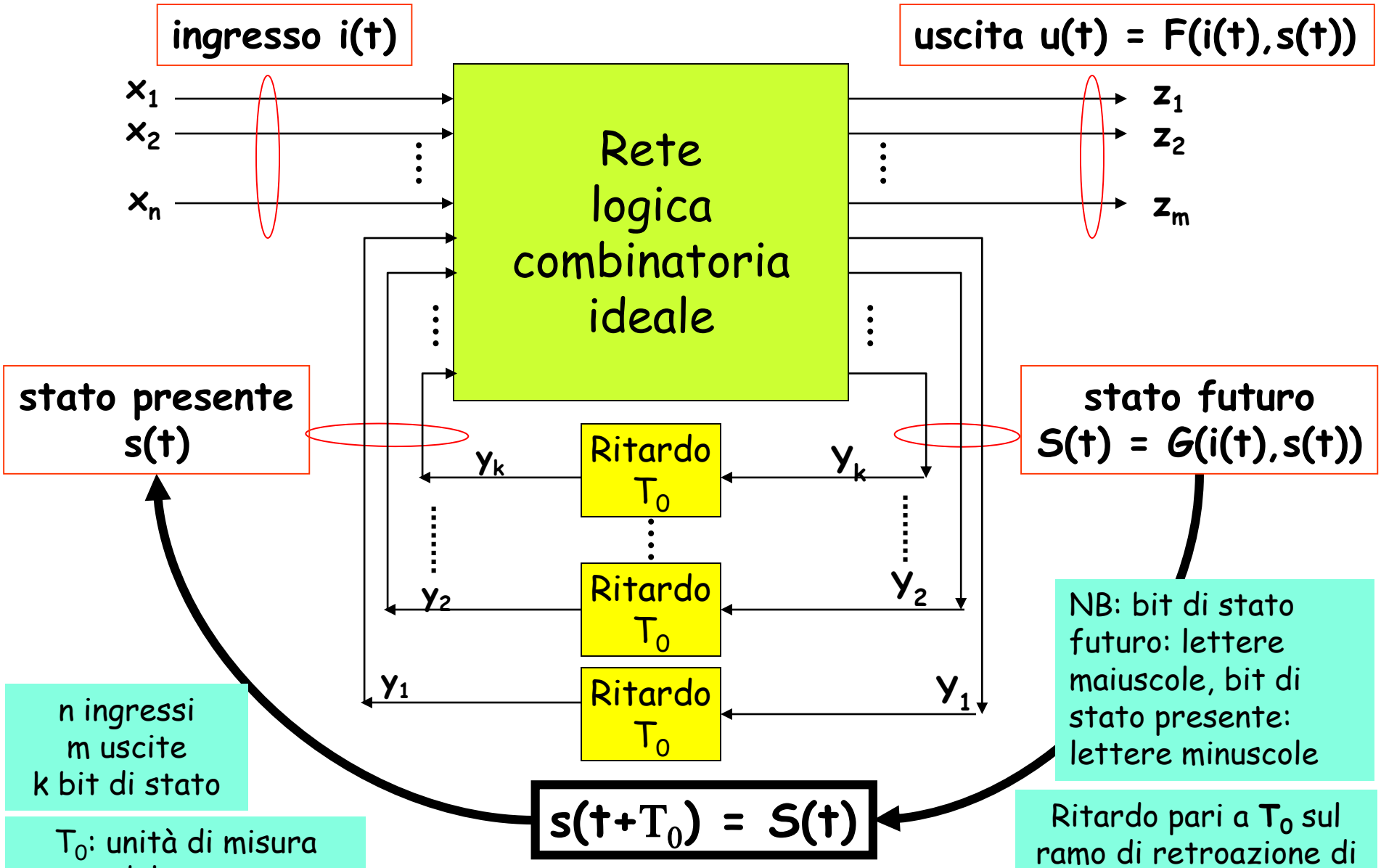
Capitolo 6
Reti Sequenziali Sincrone

Reti Logiche T

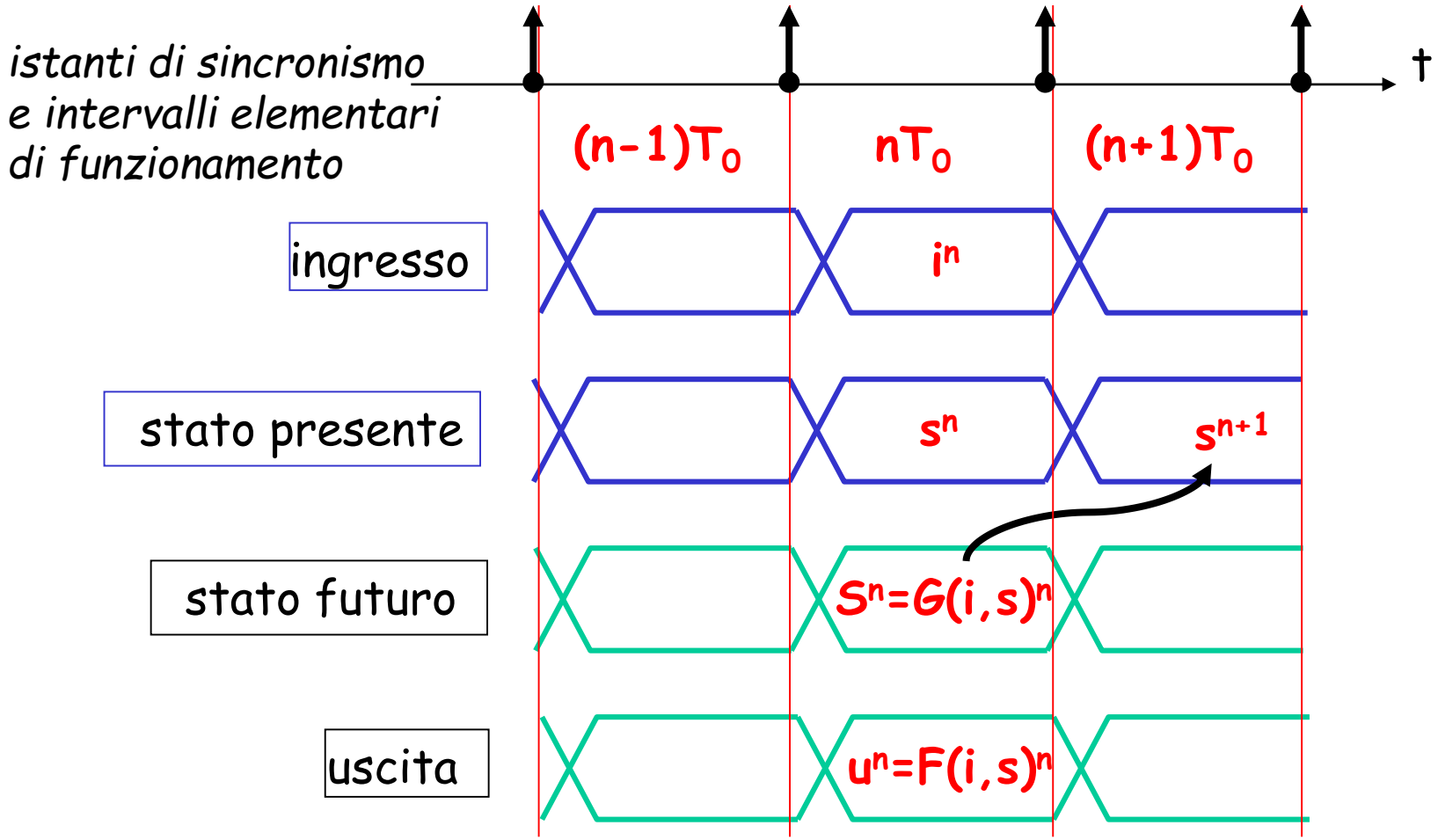
6.1 Struttura e comportamento

Struttura


ingresso i(t)
uscita u(t) = F(i(t),s(t))
x_1
x_2
x_n
z_1
z_2
z_m
Rete logica combinatoria ideale
stato presente s(t)
stato futuro S(t) = G(i(t),s(t))
y_k
y_k
Ritardo T_0
y_2
y_2
Ritardo T_0
y_1
y_1
Ritardo T_0
n ingressi
m uscite
k bit di stato
T_0: unità di misura del tempo
$s(t+T_0) = S(t)$
NB: bit di stato futuro: lettere maiuscole, bit di stato presente: lettere minuscole
Ritardo pari a T_0 sul ramo di retroazione di ogni variabile di stato

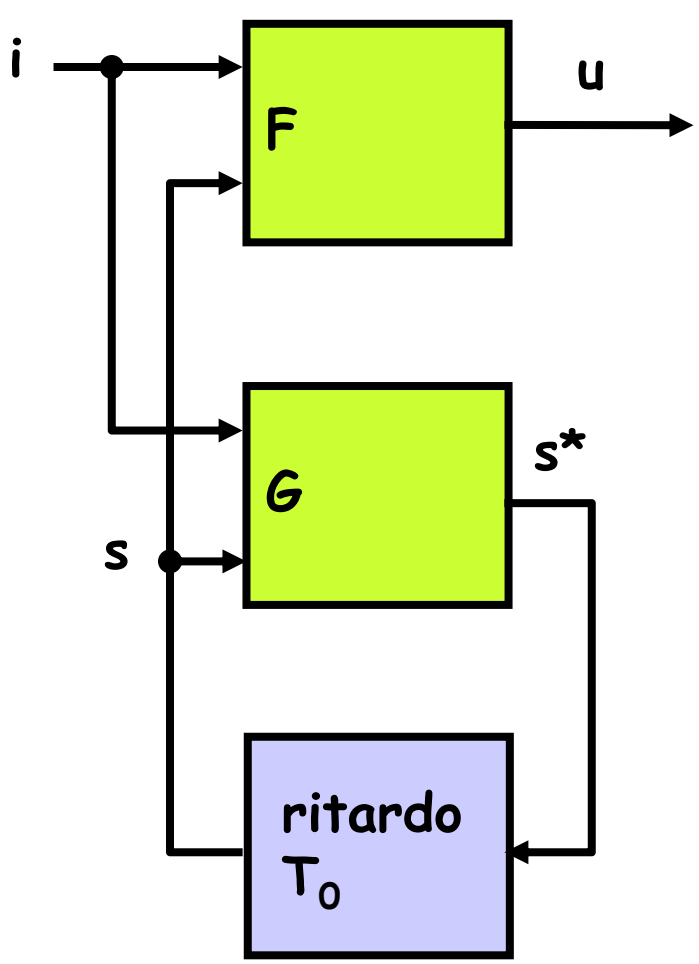
Comportamento ideale

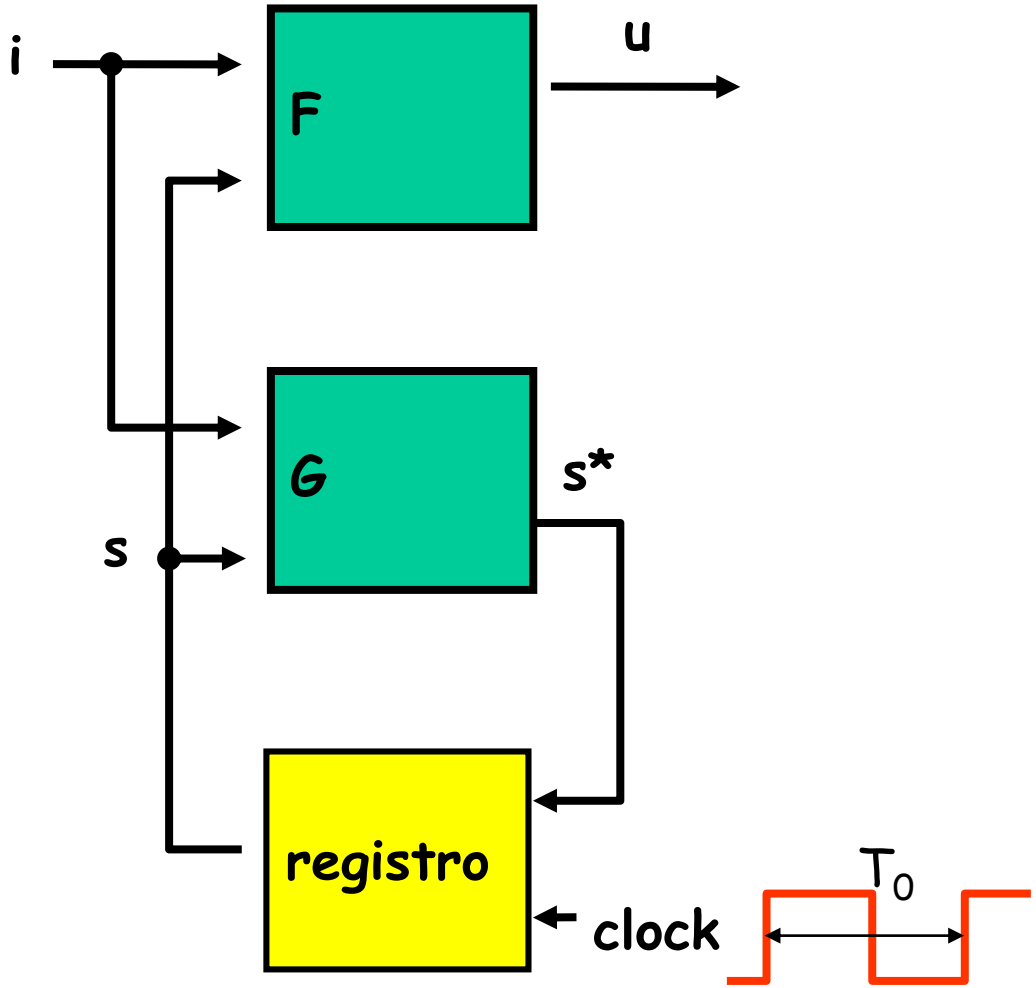
- Lo stato presente si aggiorna **solo** all'inizio di ogni periodo di clock
- Conseguentemente, stato futuro ed uscita si modificano (assumendo una RC ideale) **solo** in corrispondenza del fronte di salita del clock (*istante di sincronismo*)



Realizzazione

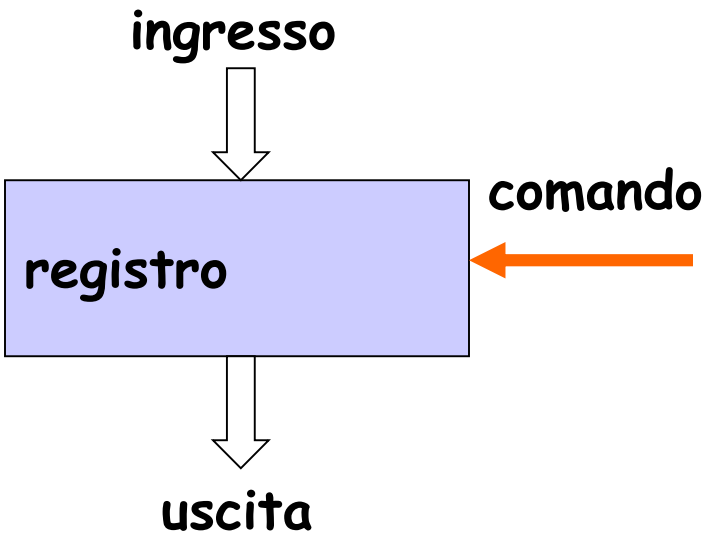
- Per realizzare un ritardo di retrazione costante e pari a T_0 si utilizza un registro ed un oscillatore con periodo T_0 (il **clock**)

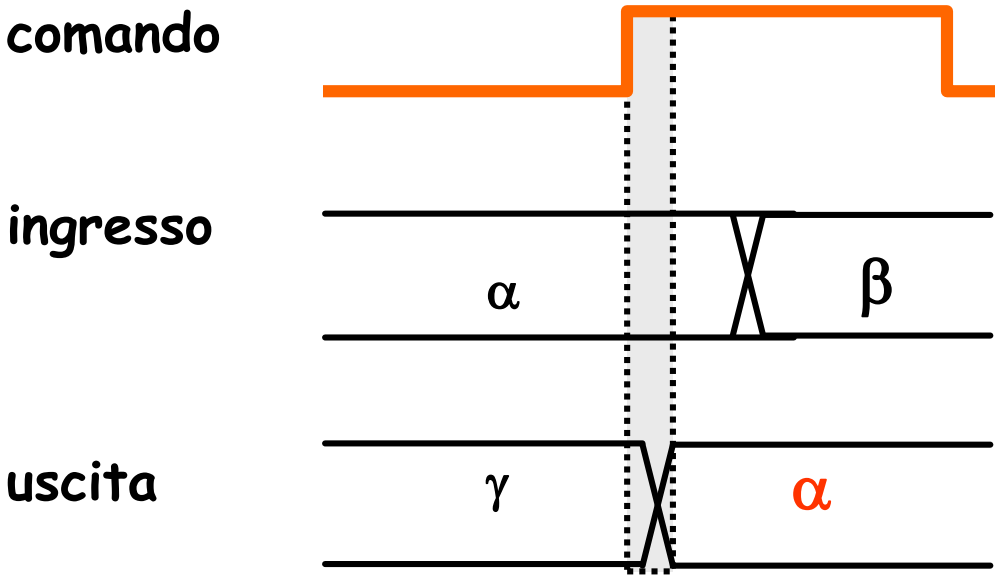




Il registro

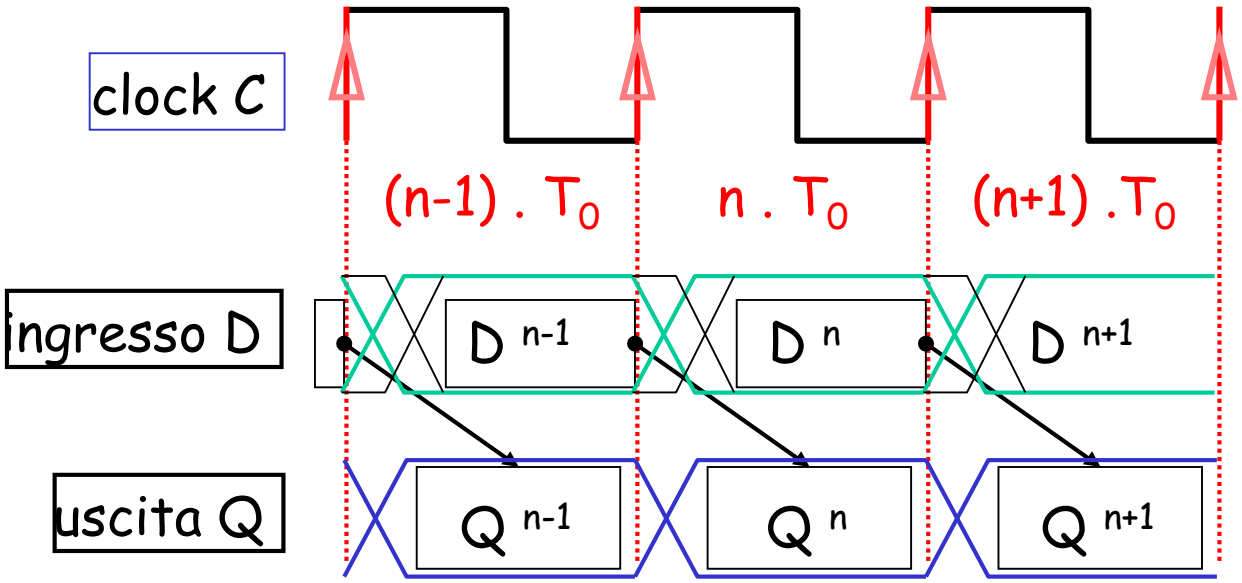
- **Registro** - Macchina sequenziale che memorizza e rende disponibile in uscita un "dato" precedentemente fornitogli in ingresso
- La scrittura di un nuovo "dato" è stabilita da un comando esterno detto "clock".
- In questo modo, il dato memorizzato nel registro viene sovrascritto ad ogni nuovo fronte del clock

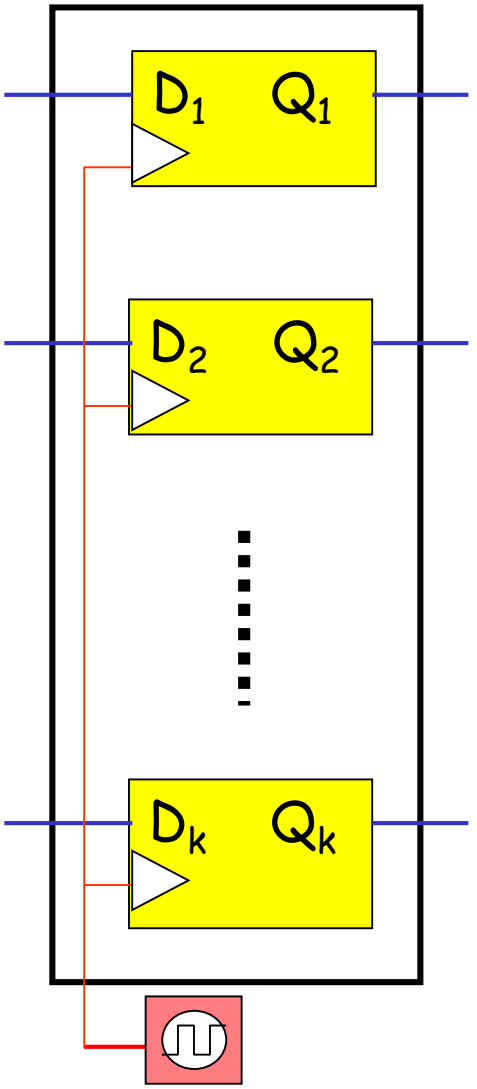




Il flip-flop D come "ritardo" di durata T_0

- Per realizzare il registro si possono utilizzare k **flip-flop D** (tanti quanti i bit di stato da memorizzare), tutti collegati al medesimo segnale di clock con periodo T_0
- Ingressi $D_1,..,D_k$: sono collegati alle uscite della RC che realizza *G* e rappresentano i bit di stato futuro
- Uscite $Q_1,..,Q_k$: sono collegate agli ingressi delle RC che realizzano F e *G* e rappresentano i bit di stato presente
- Equazione caratteristica del flip-flop D: $Q^{n+1} = D^n$





Rete sequenziale sincrona a flip-flop D

- Durante l'n-esimo intervallo elementare, a partire dagli n ingressi correnti $(x_1,..,x_n)^n$ e dai k bit di stato presente $(y_1,.., y_k)^n$ la RSS aggiorna:
  - gli m segnali di uscita z_i^n (mediante F)

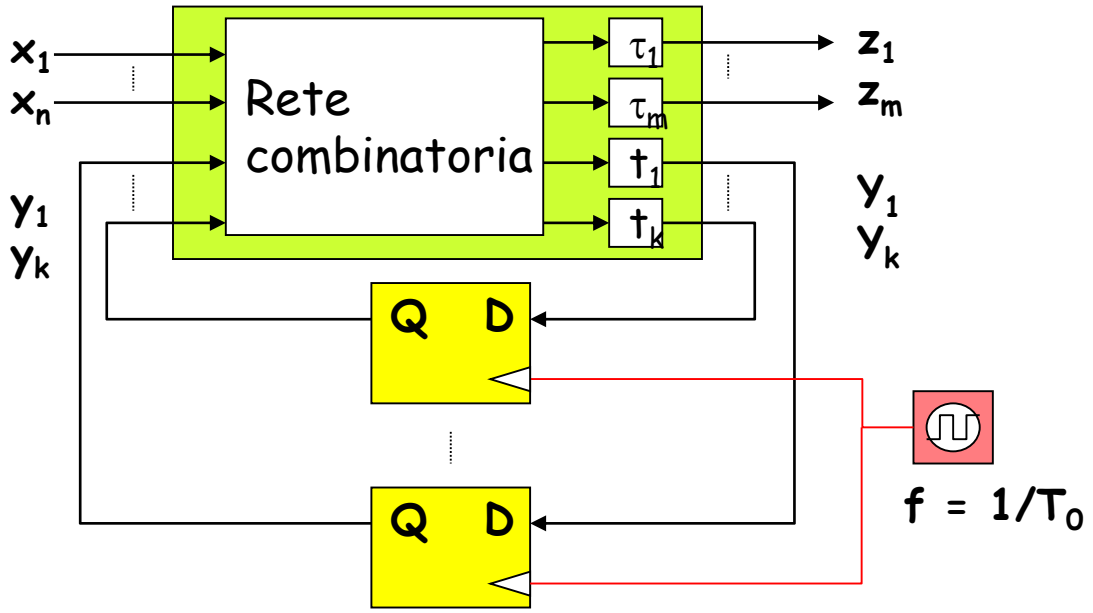
$$z_i^n = F_i\,(x_1, \ldots, x_n, y_1, \ldots, y_k)^n \quad \text{per } i = 1, \ldots, m$$

  - I k bit di stato futuro Y_i^n mediante *G*

$$Y_i^n = G_i\,(x_1, \ldots, x_n, y_1, \ldots, y_k)^n \quad \text{per } i = 1, \ldots, k$$

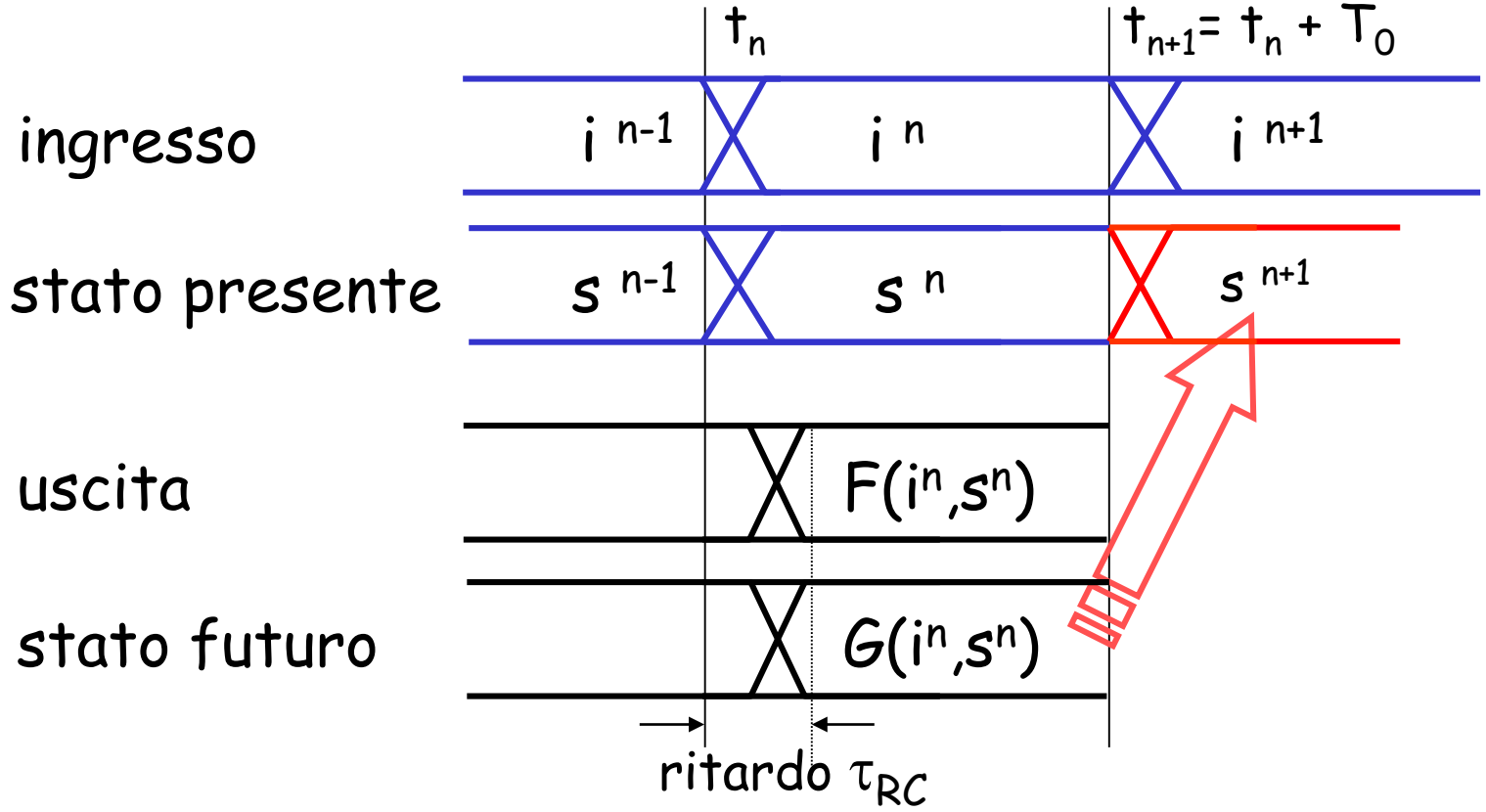
- Tramite utilizzo di k **flip-flop D**, i bit di stato futuro Y_i^n diventano i bit di stato presente y_i^{n+1} per l'intervallo successivo n+1 (v. eq. caratt. FF D)

$$y_i^{n+1} = Q_i^{n+1} = D_i^n = Y_i^n \quad \text{per } i = 1, \ldots, k$$



Vincoli sulla frequenza di funzionamento

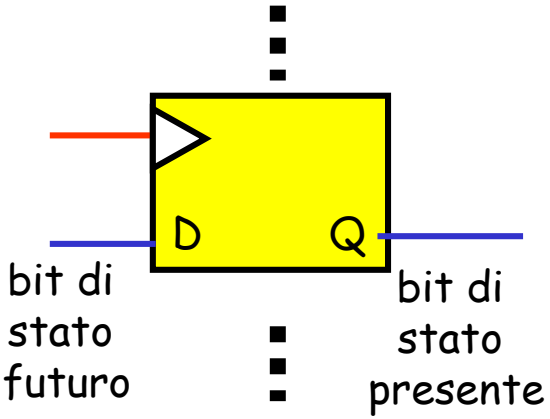
- In condizioni reali, le due reti combinatorie che realizzano le funzioni F e G introducono un ritardo
- Le uscite e lo stato futuro saranno dunque disponibili solo dopo un tempo τ_{RC} a partire dal fronte del clock
- Questa caratteristica impone il vincolo che $T_0 > \tau_{RC}$ (*più lenta è la rete, più bassa la frequenza di funzionamento*)

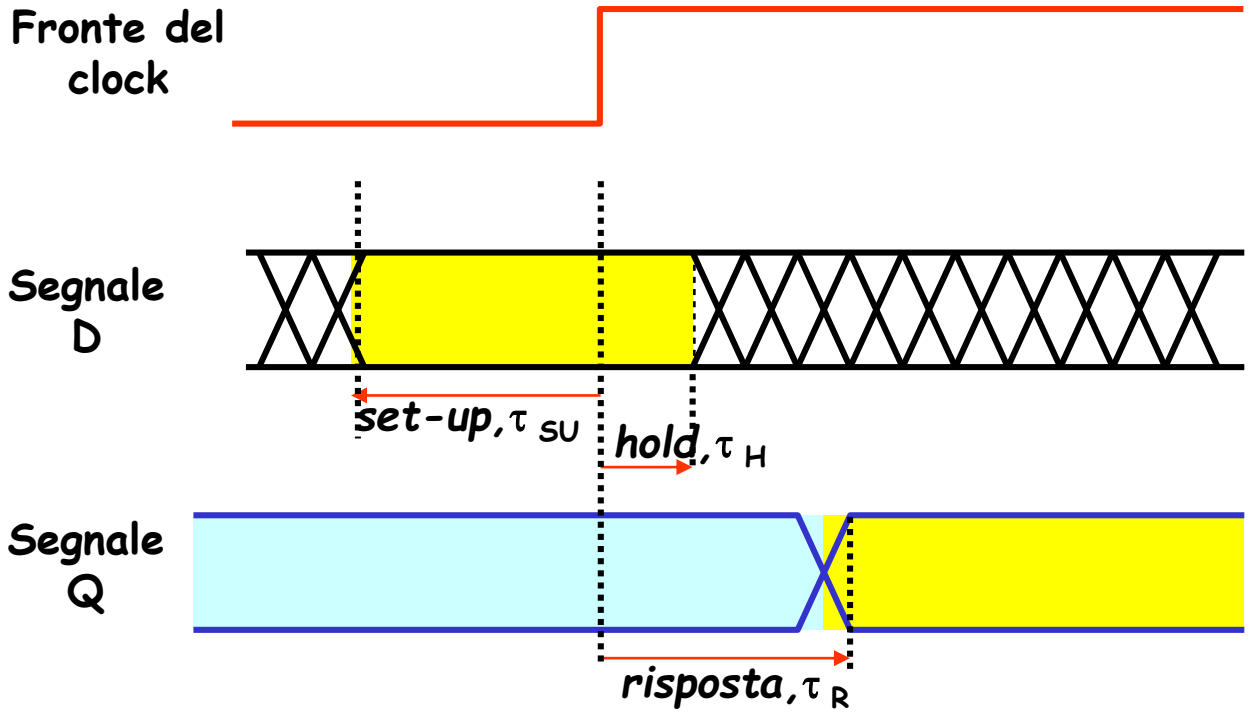


τ_{RC} : intervallo di tempo impiegato dal calcolo di F e di G

Vincoli sulla frequenza di funzionamento (2)

- In aggiunta al ritardo della RC, anche i FF D impongono dei vincoli su T_0
- Ogni FF D richiede che il segnale da campionare mantenga il valore per un tempo τ_{SU} (**set-up time**) prima e un tempo τ_H (**hold time**) dopo il fronte del clock
- Il FFD è inoltre caratterizzato da un ulteriore ritardo (**response time**, τ_R) a partire dal fronte del clock per generare il segnale Q
- Nelle realizzazioni integrate dei FFD si ha sempre che $\tau_H < \tau_R$



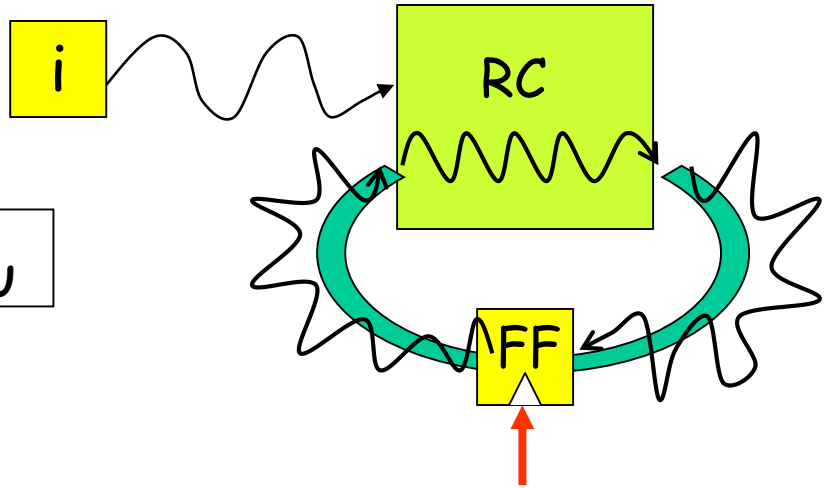


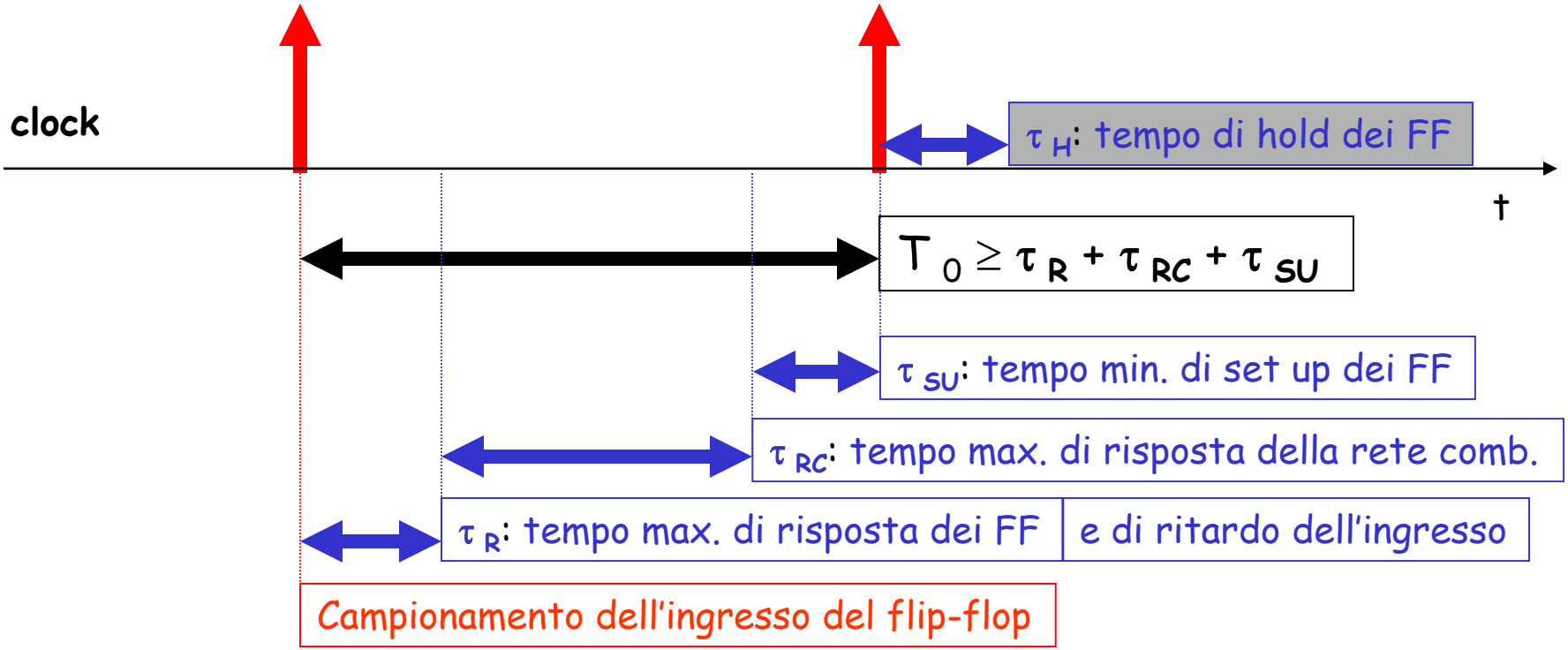
Vincoli sulla frequenza di funzionamento (3)

- Il periodo di clock deve essere dunque maggiore della somma dei ritardi di risposta del FFD, rete combinatoria e set-up del FFD:

$$T_0 \geq \tau_R + \tau_{RC} + \tau_{SU}$$

- Questo vincolo individua la massima frequenza di clock di una RSS

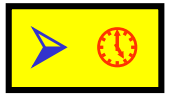




Considerazioni: sincrona vs. asincrona

- La rete sincrona è più «potente» della asincrona: con ingressi costanti, si possono avere uscite diverse ad istanti diversi (uscita può dipendere dal trascorrere del tempo)

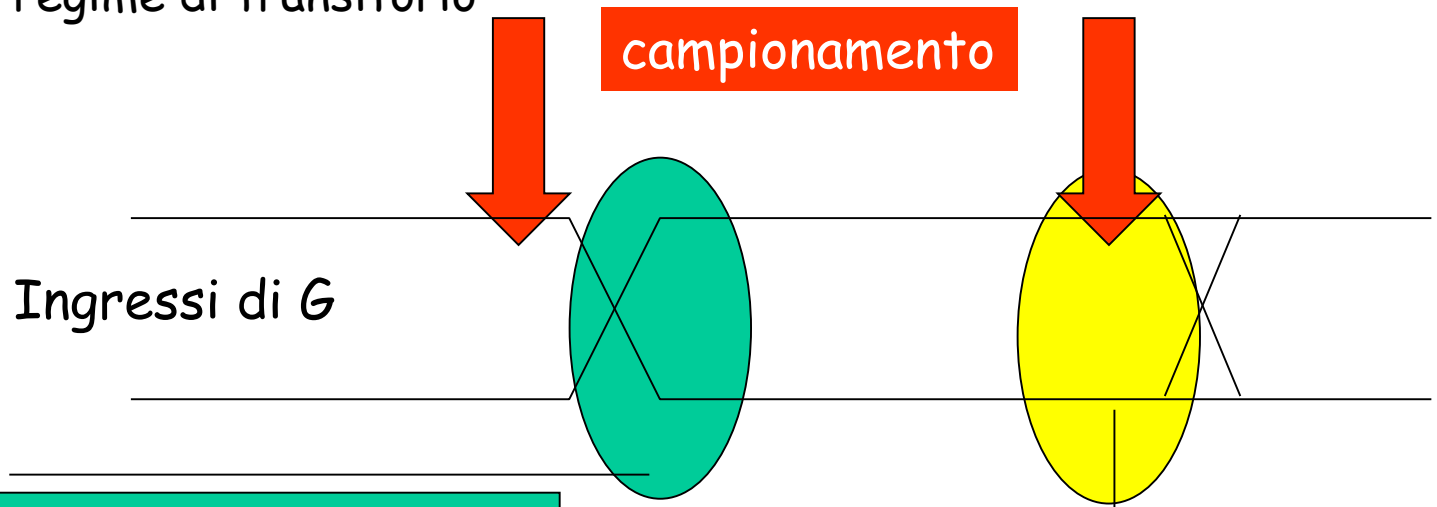
Rete asincrona 

Rete sincrona 

- la rete sincrona è piu lenta della asincrona: $T_0 > t_p$; all'interno di ciascun intervallo di tempo T_0 la macchina non modifica il suo stato
- A differenza delle reti asincrone, con le reti sincrone:
  - Non occorre preoccuparsi dei fenomeni di **alea** (statica e dinamica) dato che si verificano all'inizio di ciascun intervallo e terminano dunque prima del successivo campionamento
  - Non occorre preocuparsi dell'**adiacenza** delle configurazioni consecutive di ingresso e stato e del problema delle **corse critiche** (per lo stesso motivo)
  - È dunque possibile impiegare **espressioni minime** per le funzioni di stato e d'uscita
  - Si possono **codificare arbitrariamente** i simboli d'ingresso e uscita

Conseguenze del campionamento a regime

- L'evento che «scatena» il regime di transitorio non è più, come in una RSA, la variazione degli ingressi, bensì il fronte del segnale di clock
- Se i vincoli visti in precedenza sono soddisfatti, al contrario di una RSA, una RSS non campionerà mai G (funzione di aggiornamento dello stato interno) in regime di transitorio



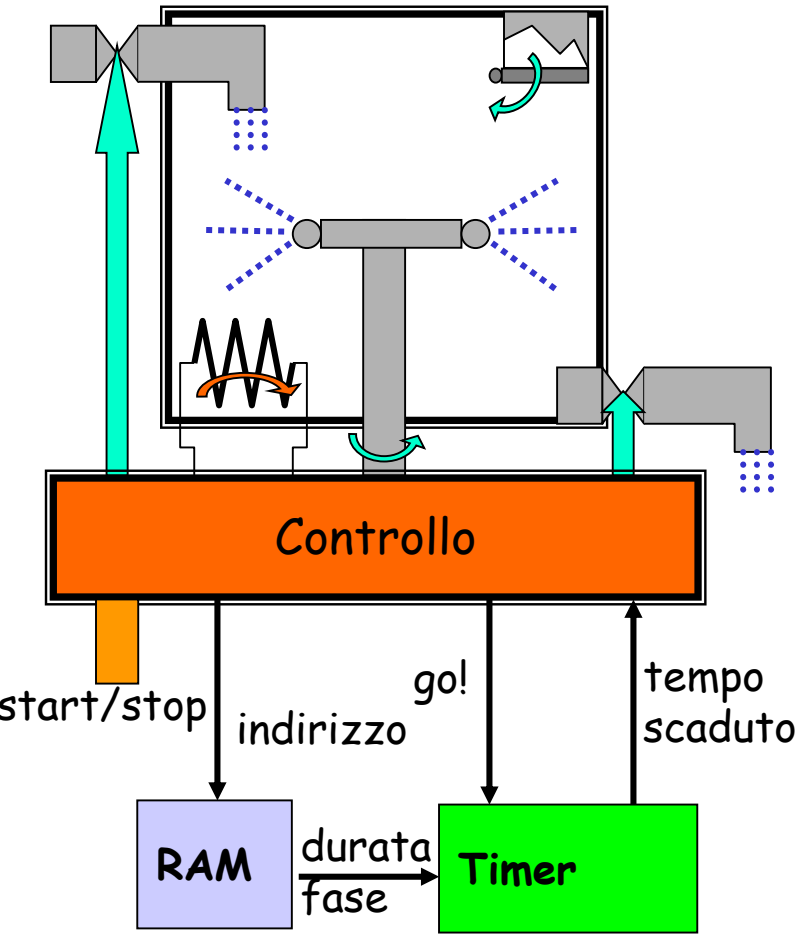
Rete asincrona
L'aggiornamento dello stato avviene qui

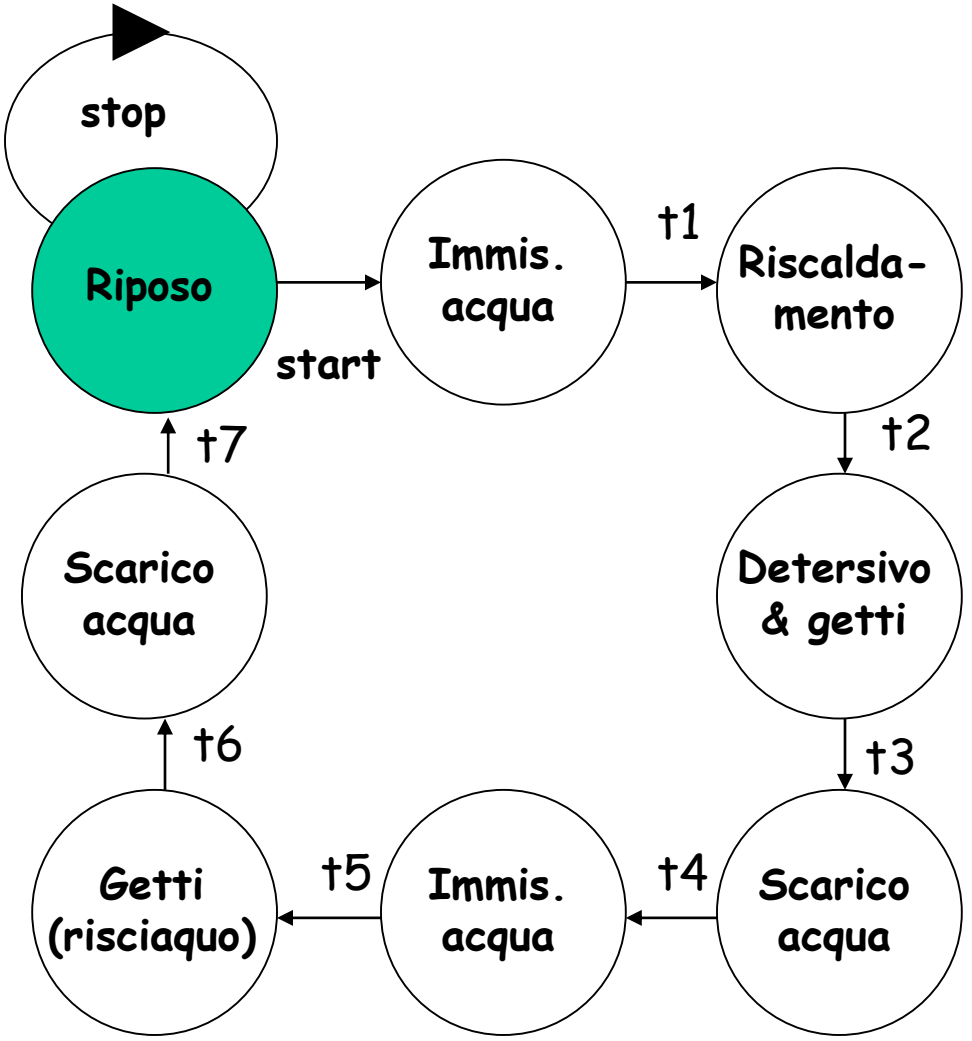
Rete sincrona
L'aggiornamento dello stato avviene qui

Vincoli RSA:
Eliminazione delle alee statiche
Adiacenza degli ingressi consecutivi
Adiacenza degli stati consecutivi

Vantaggi RSS:
Espressioni minime
Codice d'ingresso arbitrario
Codice di stato arbitrario

Esempio di macchina sincrona: la lavapiatti



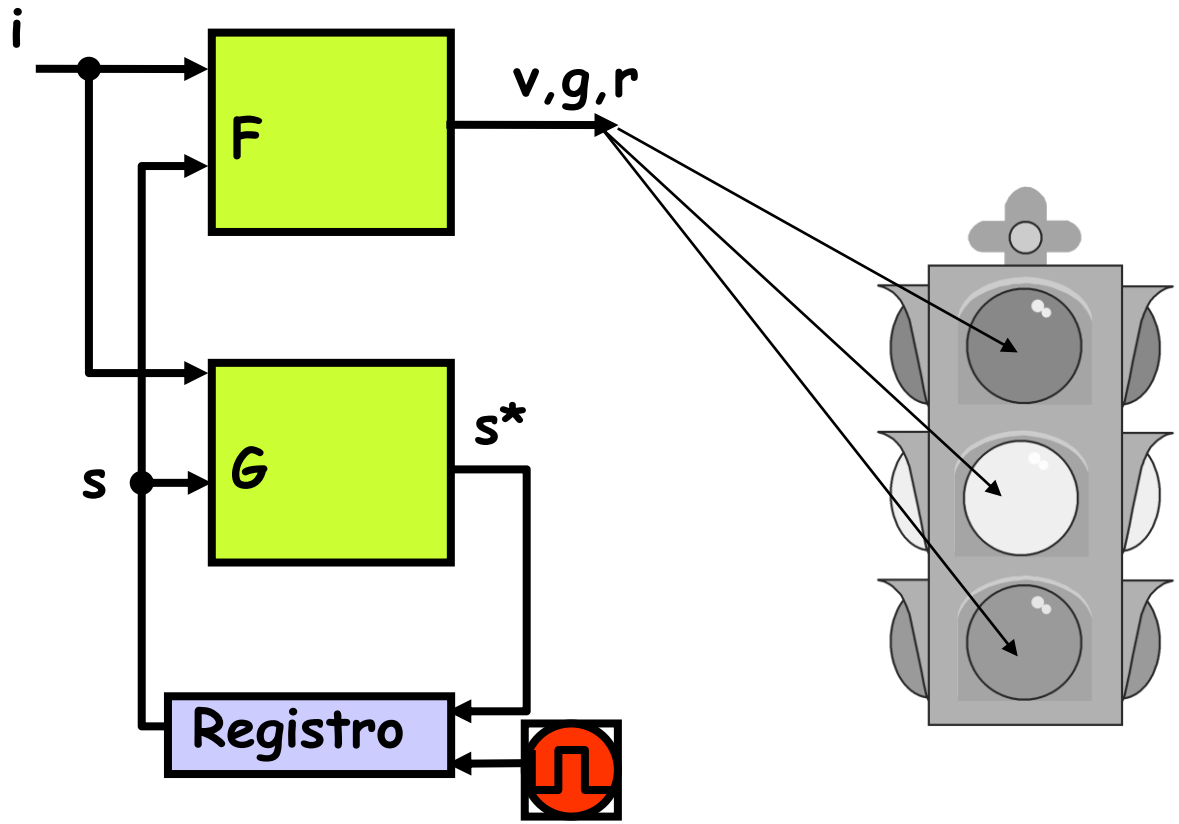


I cambiamenti di stato avvengono quando il timer segnala che è scaduto il tempo di attesa richiesto per la fase associata allo stato presente.

Esercizio - Il semaforo

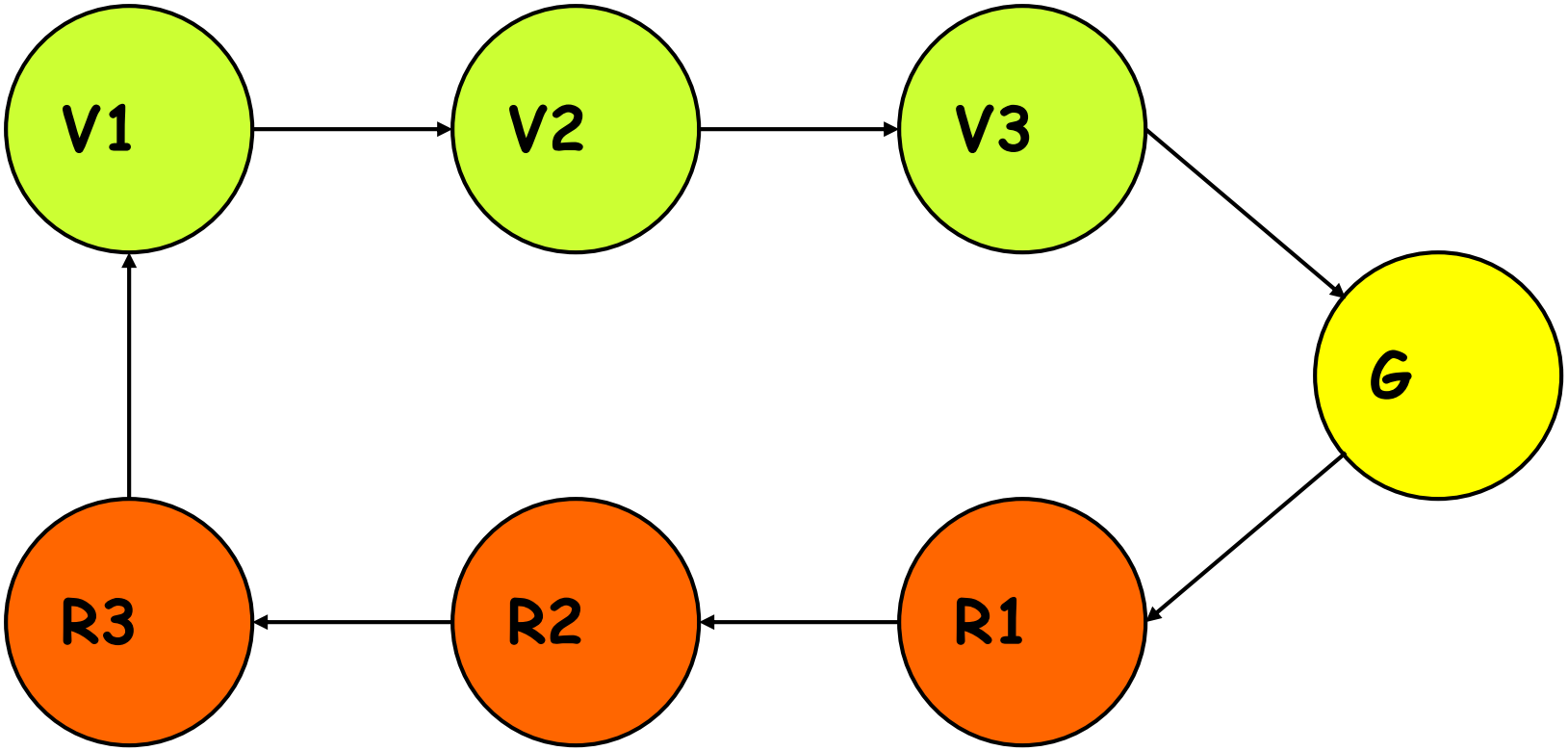
- Una RSS è adibita al controllo di un semaforo
- La rete si basa su un oscillatore con periodo di 20 s e deve soddisfare le seguenti specifiche (7 intervalli da 20s l'uno):

rosso = 60 s, *giallo* = 20 s, *verde* = 60 s



T_0 = 20 s

Semaforo - grafo degli stati



Semaforo - tabella di flusso

stato presente	stato futuro	lampada (uscite) verde	giallo	rosso
V1	V2	accesa	spenta	spenta
V2	V3	accesa	spenta	spenta
V3	G	accesa	spenta	spenta
G	R1	spenta	accesa	spenta
R1	R2	spenta	spenta	accesa
R2	R3	spenta	spenta	accesa
R3	V1	spenta	spenta	accesa

- 7 stati interni, codificabili con 3 bit, che richiedono 3 FF D
- In questo modo dalla tabella di flusso si ottiene la tabella delle transizioni (v. prossima slide)

Semaforo - la macchina sequenziale

Stato interno

$s = y_2 y_1 y_0$ (7 stati)

Uscita

$u = z_1 z_2 z_3$

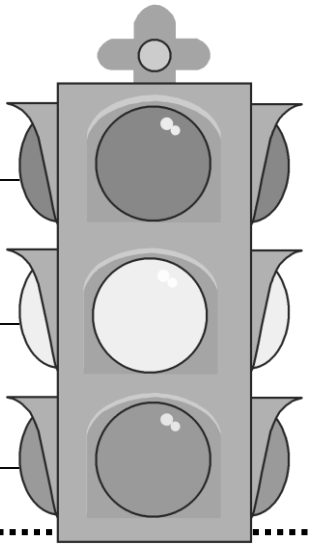
Comportamento:

$s_2 \leftarrow (s+1)_2 \bmod 7$

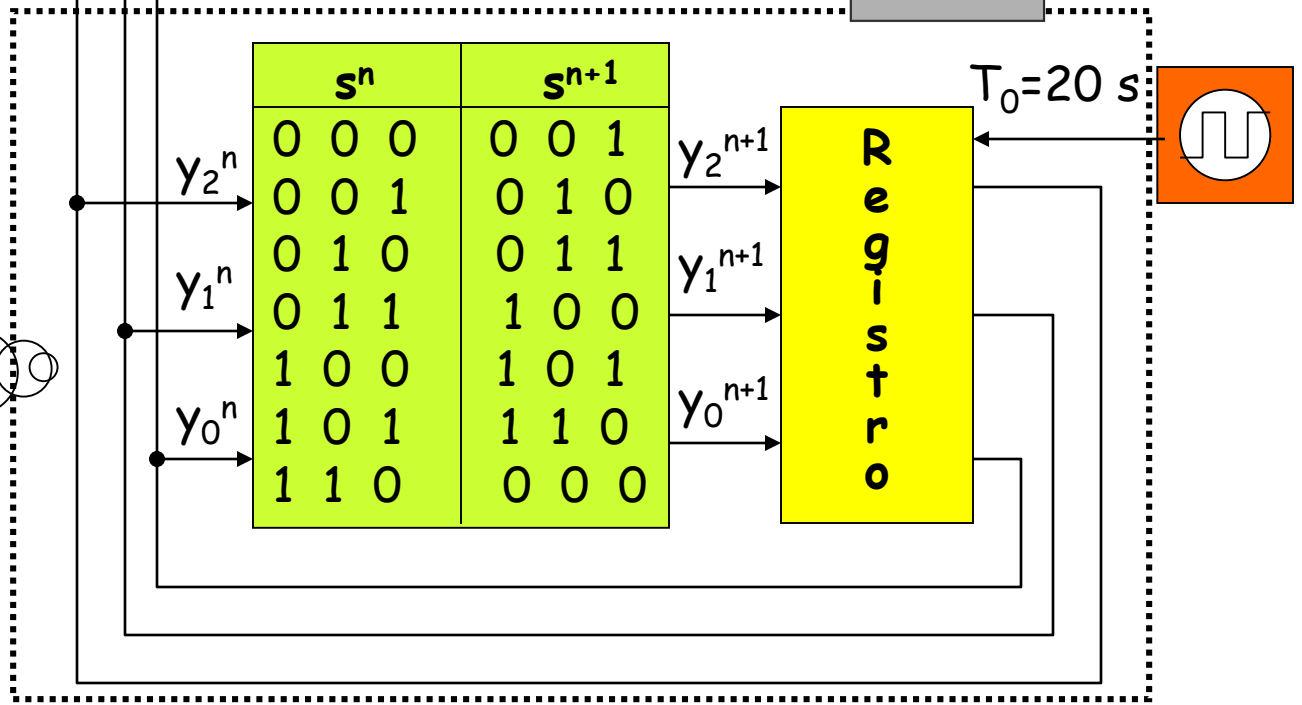
$u \leftarrow F(s)$

Il comportamento di *G* è quello di un contatore da "zero" a "sei" (modulo 7)

s^n	u^n
0 0 0	1 0 0
0 0 1	1 0 0
0 1 0	1 0 0
0 1 1	0 1 0
1 0 0	0 0 1
1 0 1	0 0 1
1 1 0	0 0 1



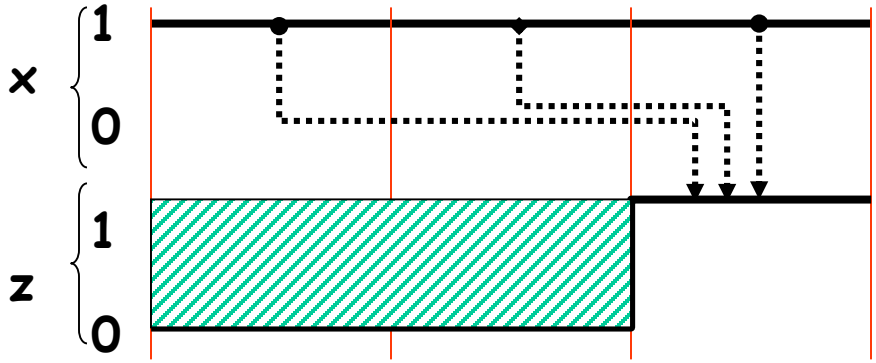
s^n	s^{n+1}
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	0 0 0



Esempio: riconoscimento di sequenze

Una rete sequenziale sincrona ha un ingresso x ed una uscita z.
La relazione ingresso/uscita è descritta dalla seguente frase:

"$z^n = 1$ quando $x^n = 1$ e solo se $x^{n-1} = x^{n-2} = 1$ "

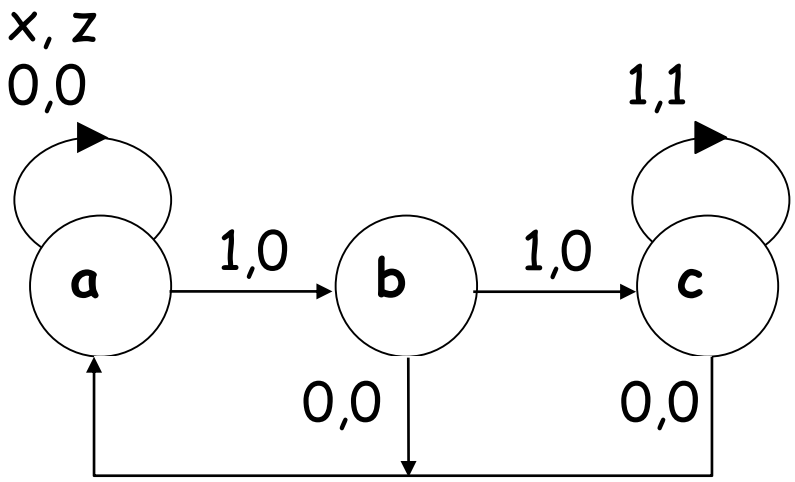


Riconoscitore di 3 "uni" consecutivi $z^n = x^n \cdot x^{n-1} \cdot x^{n-2}$

Per realizzare il grafo degli stati:
- Si traccia la parte di grafo che riconosce la sequenza assegnata, specificando su ogni ramo il valore d'uscita (durata T_0).
- Si completa il grafo, prendendo in considerazione tutte le altre possibili situazioni e curando di renderlo "strettamente connesso" (ogni stato deve poter essere raggiunto da ogni altro).

Grafo degli stati e Tabella di flusso

Modello di Mealy



Es. sequenza in ingresso=0111100

x	0	1	1	1	1	0	0
s.p.	a	a	b	c	c	c	a
s.f.	a	b	c	c	c	a	a
z	0	0	0	1	1	0	0

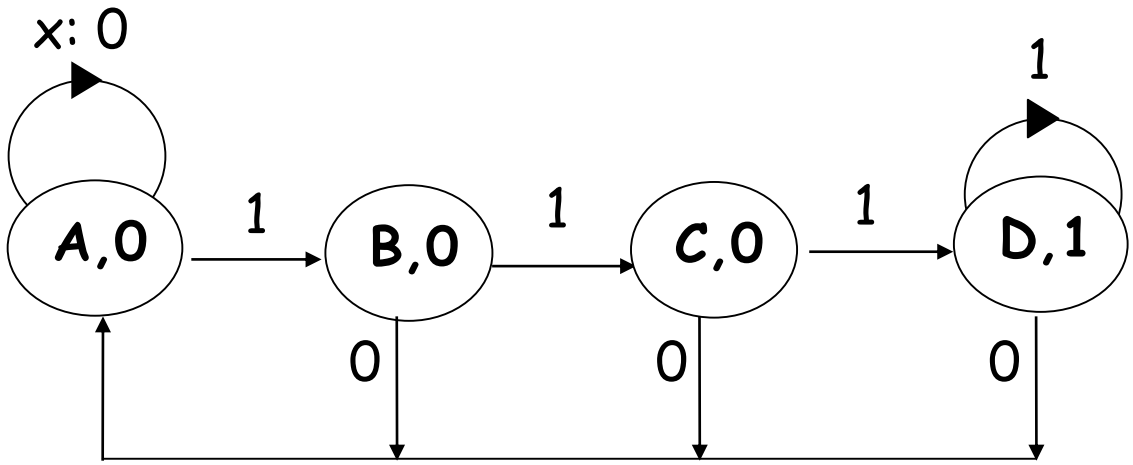
s.p. \ x	0	1
a	a,0	b,0
b	a,0	c,0
c	a,0	c,1

s.f.,z

Ogni stato resta presente per almeno un periodo di clock, ogni cambiamento di ingresso avviene all'inizio di tali intervalli ed ogni transizione si verifica al termine. La stabilità dello stato presente non è una condizione necessaria dopo una variazione di ingresso.

Grafo degli stati e Tabella di flusso

Modello di Moore



Rispetto al modello di Mealy, il modello di Moore:

- richiede uno stato in più (complessità)
- impiega un intervallo in più per riconoscere la sequenza (ritardo)

Es. sequenza in ingresso=0111100

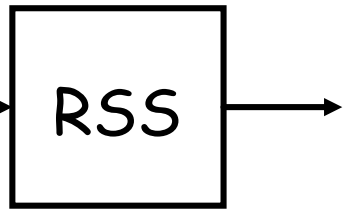
x	0	1	1	1	1	0	0
s.p.	A	A	B	C	D	D	A
s.f.	A	B	C	D	D	A	A
z	0	0	0	0	1	1	0

s.p. \ x	0	1	z
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

s.f.

Mealy vs. Moore

$i \in I$: {a, b, c, d}



$u \in U$: {∅, b, c, d}

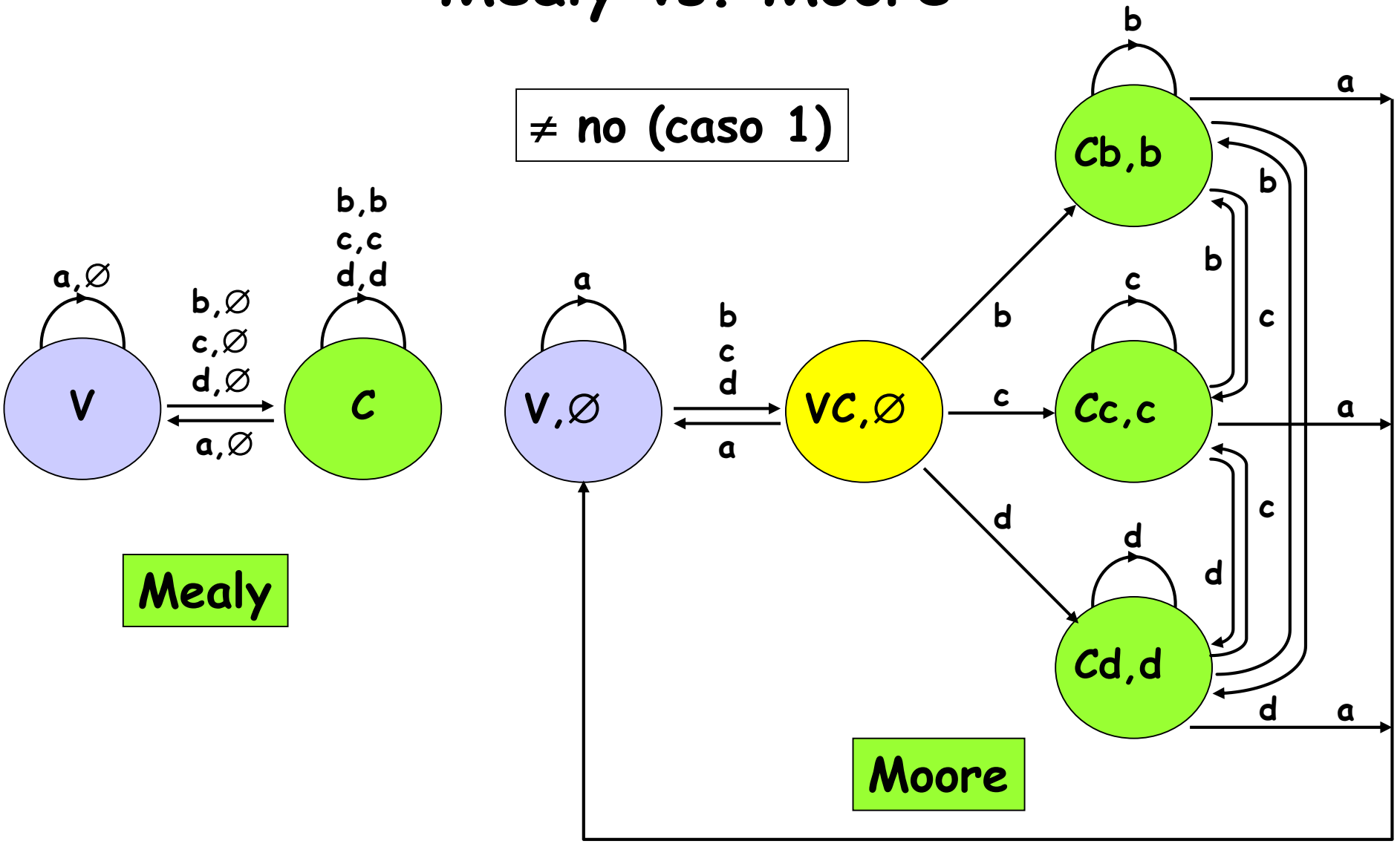
Comportamento:
il simbolo di uscita "u" deve riprodurre il simbolo "i" contestualmente presente in ingresso solo nel caso in cui "i" sia una consonante preceduta da un'altra consonante;
in ogni altro caso, u = ∅

CASO 1: u=i solo se due consonanti anche uguali (≠ no)
CASO 2: u=i solo se due consonanti e diverse (≠ sì)

Es:	a	b	b	b	c	c	d	c	b	b	d	d	d	b	c	c	a	
≠sì	∅	∅	∅	∅	c	∅	d	c	b	∅	d	∅	∅	b	c	∅	∅	**Mealy**
≠no	∅	∅	b	b	c	c	d	c	b	b	d	d	d	b	c	c	∅	
≠sì	…	∅	∅	∅	∅	c	∅	d	c	b	∅	d	∅	∅	b	c	∅	**Moore**
≠no	…	∅	∅	b	b	c	c	d	c	b	b	d	d	d	b	c	c	

Il modello di Moore comporta un «ritardo» nelle uscite pari a un periodo di clock!

Mealy vs. Moore

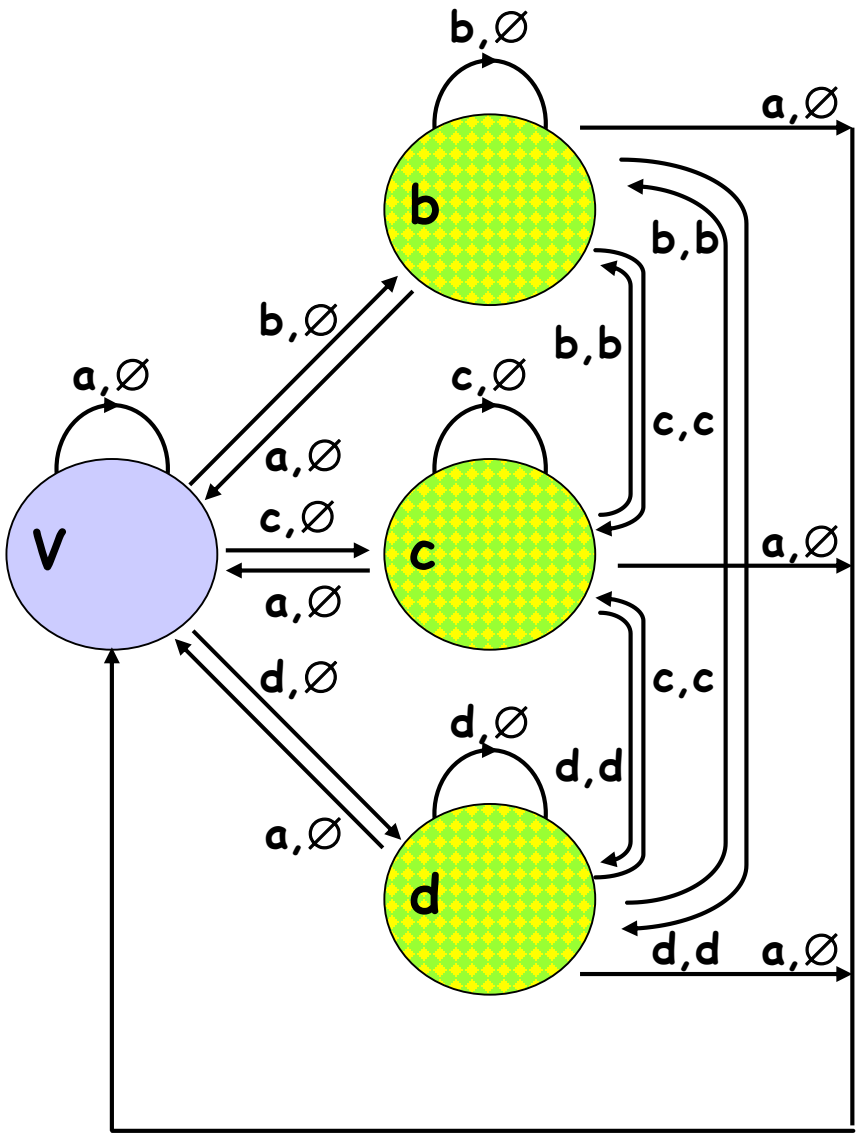


Modello di Mealy: 2 stati (anche se I ≡ alfabeto italiano/inglese)

Modello di Moore: 5 stati (18/23 se I ≡ alfabeto italiano/inglese)

Mealy vs. Moore

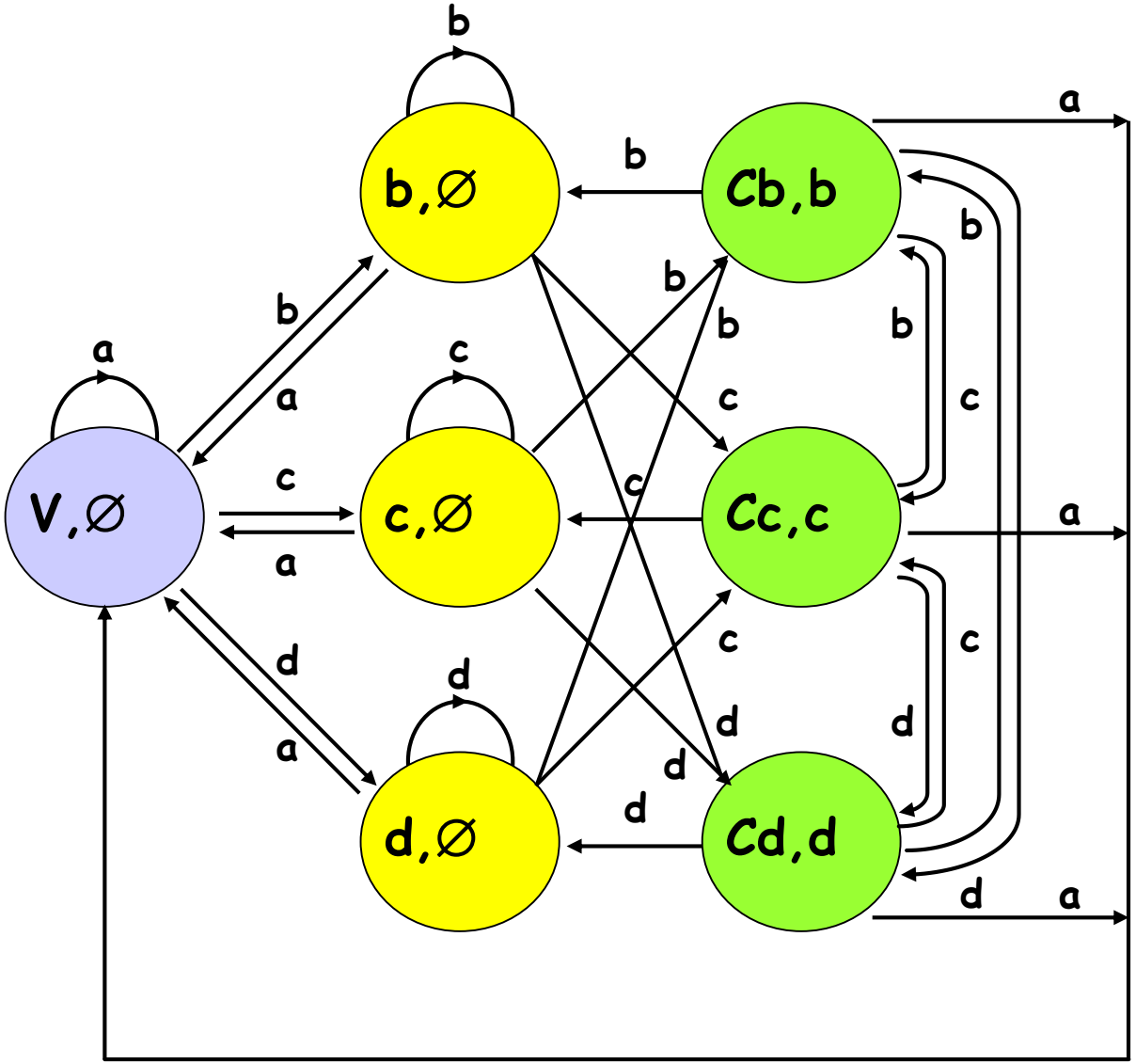
≠ sì (caso 2)



Modello di Mealy: 4 stati (17/22 se I ≡ alfabeto italiano/inglese)

Mealy vs. Moore

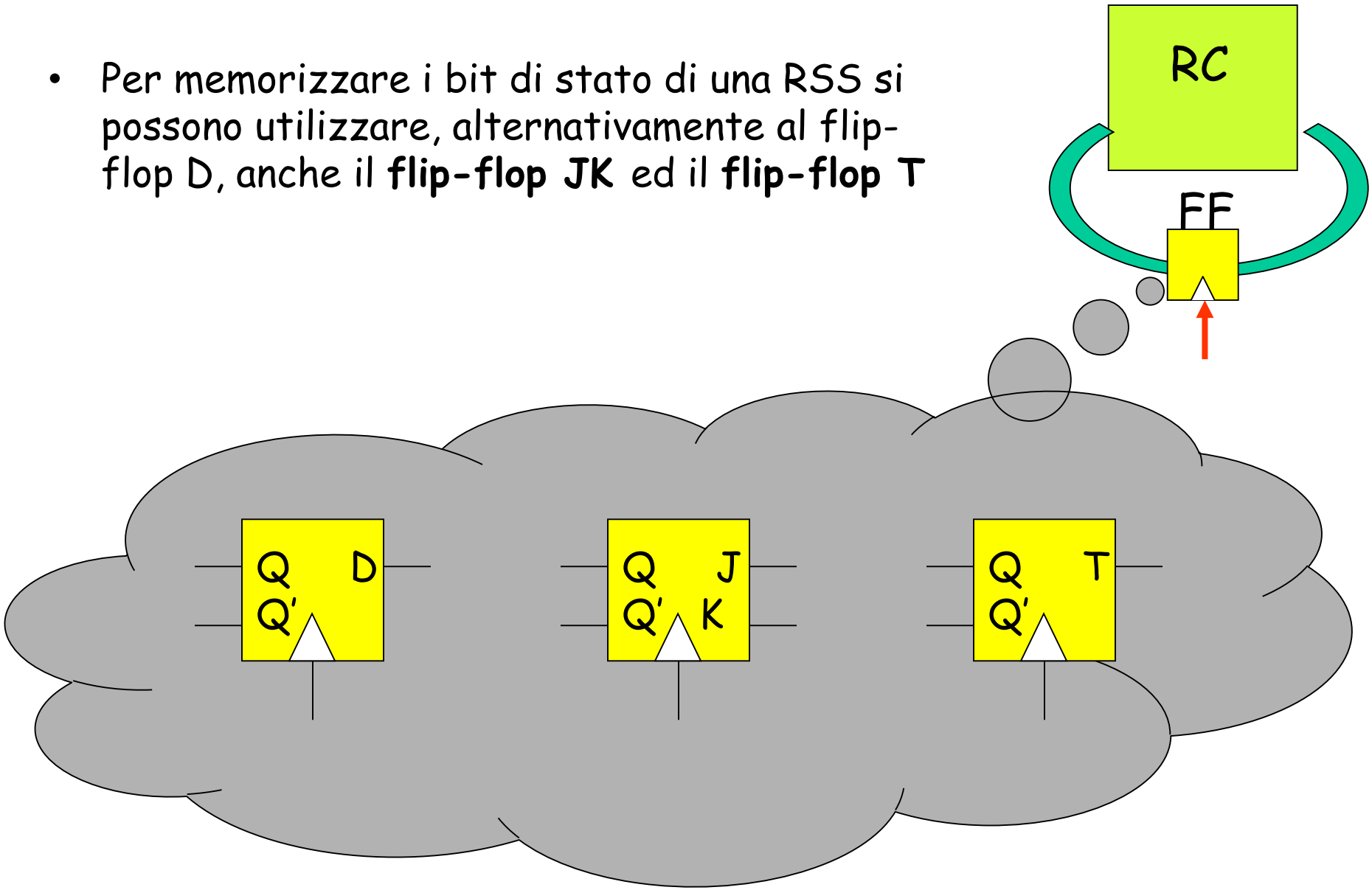
≠ sì (caso 2)



Modello di Moore: 7 stati (33/43 se I ≡ alfabeto italiano/inglese)

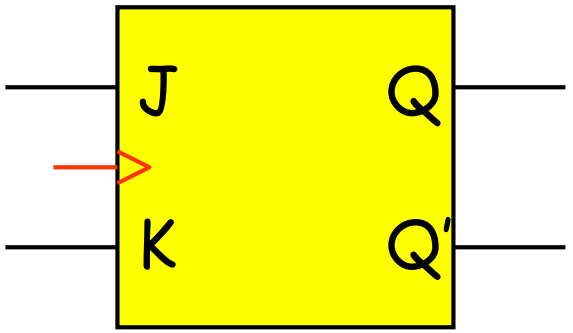
Tipi di flip-flop

- Per memorizzare i bit di stato di una RSS si possono utilizzare, alternativamente al flip-flop D, anche il **flip-flop JK** ed il **flip-flop T**



Il flip-flop JK

- Il **flip-flop JK** è anch'esso una RSA azionata dai fronti di un clock
- 2 ingressi (J,K) e 1 bit di stato interno (che memorizza «0» o «1»)
- La presenza di due ingressi impone 4 diversi comportamenti:



	J^n	K^n	Q^n	Q^{n+1}
hold	0	0	0	0
	0	0	1	1
set	1	0	0	1
	1	0	1	1
reset	0	1	0	0
	0	1	1	0
toggle	1	1	0	1
	1	1	1	0

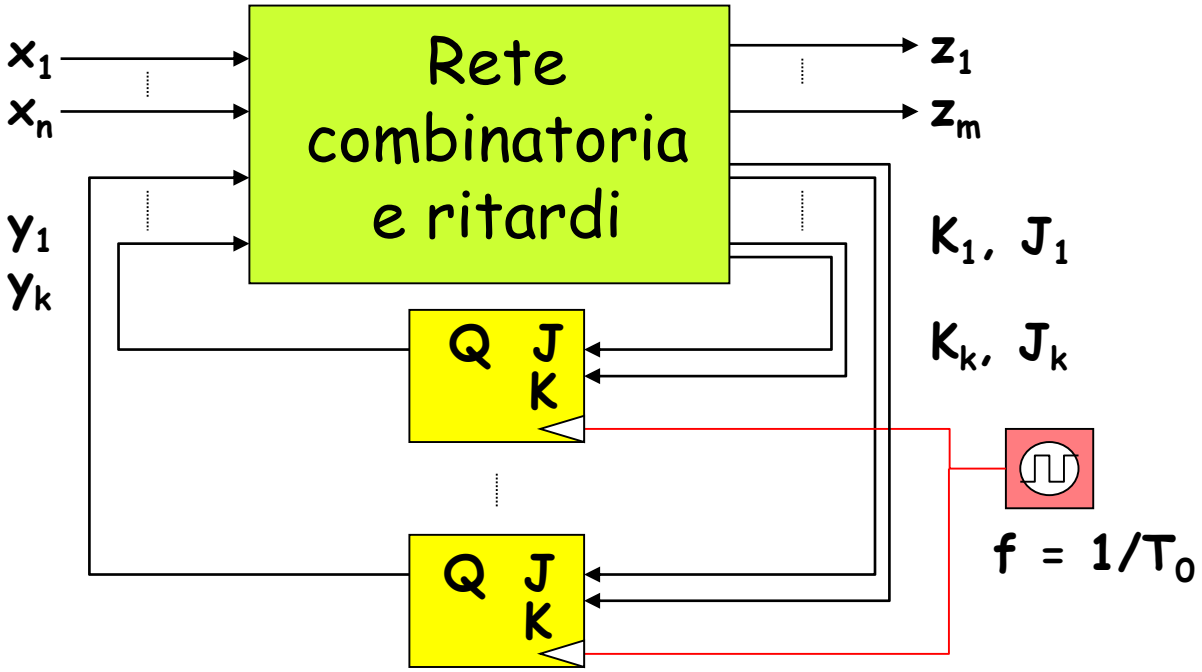
Eq. caratteristica:
$Q^{n+1} = (J.Q' + K'.Q)^n$

Q^n \ J^nK^n	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Q^{n+1}

Rete sequenziale sincrona a flip-flop JK

- Utilizzando flip-flop JK come elementi di memoria per una RSS, i bit di stato futuro $Y_1,..,Y_k$ vengono sostituiti dai segnali $K_1,J_1,..,K_k,J_k$ (realizzati tramite RC)
- Tramite l'eq. caratteristica è possibile dedurre yi a partire da Ki e Ji



$$z_i^{\,n} = F_i(x_1, \ldots, x_n, y_1, \ldots, y_k)^n \quad \text{per } i = 1, \ldots, m$$

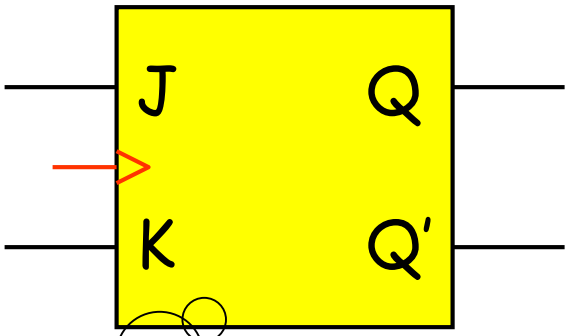
$$y_i^{\,n+1} = (J_i \cdot y_i' + K_i' \cdot y_i)^n \quad \text{per } i = 1, \ldots, k$$

$$\text{con } J_i^{\,n} = J_i(x_1, \ldots, x_n, y_1, \ldots, y_k)^n$$

$$K_i^{\,n} = K_i(x_1, \ldots, x_n, y_1, \ldots, y_k)^n$$

Il flip-flop JK (struttura con FF D)

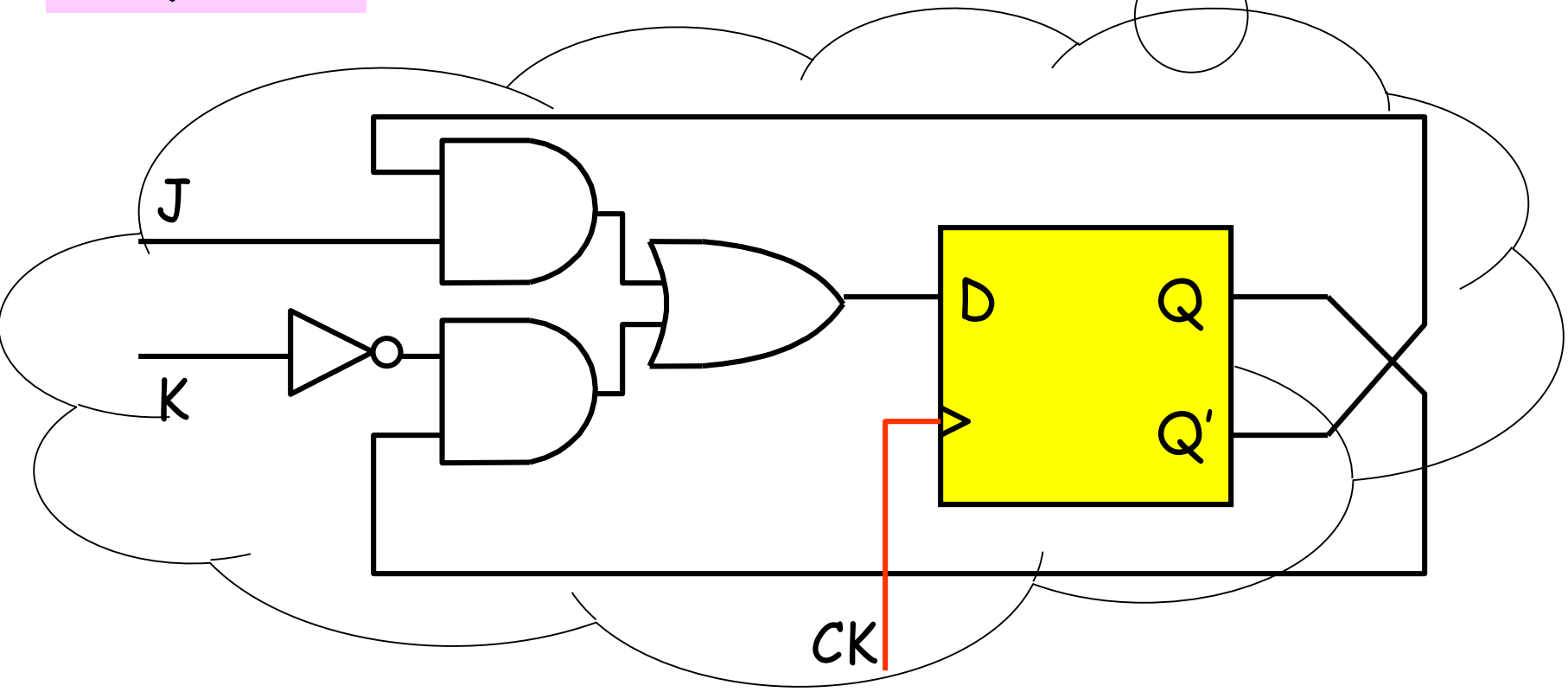
- è possibile dedurre la realizzazione di un flip-flop JK mediante flip-flop D utilizzando le relative eq. caratteristiche



$JK: Q^{n+1} = (J.Q' + K'.Q)^n$

$D: Q^{n+1} = D^n$

$D^n = (J.Q' + K'.Q)^n$



Dal FF JK al FF D

- Viceversa, a partire da un flip-flop JK è possibile costuire un flip-flop D sempre mediante l'equazione caratteristica di un flip-flop JK:

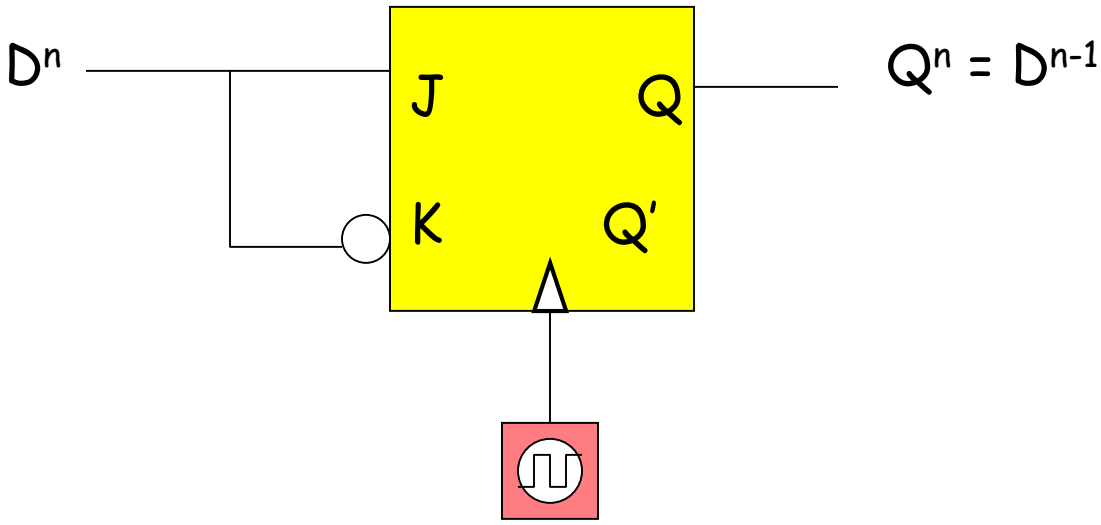
$$Q^{n+1} = (J.Q' + K'.Q)^n$$

Posto J=D e K=D'

$$Q^{n+1} = (D.Q' + D.Q)^n$$

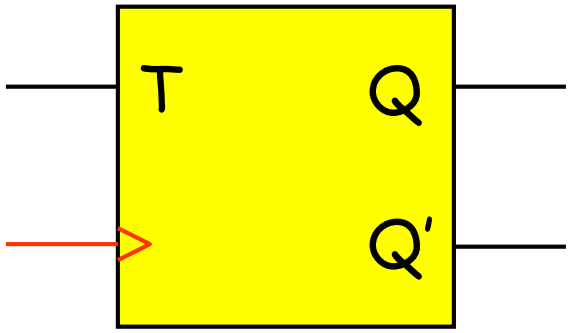
$$Q^{n+1} = D^n$$

E9 (combinazione)

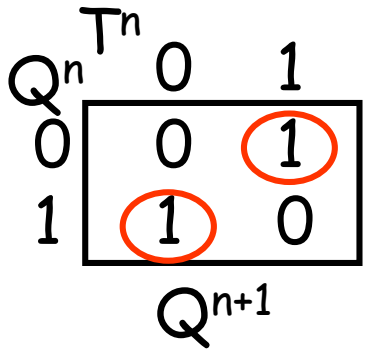


Il flip-flop T

- Il **flip-flop T** ha un unico ingresso T e dunque due sole modalità di funzionamento:



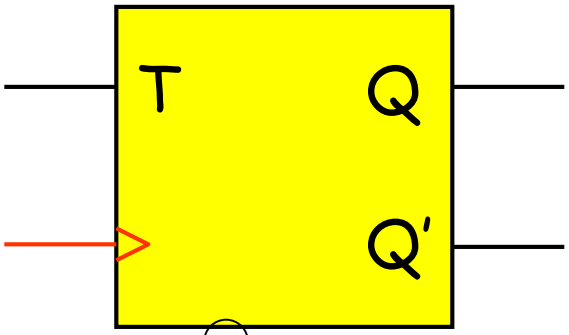
	T^n	Q^n	Q^{n+1}
hold	0	0	0
	0	1	1
toggle	1	0	1
	1	1	0



Eq. caratteristica: $Q^{n+1} = (T.Q' + T'.Q)^n = (T \oplus Q)^n$

Il flip-flop T (struttura con FF D)

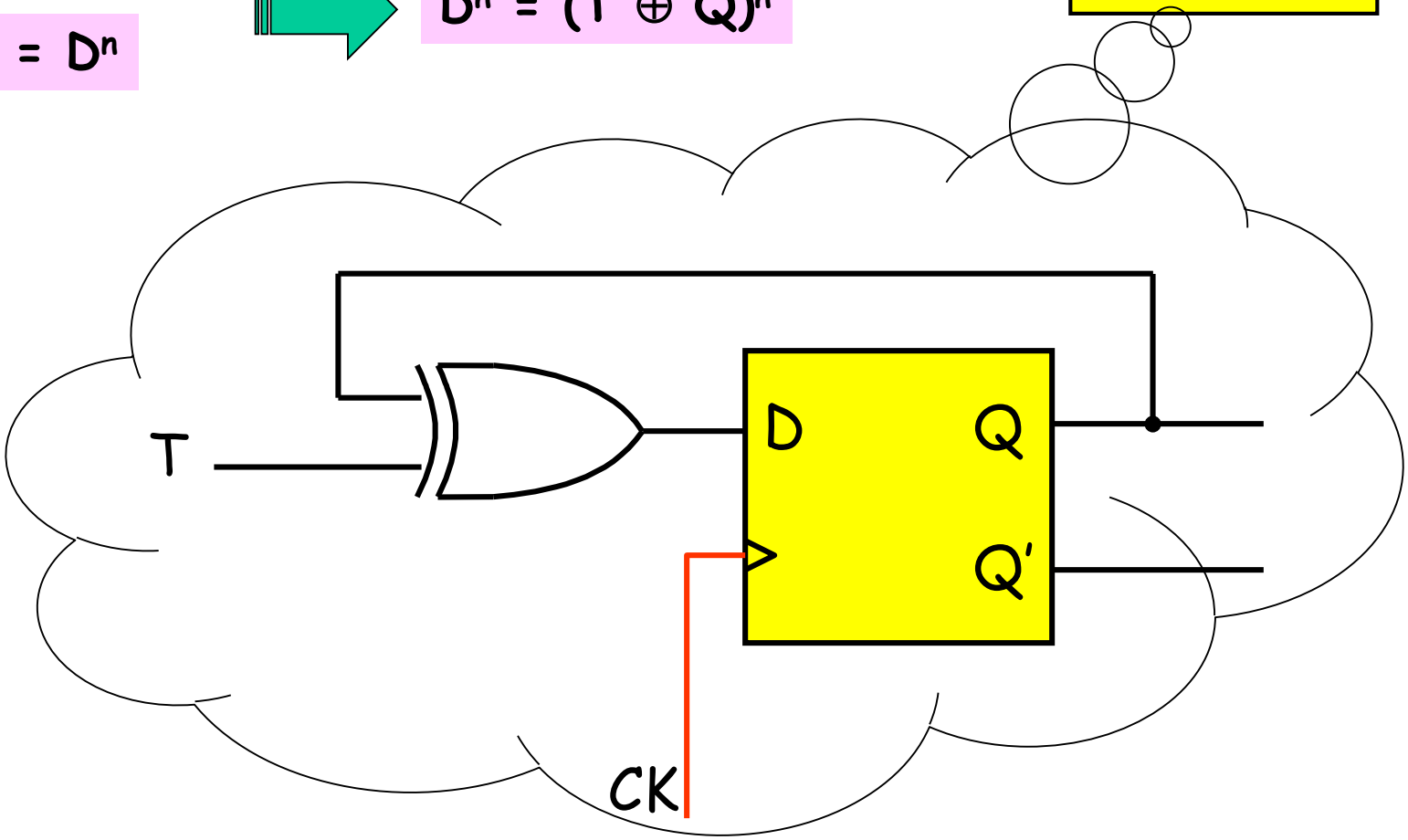
- è possibile dedurre la realizzazione di un flip-flop T mediante flip-flop D utilizzando le relative eq. caratteristiche



$T: Q^{n+1} = (T \oplus Q)^n$

$D: Q^{n+1} = D^n$

$D^n = (T \oplus Q)^n$



Dal flip-flop T al flip-flop D

- Viceversa, a partire da un flip-flop T è possibile costuire un flip-flop D sempre sfruttando l'equazione caratteristica di un flip-flop T:

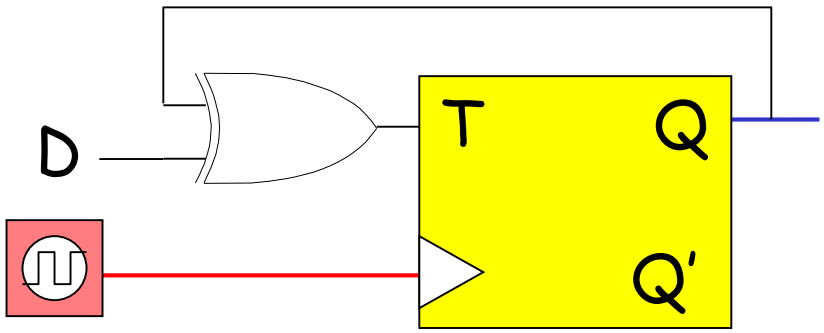
$$Q^{n+1} = (T \oplus Q)^n$$

Posto $T = D \oplus Q$

$$Q^{n+1} = ((D \oplus Q) \oplus Q)^n$$

$$Q^{n+1} = D^n$$

x	y	x⊕y	(x⊕y)⊕y
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1



Dal flip-flop T al JK e viceversa

Flip-flop T mediante flip-flop JK

Equazione caratteristica FF JK:

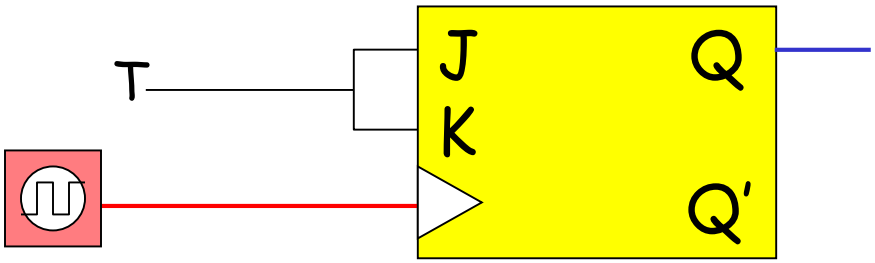
$Q^{n+1} = (J\,Q' + K'Q)^n$

Pongo $J = K = T$

$Q^{n+1} = (T \oplus Q)^n$

FF JK «universale»:

- T=J=K -> FF T
- D=K=J' -> FF D



Flip-flop JK mediante flip-flop T

Equazione caratteristica:

$Q^{n+1} = (T \cdot Q' + T' \cdot Q)^n$

Pongo $T = J \cdot Q' + K \cdot Q$

$Q^{n+1} = ((JQ'+KQ)Q' + (JQ'+KQ)'Q)^n$

$= (JQ'Q'+KQQ') + ((JQ')' \cdot (KQ)')Q)^n$

$= (JQ'+K0 + ((J'+Q) \cdot (K'+Q'))Q)^n$

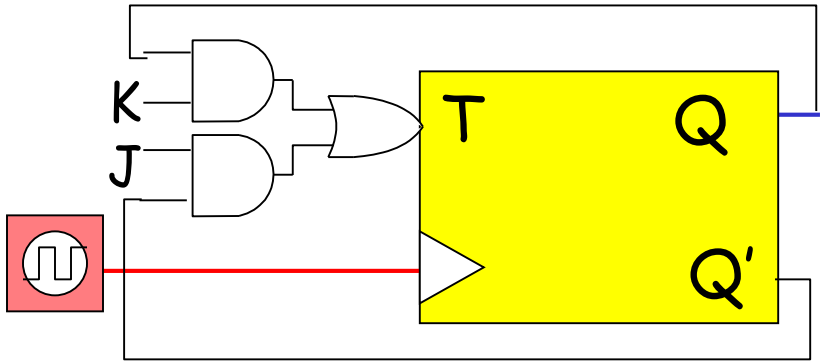
$= (JQ' + (J'K'+QK'+J'Q'+QQ')\,Q)^n$

$= (JQ' + J'K'Q + QK' + J'0 + 0Q)^n$

$= (\,JQ' + K'Q(J'+1)\,)^n$

$= (\,JQ' + K'Q\ 1\,)^n$

$Q^{n+1} = (J \cdot Q' + K' \cdot Q)^n$



6.2
Analisi e sintesi

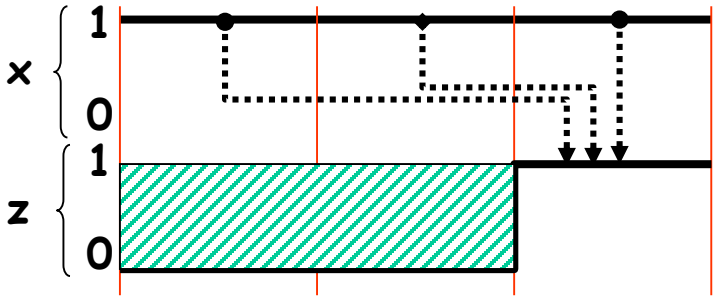
Il procedimento di sintesi

Il procedimento di sintesi di una rete sequenziale sincrona è formato da 5 passi e consente di dedurre lo schema logico dalle specifiche di comportamento:

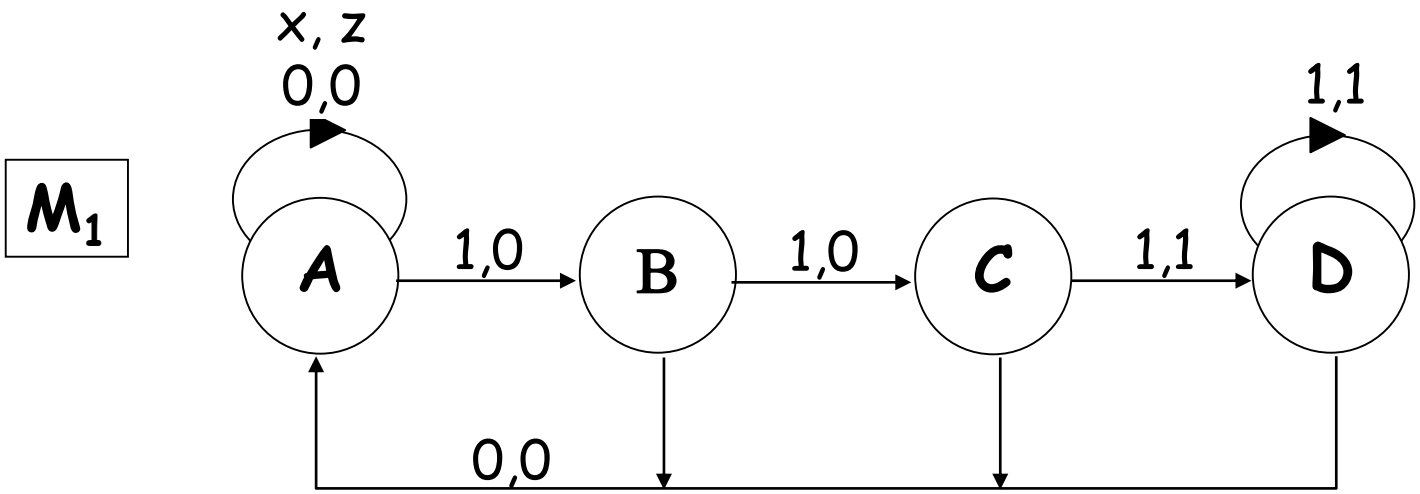
1: comprensione delle **specifiche**
2: individuazione del **grafo degli stati** (tipicamente mediante modello di Mealy)
3: definizione della **tabella di flusso** a partire dal grafo degli stati (individuando l'**automa minimo** mediante eliminazione di stati indistinguibili)
4: codifica degli stati (arbitraria, no corse critiche) e definizione della **tabella delle transizioni,**
5: scelta dei flip-flop (D,JK,T) e sintesi della parte combinatoria

Esempio: il riconoscitore di sequenza - reloaded

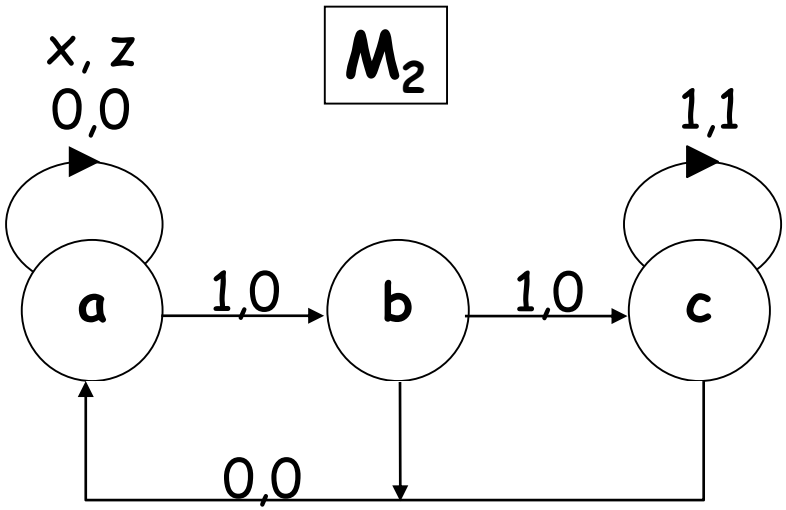
Una rete sequenziale sincrona ha un ingresso x ed una uscita z.
La relazione ingresso/uscita è descritta dalla seguente frase:
"$z^n = 1$ quando $x^n = 1$ e solo se $x^{n-1} = x^{n-2} = 1$ "
(riconoscitore di 3 "uni" consecutivi)

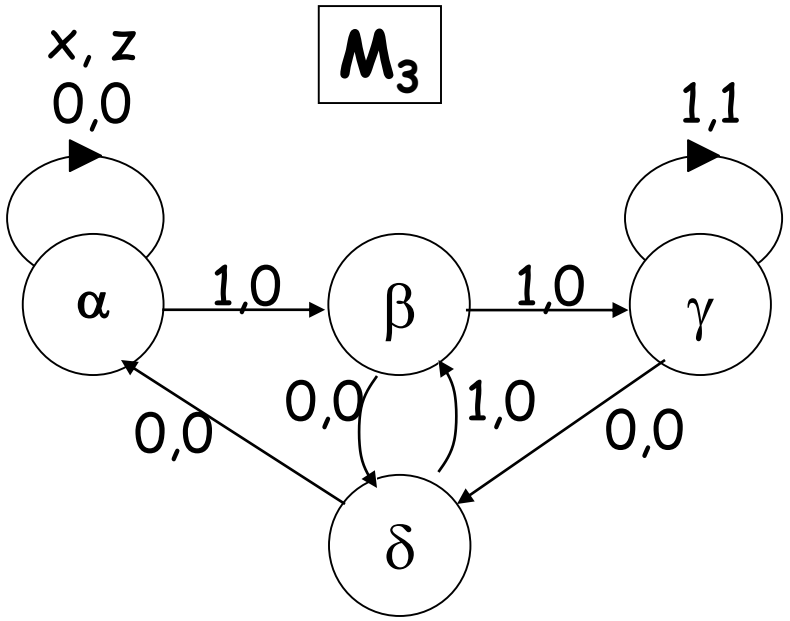


- Una possibile realizzazione del **grafo degli stati** con modello di Mealy (diverso da quello visto in prcedenza) è la seguente (M_1)



② Macchine equivalenti





Macchine equivalenti - Sono dette equivalenti macchine sequenziali che presentano uno stesso comportamento impiegando un diverso numero di stati. (Esempio: $M_1 = M_2 = M_3$)

Macchina minima : macchina che, per un dato comportamento, ha il più piccolo insieme di stati. (Esempio: M_2)

Tabelle di flusso di macchine equivalenti

M1

	0	1
A	A,0	B,0
B	A,0	C,0
C	A,0	D,1
D	A,0	D,1

Le righe C e D sono identiche

M3

	0	1
α	α,0	β,0
β	δ,0	γ,0
γ	δ,0	γ,1
δ	α,0	β,0

Le righe α e δ sono identiche

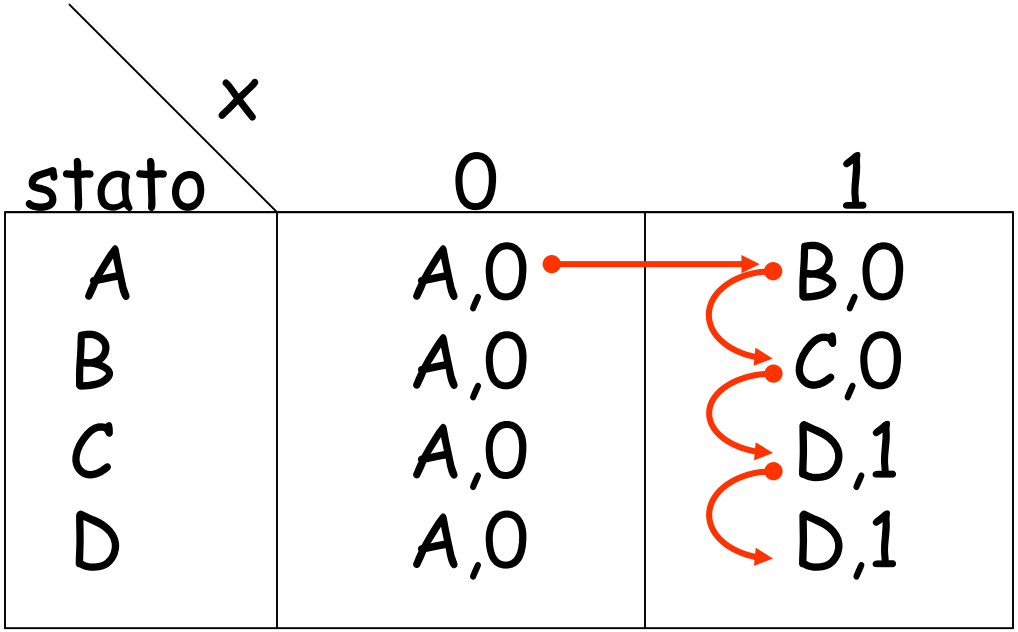
M2

	0	1
a	a,0	b,0
b	a,0	c,0
c	a,0	c,1

M2 si ottiene da M1 ponendo a = A, b = B e c = {C,D}
M2 si ottiene da M3 ponendo a = {α,δ}, b = β e c = γ

- Se l'automa minimo non è stato individuato in fase di determinazione del grafo degli stati, può essere fatto in fase di determinazione della tabella di flusso
- Nella tabella di M2 non vi sono stati equivalenti: essa descrive dunque **l'automa minimo**
- In questo caso però M2 non produce vantaggio a livello di codifica degli stati, essendo in tutti e 3 i casi necessari 2 bit (dunque 2 flip-flop)

③ Tabella di flusso di M_1



N.B. In una rete sequenziale sincrona ogni stato resta presente per almeno un periodo di clock, ogni cambiamento di ingresso avviene all'inizio di tali intervalli ed ogni transizione si verifica al termine. La stabilità dello stato presente non è una condizione necessaria dopo una variazione di ingresso. E' proprio la assenza di questo vincolo che consente di specificare comportamenti di tipo 2 o di tipo 3.

M1: codifica degli stati e tabella delle transizioni

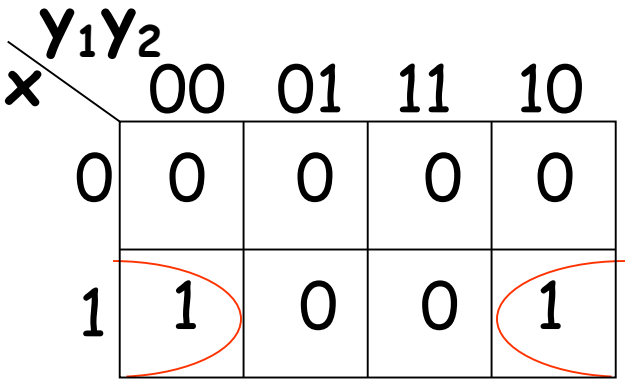
$y_1^n y_2^n$ \ x^n	0	1
A: 00	00,0	10,0
B: 10	00,0	11,0
C: 11	00,0	01,1
D: 01	00,0	01,1

$y_1^{n+1} y_2^{n+1}, z^n$

- **Codifica degli stati**: in una rete sequenziale sincrona la codifica degli stati è **arbitraria** ($2^n \geq M$, naturalmente!). Il campionamento a regime dei segnali di stato elimina infatti a priori il problema di errate interpretazioni causate dal loro iniziale disallineamento (assenza di ***corse critiche***)
- Nel caso di M1: codifica dei 4 stati con 2 bit (y1, y2)

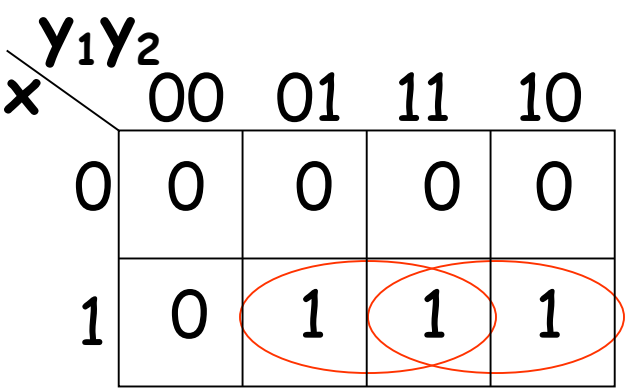
④&⑤ M1: sintesi con flip-flop D

- **Copertura delle funzioni di eccitazione dei flip-flop**: come ultimo passo del procedimento di sintesi occorre determinare i bit di stato futuro e le uscite a partire dai bit di stato presente e gli ingressi (sono le funzioni F e G della rete combinatoria, dette *funzioni di eccitazione dei flip-flop*)
- il campionamento a regime dei segnali di stato elimina a priori il pericolo di **alee statiche e dinamiche**: è possibile utilizzare le espressioni **minime** (al contrario delle reti asincrone)
  - Ipotesi: si cercano reti minime di tipo **SP**



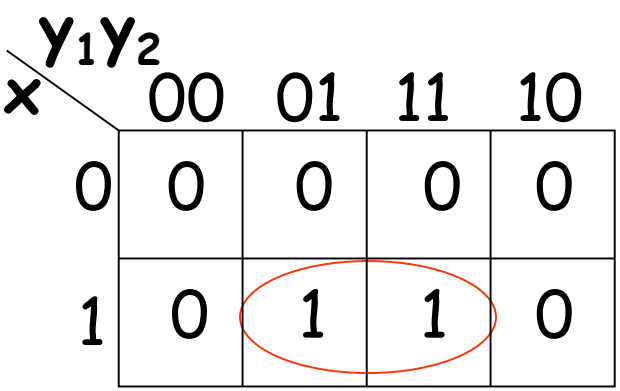
y_1^{n+1}

$D_1 = y_1^{n+1} = x.y_2'$



y_2^{n+1}

$D_2 = y_2^{n+1} = x.y_2 + x.y_1$



z^n

$z = x.y_2$

M1: sintesi con flip-flop JK

	Q^{n+1}	Q^n	J^n	K^n
0 ⇒	0	0	0	-
1 ⇒	1	1	-	0
1 ⇒	1	0	1	-
0 ⇒	0	1	-	1

Eq. caratt.: $Q^{n+1} = (J.Q' + K'Q)^n$

Procedimento di sintesi con FF JK:

- «ingrossare» i valori di y^{n+1} che differiscono da y^n
- generare J e K utilizzando la tabella a lato

Funzioni di eccitazione J,K

x \ y_1y_2	00	01	11	10
0	0	0	0	0
1	1	0	0	1

y_1^{n+1}

x \ y_1y_2	00	01	11	10
0	0	0	-	-
1	1	0	-	-

$J_1 = x.y_2'$

x \ y_1y_2	00	01	11	10
0	-	-	1	1
1	-	-	1	0

$K_1 = x' + y_2$

x \ y_1y_2	00	01	11	10
0	0	0	0	0
1	0	1	1	1

y_2^{n+1}

x \ y_1y_2	00	01	11	10
0	0	-	-	0
1	0	-	-	1

$J_2 = x.y_1$

x \ y_1y_2	00	01	11	10
0	-	1	1	-
1	-	0	0	-

$K_2 = x'$

M1: sintesi con flip-flop T

	Q^{n+1}	Q^n	T^n
0 ⇒	0	0	0
1 ⇒	1	1	0
1 ⇒	1	0	1
0 ⇒	0	1	1

Eq. caratt.: $Q^{n+1} = (T.Q' + T'Q)^n$

Procedimento di sintesi con FF T:

- «ingrossare» i valori di y^{n+1} che differiscono da y^n
- generare T utilizzando la tabella a lato (T=1 per i valori «ingrossati», T=0 altrove)

x \ y_1y_2	00	01	11	10
0	0	0	0	0
1	1	0	0	1

y_1^{n+1}

Funzioni di eccitazione T

x \ y_1y_2	00	01	11	10
0	0	0	1	1
1	1	0	1	0

$T_1 = x.\ y_2'.\ y_1' + y_1.y_2 + x'.y_1$

x \ y_1y_2	00	01	11	10
0	0	0	0	0
1	0	1	1	1

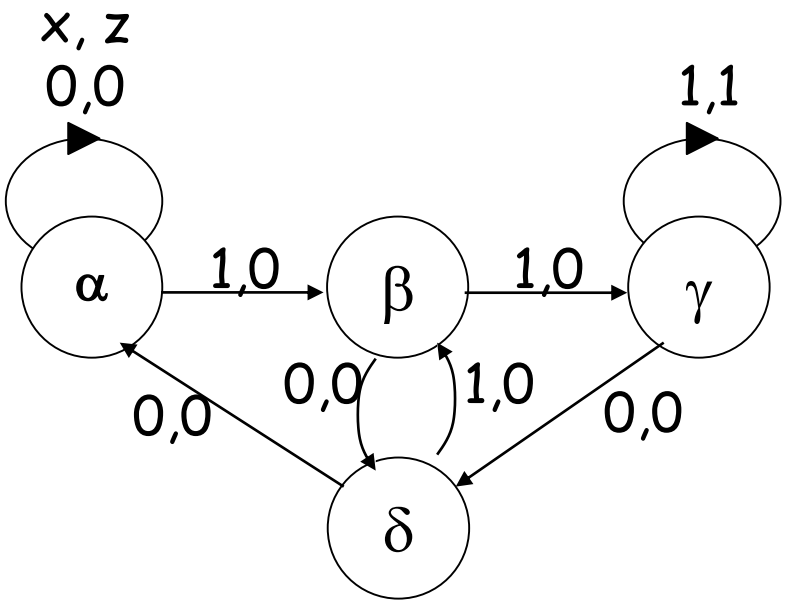
y_2^{n+1}

x \ y_1y_2	00	01	11	10
0	0	1	1	0
1	0	0	0	1

$T_2 = x.\ y_2'.\ y_1 + x'.y_2$

③&④&⑤ Grafo e tabella di flusso di M_3

- Analogamente al caso di M1, procediamo nella sintesi di M3 determinandone la tabella di flusso a partire dal suo grafo degli stati



stato \ x	0	1
α	α,0	β,0
β	δ,0	γ,0
γ	δ,0	γ,1
δ	α,0	β,0

Codifica e tabella delle transizioni di M_3

- Dal grafo degli stati si determina la tabella delle transizioni tramite codifica degli stati (si utilizza una particolare codifica che risulterà utile come mostrato nella prossima slide)

stato \ x	0	1
α	α,0	β,0
β	δ,0	γ,0
γ	δ,0	γ,1
δ	α,0	β,0

	$Q_1=0$	$Q_1=1$
$Q_2=0$	α	β
$Q_2=1$	δ	γ

$Q_1^nQ_2^n$ \ x^n	0	1
α:00	00,0	10,0
β:10	01,0	11,0
γ:11	01,0	11,1
δ:01	00,0	10,0

$Q_1^{n+1}\ Q_2^{n+1}$, z^n

M3: sintesi con flip-flop D

- Infine si determinano le coperture minime (anche in questo caso si procede mediante espressioni SP)
- La copertura utilizzata permette di avere funzioni di eccitazione che **non richiedono alcun gate**
- La rete ottenuta è un **registro a scorrimento a 2 bit** (i bit di stato memorizzano i precedenti valori del bit d'ingresso, si vedrà in seguito)

x \ Q_1Q_2	00	01	11	10
0	0	0	0	0
1	1	1	1	1

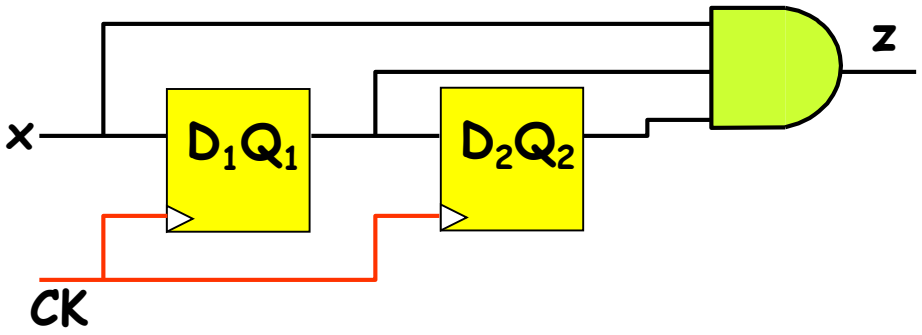
$D_1^n = Q_1^{n+1} = x^n$

x \ Q_1Q_2	00	01	11	10
0	0	0	1	1
1	0	0	1	1

$D_2^n = Q_2^{n+1} = Q_1^n$

x \ Q_1Q_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

$z^n = x^n . Q_1^n . Q_2^n$



Verifica del comportamento:

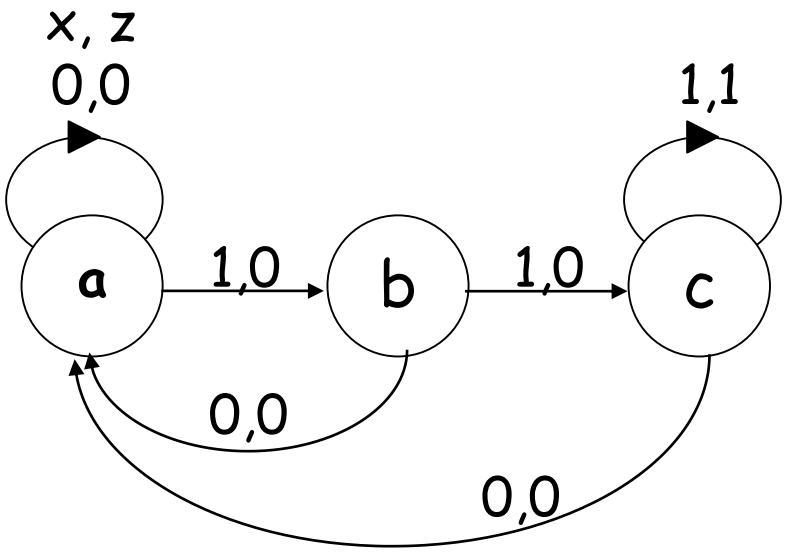
$Q_1^{n+1} = x^n$

$Q_2^{n+1} = Q_1^n = x^{n-1}$

$z^n = (x . Q_1 . Q_2)^n = x^n . x^{n-1} . x^{n-2}$

③&④&⑤ Grafo e tabella di flusso di M_2

- Analogamente ai casi precedenti procediamo nella sintesi di M2 determinandone la tabella di flusso a partire dal suo grafo degli stati



stato \ x	0	1
a	a,0	b,0
b	a,0	c,0
c	a,0	c,1

Codifica e tabella delle transizioni di M2

- Dal grafo degli stati si determina la tabella delle transizioni tramite codifica degli stati
- Stato 01 è **impossibile**: stato futuro e uscita sono quindi **indifferenti**
- **Attenzione**: se all'init del sistema si finisce in uno stato impossibile, si possono ottenere comportamenti non voluti (es. permanenza in stati impossibili per via di condizioni di stabilità imposte dalla copertura)
- per evitare ciò, si possono imporre valori appropriati dello **stato futuro** (al posto delle indifferenze) agli stati presenti impossibili, in modo da avere **instabilità per ogni ingresso** su ciascuno stato impossibile
- **Autoinizializzazione**: proprietà di una rete per cui le configurazioni impossibili vengono abbandonate per qualsiasi configurazione d'ingresso

$Q_1^n Q_2^n$ \ x^n	0	1
a:00	00,0	10,0
b:10	00,0	11,0
c:11	00,0	11,1
01	–,–	–,–

$Q_1^{n+1}\ Q_2^{n+1},\ z^n$

M2: sintesi con flip-flop D

Ipotesi: reti minime di tipo SP

Q_1Q_2 / x	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$D_1^n = Q_1^{n+1} = (x)^n$

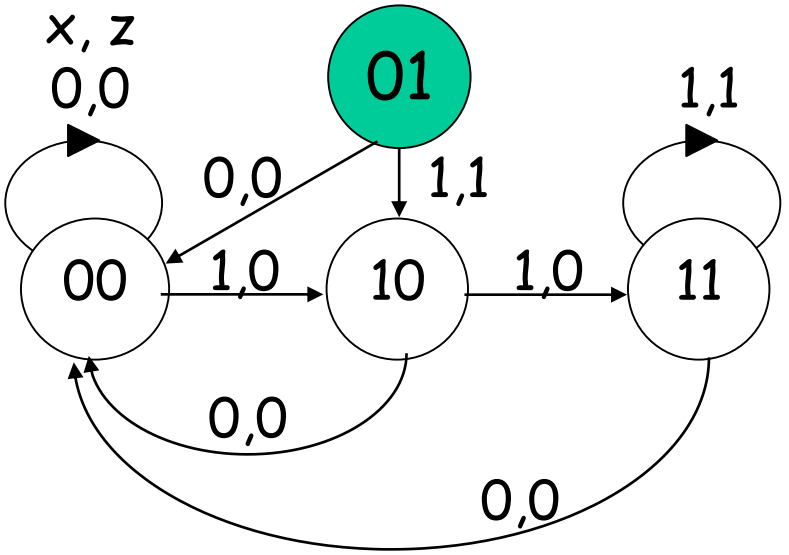
Q_1Q_2 / x	00	01	11	10
0	0	0	0	0
1	0	0	1	1

$D_2^n = Q_2^{n+1} = (x.Q_1)^n$

Q_1Q_2 / x	00	01	11	10
0	0	-	0	0
1	0	-	1	0

$z^n = (x.Q_2)^n$

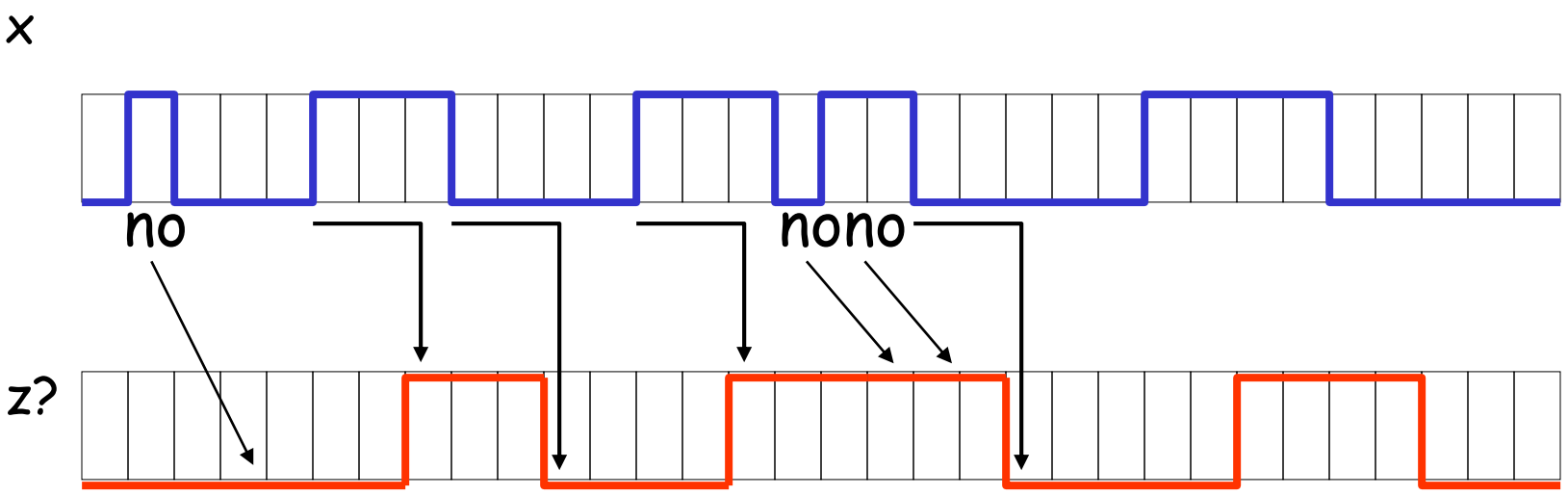
- **Autoinizializzazione**: le configurazioni indifferenti di stato futuro vengono opportunamente sostituite con zeri e uni



Esercitazione N. 19 (1)

Una RSS ha il compito di riprodurre sulla sua uscita z, con un ritardo di due intervalli di clock, il valore presente sul suo ingresso x, a condizione però che tale valore perduri per più di due intervalli.
Se il valore di x è presente solo per uno o per due intervalli, l'uscita z deve ignorarlo e mantenere il valore che aveva prima della variazione di x.

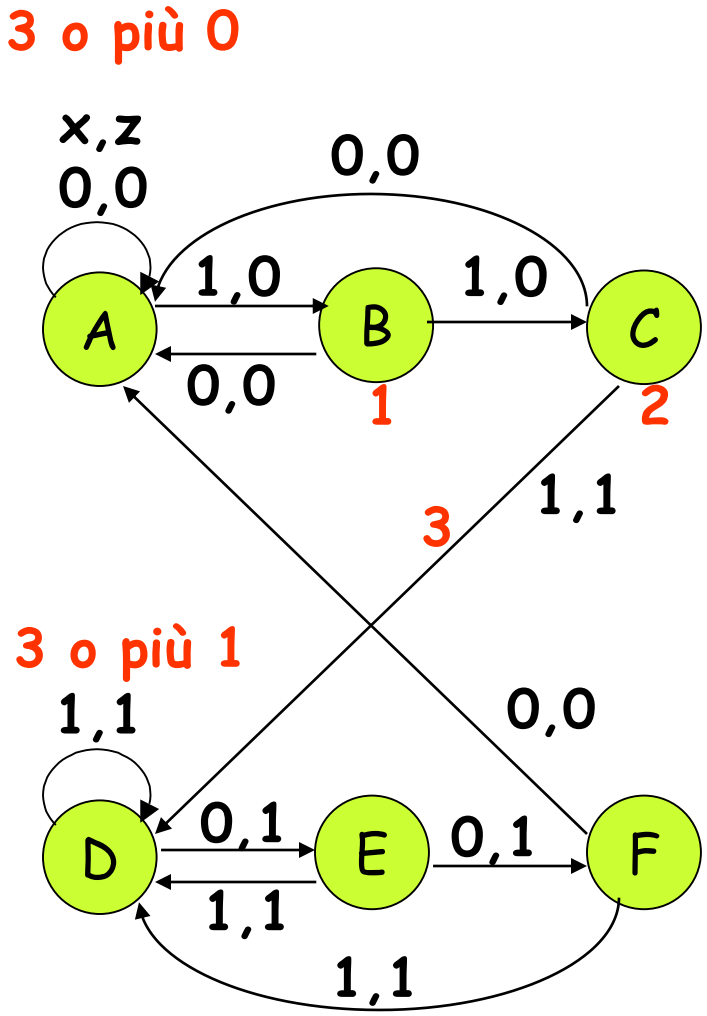
DOMANDA N.1 – indicare la forma d'onda del segnale d'uscita z



Esercitazione N. 19 (2)

DOMANDA N.2- Completare il grafo degli stati

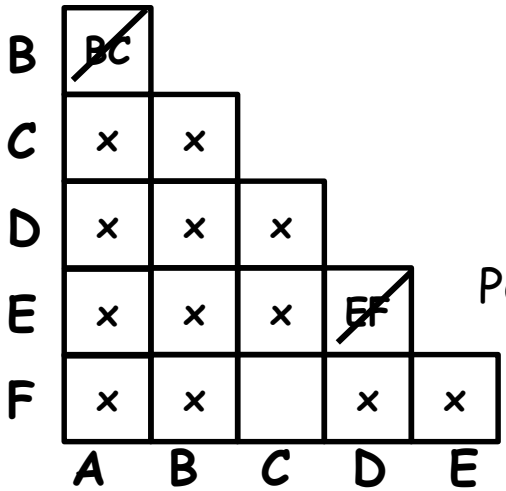
DOMANDA N.3- Tracciare la TdF, codificare gli stati e riempire la TdT



	x = 0	x = 1
A	A,0	B,0
B	A,0	C,0
C	A,0	D,1
D	E,1	D,1
E	F,1	D,1
F	A,0	D,1

Stati equivalenti?

$y_1 y_2 y_3$	x = 0	x = 1
A=0 0 0	000,0	001,0
B=0 0 1	000,0	010,0
C=0 1 0	000,0	011,1
D=0 1 1	100,1	011,1
E=1 0 0	101,1	011,1
F=1 0 1	000,0	011,1
1 1 0	---,-	---,-
1 1 1	---,-	---,-



CMC: {A},{B},{C,F},{D},{E}

Possibile riduzione a 5 stati

Esercitazione N. 19 (3)

DOMANDA N.4 – Relativamente alla TdF a 6 stati, sintetizzare la variabile di stato y_1 con FF-JK

		J	K
0	hr	0	-
1	hs	-	0
0	tr	-	1
1	ts	1	-

$y_1 y_2$ \ $y_3 x$	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	-	-	-	-
10	-	-	-	-

$J_1 = y_2 y_3 x'$

$y_1 y_2$ \ $y_3 x$	00	01	11	10
00	-	-	-	-
01	-	-	-	-
11	-	-	-	-
10	0	1	1	1

$K_1 = x + y_3$

$y_1 y_2 y_3$	x = 0	x = 1
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	0
1 0 0	1	0
1 0 1	0	0
1 1 0	-	-
1 1 1	-	-

y_1^{n+1}

Il procedimento di analisi

Il procedimento di analisi di una rete sequenziale sincrona è formato da 5 passi e consente di dedurne il comportamento dallo schema logico:

1: individuazione delle espressioni dei segnali d'ingresso a ciascun flip-flop

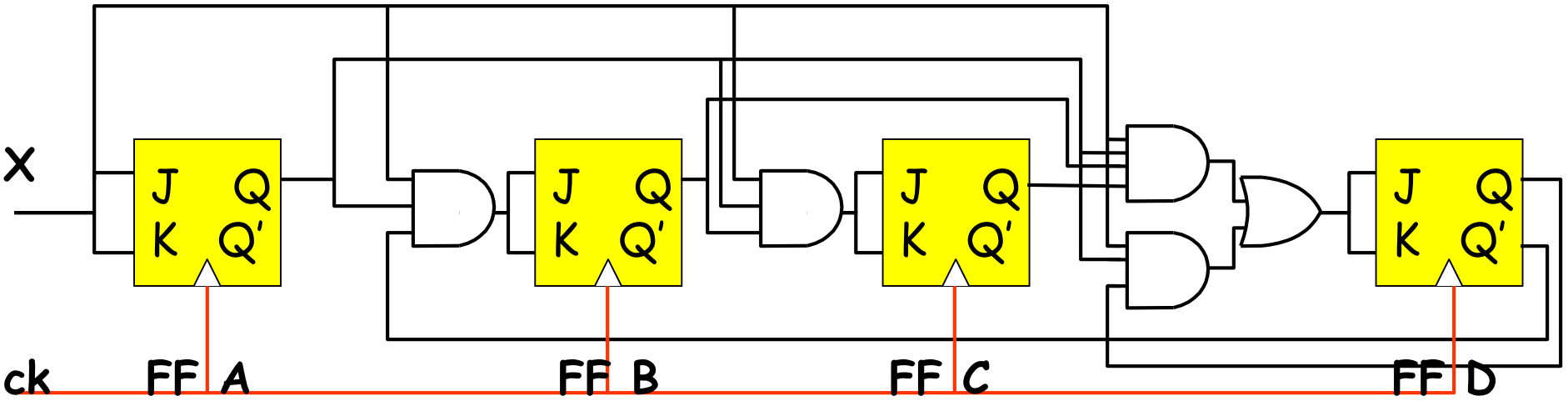
2: individuazione delle espressioni di stato futuro e d'uscita

3: individuazione della tabella delle transizioni

4: deduzione e studio della tabella di flusso

5: tracciamento e studio del grafo degli stati

Esempio: il contatore BCD



- 4 FF JK-> 4 variabili di stato, 16 stati
- Si denominano i flip-flop e si scrivono le espressioni dei loro comandi:

$$J_A^n = K_A^n = X^n$$

$$J_B^n = K_B^n = (X.\, Q_A.\, Q_D')^n$$

$$J_C^n = K_C^n = (X.\, Q_A.\, Q_B)^n$$

$$J_D^n = K_D^n = (X.\, Q_A.\, Q_B.\, Q_C + X.\, Q_A.\, Q_D)^n$$

Espressioni di stato

Tramite l'equazione caratteristica si passa dalle espressioni delle funzioni di eccitazione a quelle delle variabili di stato futuro.

$Q^{n+1} = (J.Q'+K'.Q)^n$
Nel caso J=K=T si ha
$Q^{n+1} = (T \oplus Q)^n$

$$Q_A^{n+1} = (X \oplus Q_A)^n$$

$$Q_B^{n+1} = ((X.\ Q_A.\ Q_D') \oplus Q_B)^n$$

$$Q_C^{n+1} = ((X.\ Q_A.\ Q_B) \oplus Q_C))^n$$

$$Q_D^{n+1} = ((X.\ Q_A.\ Q_B.\ Q_C + X.\ Q_A.\ Q_D) \oplus Q_D)^n$$

Tabella delle transizioni

$(Q_D Q_C Q_B Q_A)^n$	$X^n = 0$	$X^n = 1$
0 0 0 0	0 0 0 0	0 0 0 1
0 0 0 1	0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 0	0 0 1 1
0 0 1 1	0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 0	0 1 0 1
0 1 0 1	0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 0	0 1 1 1
0 1 1 1	0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 0	1 0 0 1
1 0 0 1	1 0 0 1	0 0 0 0
1 0 1 0	1 0 1 0	1 0 1 1
1 0 1 1	1 0 1 1	0 1 1 0
1 1 0 0	1 1 0 0	1 1 0 1
1 1 0 1	1 1 0 1	0 1 0 0
1 1 1 0	1 1 1 0	1 1 1 1
1 1 1 1	1 1 1 1	0 0 1 0

$(Q_D Q_C Q_B Q_A)^{n+1}$

Per X = 1 e S = 0000,..., 1001 si ha: $(S)_2^{n+1} = (S+1)_2^n \bmod 10$

Per X = 0 si ha $(S)_2^{n+1} = (S)_2^n$

Un ingresso del tipo di X è denominato **comando di ENABLE**

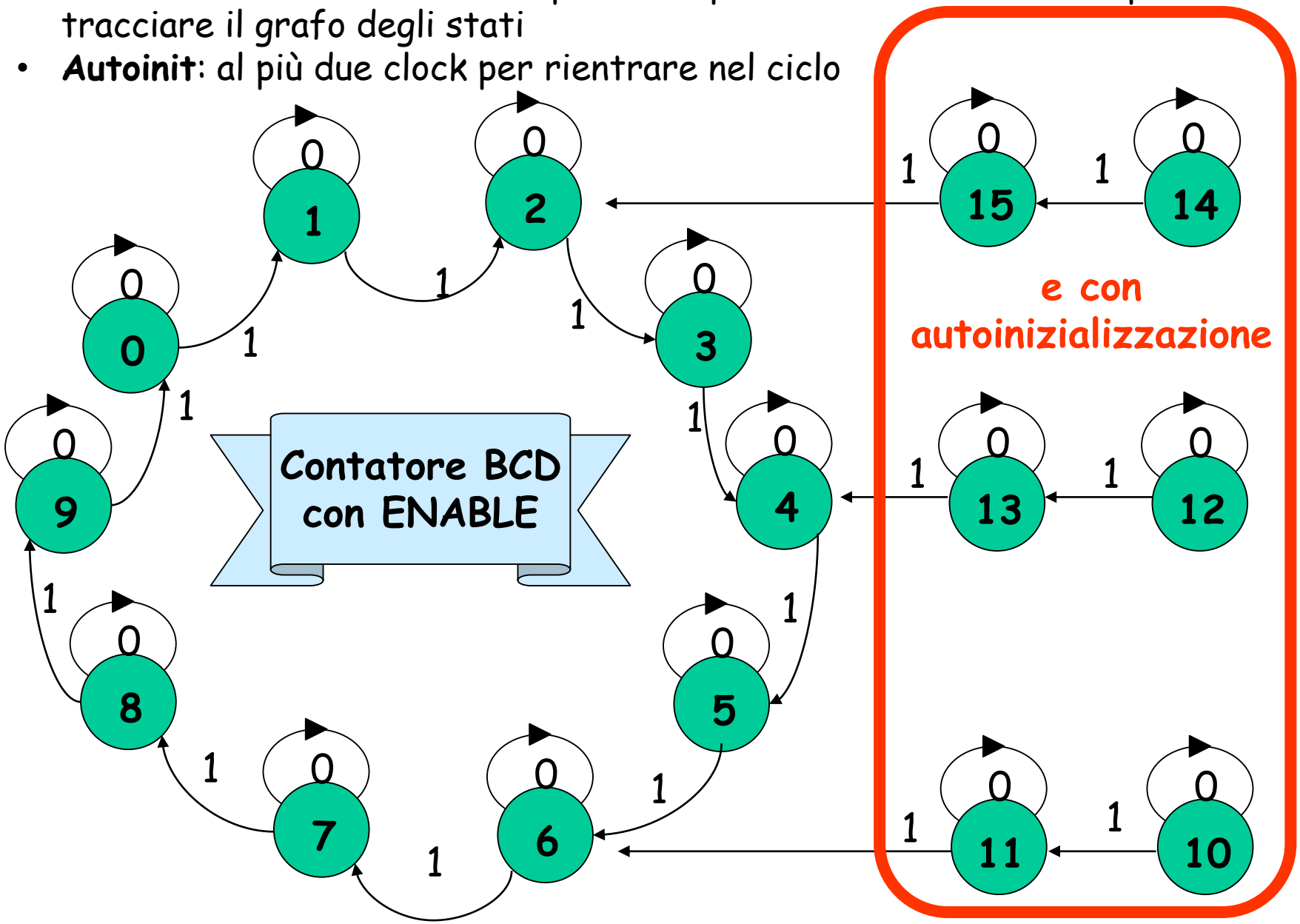
S_p	X=0	X=1
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	8
8	8	9
9	9	0
10	10	11
11	11	6
12	12	13
13	13	4
14	14	15
15	15	2

S_f

Interpretando $Q_D..Q_A$ come numeri binari a 4 bit, si deduce il comportamento di un contatore BCD in base 10 e comando di ENABLE

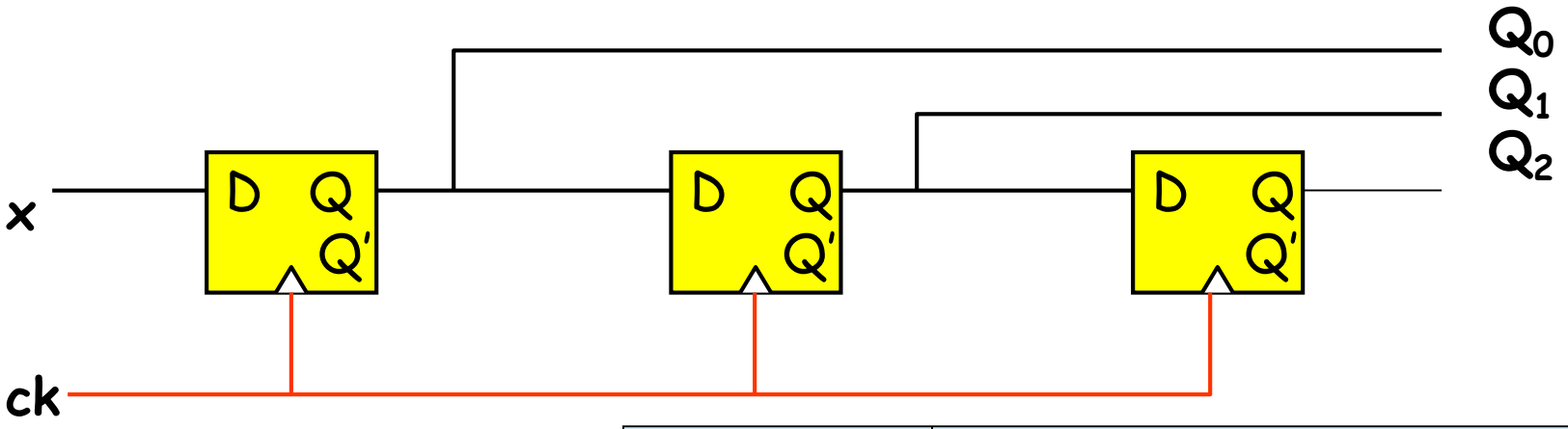
Grafo degli stati

- Dalla tabella delle transizioni passando per la tabella di flusso si può tracciare il grafo degli stati
- **Autoinit**: al più due clock per rientrare nel ciclo



Esempio: registro a scorrimento a 3 bit

- Analisi di un registro a scorrimento a 3 bit



$Q_0^{n+1} = D_0^n = x^n$

$Q_1^{n+1} = D_1^n = Q_0^n = x^{n-1}$

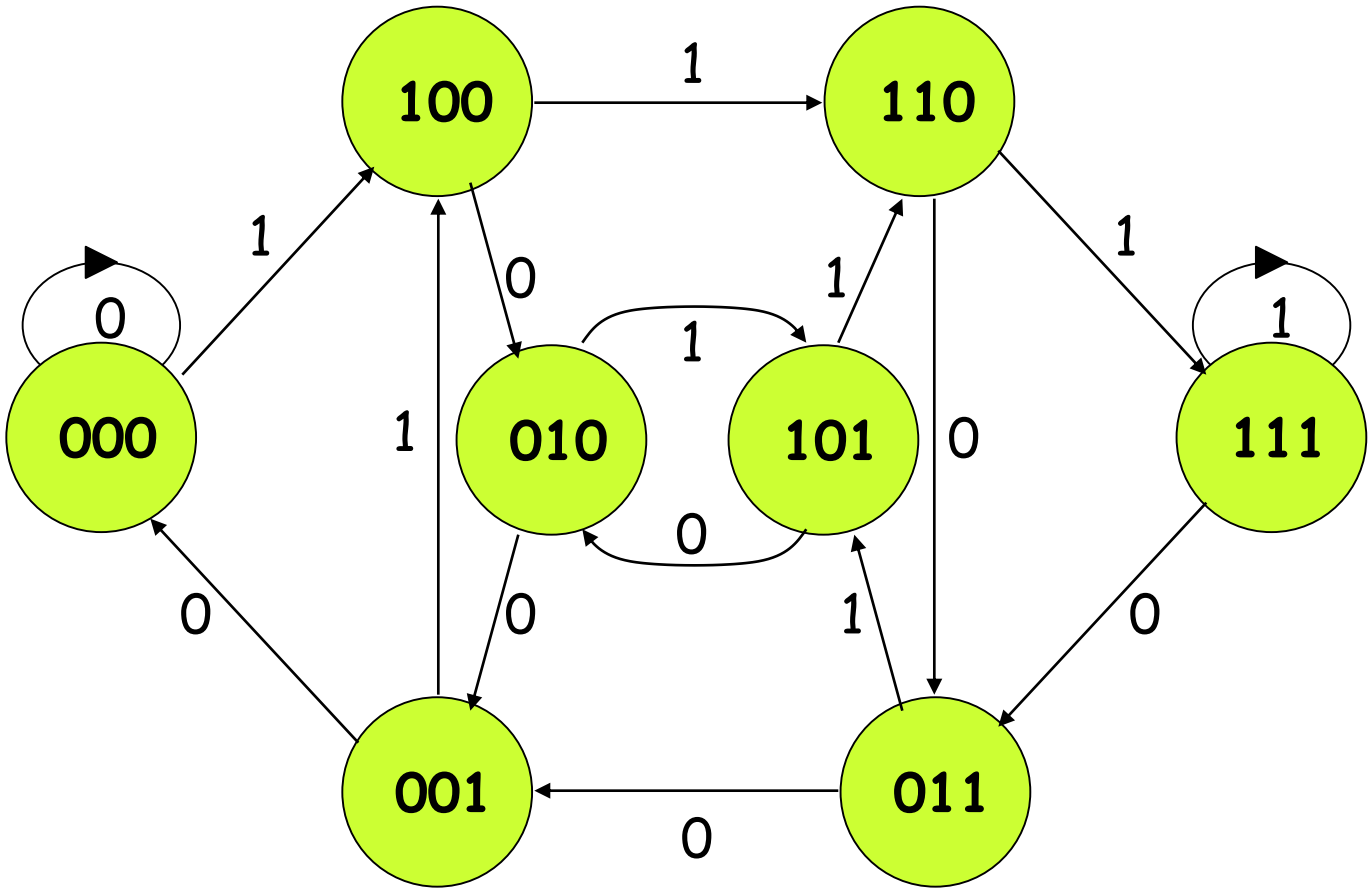
$Q_2^{n+1} = D_2^n = Q_1^n = x^{n-2}$

$(Q_0\ Q_1\ Q_2)^n$	$x^n = 0$	$x^n = 1$
0 0 0	0 0 0	1 0 0
0 0 1	0 0 0	1 0 0
0 1 0	0 0 1	1 0 1
0 1 1	0 0 1	1 0 1
1 0 0	0 1 0	1 1 0
1 0 1	0 1 0	1 1 0
1 1 0	0 1 1	1 1 1
1 1 1	0 1 1	1 1 1

$(Q_0\ Q_1\ Q_2)^{n+1}$

Q_0, Q_1, Q_2

Grafo degli stati

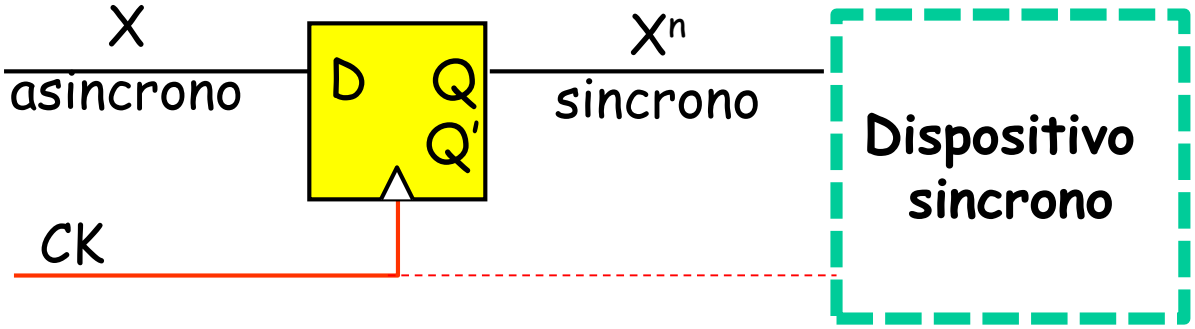


- Ogni stato è codificato dalla configurazione corrispondente agli ultimi tre valori dell'ingresso
- Ad ogni colpo di clock il bit in ingresso entra in Q_0, il bit in Q_0 "scorre" o "trasla" su Q_1, il bit in Q_1 su Q_2
- Due stabilità (tutti «uni», tutti «zeri»)

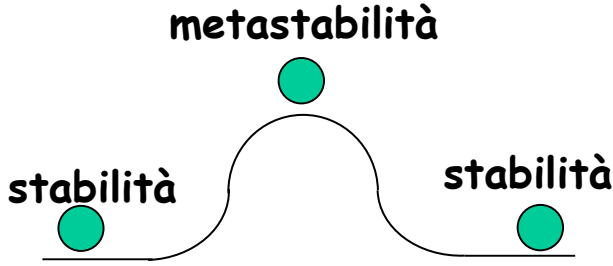
Sincronizzazione di un segnale asincrono

- X è un segnale asincrono rispetto al clock: come sincronizzarlo?
- Problema ricorrente in cui occorre interfacciare un segnale di input asincrono ad un dispositivo sincrono

- Soluzione : individuare la massima frequenza di variazione di X e fissare la frequenza di campionamento del clock ad un valore più alto
- Utilizzando un FF D si ottiene un segnale ritardato ma con variazioni **sincrone** rispetto a CK



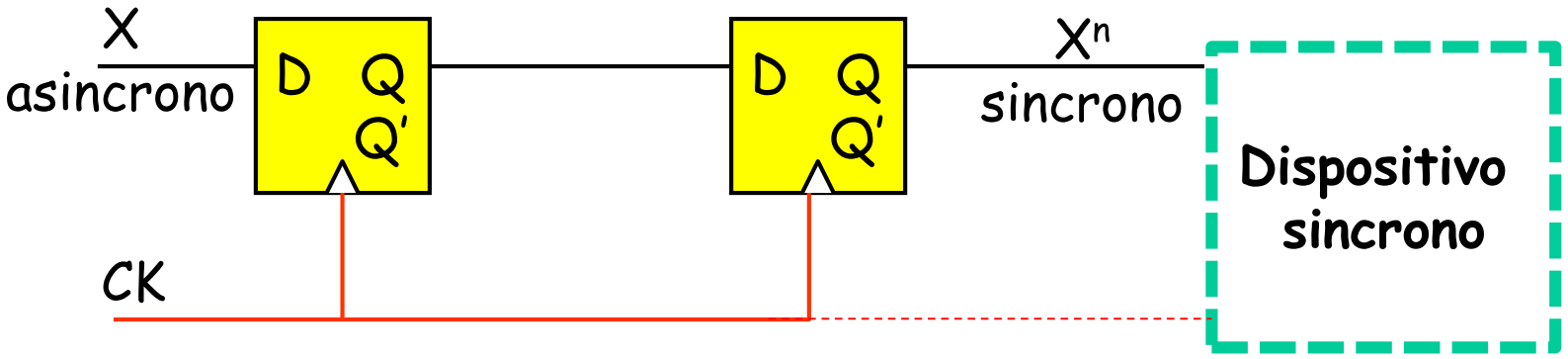
- Se la variazione di X capita nell'intervallo $\tau_{su}+\tau_h$ -> **rischio di metastabilità**:
  - Dallo stato di metastabilità, lo stato interno del FF verrà prima o poi riportato su uno dei due stati stabili, ma in un tempo non prevedibile (potenzialmente anche molto lungo)



Sincronizzazione di un segnale asincrono (2)

- La metastabilità del FF può causare problemi ai dispositivi sincroni a valle che utilizzano tale segnale

- Soluzione: **ritardare il segnale** di un intervallo di tempo sufficiente per risolvere la condizione di metastabilità nella maggior parte dei casi
- In molti casi pratici, è sufficiente ritardare il segnale di un unico ciclo di clock



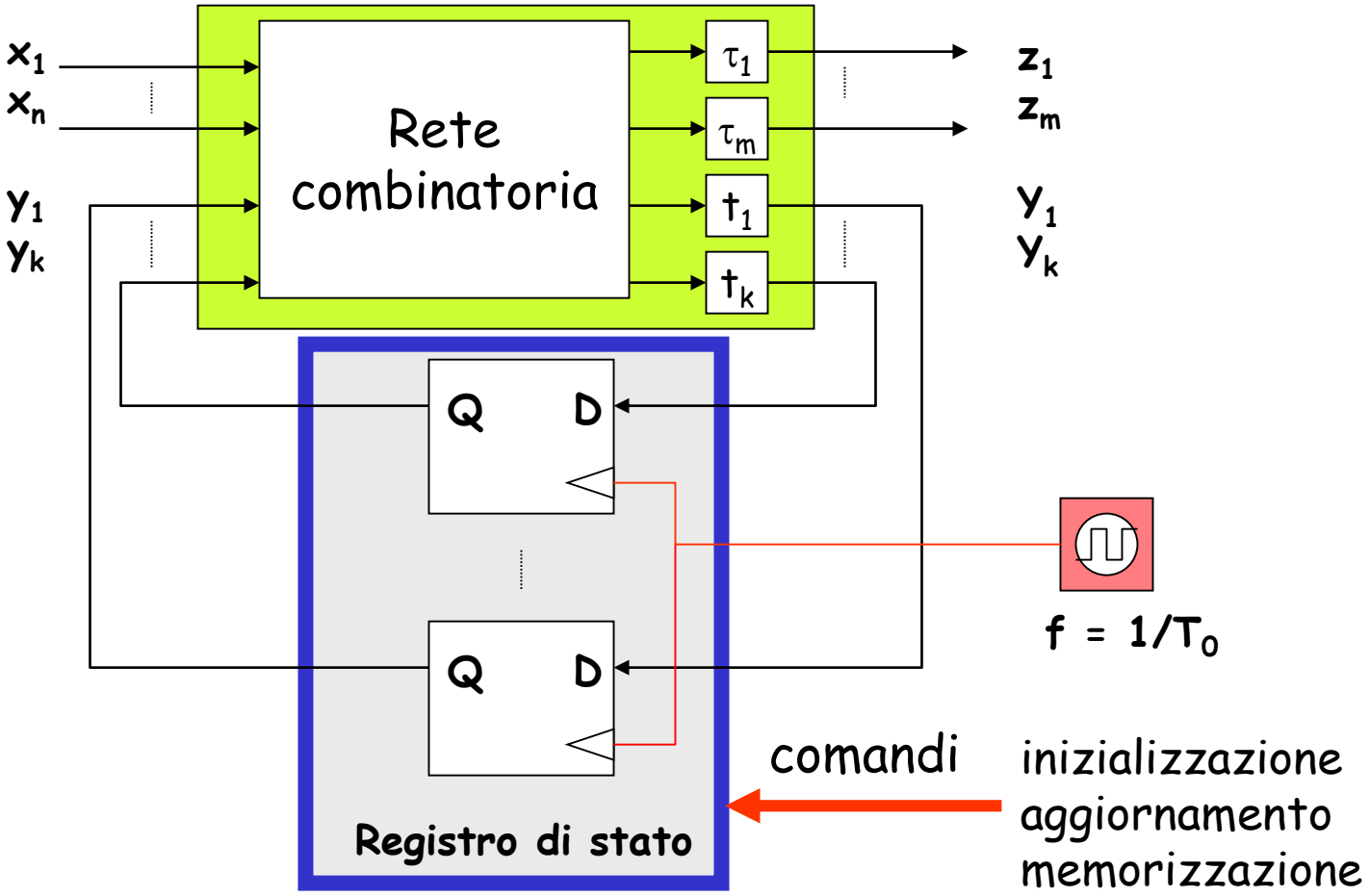
- Tale approccio è risolutivo in tutti quei casi in cui il periodo di clock è maggiore del tempo di risoluzione della metastabilità + il tempo di set-up del FF
- Alternativa: utilizzare più FF in cascata (aumentando così il ritardo)

6.3
Registri e contatori

Circuiti sequenziali ad elevata complessità

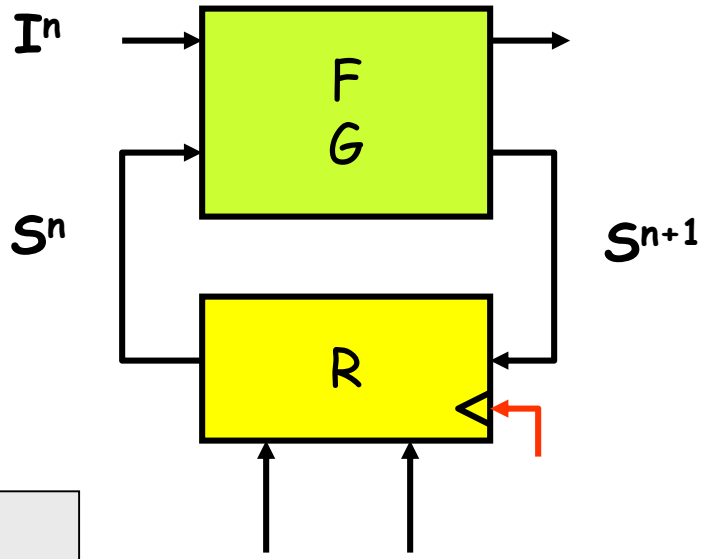
- Abbiamo già visto nel caso delle reti combinatorie la tendenza ad avere componenti a sempre maggiore **complessità** per fare fronte alla necessità di un **maggiore livello di integrazione**
- Similmente per le reti sequenziali sincrone si sono nel tempo resi disponibili moduli sequenziali programmabili, ovvero circuiti ad elevata complessità, a partire dai **registri di stato**, **contatori** e **registri a scorrimento**, da utilizzare all'interno di interfacce, memorie, processori, ...
- L'utilizzo di segnali di comando permette flessibilità di impiego da parte dell'utilizzatore che può selezionare uno tra diversi comportamenti

Il registro di stato (o accumulatore)



- modulo che contiene un certo numero di flip-flop azionati dallo stesso clock e singolarmente dotati di un ingresso e di una uscita.
- tramite i comandi di WE e RES permette di realizzare le tre modalità di inizializzazione, aggiornamento e memorizzazione

I comandi WE e RES



RES prioritario

segnali sincroni

WE^n RE ^{n}	*fase*	*comportamento*
- 1	inizializzazione	$S^{n+1} = 0$
1 0	aggiornamento	$S^{n+1} = G(S^n, I^n)$
0 0	memorizzazione	$S^{n+1} = S^n$

- I segnali WE e RES sono sincroni: ad ogni intervallo di clock lo stato futuro S^{n+1} dipende dal suo ingresso ma anche da WE e RES

I comandi WE e RES (Registro con FF D)

RES^n, Q^n \ WE^n, I^n	00	01	11	10
00	0	0	1	0
01	1	1	1	0
11	0	0	0	0
10	0	0	0	0

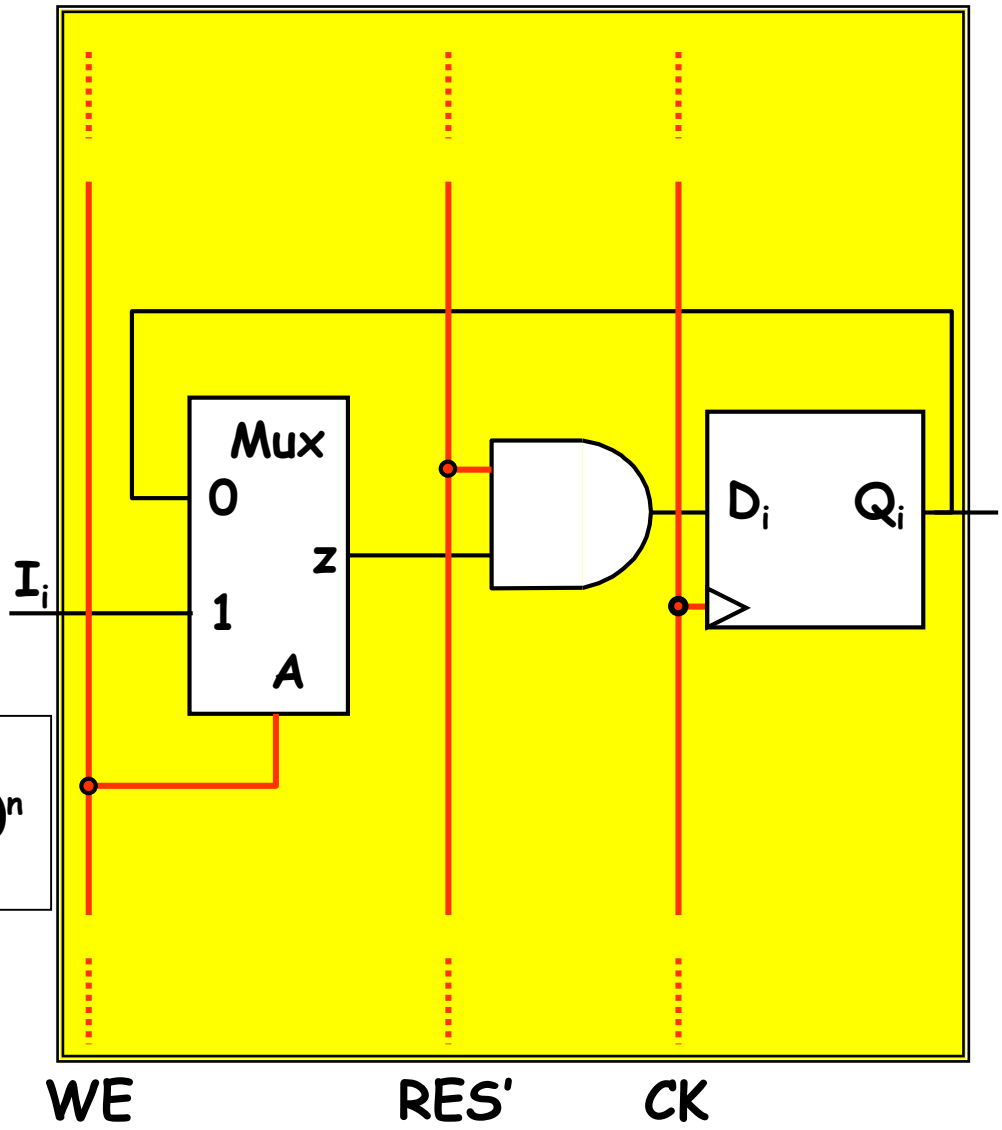
Q^{n+1}

- Con RES=1 -> $Q^{n+1}=0$
- Con RES=0, WE=1-> $Q^{n+1}=I^n$
- Altrimenti: $Q^{n+1}=Q^n$

Per i = 0, 1, .., N-1

$$Q_i^{n+1} = (RES' \cdot I_i \cdot WE + RES' \cdot Q_i \cdot WE'))^n$$
$$= (RES' \cdot (I_i \cdot WE + Q_i \cdot WE'))^n$$

- L'i-esima cella di un registro (con FF D) contiene un selettore a 2 vie (azionato dal WE) seguito da un AND (azionato dal RES)



I comandi WE e RES (Registro con FF JK)

RES^n, Q^n \ WE^n, I^n	00	01	11	10
00	0	0	**1**	0
01	1	1	1	**0**
11	**0**	**0**	**0**	**0**
10	0	0	0	0

Q^{n+1}

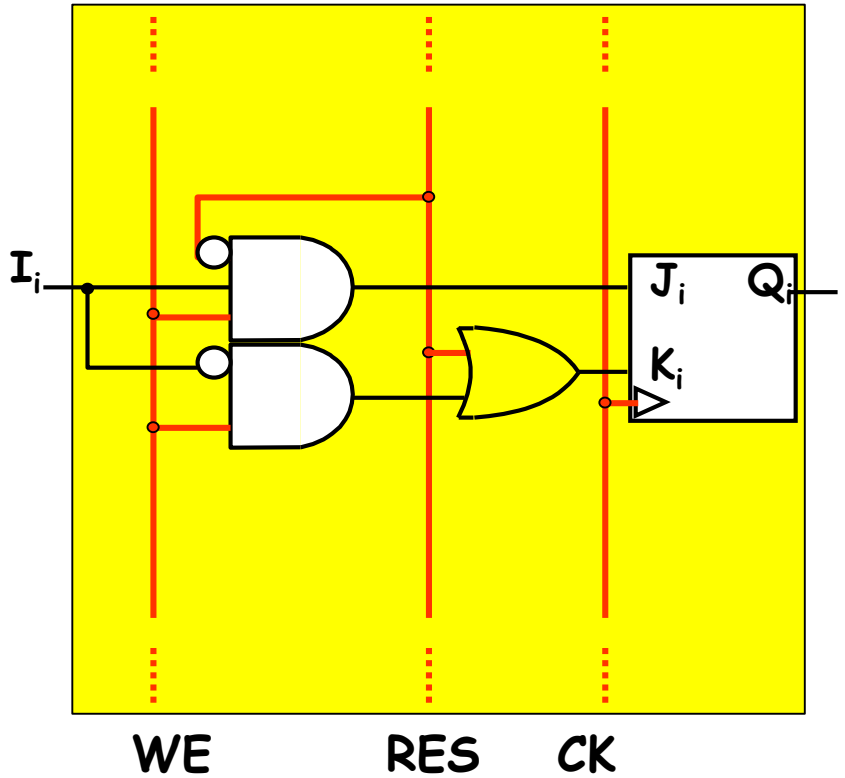
		J	K
0	hr	0	-
1	hs	-	0
0	tr	-	1
1	ts	1	-

RES^n, Q^n \ WE^n, I^n	00	01	11	10
00	0	0	1	0
01	-	-	-	-
11	-	-	-	-
10	0	0	0	0

$J^n = RES' \cdot WE \cdot I_i$

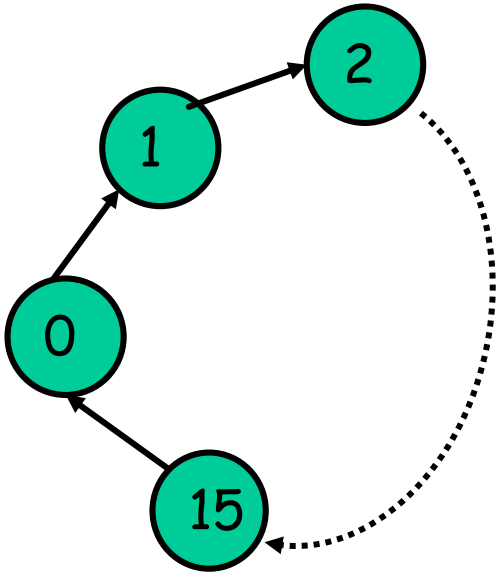
RES^n, Q^n \ WE^n, I^n	00	01	11	10
00	-	-	-	-
01	0	0	0	1
11	1	1	1	1
10	-	-	-	-

$K^n = RES + WE \cdot I_i'$

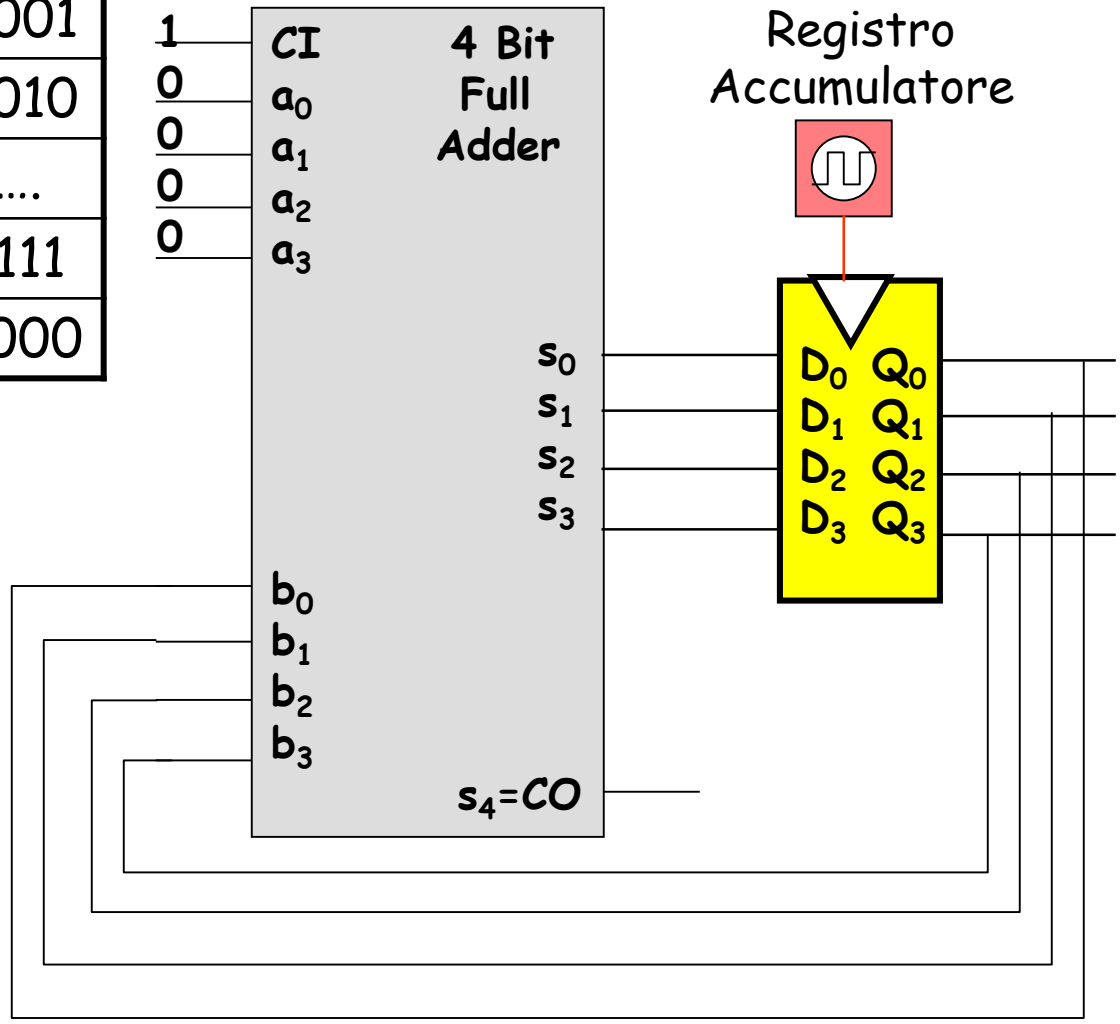


- L'i-esima cella di un registro con FF JK è più semplice di quella di un registro con FF D

Il contatore binario x16



s^n	s^{n+1}
0000	0001
0001	0010
....	....
1110	1111
1111	0000

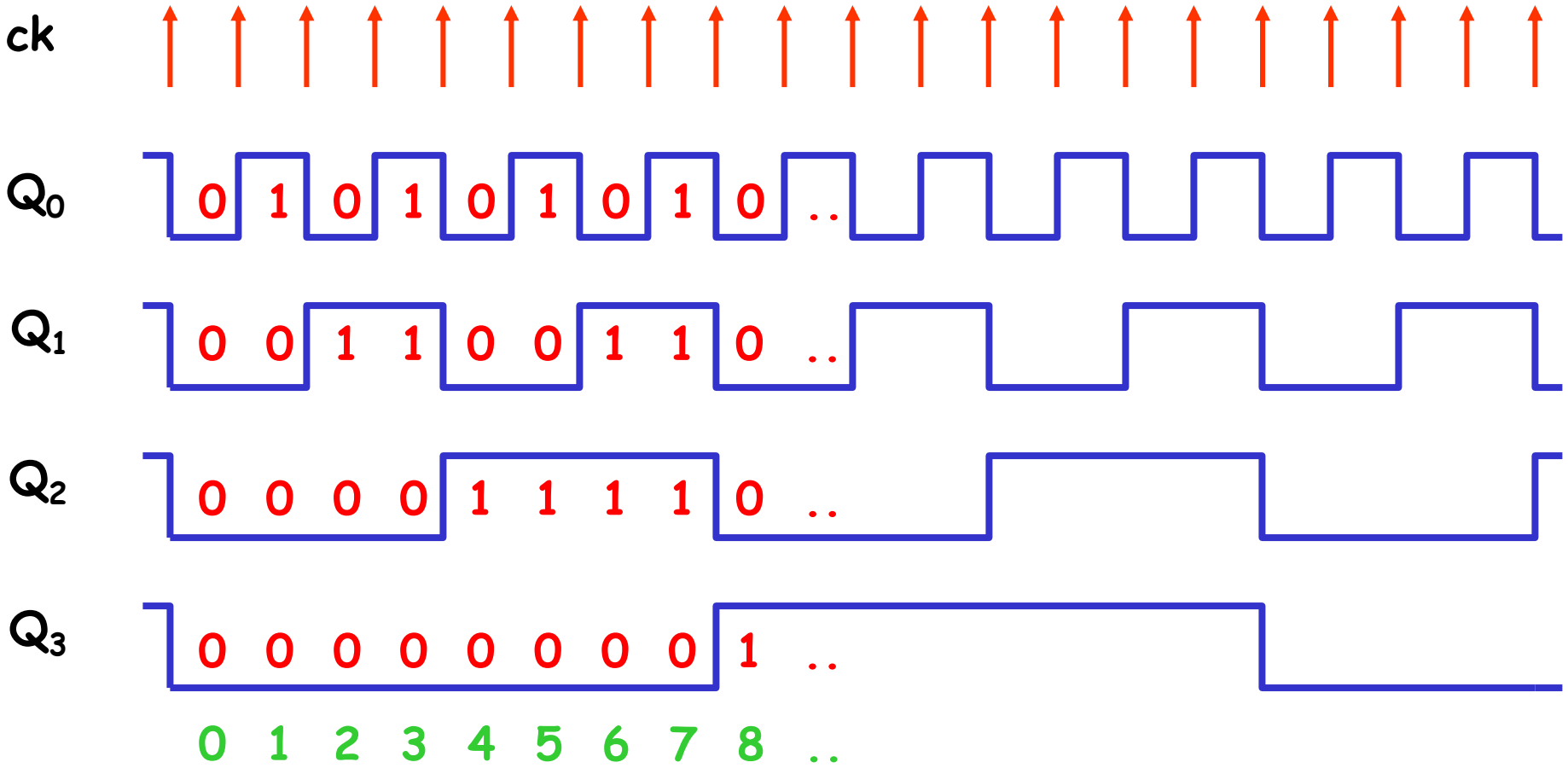


- Il grafo di un contatore è caratterizzato da un ciclo contenente tutti (o quasi) gli stati interni
- Il n° di stati rappresenta la base di conteggio
- Con k FF si realizza un contatore a base 2^k
  - Q_0: bit di minor peso
  - Q_{k-1}: bit di maggior peso

$$s^{n+1} = (s+1)^n \bmod 16$$

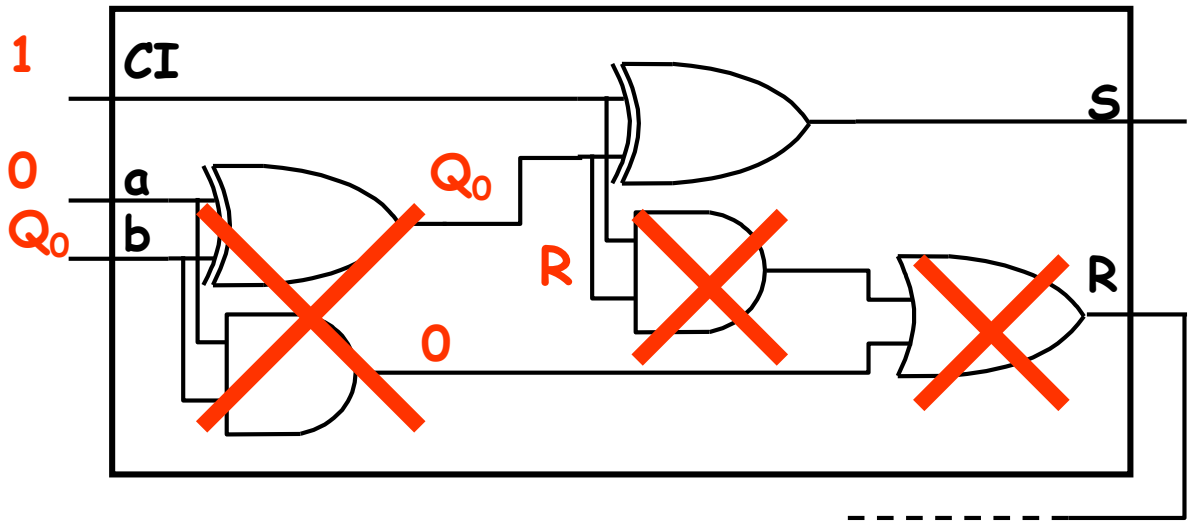
Forme d'onda

- Proprietà del contatore binario
  - tutte le uscite dei flip-flop sono onde quadre
  - l'uscita Q_i del flip-flop che memorizza il bit di peso 2^i ha periodo doppio di quella presente sull'uscita Q_{i-1} del flip-flop che memorizza il bit di peso 2^{i-1}
  - L'uscita del primo flip-flop Q_0 divide per due la frequenza del clock



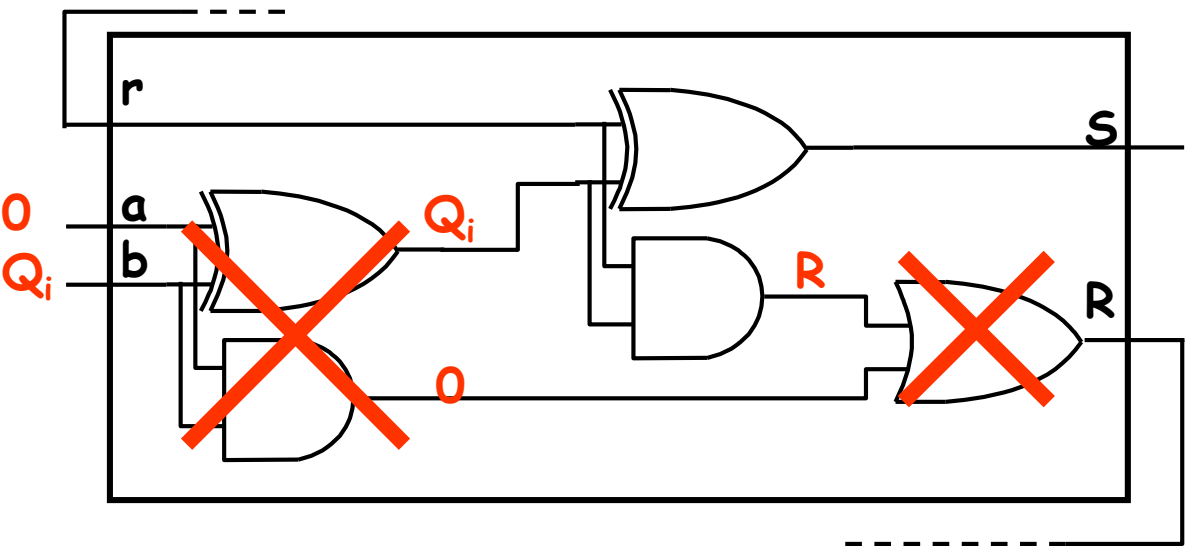
Una rete più semplice per l'incremento

- Uno degli operandi è sempre = 0: il sommatore è un «cannone per mosche»?



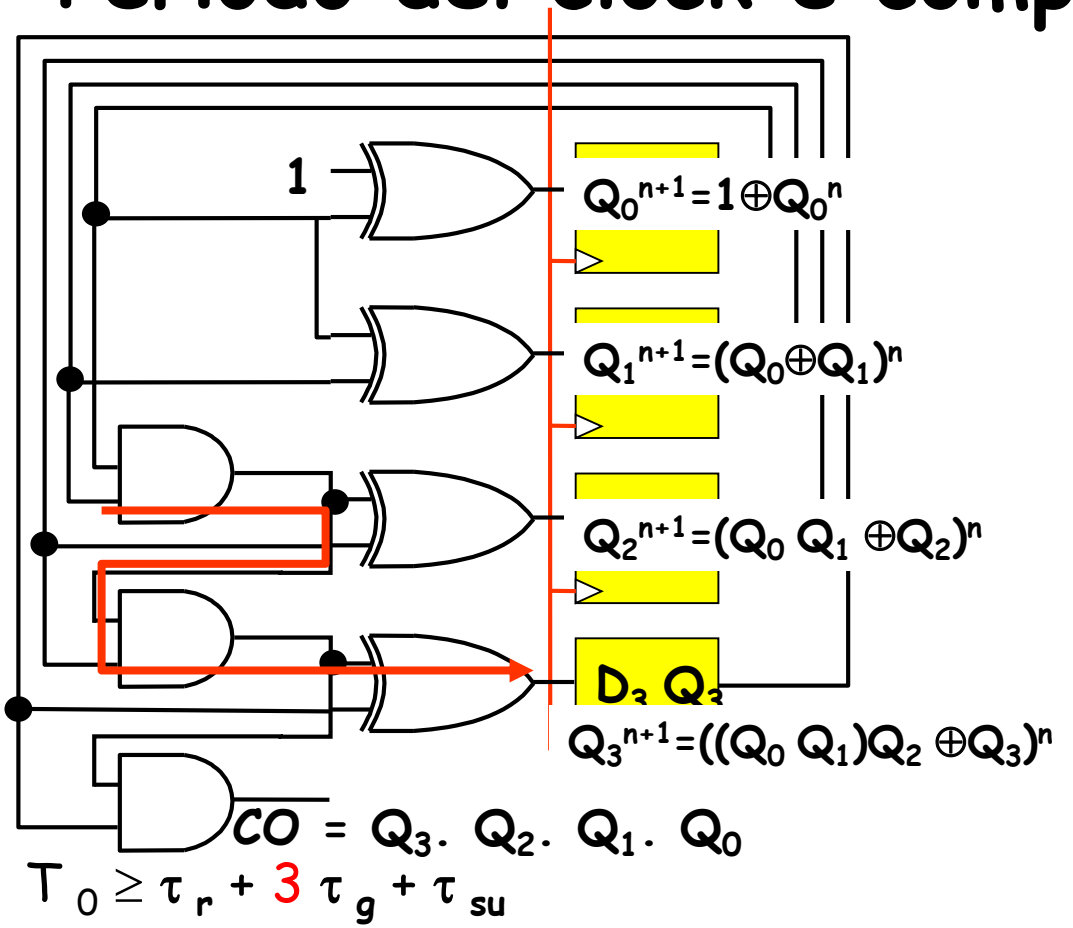
- Sì, è sufficiente una rete più semplice (1 AND e 1 EX-OR ad ogni stadio)
- $S_0 = Q_0 \oplus 1 \; (= Q_0')$
- $R_0 = Q_0$

Eliminabile se Q_0' disponibile

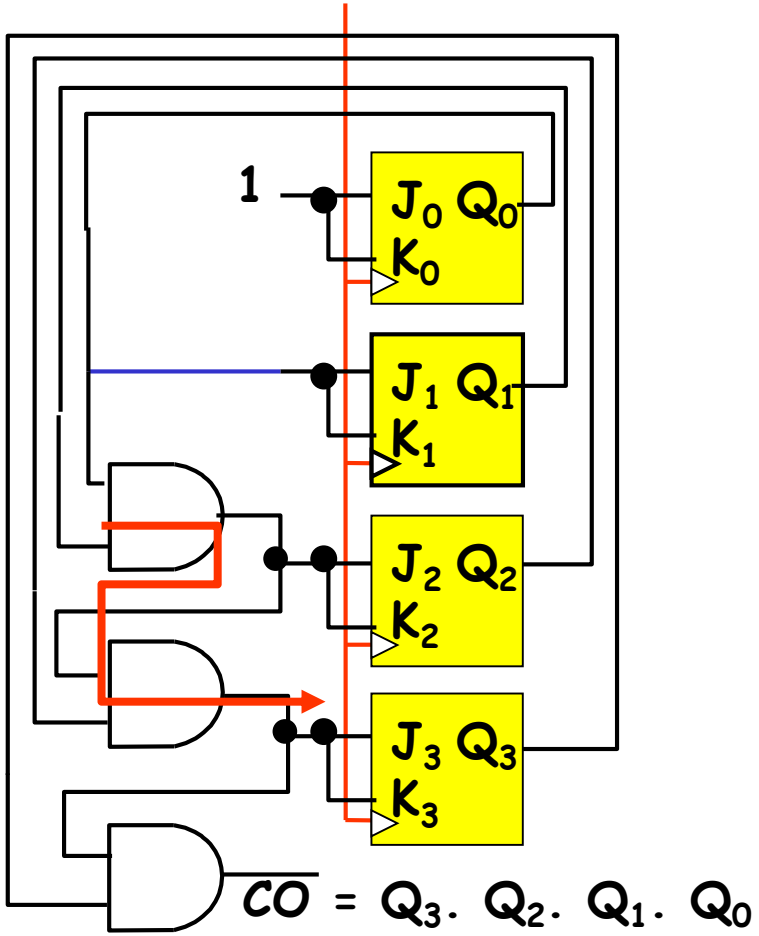


- $S_i = Q_i \oplus r_i$
- $R_i = Q_i \, r_i$
- $S_{i+1} = Q_{i+1} \oplus r_{i+1}$
  $= Q_{i+1} \oplus (Q_i \, r_i)$
  $= Q_{i+1} \oplus (Q_i Q_{i-1} r_{i-1})$
  $= Q_{i+1} \oplus (Q_i Q_{i-1} .. Q_0)$
- $R_{i+1} = Q_{i+1} \, r_{i+1}$
  $= Q_{i+1} \, Q_i \, r_i$
  $= Q_{i+1} Q_i Q_{i-1} r_{i-1}$
  $= Q_{i+1} Q_i Q_{i-1} .. Q_0$

Periodo del clock e complessità della rete



$T_0 \geq \tau_r + 3\,\tau_g + \tau_{su}$



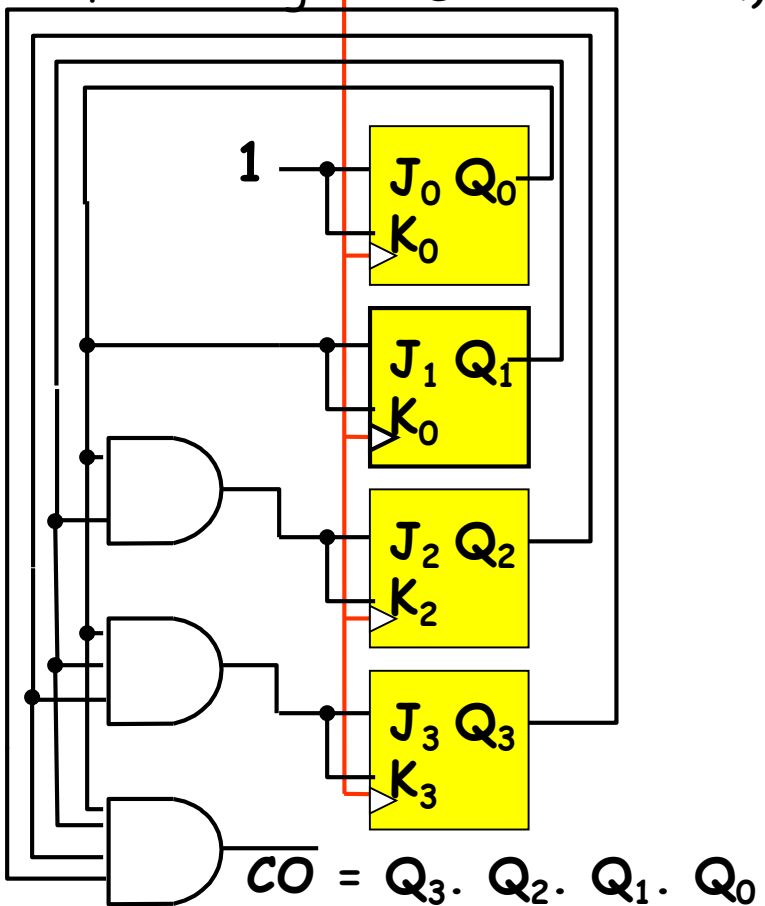
$T_0 \geq \tau_r + 2\,\tau_g + \tau_{su}$

Eq. caratteristica FF JK: $Q^{n+1} = (J.Q' + K'.Q)^n$
Se J=K=1 -> $Q_0^{n+1} = (Q_0')^n$ (toggle)
Se J=K=Q_0^n -> $Q_1^{n+1} = (Q_0Q_1'+Q_0'Q_1)^n = (Q_0\oplus Q_1)^n$

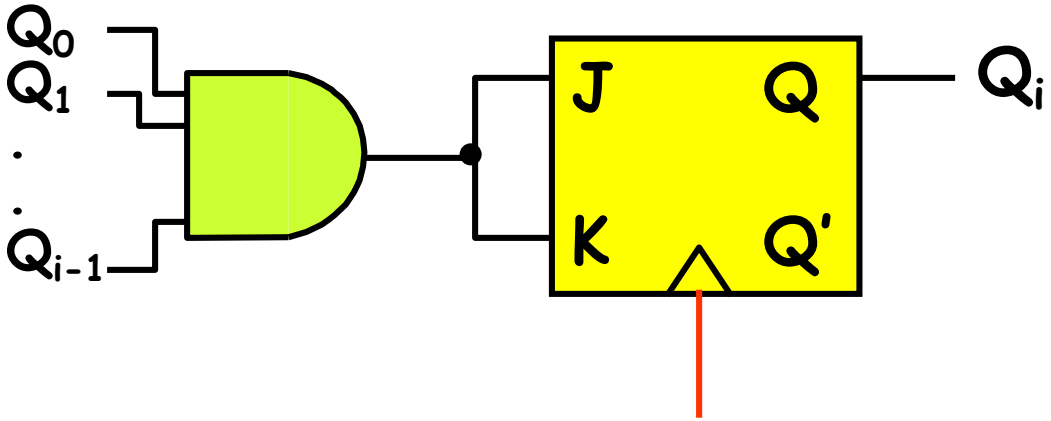
- Utilizzando FF D, nel caso di contatore a 4 bit il percorso di elaborazione più lungo è formato da 3 gate
- Utilizzando FF JK si eliminano gli ex-or e il percorso di elaborazione più lungo ha meno gate -> utilizzabile con frequenza di clock più alta!

Un contatore ancora più veloce

- Per la proprietà associativa del prodotto logico, gli AND in cascata possono essere sostituiti da un unico AND a più ingressi (allo stadio i-esimo, tutti i Q_j con j=0, .., i-1 più il riporto i-esimo)
- Intuitivamente: il bit di peso i deve incrementare il suo valore se e solo se tutti i bit di minor peso hanno valore 1: in quel caso, J=K=1 («toggle»); in caso contrario: J=K=0 («hold»).
- Vantaggio: ritardi ancora più ridotti sul percorso più lungo, **rete più veloce** (ma il fan-in degli AND cresce con i!)



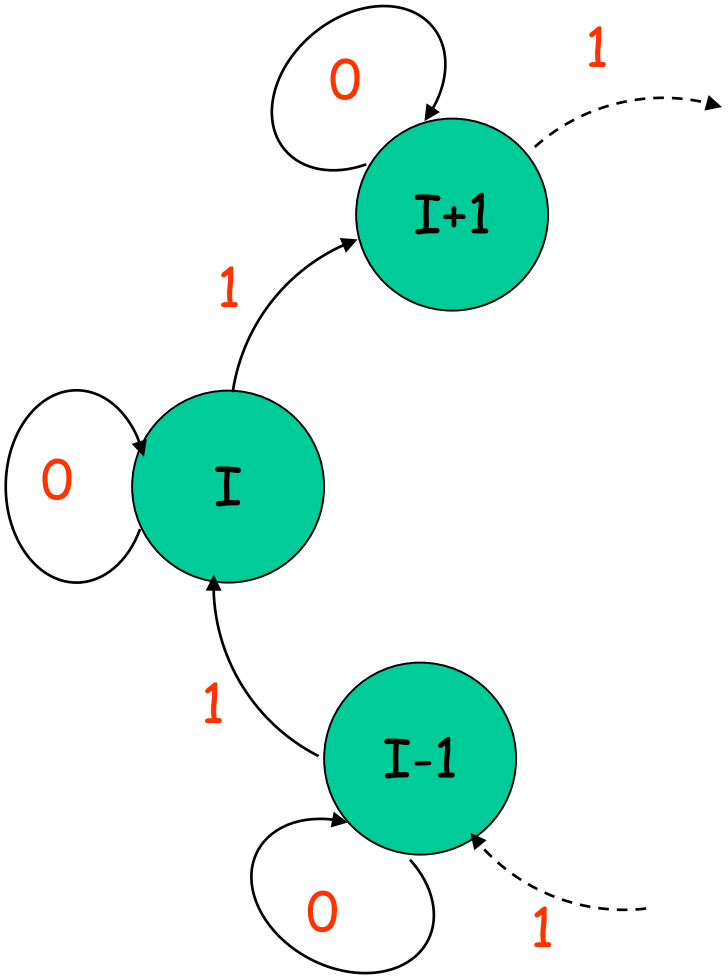
$$Q_i^{n+1} = (..((Q_0\ Q_1)Q_2)..Q_{i-1}) \oplus Q_i)^n$$
$$= (Q_0\ Q_1 Q_2 Q_{i-1} \oplus Q_i)^n$$

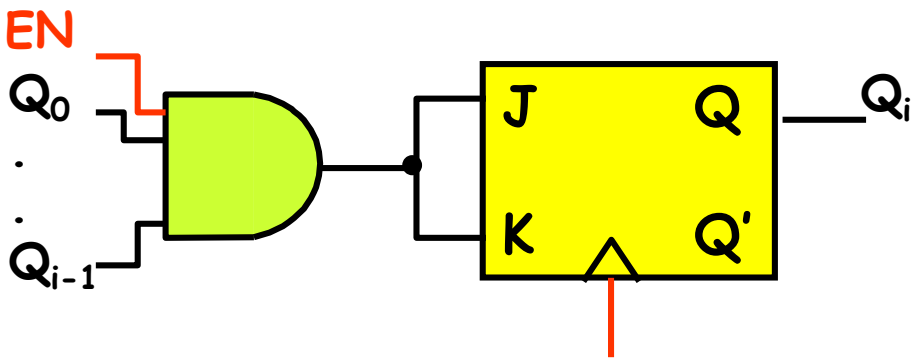


$$T_0 \geq \tau_r + \tau_g + \tau_{su}$$

Il comando di ENABLE

- Anche i contatori sono spesso dotati di segnali di comando sincroni.
- ENABLE: abilita/disabilita la modifica dello stato interno del contatore



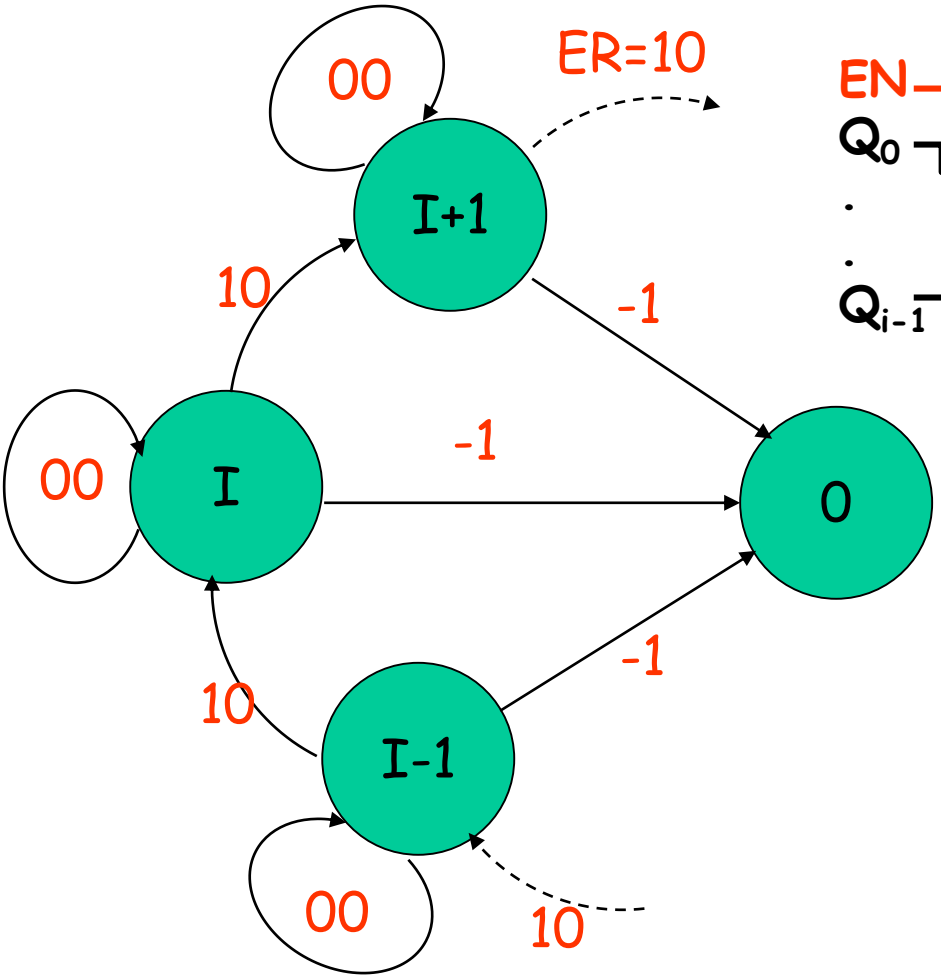


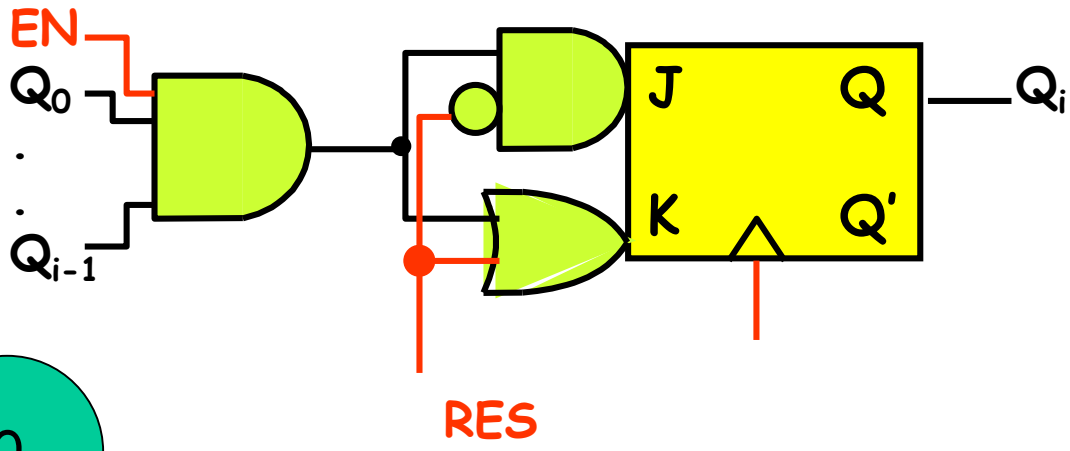
EN = 0: $S^{n+1} = S^n$

EN = 1: $S^{n+1} = (S+1)^n \bmod 2^n$

ENABLE & RESET

- RESET: impone allo stato interno la configurazione di tutti «zeri»
- è tipicamente **prioritario** rispetto all'ENABLE
- nella rete mostrata, RES=1 impone la modalità d'ingresso «reset» al FF JK (J=0, K=1)





RES = 1: $S^{n+1} = 0$

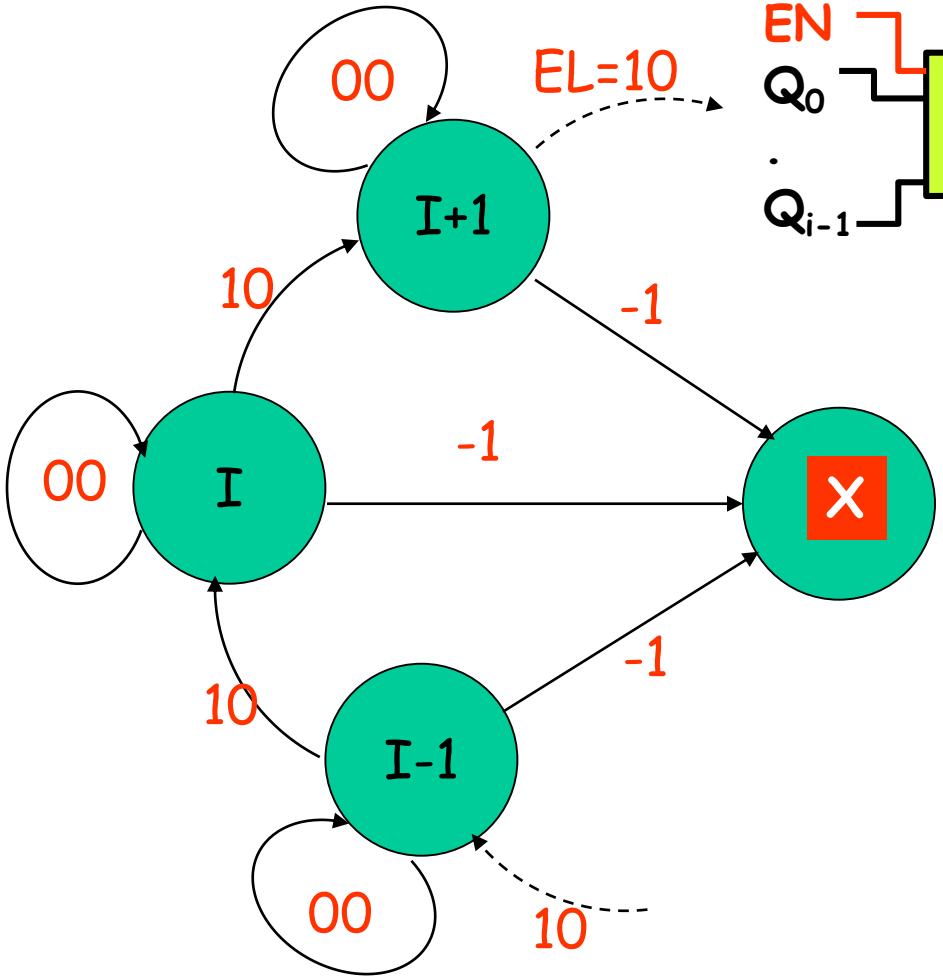
RES = 0 e

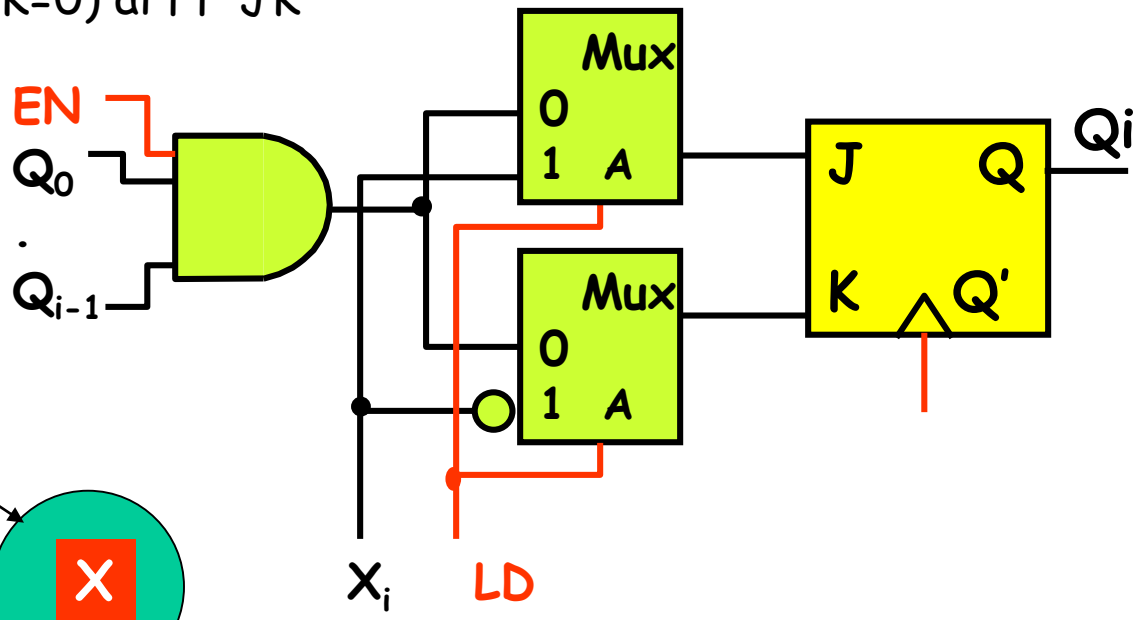
EN = 0: $S^{n+1} = S^n$

EN = 1: $S^{n+1} = (S+1)^n \bmod 2^n$

ENABLE & LOAD

- LOAD: setta lo stato interno ad una configurazione imposta dall'esterno tramite opportuni segnali $x_0, \ldots, x_{k-1}$
- Tipicamente prioritario rispetto all'ENABLE
- nella rete mostrata, LD=1 impone le modalità «reset» (X_i = 0 -> J=0,K=1) oppure «set» (X_i = 1 -> J=1,K=0) al FF JK





LD = 1: $S^{n+1} = X$

LD = 0 e

EN = 0: $S^{n+1} = S^n$

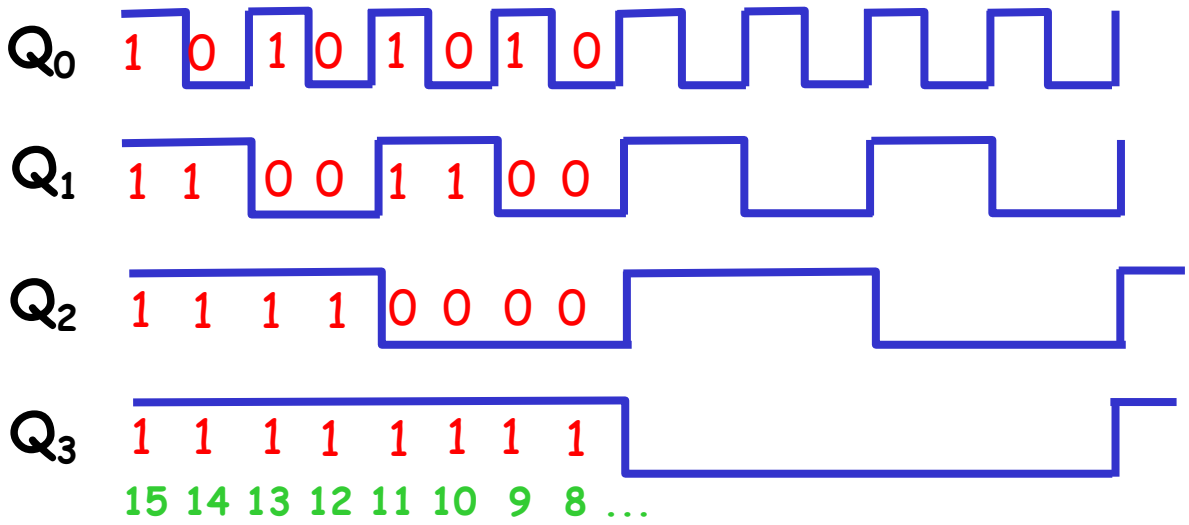
EN = 1: $S^{n+1} = (S+1)^n \bmod 2^n$

Il contatore "all'indietro"

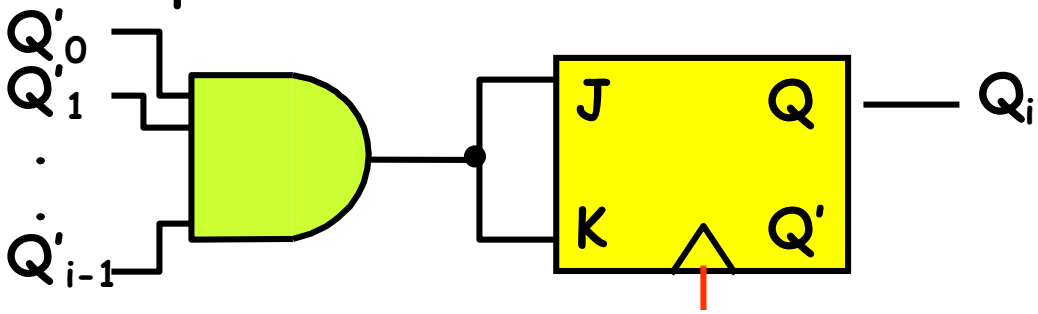
- Per decrementare anzichè incrementare: è possibile utilizzare una rete del tutto analoga alla precedente in cui si complementano tutti i bit d'ingresso dell'AND
- Intuitivamente: il bit di stato i-esimo deve commutare il suo stato (*toggle*, *J=K=1*) quando tutti i bit precedenti sono a 0

prestito

$Q_i\ Q_{i-1}\ \ldots\ Q_1\ Q_0$ **-**
1

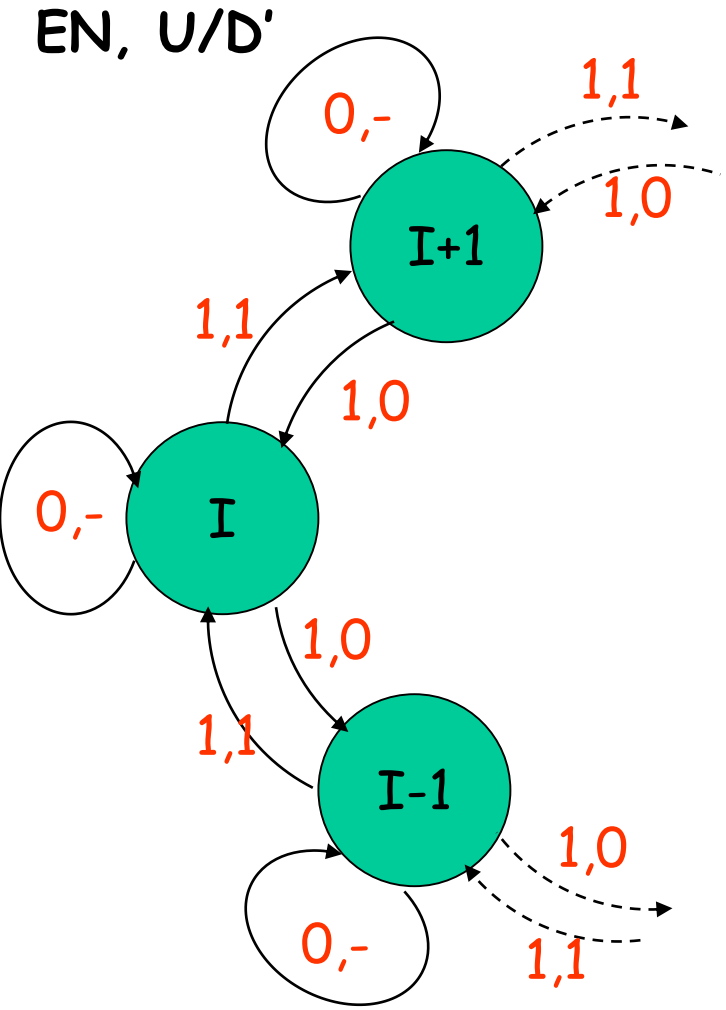


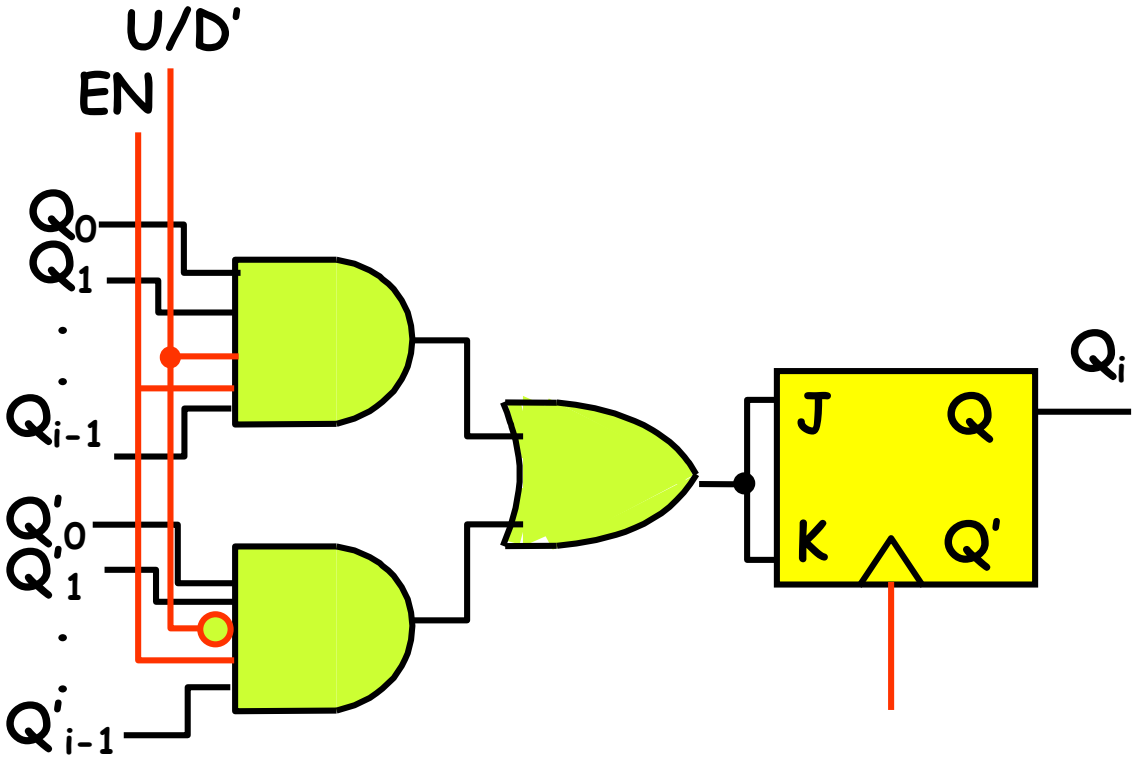
i-esimo prestito



ENABLE & Up/Down (U/D')

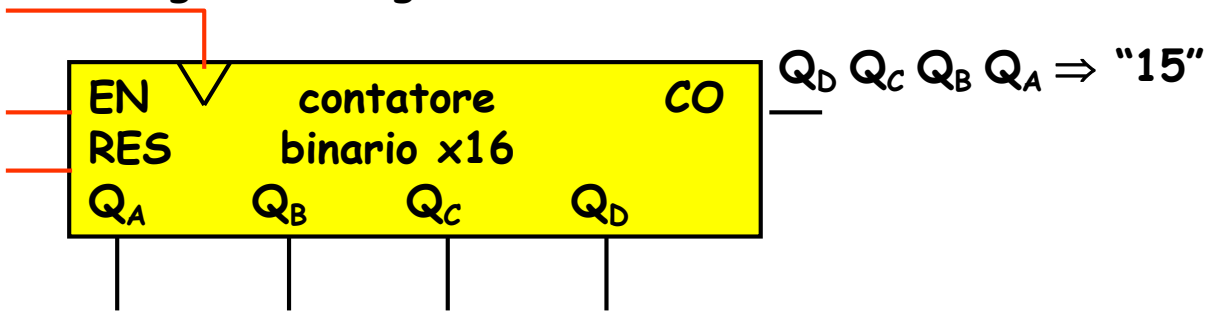
- Comando di **UP/DOWN**: stabilisce se il conteggio va effettuato in avanti o all'indietro (incremento/decremento dello stato interno)
- Nell'esempio: ENABLE proritario rispetto a U/D



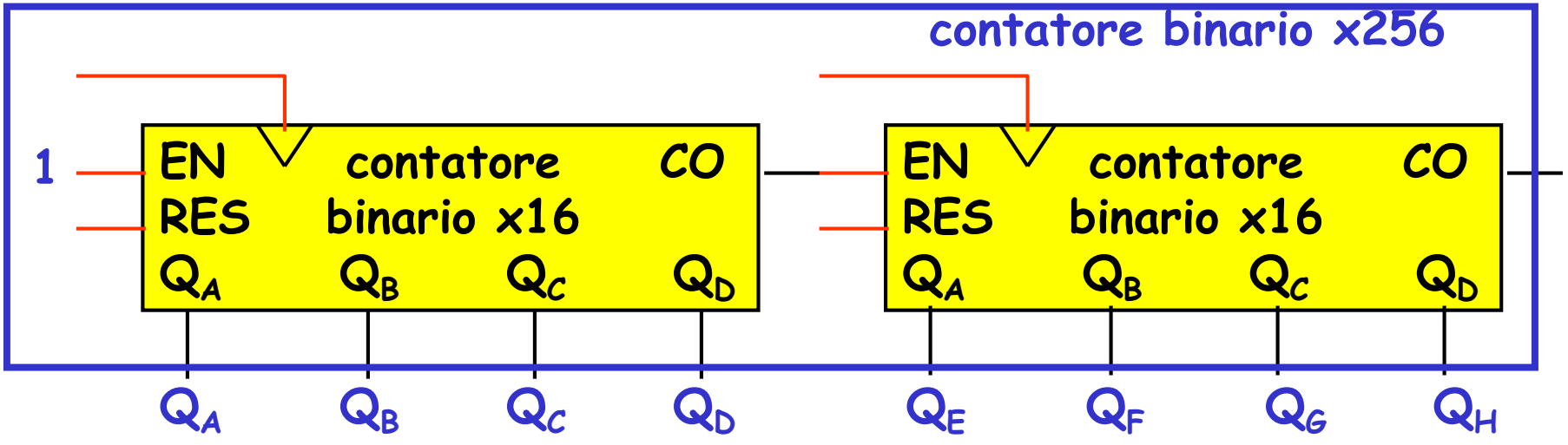


Aumento della base di conteggio

- Disponendo più contatori in cascata si può aumentare la base di conteggio
- Risulta dunque utile utilizzare il segnale CO (precedentemente non utilizzato) per collegarlo all'ingresso di ENABLE del modulo successivo

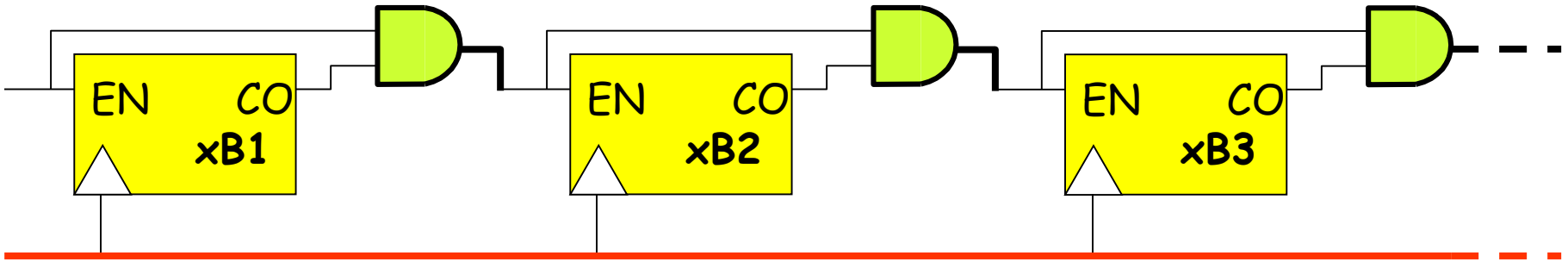


- Esempio: il contatore a destra (che realizza i bit più significativi del conteggio complessivo) «conta» solo in corrispondenza della configurazione 1111 del contatore a sinistra (bit meno significativi del conteggio complessivo)



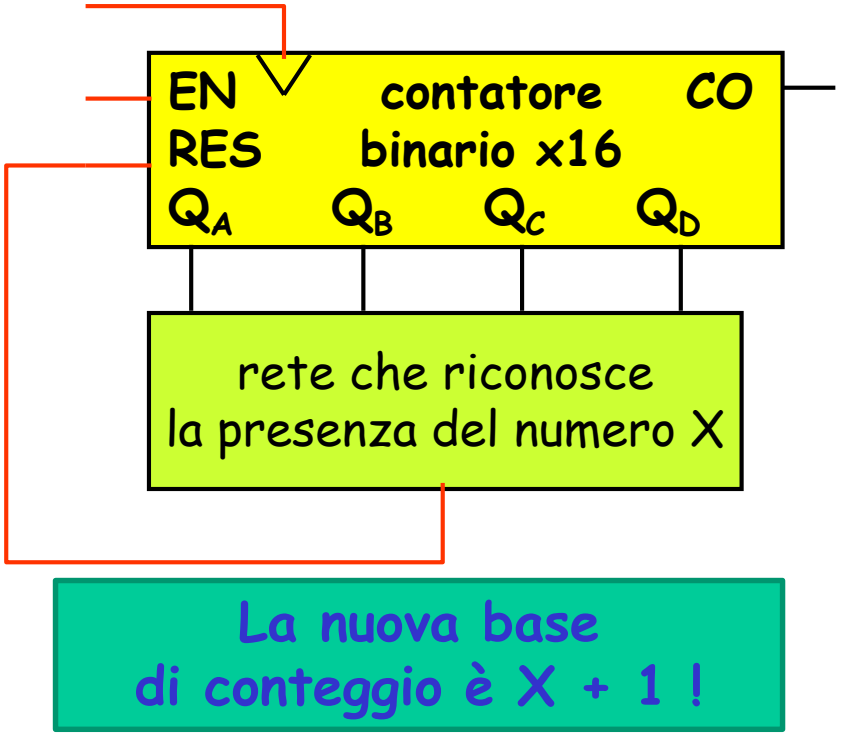
Disposizione in cascata di contatori

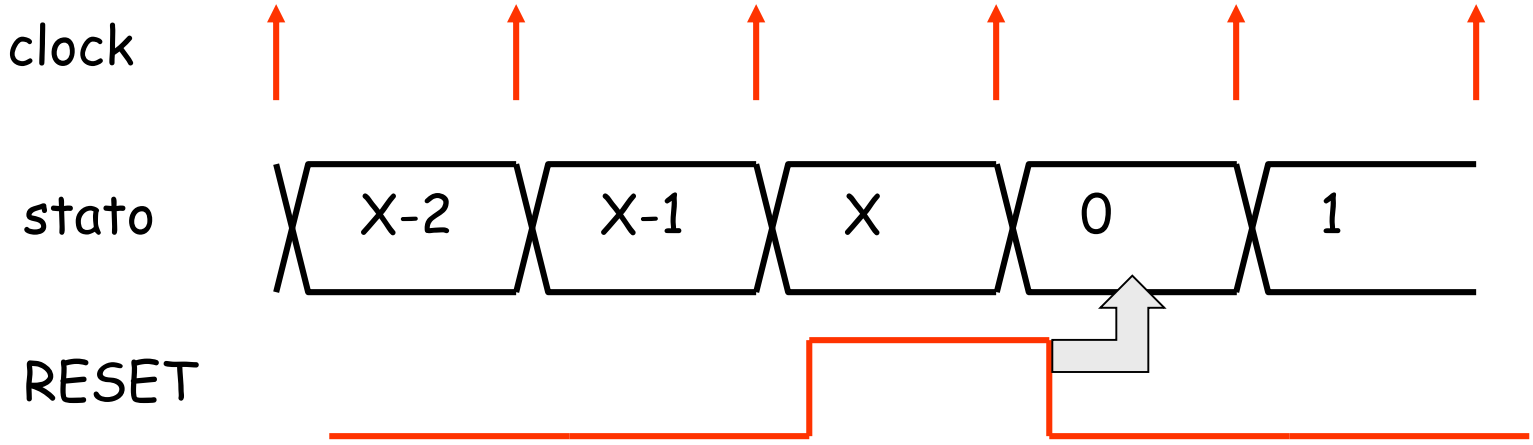
In generale, vale la proprietà che:
la disposizione in cascata di **n** moduli di conteggio,
rispettivamente con base **B1**, **B2**,.., **Bn**,
fornisce un contatore con base **B = B1 × B2 × …… × Bn**



Diminuzione della base di conteggio

- Per ottenere un contatore in base X+1 (con $X<2^k$) a partire da un contatore a k bit, occorre una rete combinatoria che generi il valore 1 quando lo stato presente è = X (0 altrimenti)
- L'uscita di questa rete viene connessa all'ingresso di RESET
- Si ottiene così un contatore ad X+1 stati





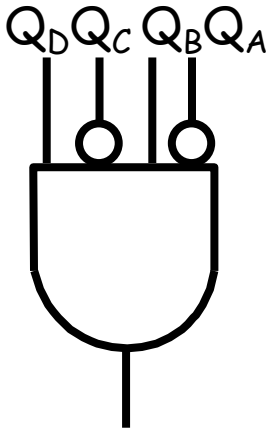
Sintesi del RESET per diminuire la base di conteggio

- Per realizzare la rete combinatoria che riconosce il numero X vi sono varie soluzioni

- Sol. 1) realizzo il mintermine corrispondente a X
  - soluzione rapidamente implementabile

1) Mintermine

Es. $X=(10)_{10} = (1010)_2$ -> RES = $Q_D Q'_C Q_B Q'_A$

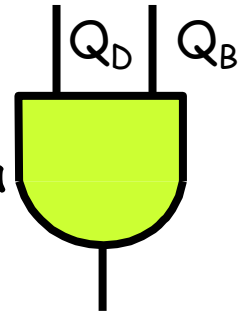


Sintesi del RESET per diminuire la base di conteggio (cont.)

- Sol. 2) tutti gli stati relativi ad un conteggio di valore > X costituiscono condizione di indifferenza (in quanto irraggiungibili) -> realizzo una sintesi tramite rete di costo minimo
- Vantaggio rispetto a soluzione precedente: utilizzo AND a fan-in minore
- In generale: sintesi minima equivale all'AND tra i bit a «uno» nella rappr. binaria di X; ovvero si connette al RESET un AND avente in ingresso i soli bit che hanno valore 1 in X

Q_D^n, Q_C^n \ Q_B^n, Q_A^n	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	-	-	-	-
10	0	0	-	1

RES^n



2) rete minima
$RES = Q_B Q_D$

Es. formulazione generale:
contatore in base X+1=53

$X=52_2 = 1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0 = 110100$

110100 è minore di
111100
110110
110101
ecc.

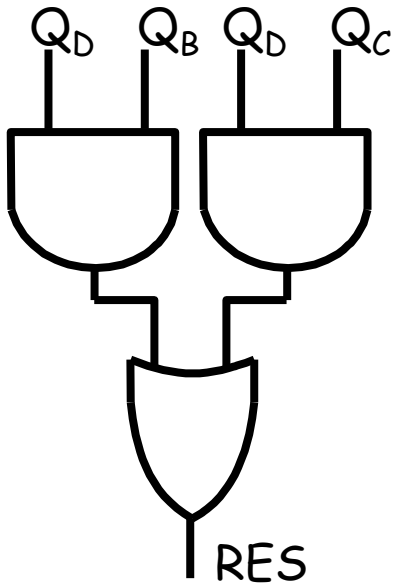
$RES = b_5 \cdot b_4 \cdot b_2$

Sintesi del RESET per diminuire la base di conteggio (cont.)

- 3) Progetto di una rete che fornisce 1 in corrispondenza di X e di ogni numero di valore più grande
  - equivale alla sintesi minima con copertura di tutte le condizioni di indifferenza
  - svantaggio: utilizzo di un maggior numero di gates
  - vantaggio: consente l'**autoinizializzazione** del contatore in un unico intervallo di clock (ogni configurazione non utilizzata attiva immediatamente il RESET)

Q_D^n, Q_C^n \ Q_B^n, Q_A^n	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	-	-	-	-
10	0	0	-	1

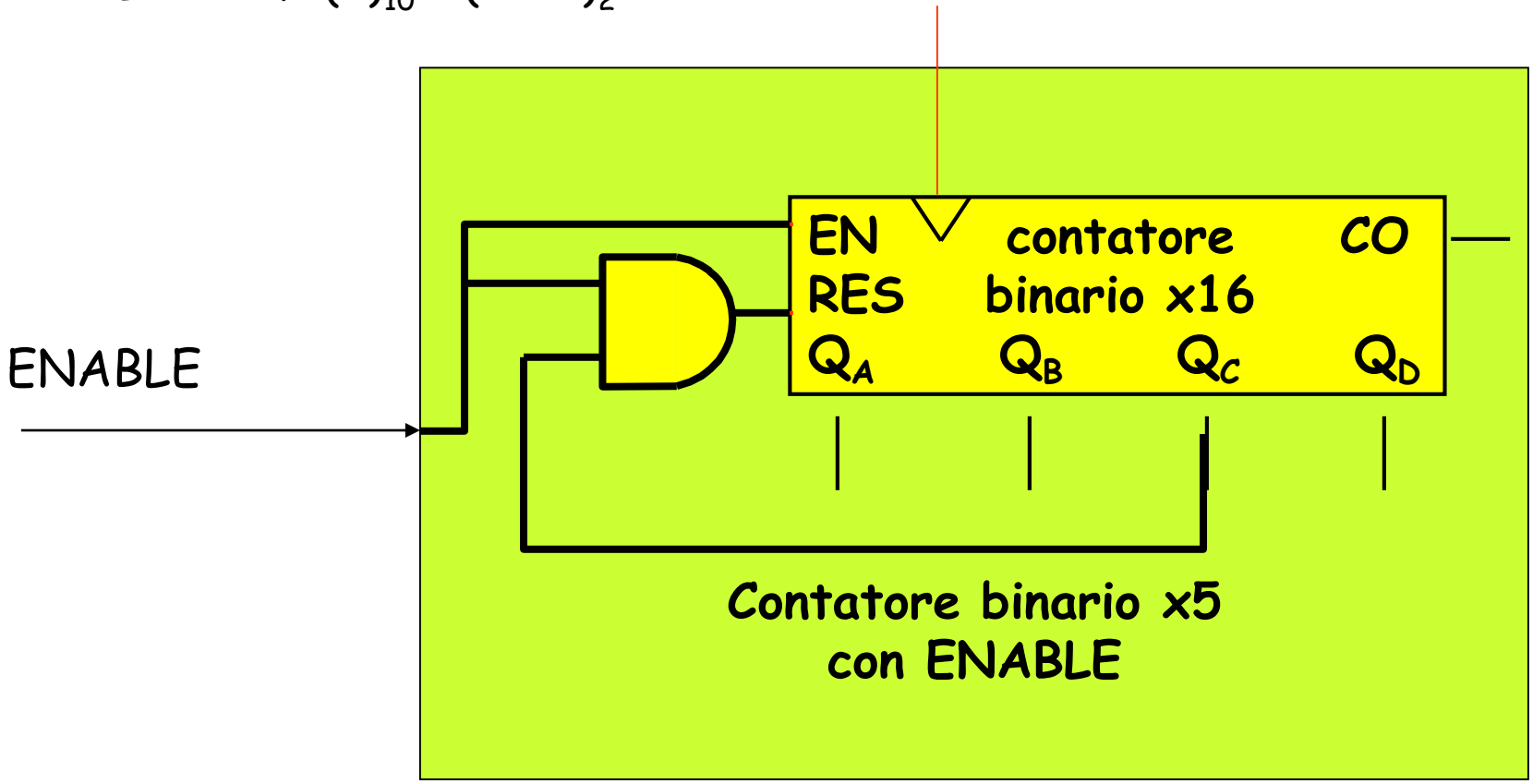
RES^n



3) autoinizializzazione in un solo clock

Esercizio

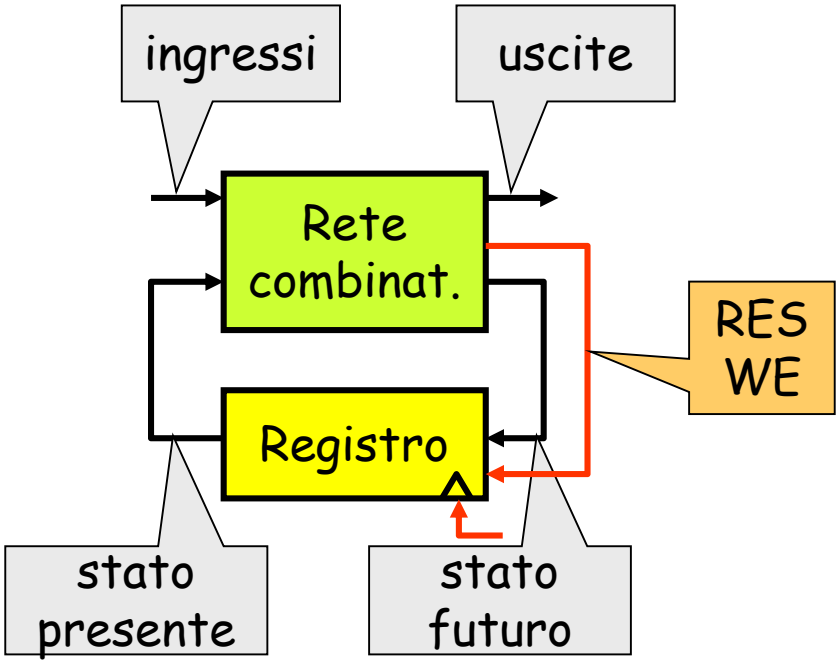
- Dato un contatore in base 16 con segnale di RESET e ENABLE, realizzare un contatore in base 5
- Se B=5 -> X =$(4)_{10}$ = $(0100)_2$

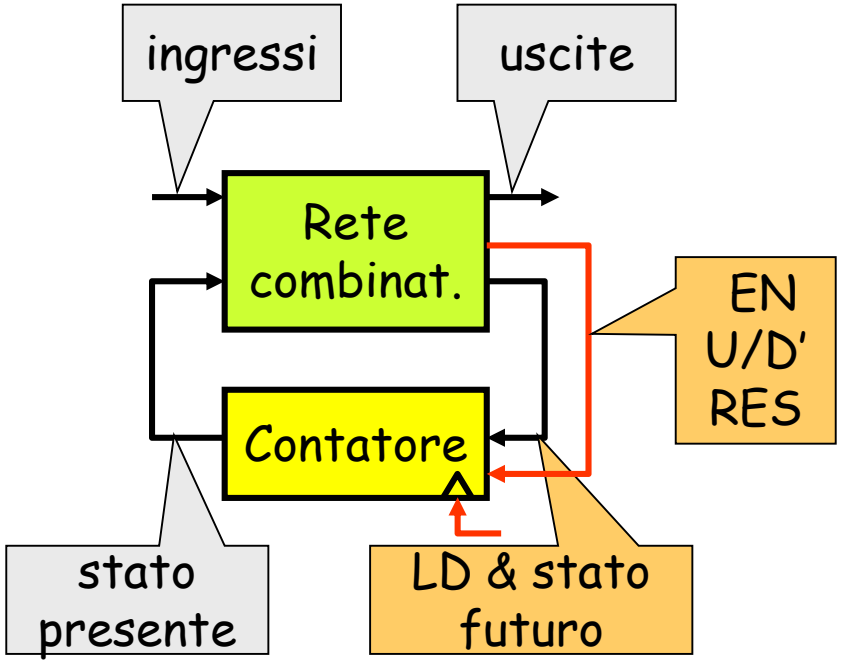


- L'AND va condizionato con l'ingresso di ENABLE per evitare che il contatore resetti il suo stato interno quando ENABLE=0 e Q_C=1 (RESET è prioritario rispetto all'ENABLE)

Sintesi con contatori

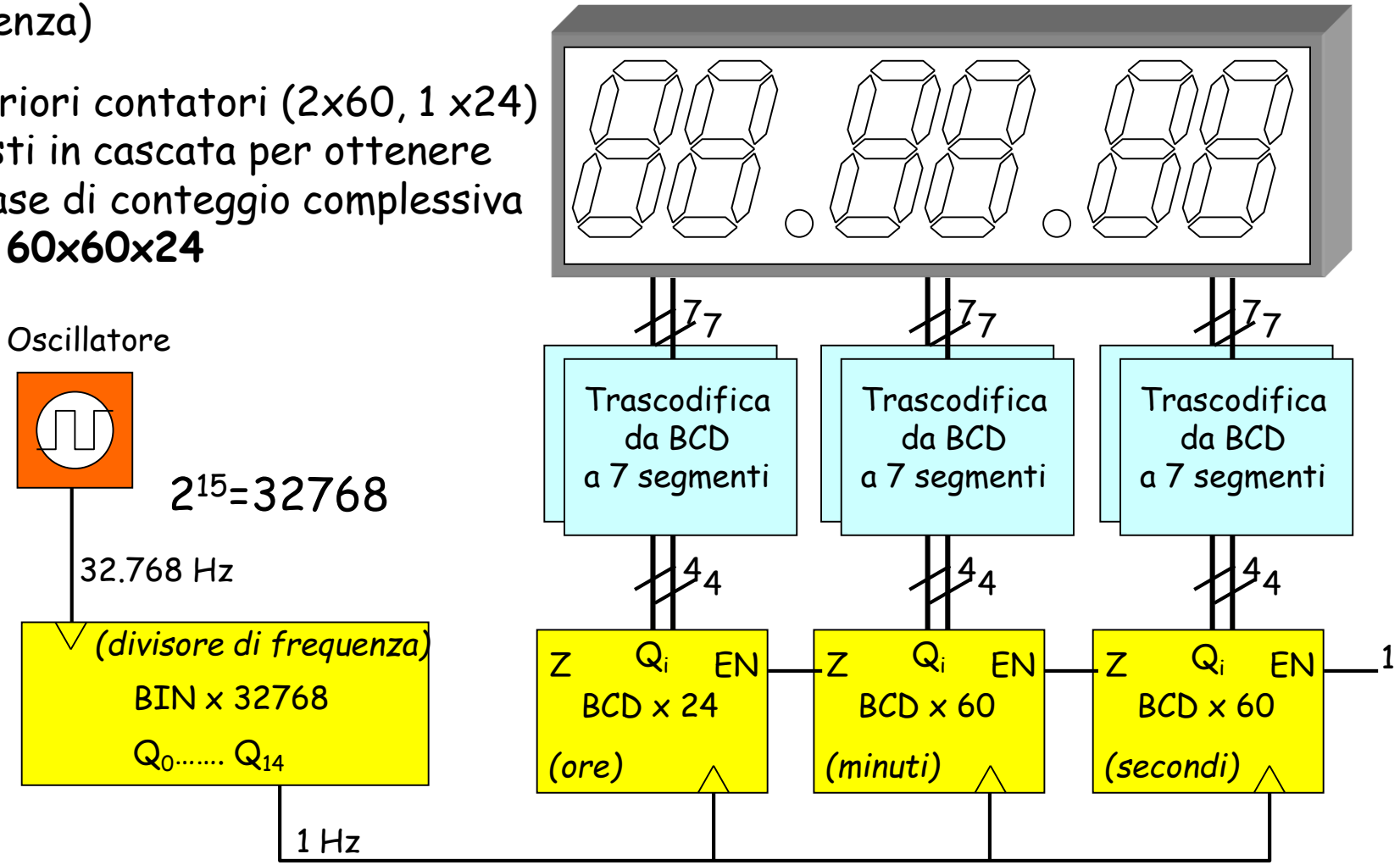
- I contatori possono essere utilizzati nelle reti sincrone per la memorizzazione dello **stato interno** al posto dei registri di stato
- Necessario utilizzare contatori con comando di LOAD (posto a 1) e ingressi di LOAD collegati ai segnali di stato futuro
- Un utilizzo opportuno dei comandi di EN, RES, U/D e LD permette spesso di ottenere **realizzazioni più semplici** della RSS
- L'utilizzo di contatori risulta in generale utile nel caso di
  - circuiti dedicati alla misura del **trascorrere del tempo** e al **conteggio di eventi**
  - circuiti di controllo di altri circuiti, ove è necessario inviare segnali con **forme d'onda periodiche o aperiodiche**





Esempio 1/3: orologio digitale

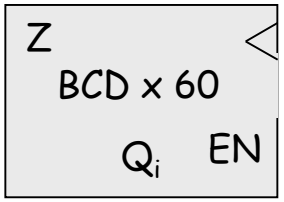
- Progettare un orologio digitale in grado di visualizzare ore, minuti e secondi tramite sei display a 7 segmenti a partire da un oscillatore a 32768 Hz
- Segnale sull'uscita Q14 del contatore x32768(=2^{15}) ha la frequenza di **1 Hz** (utilizzabile come segnale di clock, impiego del contatore come divisore di frequenza)
- 3 ulteriori contatori (2x60, 1 x24) disposti in cascata per ottenere una base di conteggio complessiva pari a **60x60x24**

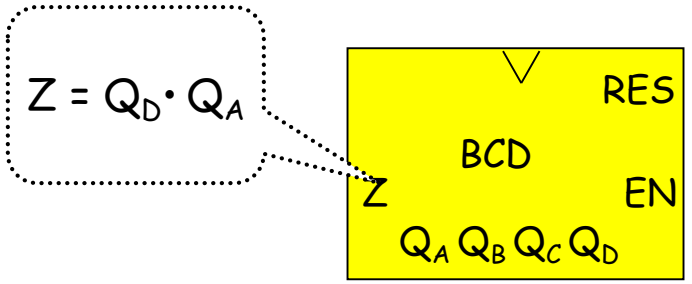


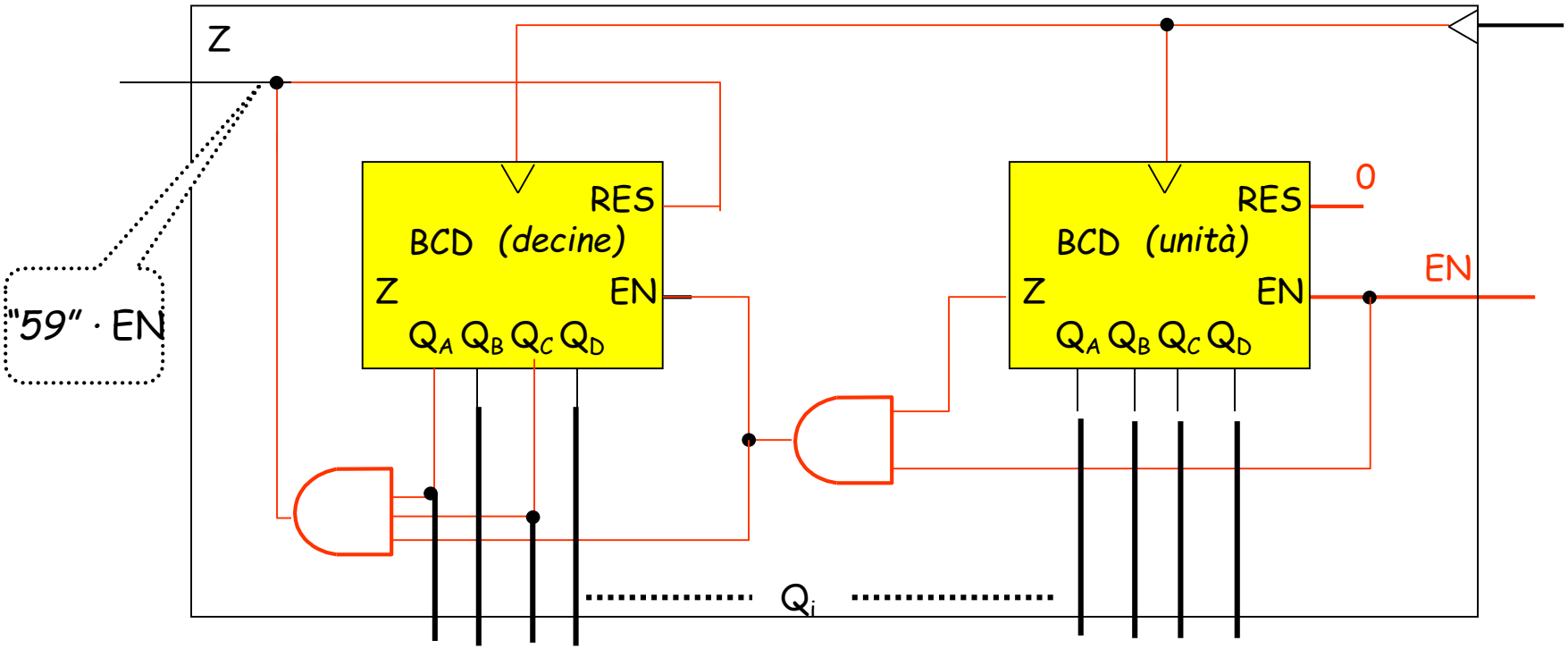
Il contatore dei minuti/secondi

- Conteggio base 60: due contatori BCD in cascata con **diminuzione della base di conteggio** (X=59=0101 1001)
- Contatore BCD: *CO=Z=1 se lo stato = 9* (quindi $Z = Q_D \cdot Q_A$)

0000 0000
0000 0001
……..
0000 1001
0001 0000
……
0101 1001

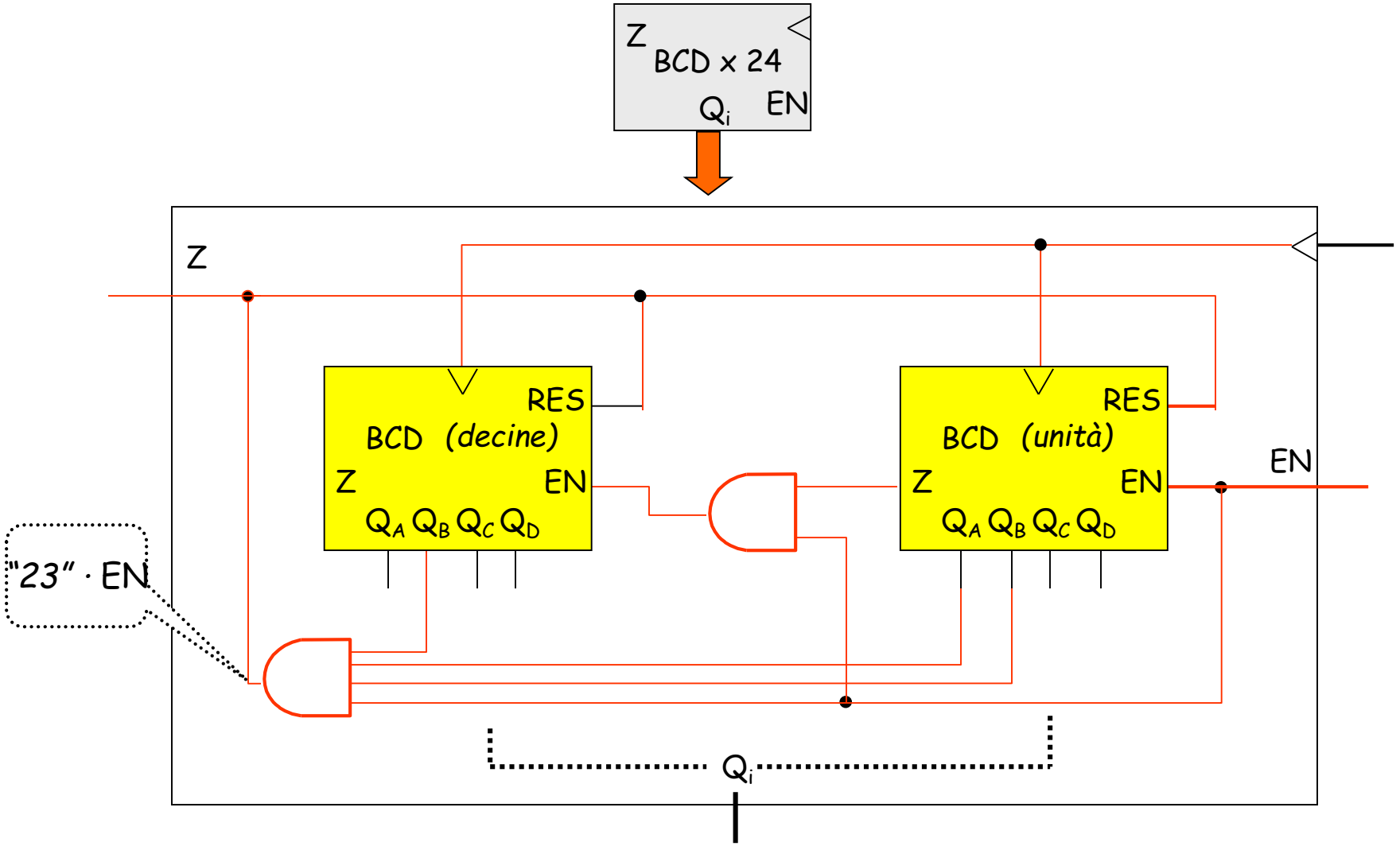






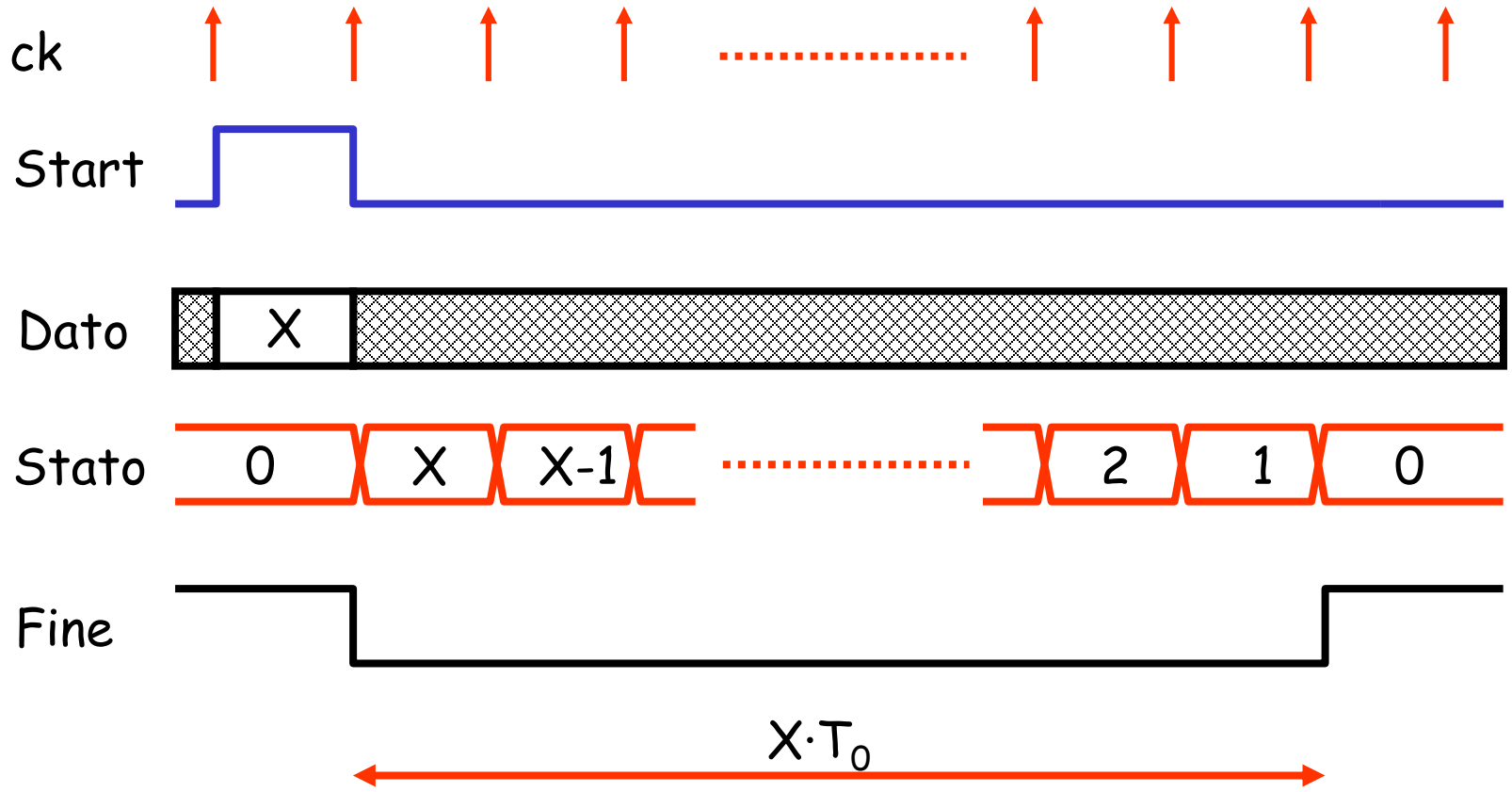
Il contatore delle ore

- Simile al caso precedente (due contatori BCD in cascata) ma realizza un contatore a base 24 (X=23=0010 0011)
- RES = $(Q_A Q_B)_1$ $(Q_B)_2$ EN



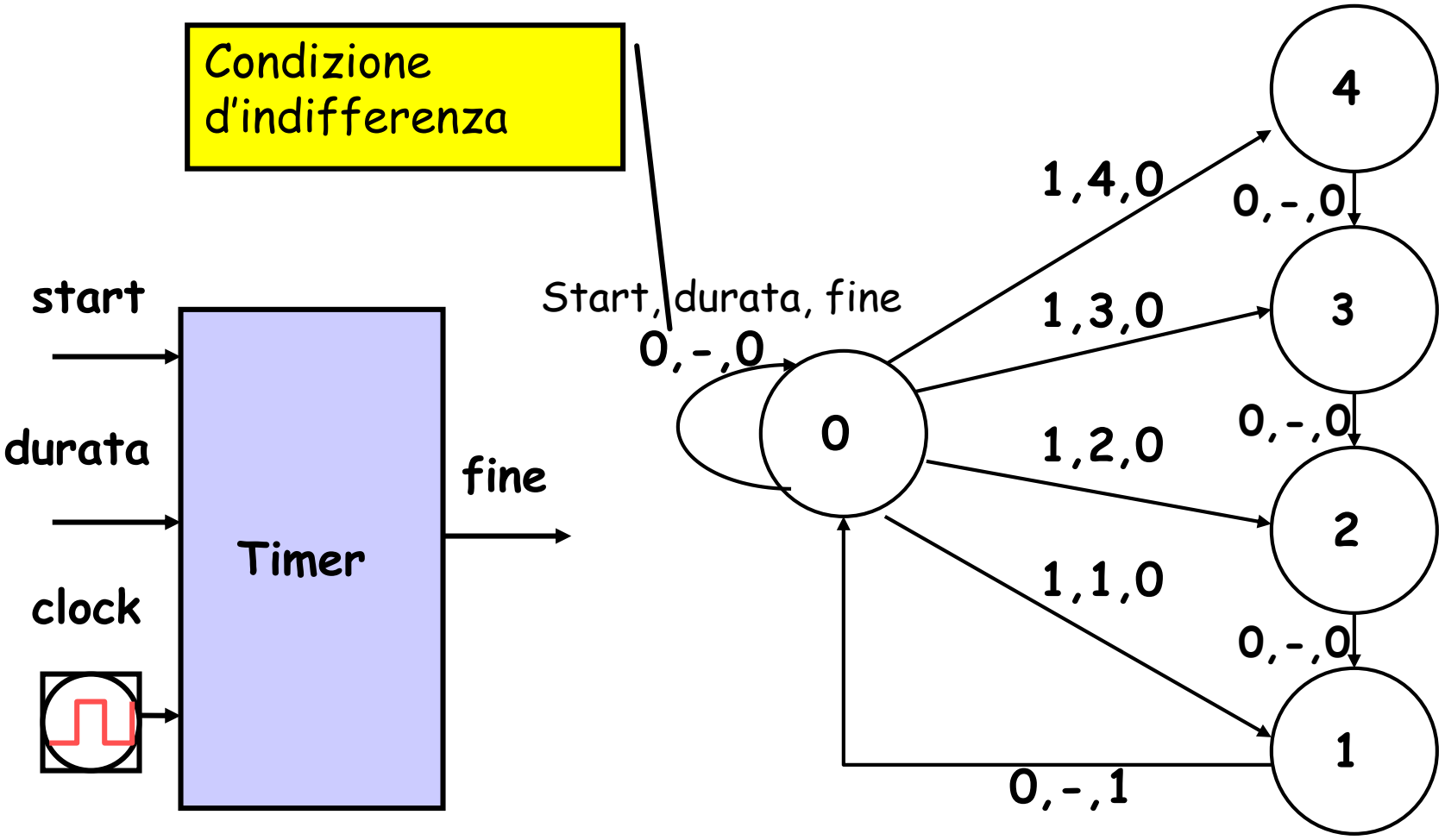
Esempio 2/3: il watch dog (o timer)

- RSS che, dato un comando di *start* e la durata di un intervallo di tempo da misurare X (multiplo di T_0) genera una segnalazione di fine intervallo al trascorrere di D (nell'esempio, attivazione del segnale *fine*)
- Con start=1, viene eseguito un **conteggio all'indietro** (da X a 0)



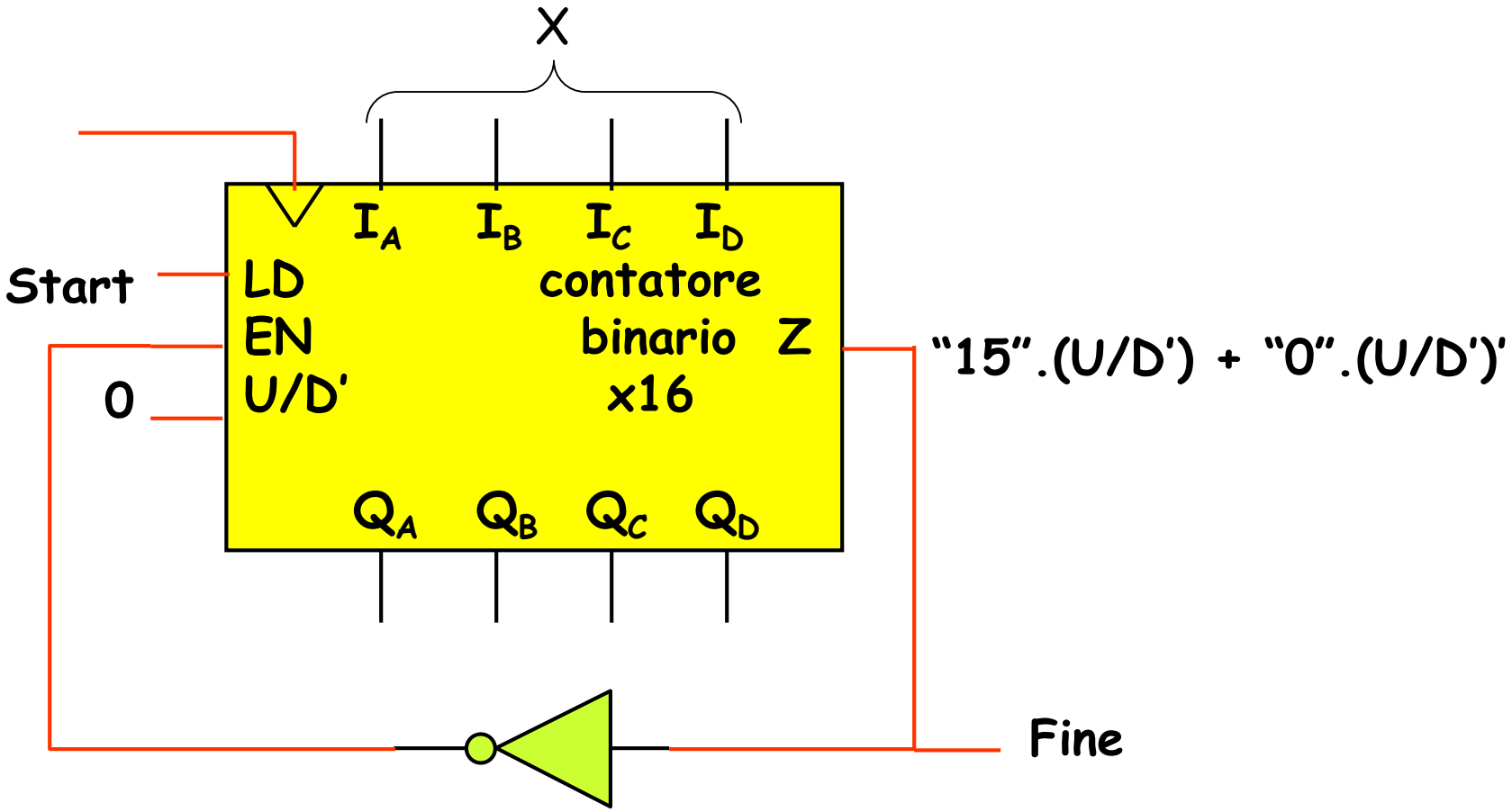
Grafo degli stati

- La macchina usa il periodo del clock come "unità di misura" del tempo T_0
- Nel caso in cui D vale al max 4, il grafo degli stati ha 5 stati di cui uno stabile (codificabili con 3 bit -> 3 FF)



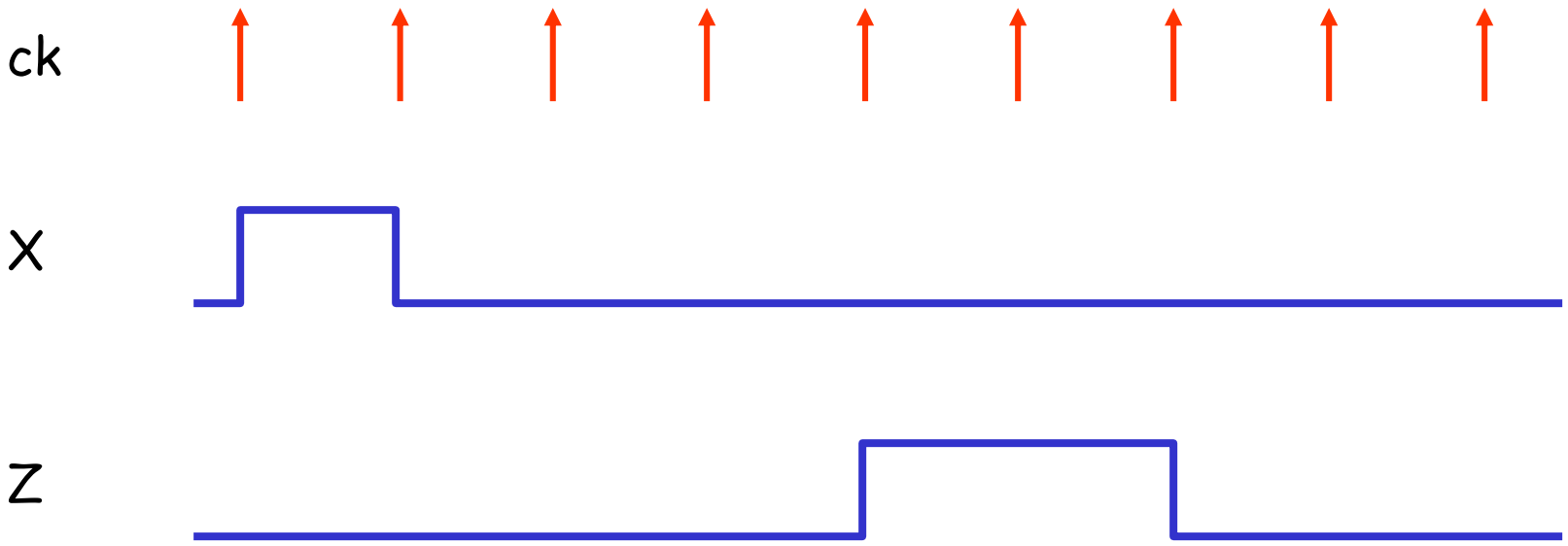
Schema logico

- Utilizzato un contatore x16 con LOAD, ENABLE, UP/DOWN
- Z in questo caso segnala il trabocco anche in conteggio all'indietro: Z=1 se U/D'=1 e S=15 ma anche se U/D'=0 e S=0
- U/D' = 0 per contare sempre all'indietro
- LD collegato a START per caricare X quando START=1
- Z' collegato a EN per arrestare il conteggio quando $Q_A..Q_D$ = 0000



Esempio 3/3: sintesi con contatori

- Una rete sequenziale sincrona ha un ingresso X che assume il valore 1 molto di rado e comunque sempre per un solo periodo di clock.
- L'uscita Z deve sia ritardare l'impulso di ingresso di quattro unità di tempo, sia raddoppiarne la durata.



Grafo e codifica degli stati

- Il comportamento è quello di un contatore con base 6 che compie un intero ciclo per ogni evento X = 1
- Per realizzarlo si può impiegare un contatore binario x8 dotato di comandi di ENABLE e di RESET.
- Per raddoppiare la durata del segnale, in concomitanza degli stati «4» e «5» l'uscita dovrà essere pari a 1

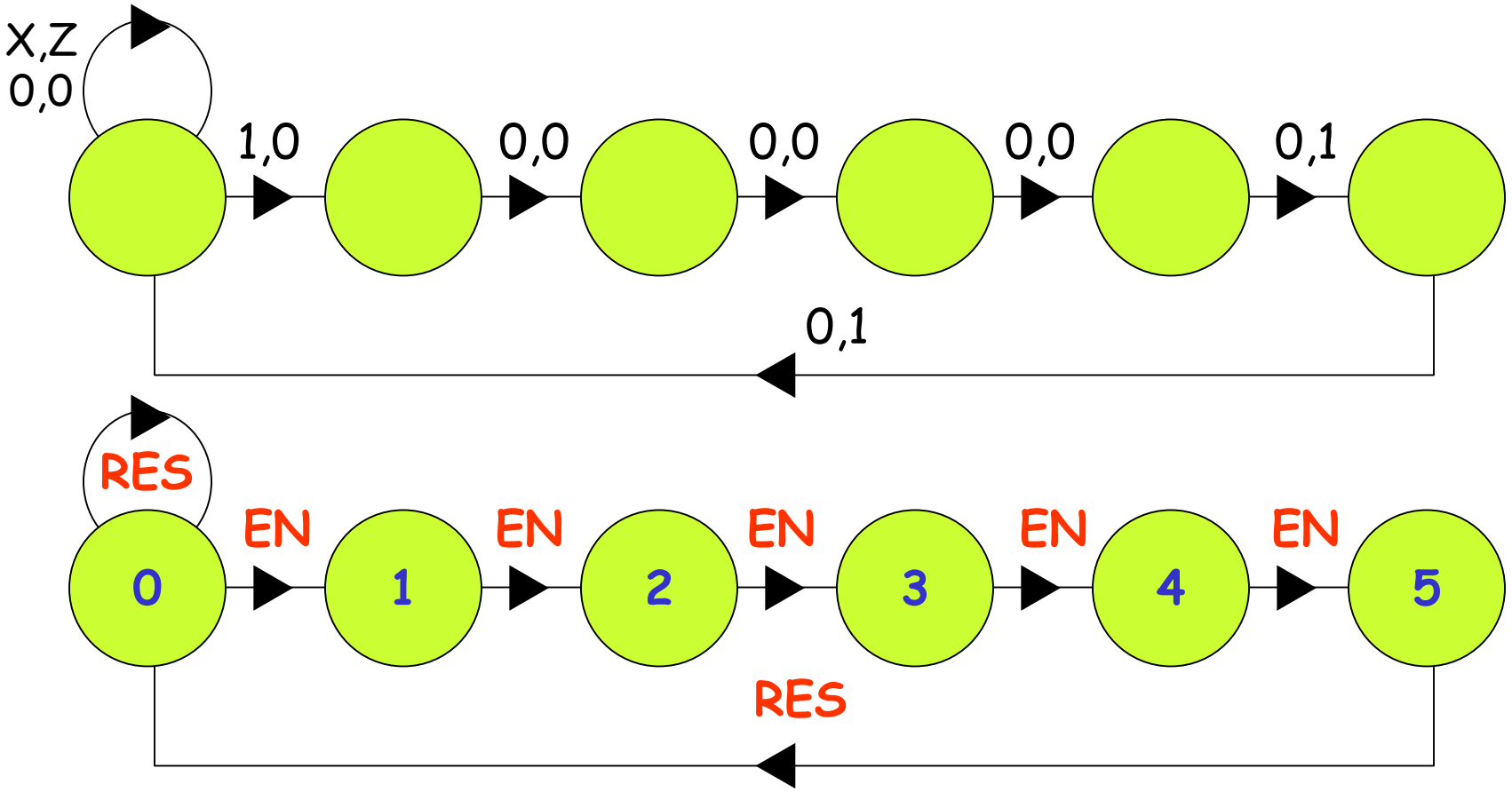


Tabella delle transizioni

	stato	X=0	X=1	Z
riposo	000	000	001	0
	001	010	---	0
	010	011	---	0
	011	100	---	0
	100	101	---	1
	101	000	---	1
	110	---	---	-
	111	---	---	-

stato	X=0	X=1
000	0,0,0	1,0,0
001	1,0,0	-,-,-
010	1,0,0	-,-,-
011	1,0,0	-,-,-
100	1,0,1	-,-,-
101	-,1,1	-,-,-
110	-,-,-	-,-,-
111	-,-,-	-,-,-

EN, RES, Z

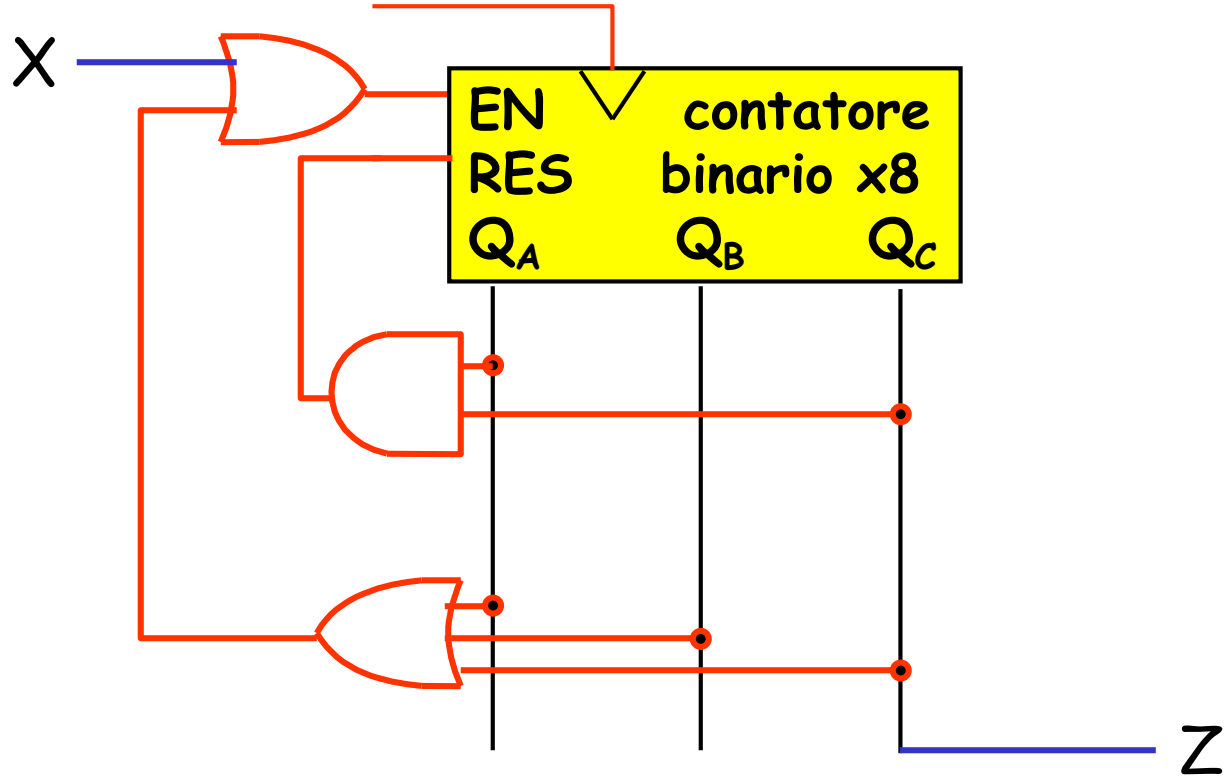
- Con stato 101 e X=0, EN è indifferente in quanto il RESET è prioritario
- Data la semplicità non è necessario utilizzare le mappe per ottenere le espressioni minime
  - EN = 1 se X=1 o se lo stato non è 0
  - RES = 1 se lo stato è 5
  - Z = 1 se lo stato è >= 4

$EN = X + (Q_C'Q_B'Q_A')'$

$RES = Q_C\,Q_A$

$Z = Q_C$

Schema logico



$EN = X + (Q_C'Q_B'Q_A')' = X + (Q_C + Q_B + Q_A)$

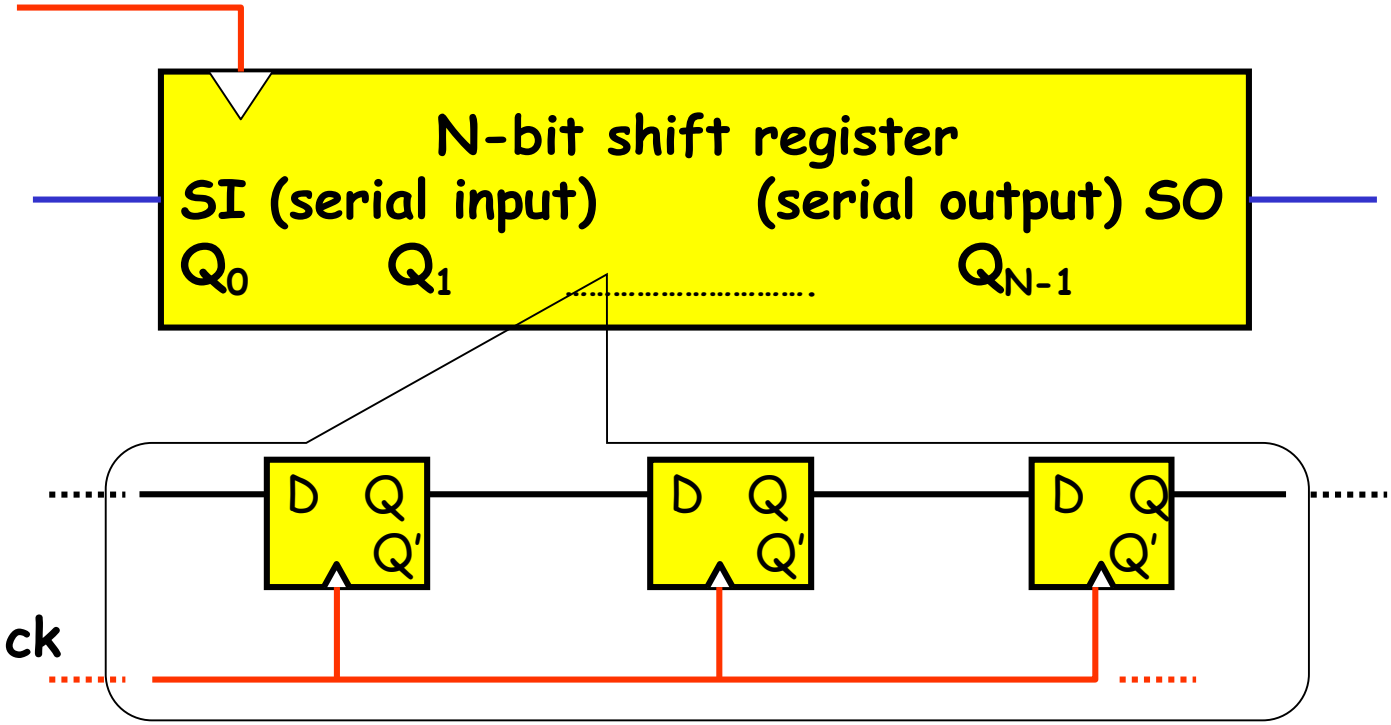
$RES = Q_C\,Q_A$

$Z = Q_C$

- Essendo le uscite del contatore solo in forma vera, per realizzare l'ENABLE conviene trasformare eventuali termini complementati nella equivalente forma vera tramite De Morgan

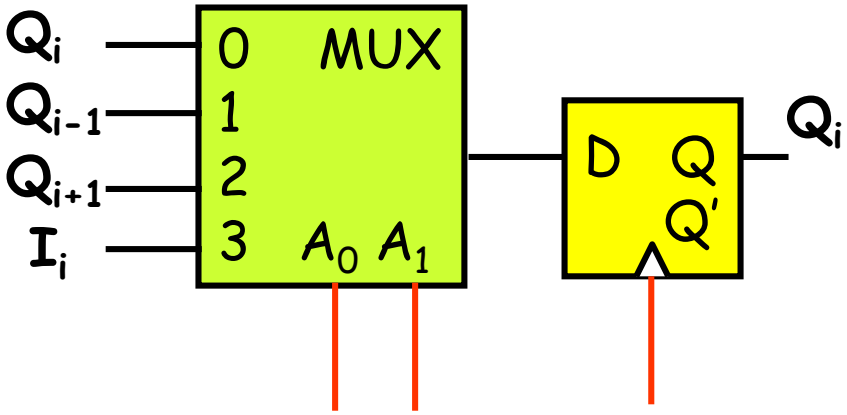
Registri a scorrimento

- **Shift register o registro a scorrimento:** rete sequenziale sincrona formata da N flip-flop D disposti in cascata.
- Componente utile per
  - ritardare da 1 a N intervalli di tempo la forma d'onda di un segnale
  - riconoscere il verificarsi di stringhe d'ingresso
  - convertitore S/P e P/S
  - conteggio
  - rotazione verso destra/sinistra
  - moltiplicazione/divsione per una potenza di 2



Universal shift register

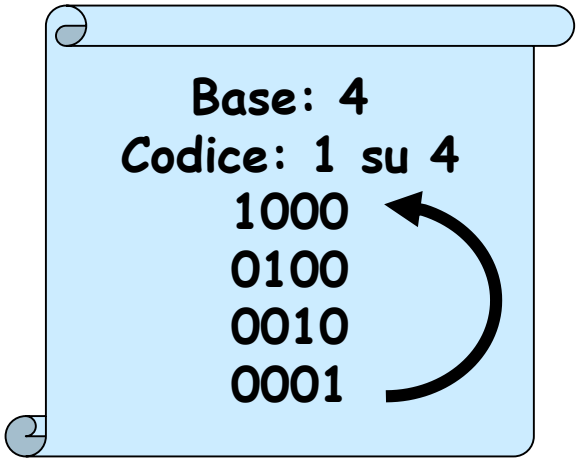
- **Universal Shift Register:** circuito a 4 bit che realizza le varie funzioni attribuibili a uno shift register
- Ogni FF della cascata è preceduto da un MUX avente in ingresso la sua uscita ($\mathbf{Q_i}$), quella del precedente ($\mathbf{Q_{i-1}}$), quella del successivo ($\mathbf{Q_{i+1}}$), un bit esterno ($\mathbf{I_i}$)
- Quattro comportamenti possibili mediante i due bit d'indirizzo A_0A_1:
  - Mantenimento dello stato (*hold*)
  - Scorrimento a destra (*right*) e a sinistra (*left*)
  - Caricamento di un dato (*load*)



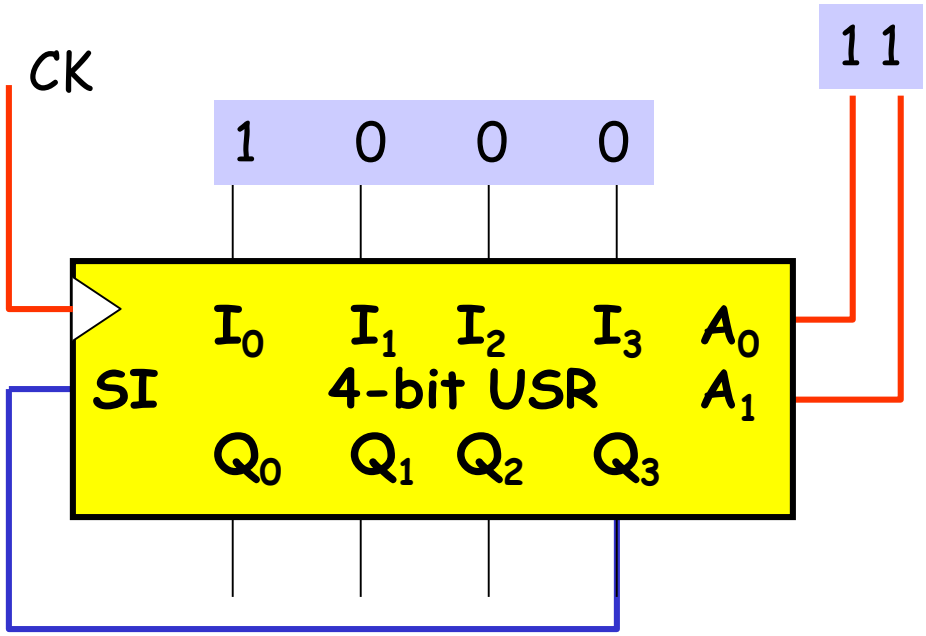
	A_0^n	A_1^n	Q_i^{n+1}
hold	0	0	Q_i^n
right	1	0	Q_{i-1}^n
left	0	1	Q_{i+1}^n
load	1	1	I_i^n

Esempio: contatore ad anello

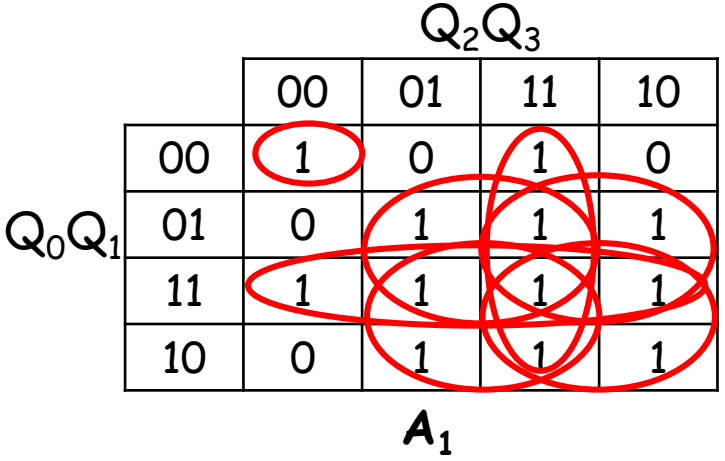
- Realizza un ciclo di conteggio in base 4 (in base N se utilizzo N FF)
- Facendo scorrere un solo «1» scorre le configurazioni del codice «uno su quattro»
- Basta collegare SI a SO (Q_3) e programmarlo in modalità «shift right» (A_0A_1=10)



N flip-flop → Base N



- … e se sta circolando una delle 12 configurazioni non utilizzate?
- Occorre inizializzare («load», A_1A_0=11) il circuito con una delle 4 configurazioni lecite (es. 1000)

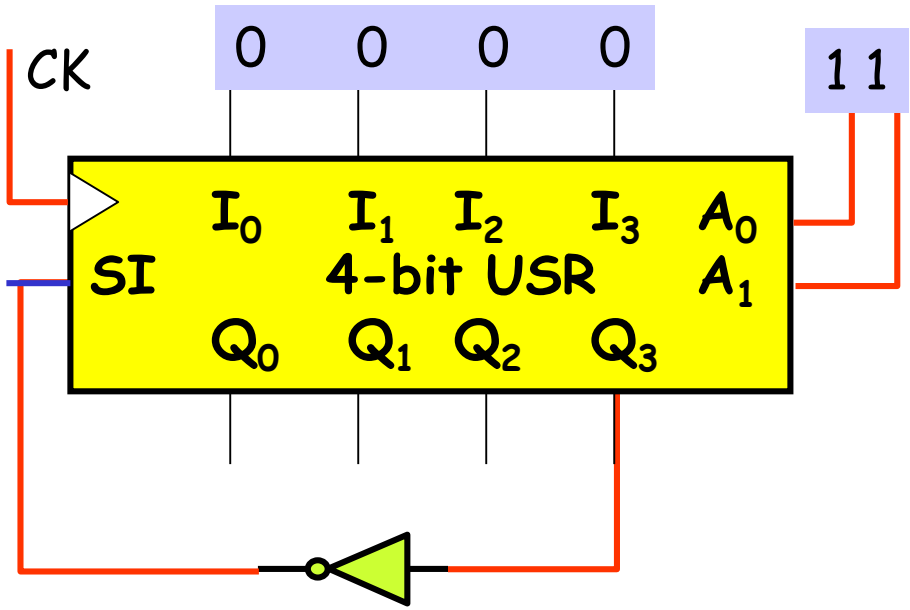


Esempio: contatore di Johnson

- Contatore di Johnson o a riempimento/svuotamento
- Realizza un ciclo di conteggio in base 8 (in base 2xN se utilizzo N FF)
- Fa scorrere «uni» fino a raggiungere la configurazione 1111, poi «zeri» fino a 0000
- Basta collegare a SI il complemento di SO (Q_3) e inizializzare il circuito con una delle 8 configurazioni lecite (es. 0000)

Base: 8
Codice: Johnson

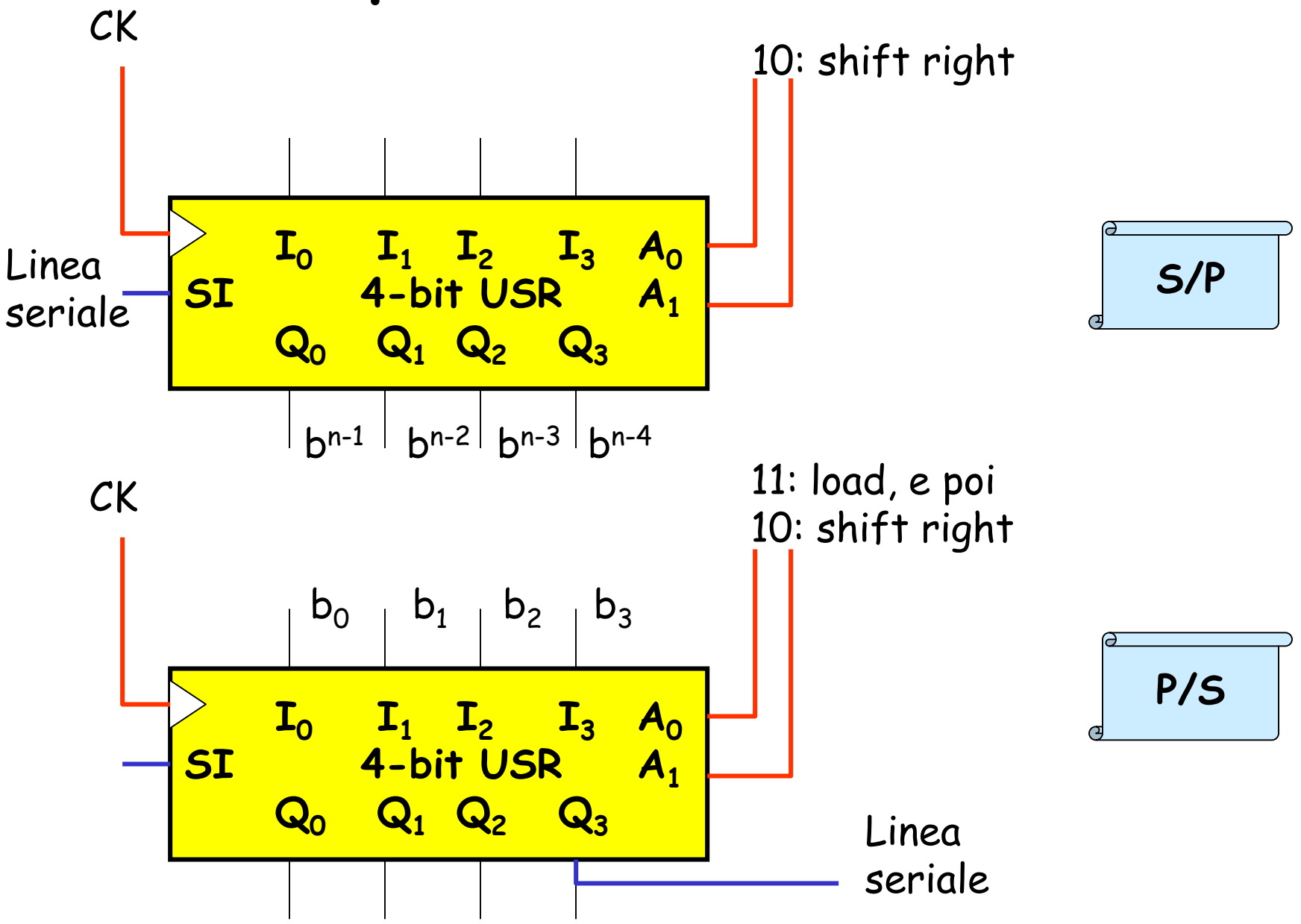
0000
1000
1100
1110
1111
0111
0011
0001



N flip-flop → Base 2 × N

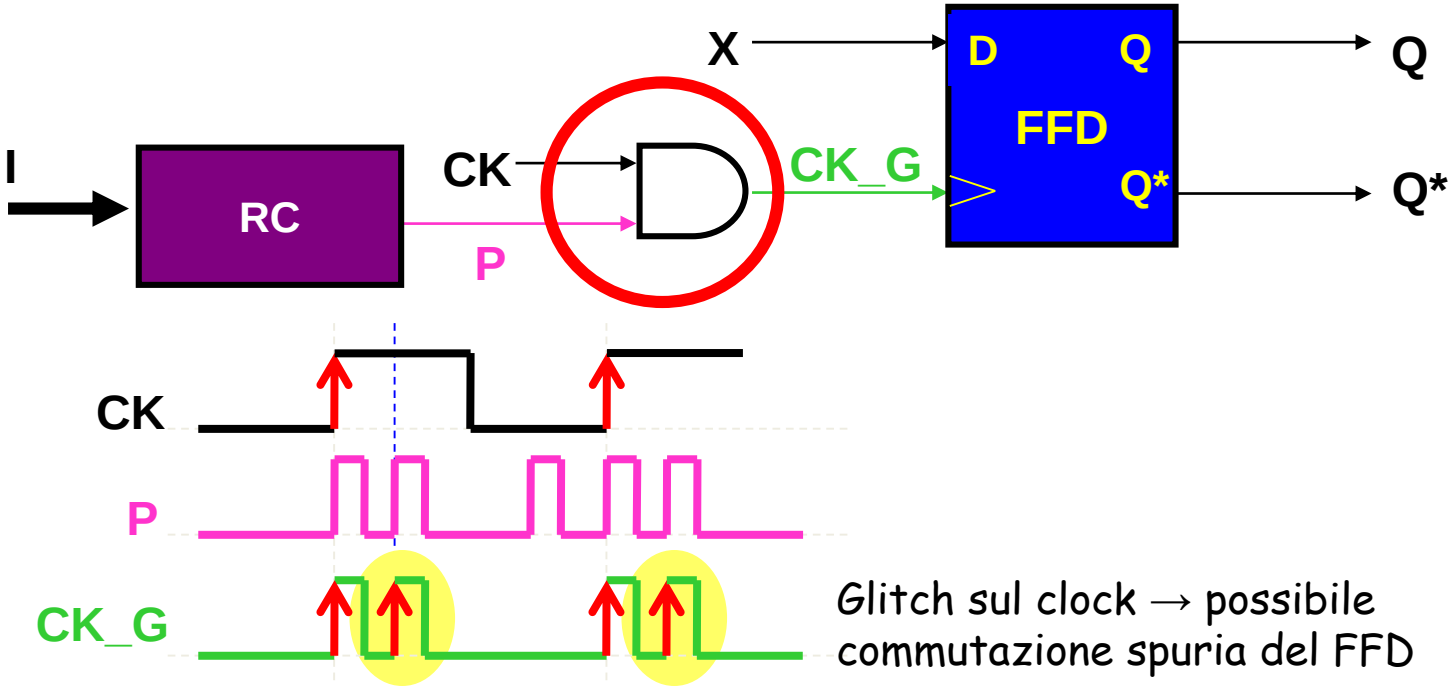
Q_0Q_1 \ Q_2Q_3	00	01	11	10
00	0	0	0	1
01	1	1	0	1
11	0	1	0	0
10	0	1	1	1

Esempio: conversioni S/P e P/S



Clock gating e glitch sul clock

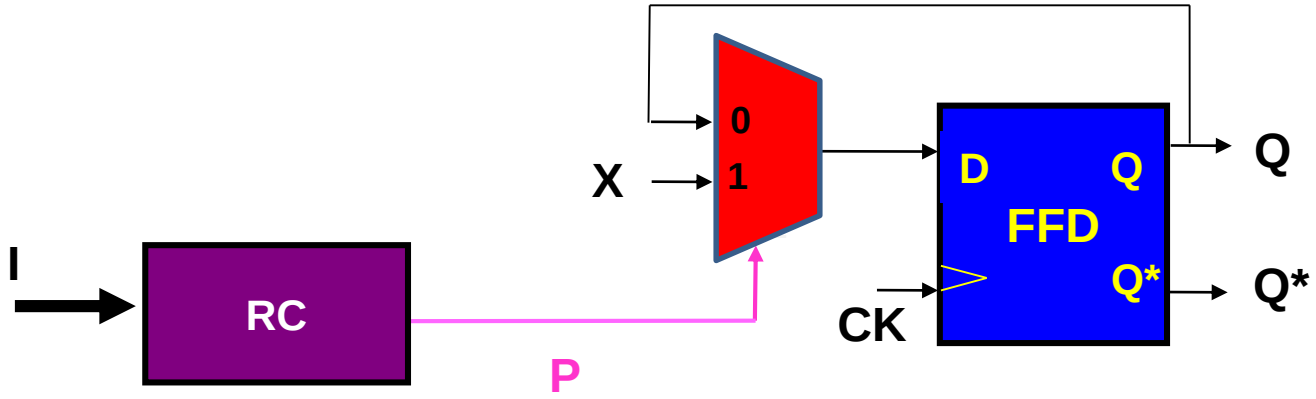
- Nelle reti sincrone è necessario evitare variazioni spurie *(glitches)* del segnale di clock che possono provocare commutazioni indesiderate dei FFD.
- Nell'esempio, P viene utilizzato per «fermare» il clock a seconda dell'elaborazione di una rete combinatoria e dei suoi ingressi I.
- In presenza di alee introdotte dalla rete combinatoria, a causa del "*clock gating*" possono verificarsi fronti di salita spuri del segnale CK_G



- La presenza di tali fronti indesiderati dipende dalla larghezza dell'impulso spurio: se molto stretto il FF potrebbe non sentirlo
- Il clock gating non è dunque proibito, ma da evitare in caso di incertezza.

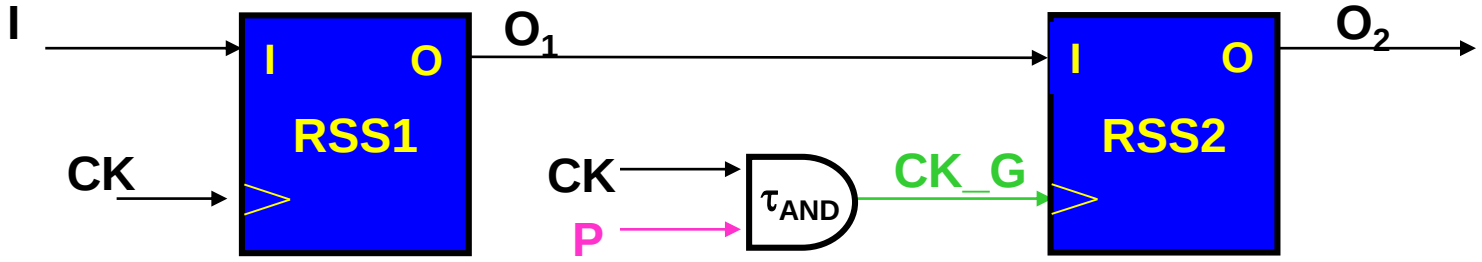
Clock gating e glitch sul clock

- Una soluzione alternativa al clock gating prevede l'utilizzo di un Mux a 2 vie in cui il segnale P (soggetto a glitch) viene utilizzato come segnale di selezione
- Il segnale d'uscita del Mux viene portato in ingresso al FF D, mentre l'altro ingresso del Mux (oltre a X) è rappresentato da Q
- Se la rete combinatoria genera il valore P=1, l'uscita Q assume (al successivo fronte di salita del clock) il valore di X, altrimenti l'uscita Q rimane invariata
- Eventuali glitch di P non in presenza di fronti di salita di CK non creano problemi alla rete

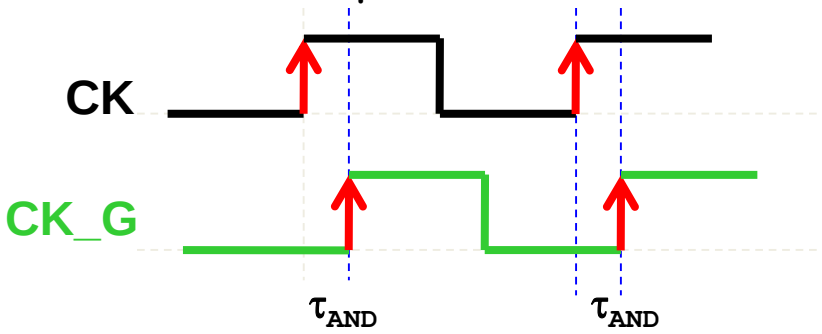


Clock gating e clock-skew

- Il clock gating, oltre a generare potenziali glitch, può portare anche al fenomeno del "*clock-skew*" (disallineamento).
- Consideriamo ad esempio due RSS «1» e «2» in cascata:



- I clock delle due reti (*CK* e *CK_G*) sono sfasati di un tempo pari al ritardo introdotto dalla RC utilizzata per il clock gating (nell'esempio, un AND).
- Il clock-skew è potenzialmente dannoso in quanto la RSS «2» potrebbe campionare *in presenza dello stesso fronte di clock* i nuovi valori prodotti dalla RSS «1» (potenzialmente ancora in regime di transitorio)



Comportamento desiderato: $O_2^n = f(O_1^{n-1})$
Comportamento effettivo: $O_2^n = f(O_1^n)$

- Il "clock-skew" può essere generato, oltre che da clock gating, anche da percorsi elettrici di lunghezza diversa (con differenza significativa rispetto alla frequenza di clock)