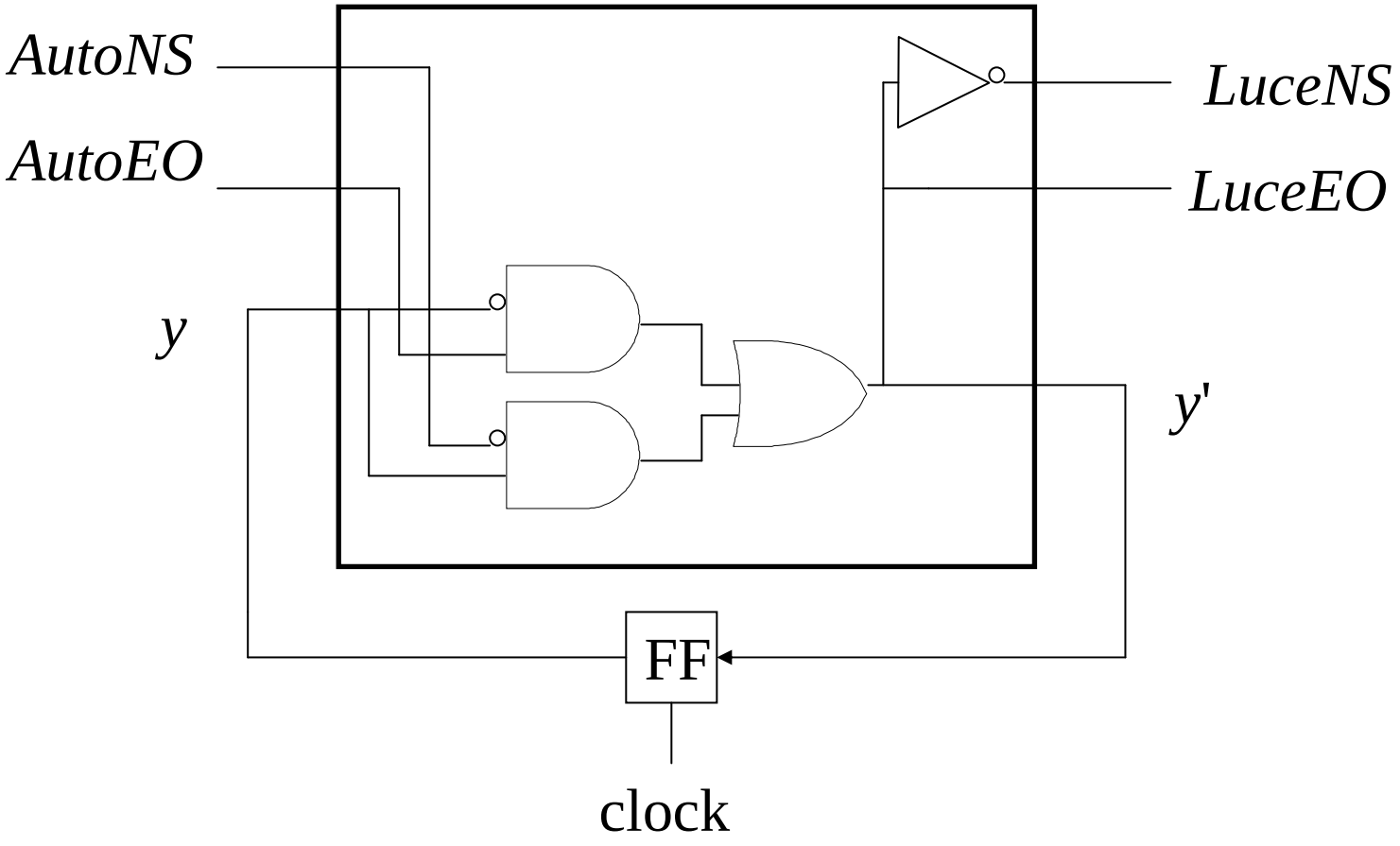
Tabelle di transizione e delle uscite e formule

Stato presente	***Ingressi***		***Stato prossimo***	***Uscite***	
y	***AutoNS***	***AutoEO***	***y'***	***LuceNS***	***LuceEO***
0	0	0	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	0	1	0
1	1	1	0	1	0

$$\boldsymbol{y'} = \overline{y}\,\overline{AutoNS}\,AutoEO + \overline{y}\,AutoNS\,AutoEO + y\,\overline{AutoNS}\,\overline{AutoEO} + y\,\overline{AutoNS}\,AutoEO = y\,\overline{AutoNS} + \overline{y}\,AutoEO$$

$$\boldsymbol{LuceEO} = y' \qquad \boldsymbol{LuceNS} = \overline{y'}$$

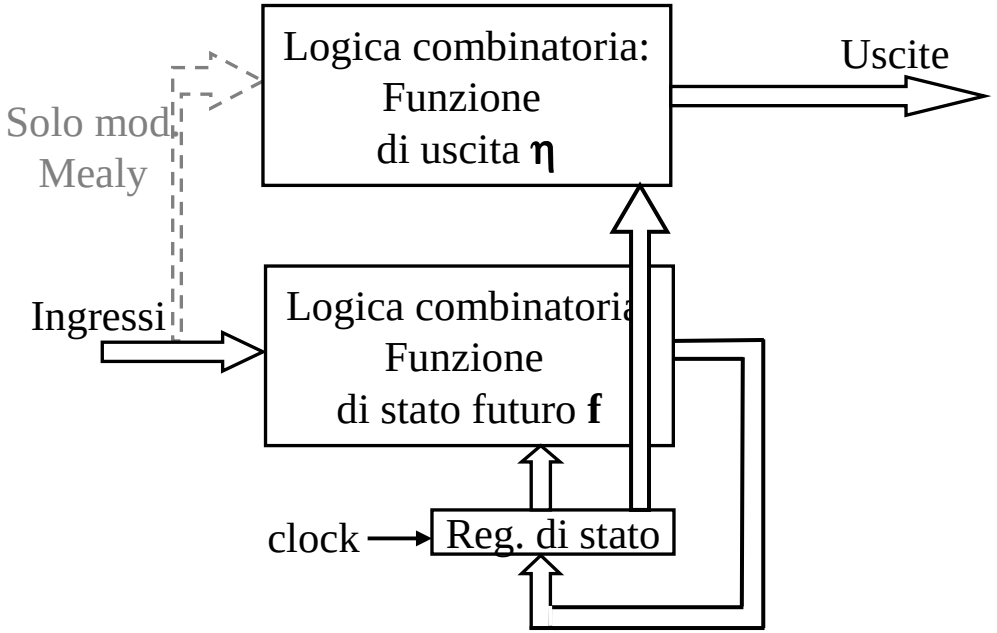
Realizzazione circuitale

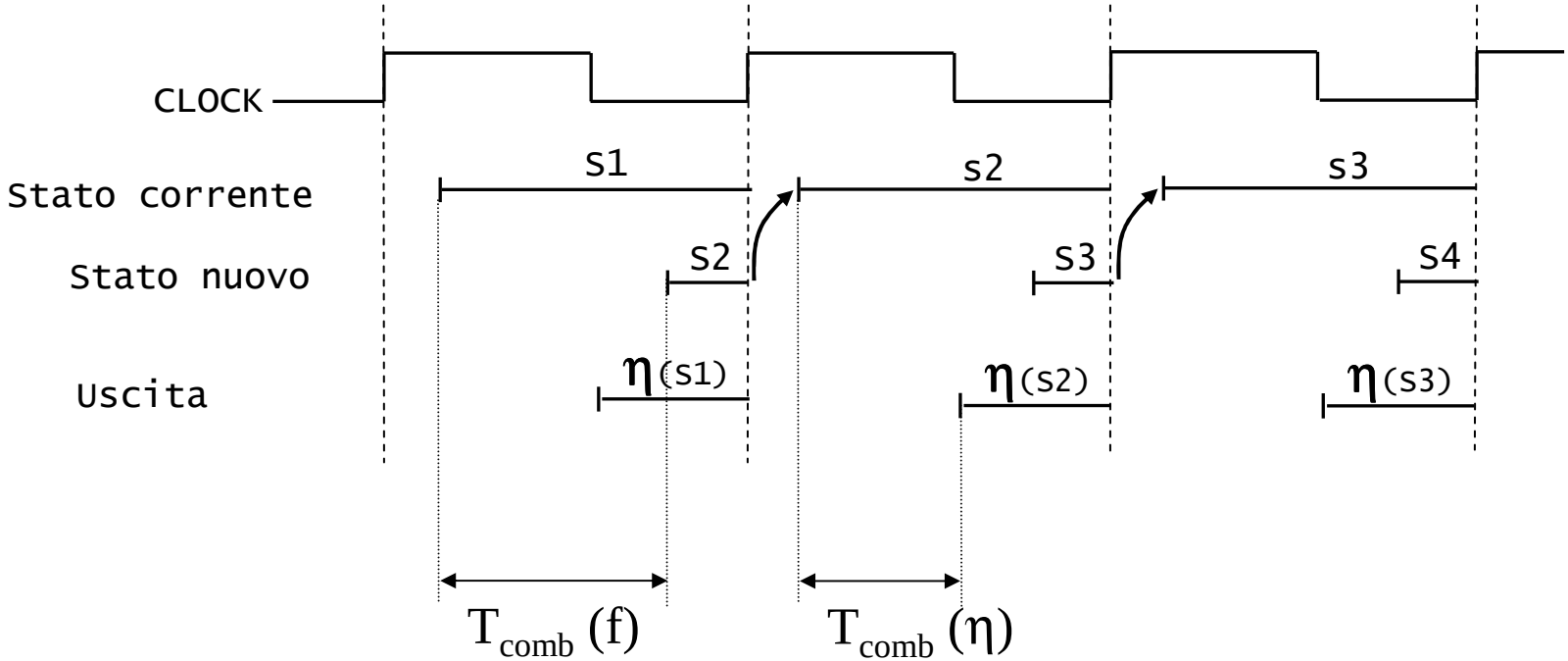


RICAPITOLANDO

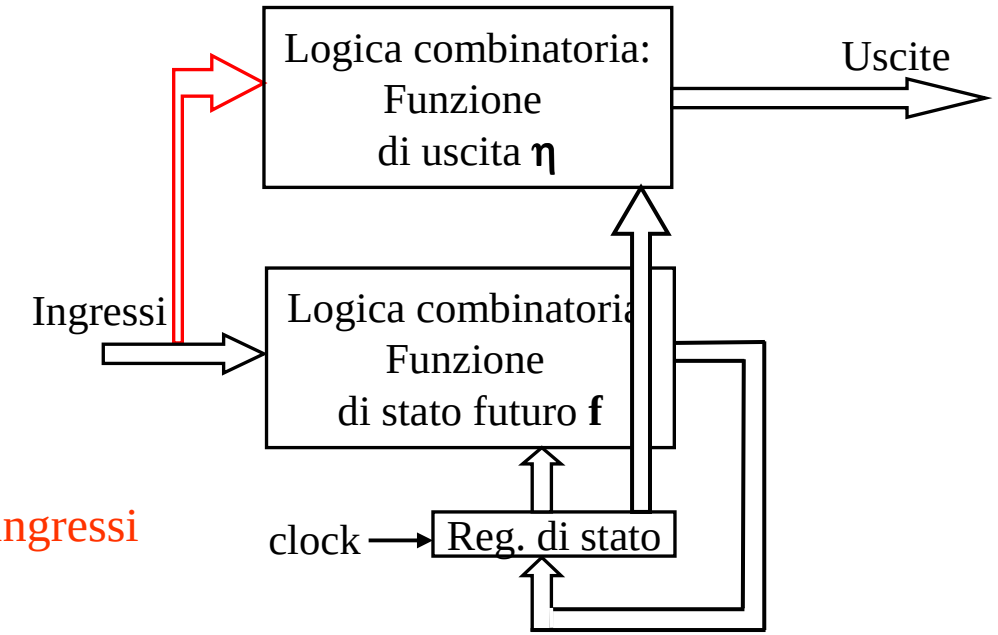
- Occorre fare **2 tabelle di verità**:
  - una tabella rappresenta la ***funzione di stato futuro***, dove lo stato futuro è funzione degli ingressi e dello stato presente
  - l'altra tabella rappresenta ***la funzione di uscita***, dove l'uscita è funzione solo dello stato presente (Moore) oppure degli ingressi e dello stato presente (Mealy)
- Nella macchina sequenziale la logica combinatoria può essere divisa in 2 parti: la prima determina le uscite e la seconda determina lo stato futuro
- Moore:
  - vantaggio in termini di **velocità** (se le uscite devono essere disponibili il più presto possibile: si pensi al controllo del processore multi-ciclo)
  - svantaggio: il numero degli stati può essere maggiore

Ancora sulla temporizzazione (Mod. Moore)





Ancora sulla temporizzazione (Mod. Mealy)



- Risente degli ingressi
- Stabilità in funzione della stabilità ingressi