



Architetture sincrone e asincrone

Sintesi di circuiti sequenziali

Federico Pedersini

Dipartimento di Informatica
Università degli Studi di Milano

Circuiti combinatori



Circuiti combinatori = circuiti senza memoria:

Le uscite al tempo t dipendono unicamente dagli ingressi al tempo t

$$out_i = f_i(in_1, in_2, \dots, in_M) \quad , \quad i = 1 \dots N$$

- Ogni uscita **out_i** è una **funzione logica** degli ingressi **in₁ ... in_M**
- Indipendenza dal tempo
- È impossibile modificare la "storia" del circuito → **non c'è memoria**



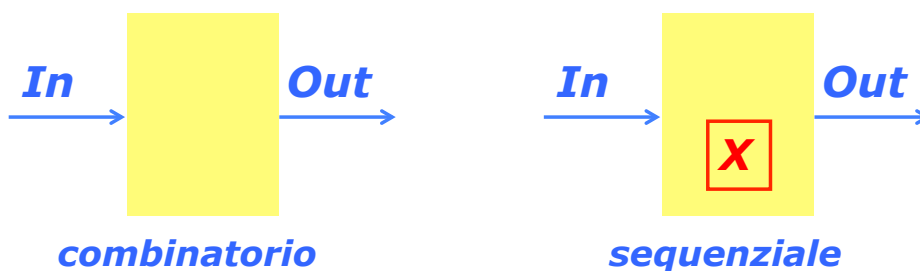
Circuito combinatorio



- ❖ **Circuiti combinatori** = circuiti senza memoria
 - Gli output al tempo t dipendono unicamente dagli input al tempo t

$$Out = f (In)$$
 - Per consentire ad un dispositivo di mantenere le informazioni, sono necessari circuiti con memoria

- ❖ **Circuiti sequenziali** = circuiti con memoria (stato)
 - La memoria contiene lo **stato (X)** del sistema:
$$Out = f (In , X)$$



Circuiti sequenziali, o con memoria

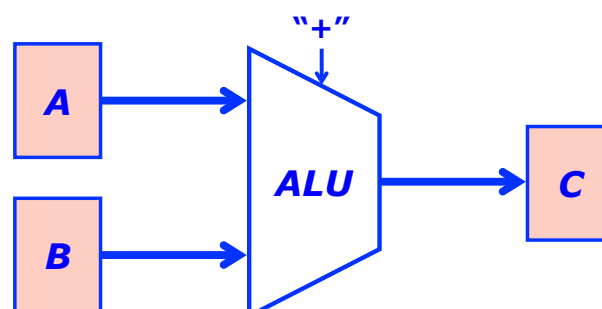
- ❖ **Sono necessari perché...**
 - In un sistema di elaborazione di informazioni è **necessario poter conservare** per un certo tempo le **informazioni**

$$C = A + B;$$

...

$$C = C + 1;$$

- “conservare per un certo tempo” → **MEMORIZZARE**

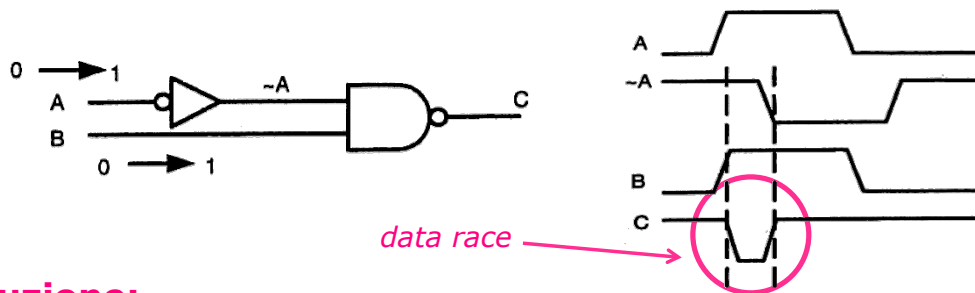


Sono necessari perché...

al crescere della complessità dei circuiti combinatori, si verificano fenomeni critici:

- ❖ **"Data races"**: transizioni spurie (glitch) su un segnale che deve rimanere costante

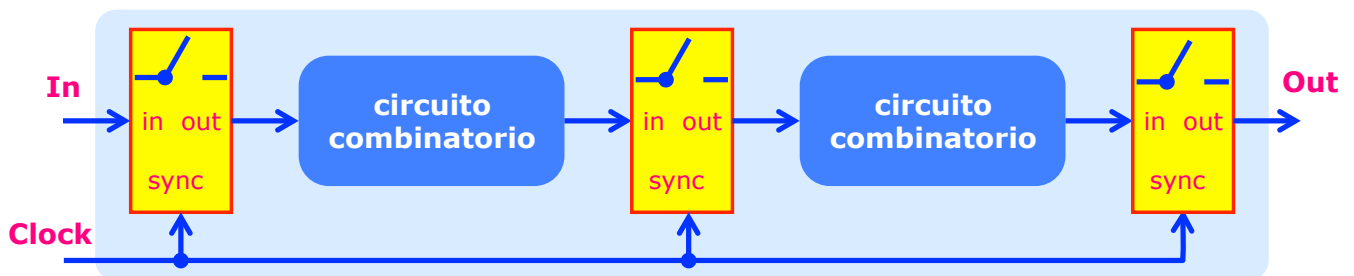
→ C'è un **limite alla complessità** dei circuiti combinatori!



Soluzione:

- ❖ spezzo il circuito in **SEZIONI**
- ❖ inserisco tra le sezioni delle **BARRIERE** al segnale, che "APRO" solo quando i segnali sono **STABILI**.

Struttura di architetture sincrone



Cancello → Circuito combinatorio → Cancello

- ❖ **Sincronizzazione**: la logica combinatoria deve terminare la propria commutazione in tempo utile
- ❖ **Dimensionamento** del periodo di clock **T**:
 - La commutazione del clock deve avvenire **dopo** che la logica combinatoria abbia terminato tutte le commutazioni
 - Il tempo necessario alla logica combinatoria per commutare dipende dal *cammino critico*



Dove il "cancello" viene inserito, si **sincronizza** l'attività di elaborazione

- Altrimenti i cammini critici renderebbero il funzionamento del circuito sempre più casuale.

"**cancelli**" = barriere di sincronizzazione tra circuiti

- Memorizzano l'ingresso ad un certo istante
→ dispositivi di memorizzazione: **registri**
- Lo mantengono stabile in uscita, per un certo periodo
→ orologio in comune: **segnale di clock**

orologio comune → **circuiti sincroni**
presenza di registri → **circuiti sequenziali** (con memoria)



❖ **Architettura sincrona:**

l'elaborazione e propagazione dei segnali è scandita da un **orologio comune a tutto il circuito (clock)**

- Il Clock regola l'attività dei "cancelli"
- Il circuito ha il tempo di stabilizzarsi (transitori critici) fino al successivo impulso di Clock

❖ **Architettura asincrona:**

l'elaborazione e propagazione dei segnali avviene **in modo incontrollato**, secondo le velocità di reazione dei circuiti

- Non ci sono cancelli
- Non devo mai aspettare l'impulso di clock → **massima velocità**

❖ **Progettazione sincrona**

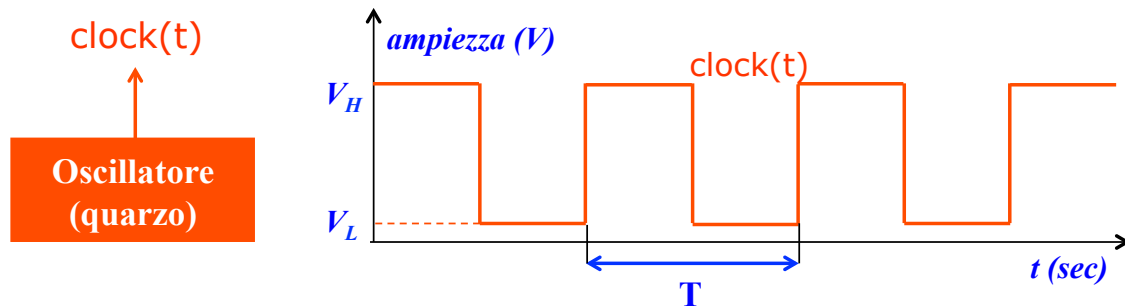
- il controllo dei transitori/cammini critici è limitato alla parte di circuito tra due **cancelli** (porte di **sincronizzazione**)

❖ **Progettazione asincrona**

- Devo progettare il circuito in modo che nessun transitorio/cammino critico causi problemi → analisi di tutti i transitori critici possibili.

Eccezioni: TRANSPUTER (INMOS, UK)

- Architettura di calcolatore con CPU asincrona
- Complessità enorme, sia di progettazione che di programmazione → progetto abbandonato.

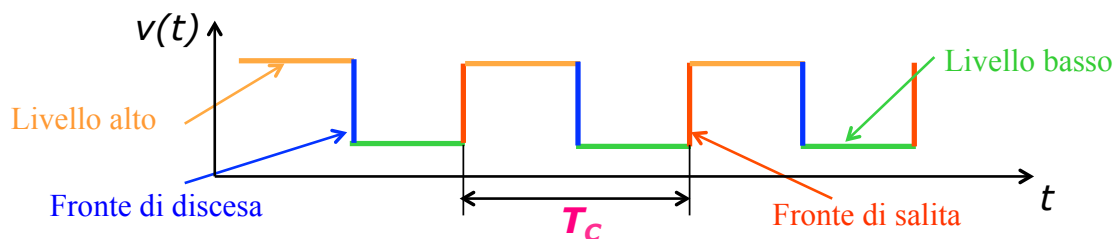


Periodo: T [s] durata di 1 ciclo (secondi)

Frequenza: f [Hz]=[s⁻¹] numero di cicli al secondo

$$T = 1/f$$

❖ Tempo di salita e discesa trascurabile, rispetto al periodo T .



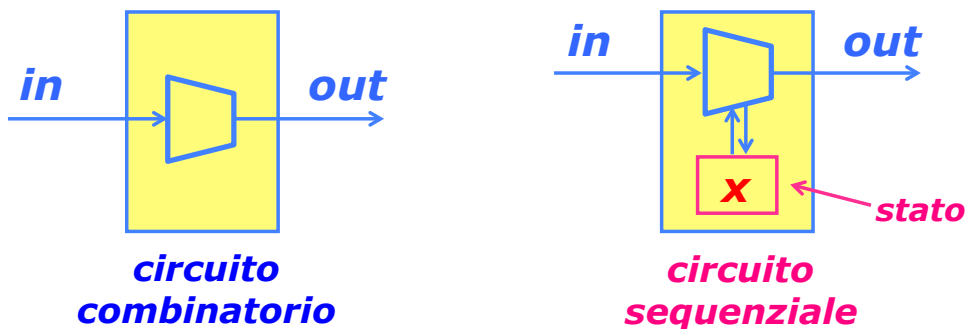
❖ **Architettura sensibile ai livelli:**

- Le variazioni di stato avvengono quando il clock è alto (basso). La parte bassa (alta) del ciclo di clock permette la propagazione tra sottocircuiti, così che i segnali si siano stabilizzati quando il clock cambia livello.

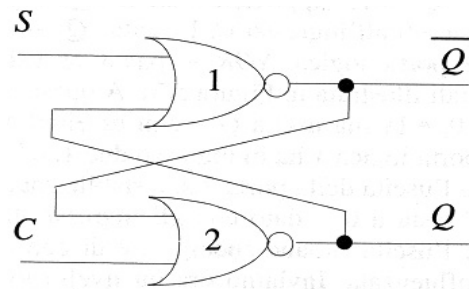
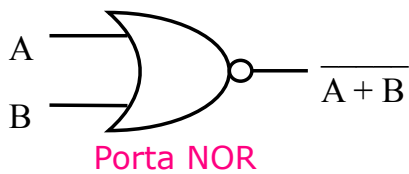
❖ **Architettura sensibile ai fronti:**

- Le variazioni di stato avvengono in corrispondenza di un fronte di clock (salita o discesa).

- ❖ **Circuiti combinatori** = circuiti **senza memoria**
 - Gli output al tempo t dipendono unicamente dagli input al tempo t
 $Out = f (In)$
 - Per consentire ad un dispositivo di mantenere le informazioni, sono necessari circuiti con memoria
- ❖ **Circuiti sequenziali** = circuiti **con memoria (stato)**
 - La memoria contiene lo **stato (X)** del sistema:
 $Out = f (In , X)$

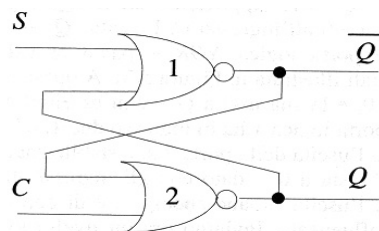
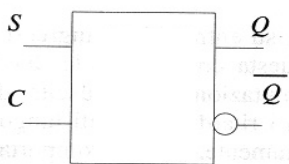


- ❖ Elemento cardine dei circuiti sequenziali è lo **stato**
 - Lo stato riassume il funzionamento negli istanti precedenti e deve essere immagazzinato (memorizzato)
- ❖ Elemento base della **memoria** è il **bistabile**
 - dispositivo in grado di **mantenere indefinitamente un valore in ingresso**
 - Il suo valore di uscita coincide con lo stato
- ❖ **Tipologie di bistabile**
 - Bistabili non temporizzati (**asincroni**) / temporizzati (**sincroni**).
 - Bistabili (sincroni) che commutano sul **livello (latch)** o sul **fronte (flip-flop)**
 - Memoria: insieme di bistabili (bistabili → registri → memorie)

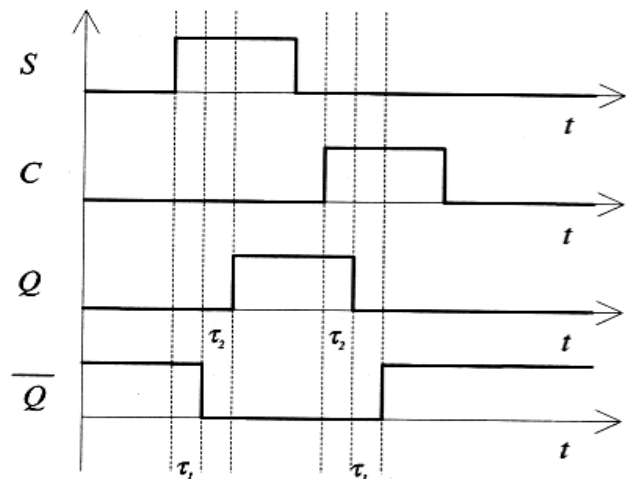
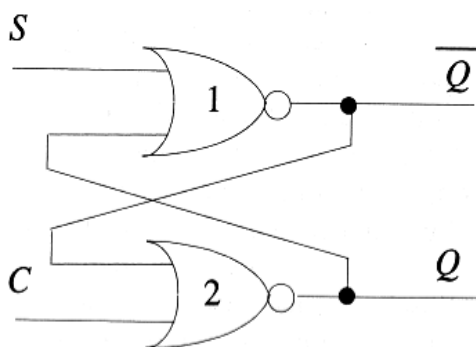


❖ Una coppia di porte NOR retroazionate può memorizzare un bit!

Latch Set-Clear



Latch Set-Reset (Set-Clear)



❖ Funzionamento:

- Set: $C = 0, S \rightarrow 1 \quad Q \rightarrow 1 \quad (\sim Q \rightarrow 0)$
- Reset: $S = 0, C \rightarrow 1 \quad Q \rightarrow 0 \quad (\sim Q \rightarrow 1)$
- Comportamenti anomali:
- $S = 1, C: 0 \rightarrow 1 \quad Q = \sim Q = 0$ (anomalia)



- ❖ Un circuito sequenziale **non** si può rappresentare mediante una **tabella di verità**
- ❖ Per i circuiti sequenziali si definisce la **tabella delle transizioni**

Tabella delle transizioni: $Q^* = f(Q, I) = f(Q, S, C)$
Q: valore dell'uscita attuale: stato **corrente**
Q*: uscita al tempo successivo: stato **prossimo**

$$Q^* = f(Q, I)$$

Q \ I	SC = 00	SC = 01	SC = 11	SC = 10
0	0	0	X	1
1	1	0	X	1

↑ $Q^* = q$
↑ Clear/Reset
↑ SET



- ❖ Tabella delle transizioni **f** :

$$Q^* = f(I, Q) = f(S, C, Q)$$

SC	SC=00	SC=01	SC=11	SC=10
Q				
Q=0	0	0	X	1
Q=1	1	0	X	1

S	C	Q	Q*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

- ❖ Considerando lo stato Q come ingresso ottengo la **tabella delle verità di Q***:

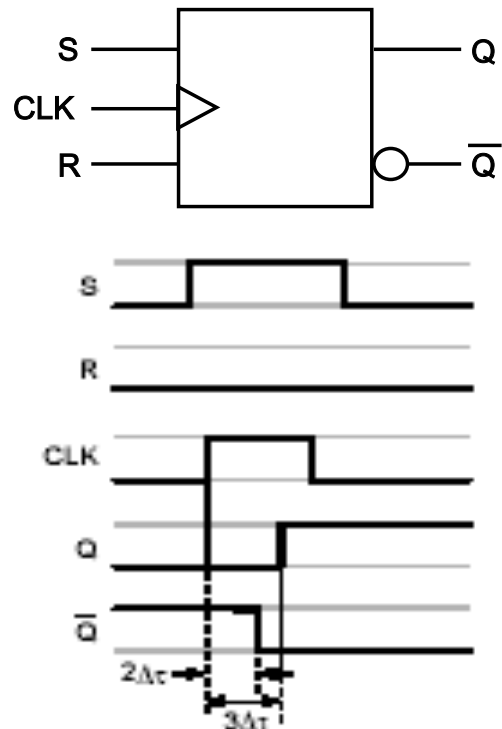
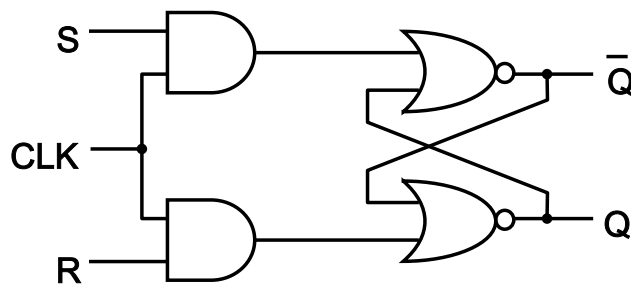
$$Q^* = S + Q\bar{C}$$



Latch Set-Reset (SR) sincrono

Struttura: **Latch SR + porte AND** tra il clock e gli ingressi.

- Solo quando il clock è alto i "cancelli" (porte AND) fanno passare gli input → **LATCH**



Sintesi della funzione logica



Tabella delle transizioni: $Q^* = f(S, R, T, Q)$

T	Q	S	R	Q*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	X

TQ	SR = 00	SR = 01	SR = 11	SR = 10
00	0	0	0	0
01	1	1	1	1
11	1	0	X	1
10	0	0	X	1

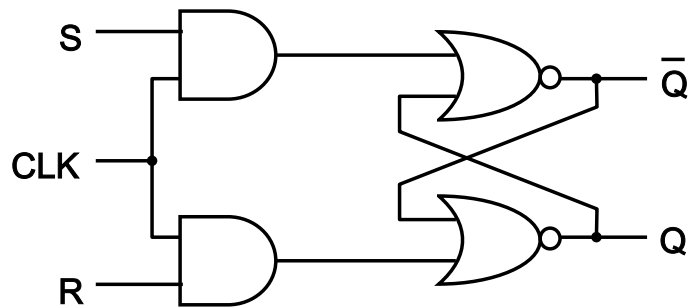
$$Q^* = \bar{T}Q + TQ\bar{R} + TS = \bar{T}Q + T(Q\bar{R} + S)$$

↑ clock alto → $Q^* = Q \sim R + S$ (bistabile)
↑ clock basso → $Q^* = Q$ (status quo)



Latch Set-Reset (SR) sincrono

$$Q^* = \bar{T}Q + T(Q\bar{R} + S)$$



$$T = 1 \rightarrow Q^* = Q\bar{R} + S: \begin{cases} \text{Set : } S = 1, R = 0 & Q^* = Q + 1 = 1 \\ \text{Reset : } S = 0, R = 1 & Q^* = 0 + 0 = 0 \end{cases}$$

$$T = 0 \rightarrow Q^* = Q: \text{ "status quo"}$$

Comportamento del Latch D

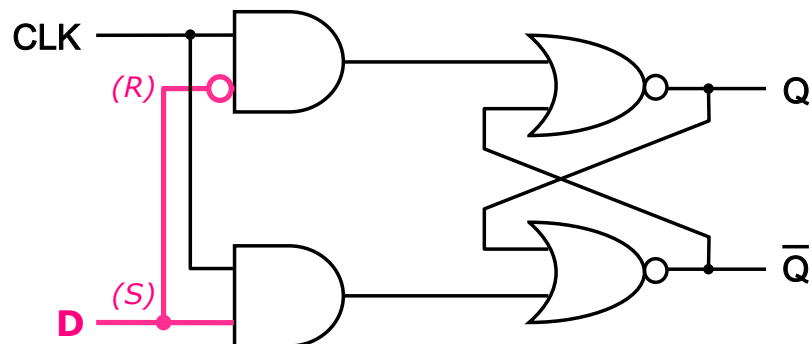


Latch SR un po' scomodo: 2 ingressi separati per memorizzare "0" o "1"

IDEA: pilota **S** e **R** con un solo ingresso: **D**

D = 1 → S=1, R=0 (Set) → **Q=1**

D = 0 → S=0, R=1 (Reset) → **Q=0**



LATCH D:

Clock ALTO (CLK=1):

- L'uscita Q insegue l'ingresso D (con $3t_p$ di ritardo)

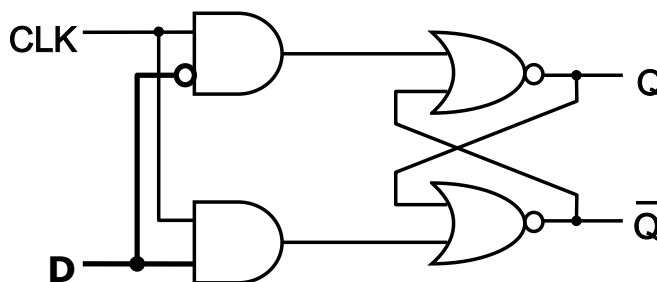
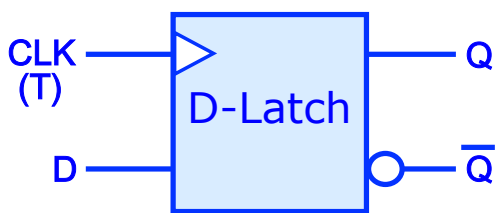
Clock BASSO (CLK=0):

- L'uscita Q rimane bloccata sull'ultimo valore assunto

LATCH D sincrono:

Memorizza il valore presente all'ingresso dati quando il clock è alto, altrimenti (clock=0) si mantiene sul valore memorizzato

```
if CLK = 1 then Q*=D else Q*=Q
```



La struttura del latch D

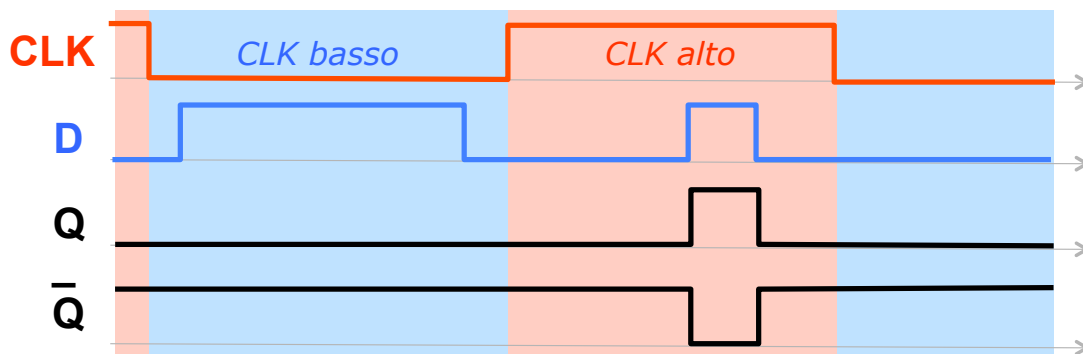
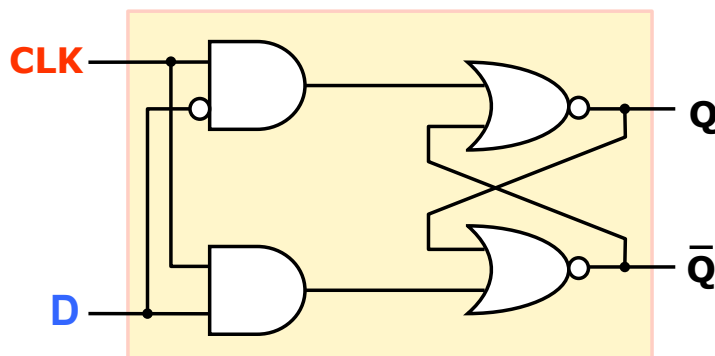


Tabella delle transizioni

- ❖ Dalla tabella delle transizioni calcolo l'espressione logica della funzione stato prossimo $Q^* = f(T, Q, D)$:

Tabella delle transizioni

T	Q	D	Q*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Funzione stato prossimo

$$Q^* = f(Q, T, D):$$

$$Q^* = \bar{T}Q\bar{D} + \bar{T}QD + T\bar{Q}D + TQD = \bar{T}Q + TD$$

Clock T alto:
→ $Q^* = D$ (input)

Clock T basso
→ $Q^* = Q$ (status quo)

$$T = 1 \rightarrow Q^* = D$$

$$T = 0 \rightarrow Q^* = Q$$

Latch: Bistabili level-sensitive

I latch sono dispositivi **trasparenti**: per tutto il tempo in cui il clock è a livello alto, il valore di D viene riportato in uscita

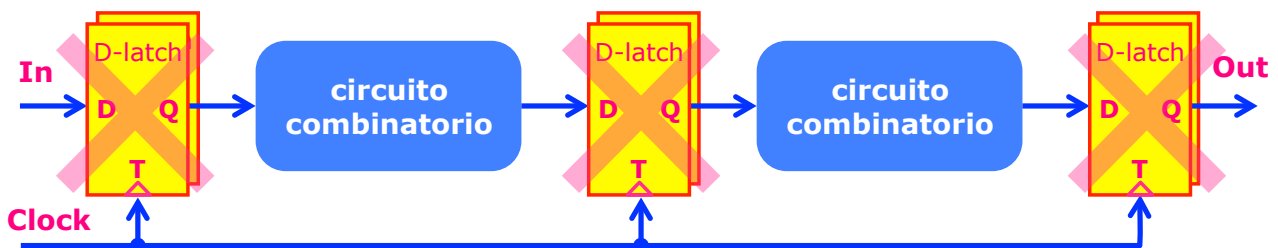
$$\text{Clock} = 1 \rightarrow Q^* = D \text{ (uscita collegata all'ingresso)}$$

Possono funzionare come "cancelli"?

NO! Nel momento in cui "apro" (T=1), i segnali possono attraversare **TUTTI** gli stadi.

Soluzione: "cancello doppio"

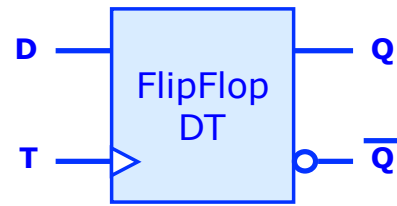
Quando apro in ingresso, l'uscita è chiusa, e viceversa.



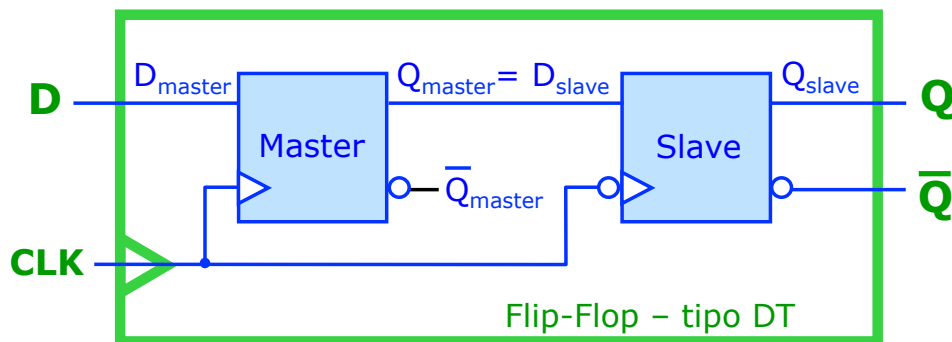


FLIP-FLOP: bistabile *edge sensitive*
(attivo sui fronti del clock):

lo stato (uscita) commuta solo in corrispondenza del fronte di salita o di discesa del clock.



Flip-Flop tipo DT
configurazione "Master-Slave":

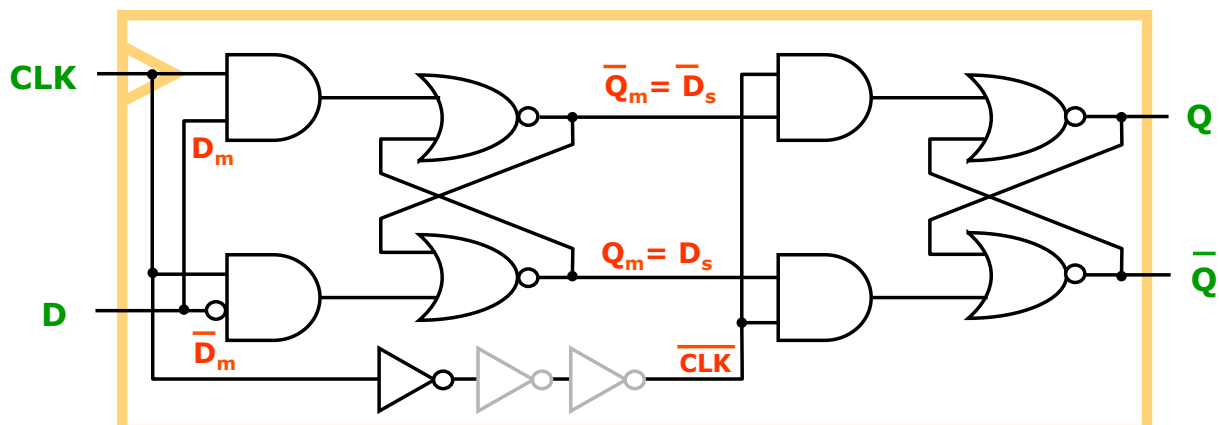


"FLIP" (clock ALTO): aperto lo stadio MASTER, chiuso lo stadio SLAVE

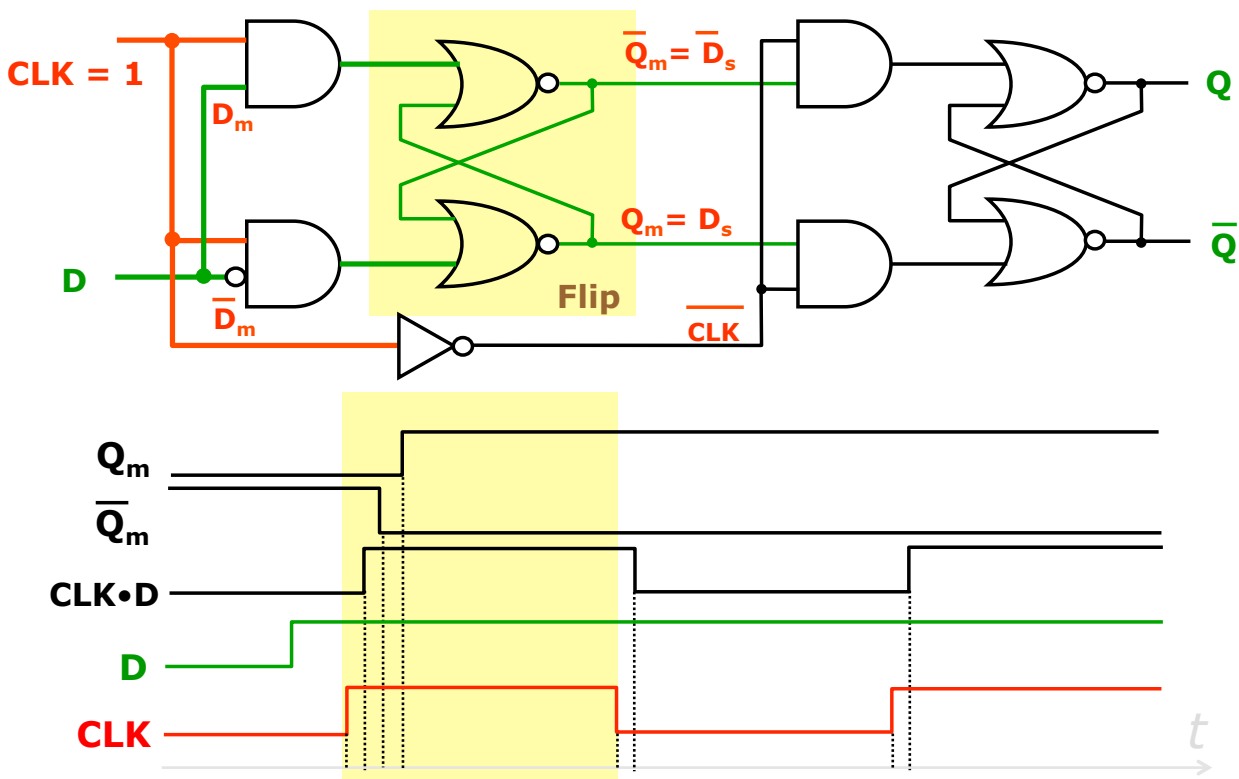
- Uscita Q bloccata su un valore precedentemente memorizzato

"FLOP" (clock BASSO): chiuso lo stadio MASTER, aperto lo stadio SLAVE

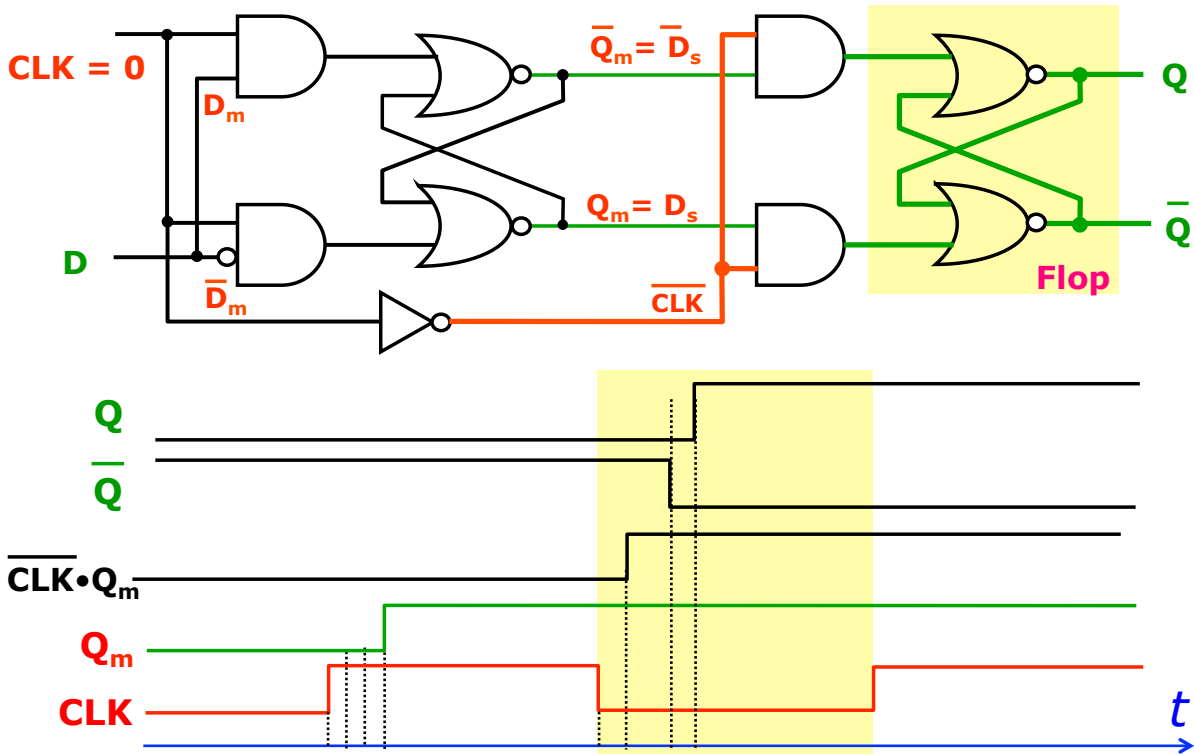
- Ingresso "chiuso", uscita bloccata sul valore contenuto in MASTER



Funzionamento: FLIP

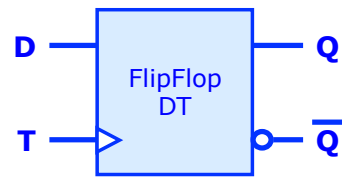


Funzionamento: FLOP

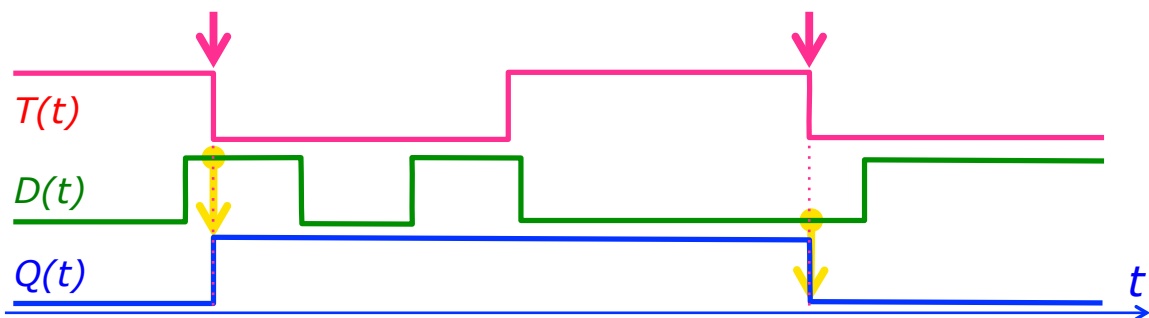




- ❖ Fronte di **SALITA** – **FLIP**
 - Attivato lo stadio **MASTER**
 - Uscita (stadio slave) invariata



- ❖ Fronte di **DISCESA** – **FLOP**
 - Attivato stadio **SLAVE**
 - Memorizzato il dato sull'ingresso:
 - Presenta il dato memorizzato in uscita:
 - Ingresso isolato



Latch: Bistabili level-sensitive



I latch sono dispositivi **trasparenti**: per tutto il tempo in cui il clock è a livello alto, il valore di D viene riportato in uscita

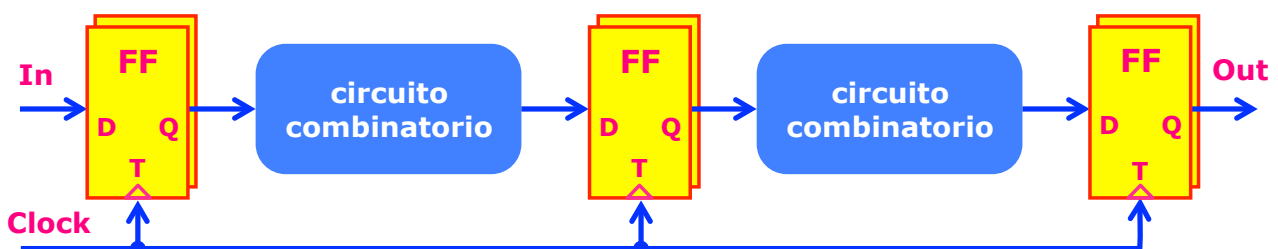
Clock = 1 → **$Q^* = D$** (**uscita collegata all'ingresso**)

Possono funzionare come **"cancelli"**?

Sì! Nel momento in cui "apro" (T=1), i segnali possono attraversare **TUTTI** gli stadi.

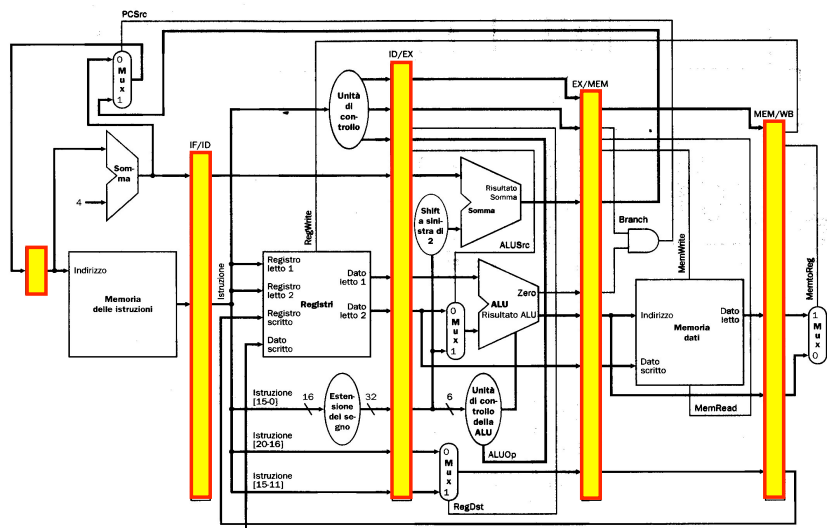
Soluzione: "cancello doppio"

Quando apro in ingresso, l'uscita è chiusa, e viceversa.



- ❖ I "cancelli" devono **disaccoppiare** i diversi sottosistemi logici
 - "memorizzare" i segnali e "rilanciarli", tutto ad un determinato istante
 - **prima e dopo**, uscite **non sensibili** agli ingressi
 - **Cancello doppio: ingresso e uscita**, mai aperti contemporaneamente

Cancelli:
registri costituiti da gruppi di flip-flop



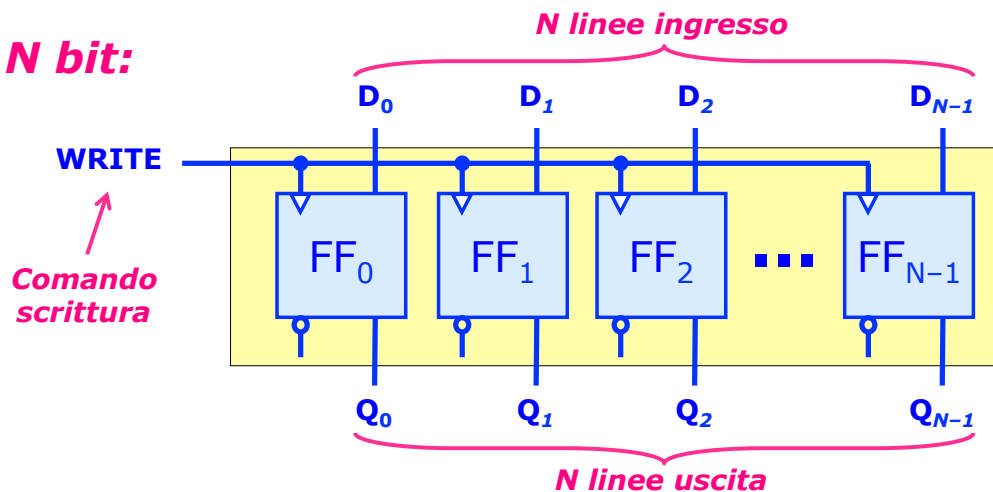
Registri

Registro: unità di memorizzazione di parole di **N bit**

Struttura: **N Flip-flop tipo DT**

Operazioni: **LETTURA:** I dati memorizzati sono **sempre leggibili** sulle uscite Q.
SCRITTURA: un **fronte** di salita (o discesa) su **WRITE** memorizza i valori presenti sugli ingressi D

Registro N bit:





Circuiti sequenziali: macchine a stati finiti

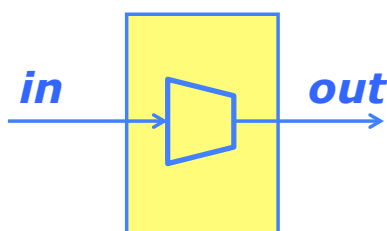
Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

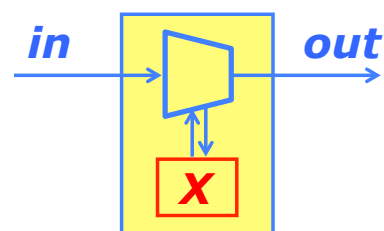
Macchine sequenziali



- ❖ Macchina combinatoria: $\mathbf{U} = f(\mathbf{I})$
 - senza memoria, uscita dipende solo dagli ingressi
- ❖ Macchina sequenziale:
 - $\mathbf{X}^* = f(\mathbf{X}, \mathbf{I})$
 - $\mathbf{U} = g(\mathbf{X})$
 - 2 funzioni: **uscita** e **stato prossimo**
 - esiste la memoria: lo **STATO**



combinatorio



sequenziale

- ❖ Una **Macchina a Stati Finiti (FSM)** è definita dalla **quintupla**:

$$\langle X, I, Y, f(), g() \rangle$$

3 insiemi e 2 funzioni logiche:

X: insieme degli stati (in numero finito).

I: alfabeto di ingresso: l'insieme dei simboli che si possono presentare in ingresso. Con **n** ingressi, avremo **2ⁿ** possibili configurazioni.

Y: alfabeto di uscita: l'insieme dei simboli che si possono generare in uscita. Con **m** uscite, avremo **2^m** possibili configurazioni

f(·): funzione stato prossimo: $X^* = f(X, I)$

Definisce l'evoluzione della macchina nel tempo, in modo deterministico

g(·): funzione di uscita: $Y = g(X)$ (macchina di **Moore**)

$Y = g(X, I)$ (macchina di **Mealy**)

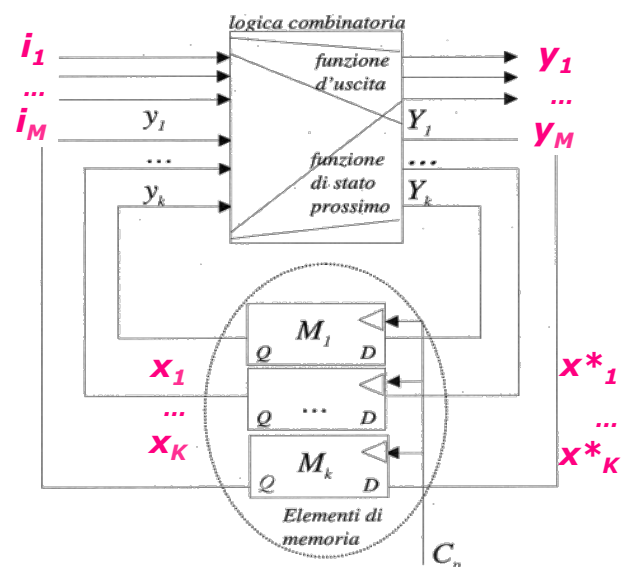
- Per il buon funzionamento della macchina è previsto talvolta anche uno **stato iniziale**, nel quale si trova la FSM **all'accensione** o dopo un comando di **RESET**

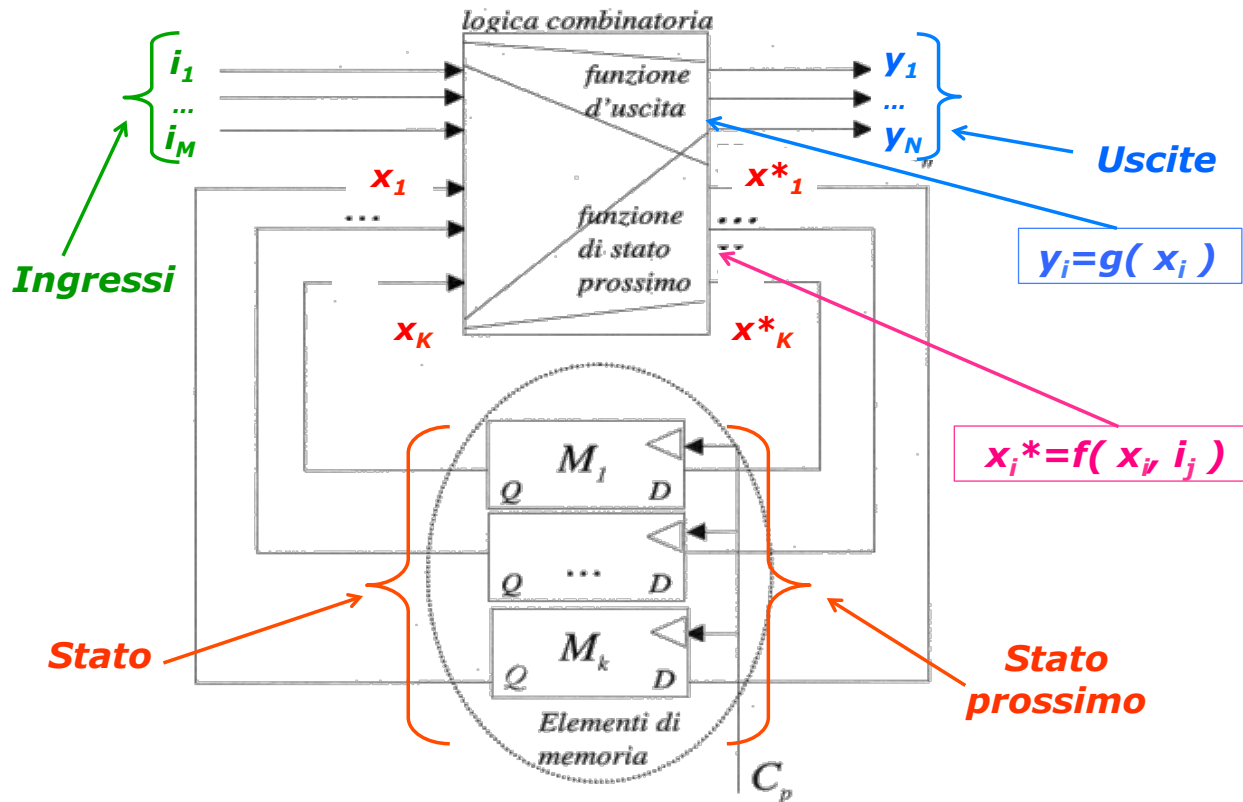
La macchina sequenziale di **Huffman**

Macchina sequenziale di Huffman

è la realizzazione pratica di una **FSM** mediante **elettronica digitale**

- ❖ Codifica **BINARIA** di: ingressi, uscite e stati
- ❖ Le funzioni: stato prossimo $f^* = f(X, I)$ e uscita $y = g(X)$ sono **funzioni logiche**.





Macchina di Moore: State Transition Table (STT)

STT: State Transition Table (Tabella delle transizioni di stato)

- Per ogni coppia: <stato attuale, ingresso> definisco stato prossimo x^*
- Per ogni stato attuale, definisco l'uscita y

$$(x_i \in X, i_j \in I) \rightarrow y = g(x_i); x^* = f(x_i, i_j)$$

Esempio: macchina di Moore \rightarrow

- M stati ($\log_2 M$ bit di stato), N ingressi ($\log_2 M$ bit d'ingresso):

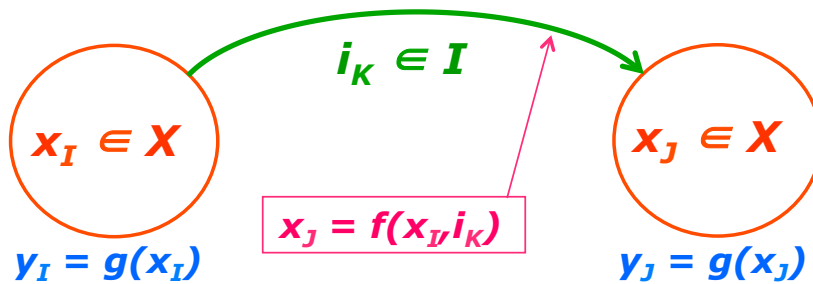
	i_1	i_2	...	i_N	Y
X_1	$x^*(x_1, i_1)$	$x^*(x_1, i_2)$		$x^*(x_1, i_N)$	$y(x_1)$
X_2	$x^*(x_2, i_1)$	$x^*(x_2, i_2)$		$x^*(x_2, i_N)$	$y(x_2)$
...					...
X_M	$x^*(x_M, i_1)$	$x^*(x_M, i_2)$		$x^*(x_M, i_N)$	$y(x_M)$



STG: State Transition Graph

(Diagramma degli Stati o Grafo delle transizioni)

- Ad ogni nodo è associato uno stato: $x_i \in X$
- ... ed un valore della funzione d'uscita: $y_i \in Y, y_i = g(x_i)$
- Un arco orientato da uno stato x_i ad uno stato x_j , contrassegnato da un simbolo (di ingresso) i_k , rappresenta una transizione che si verifica quando la macchina, essendo nello stato x_i , riceve come ingresso i_k



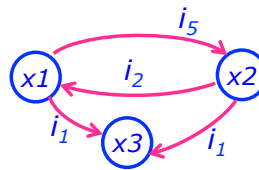
Sintesi circuito sequenziale



Sequenza operazioni di sintesi

di un **circuito sequenziale** mediante **modello a FSM**:

1. Definizione insiemi **ingressi I** e **uscite Y**
2. Costruzione grafo delle transizioni **STG**
➔ definizione insieme $X \rightarrow f(X, I)$ e $g(X)$
3. Traduzione **STG** ➔ **STT**
4. **Codifica binaria** degli elementi di X, Y, I nella **STT**
➔ gli elementi di I, Y e X diventano **parole binarie**
➔ $f(X, I)$ e $g(X)$ diventano **funzioni logiche!**
5. **Sintesi delle funzioni logiche** $f_i(X, I)$ e $g_j(X)$
➔ circuito finale con struttura di **Huffman**

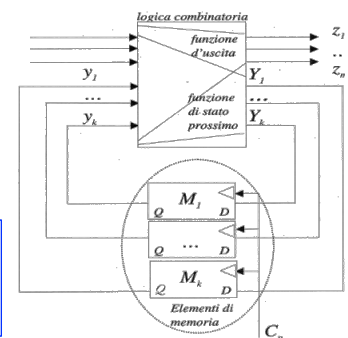


$$I = \{i_1, i_2, \dots, i_M\},$$

$$Y = \{y_1, y_2, \dots, y_P\}$$

$$X = \{x_1, x_2, \dots, x_Q\}$$

	i_1	...	i_N	Y
x_1	$x^*(x_1, i_1)$		$x^*(x_1, i_N)$	$y(x_1)$
x_2	$x^*(x_2, i_1)$		$x^*(x_2, i_N)$	$y(x_2)$
...				...
x_M	$x^*(x_M, i_1)$		$x^*(x_M, i_N)$	$y(x_M)$



n.bit stato: $r \geq \log_2 Q$

n.bit ingresso: $s \geq \log_2 M$

n.bit uscita: $t \geq \log_2 P$

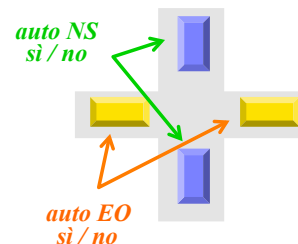
$$x_i^* = f_i(x_0, \dots, x_{r-1}, i_0, \dots, i_{s-1}), \quad i = 0 \dots r-1$$

$$y_j = g_j(x_0, \dots, x_{r-1}), \quad j = 0 \dots t-1$$

Esempio di SINTESI di FSM: controllore di un semaforo

SEMAFORO:

- ❖ **Incroccio tra 2 strade:** nord-sud (NS) ed est-ovest (EO) controllate da un semaforo
 - per semplicità consideriamo solamente rosso e verde
- ❖ Il semaforo può commutare ogni **30 secondi**
 - Macchina sincrona, clock con frequenza = ?
- ❖ E' presente un **sensore** in grado di "leggere", per ogni direttrice, se **esiste almeno un'auto in attesa**, oppure un'auto che si accinga ad attraversare (condizioni trattate allo stesso modo).
- ❖ Il semaforo deve cambiare colore, **da rosso a verde, quando esiste un'auto in attesa** sulla sua direttrice.
- ❖ Se ci sono auto in attesa sulle entrambe le direttrici il semaforo deve cambiare colore (al termine del tempo di commutazione)

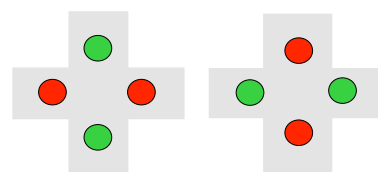


SEMAFORO: Stato, Ingresso, Uscita

INGRESSI

- Auto NS presente / non presente
 - ✦ **AutoNS=1/0**
- Auto EO presente / non presente
 - ✦ **AutoEO = 1/0**

→ 2 bit di INGRESSO → 4 configurazioni d'ingresso:
 $I = \{00, 01, 10, 11\}$



USCITE: = **STATO:** **$Y = X$**

- LuceEO verde (LuceNS rossa)
- LuceNS verde (LuceEO rossa)

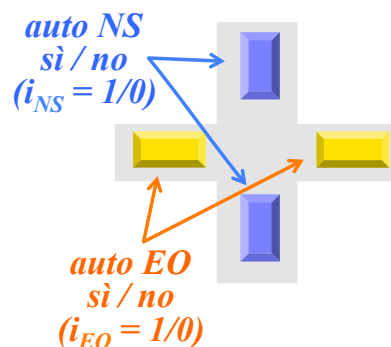
→ 2 configurazioni d'uscita → 1 bit di USCITA

STATO

- Semaforo NS VERDE, semaforo EO ROSSO
- Semaforo NS ROSSO, semaforo EO VERDE

$X = \{ \text{"VerdeNS"}, \text{"VerdeEO"} \}$

→ **1 bit di STATO** (1 flip-flop)





Funzione uscita

- ❖ Funzione uscita: $Y = g(X)$
 - Per ogni stato, definire l'uscita della macchina
- ❖ Uscita \leftrightarrow STATO $\rightarrow Y = X$
 - VerdeNS \rightarrow Verde sulla direttrice NS, rosso sulla direttrice EO
 - VerdeEO \rightarrow Verde sulla direttrice EO, rosso sulla direttrice NS
- ❖ Luce Verde NS / Rosso EO = **VerdeNS**
- ❖ Luce Verde EO / Rosso NS = **VerdeEO**

$$Y = \{ \text{"LuceVerdeNS"}, \text{"LuceVerdeEO"} \}$$



Funzione stato prossimo

- ❖ **Stato prossimo:** evoluzione dello stato, in funzione dello stato attuale e degli ingressi attuali

$$X(t+1) = X^* = f(X(t), I)$$

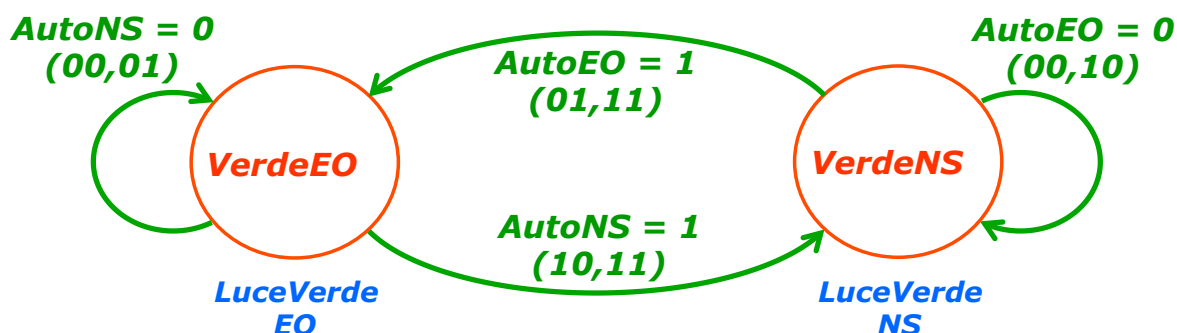
- ❖ $X_{i+1} = X_i^* = \text{"VerdeNS"}$
 - Se $X(t) = \text{"VerdeNS"}$ AND non ci sono auto sulla direttrice EO
 - Se $X(t) = \text{"VerdeEO"}$ AND ci sono auto sulla direttrice NS

VerdeNS · \sim autoEO + VerdeEO · autoNS $\rightarrow X^* = \text{VerdeNS}$
- ❖ $X_{i+1} = X_i^* = \text{"VerdeEO"}$
 - Se $X(t) = \text{"VerdeEO"}$ AND non ci sono auto sulla direttrice NS
 - Se $X(t) = \text{"VerdeNS"}$ AND ci sono auto sulla direttrice EO

VerdeEO · \sim autoNS + VerdeNS · autoEO $\rightarrow X^* = \text{VerdeEO}$



Sintesi State Transition Graph (STG):



Risultato:

❖ Funzione stato prossimo:

$$\text{VerdeNS}^* = \text{VerdeNS} \cdot \sim \text{autoEO} + \text{VerdeEO} \cdot \text{autoNS}$$

$$\text{VerdeEO}^* = \text{VerdeEO} \cdot \sim \text{autoNS} + \text{VerdeNS} \cdot \text{autoEO}$$

❖ Funzione uscita: $Y = X$



Sintesi State Transition Table (STT):

$X \backslash I(i_{EO}, i_{NS})$	$i_{EO}=0$ $i_{NS}=0$	$i_{EO}=0$ $i_{NS}=1$	$i_{EO}=1$ $i_{NS}=0$	$i_{EO}=1$ $i_{NS}=1$	Uscita
VerdeNS	VerdeNS	VerdeNS	VerdeEO	VerdeEO	Luce VerdeNS
VerdeEO	VerdeEO	VerdeNS	VerdeEO	VerdeNS	Luce VerdeEO

Funzione stato prossimo:
 $X^* = f(X, I)$

Funzione uscita:
 $Y = g(X)$



Codifica binaria della STT:

Stato:

- (VerdeNS/RossoEO, RossoNS/VerdeEO)

VerdeNS/RossoEO → "0"

RossoNS/VerdeEO → "1"

$$X = \{0, 1\}$$

Ingresso:

- 2 Variabili: (AutoEO, AutoNS) → 1 = "presente", 0 = "assente"
- 4 Configurazioni: $I = (i_{EO}, i_{NS}) = \{00, 01, 10, 11\}$

Uscita:

- (Luce_VerdeNS, Luce_VerdeEO) → $Y = \{0, 1\}$

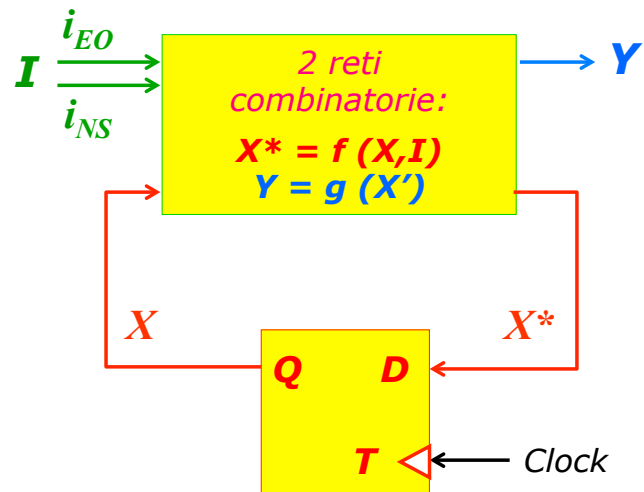
$X \backslash I(i_{EO}, i_{NS})$	00	01	10	11	Uscita Y
0	0	0	1	1	0
1	1	0	1	0	1

Sintesi della MSF del semaforo



Sintesi del circuito:

- 2 stati (0,1) → **1 flip-flop DT**
- 2 linee di ingresso
- 1 linea d'uscita



Architettura di Huffman:

❖ Rete combinatoria che implementa:

- stato prossimo: $f(X, I)$
- uscita: $g(X)$



- Mediante la STT codificata in binario, posso esprimere X^* e Y come somma di prodotti:

> cerco i mintermini:

$$X^* = \bar{X} i_{EO} \bar{i}_{NS} + \bar{X} i_{EO} i_{NS} + X i_{EO} \bar{i}_{NS} + X \bar{i}_{EO} \bar{i}_{NS} = \bar{X} i_{EO} + X \bar{i}_{NS}$$

$$Y = X$$

$I(i_{EO}, i_{NS})$ X	00	01	11	10	Y
0	0	0	1	1	0
1	1	0	0	1	1

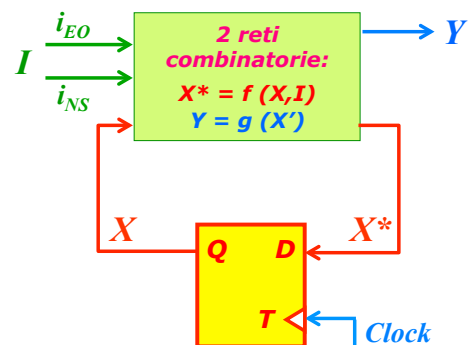
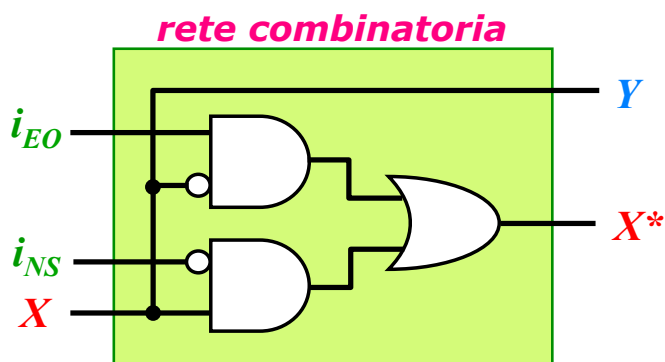
MSF del semaforo: sintesi del circuito



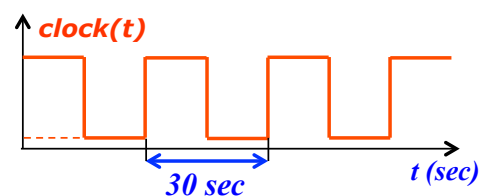
- funzioni logiche rete combinatoria:

$$X^* = \bar{X} i_{EO} \bar{i}_{NS} + \bar{X} i_{EO} i_{NS} + X i_{EO} \bar{i}_{NS} + X \bar{i}_{EO} \bar{i}_{NS} = \bar{X} i_{EO} + X \bar{i}_{NS}$$

$$Y = X$$



$$T_{CLK} = 30 \text{ sec}$$





- ❖ *Si progetti una macchina a stati finiti (di Moore) caratterizzata da una linea d'ingresso che viene letta ogni secondo e da un'uscita che vale "1" solo quando in ingresso si sia presentata la sequenza: "001".
Si determinino: STG, STT, STT codificata e il circuito completo.*
- ❖ *Si progetti un contatore binario modulo 4 che presenta, su due linee d'uscita, il valore binario di conteggio ($0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0 \dots$).
Il contatore viene incrementato ogni qualvolta sulla linea d'ingresso I si presenta un fronte di salita. Si suppongano inizialmente sia l'ingresso che le uscite a '0'.
Si determinino: STG, STT, STT codificata e il circuito completo.*
- ❖ *Si sintetizzi una macchina a stati finiti di Moore caratterizzata da una linea d'ingresso IN e due linee di uscita Y_0 e Y_1 . La macchina si comporta così: ogni volta che all'ingresso IN si presenta la sequenza "011", l'uscita Y cambia valore mentre l'uscita Z, normalmente a '0', si porta a '1' per un ciclo di clock e poi torna a '0'. Si suppongano inizialmente ingressi e uscite tutti a 0. Si determinino: STG, STT, STT codificata e struttura circuitale del sistema completo, avendo cura di semplificare il più possibile le funzioni prima di tradurle in circuito.*