Corso di Calcolatori Elettronici I

Analisi e Progetto di Macchine Sequenziali

ing. Alessandro Cilardo

Corso di Laurea in Ingegneria Biomedica

Macchine sequenziali

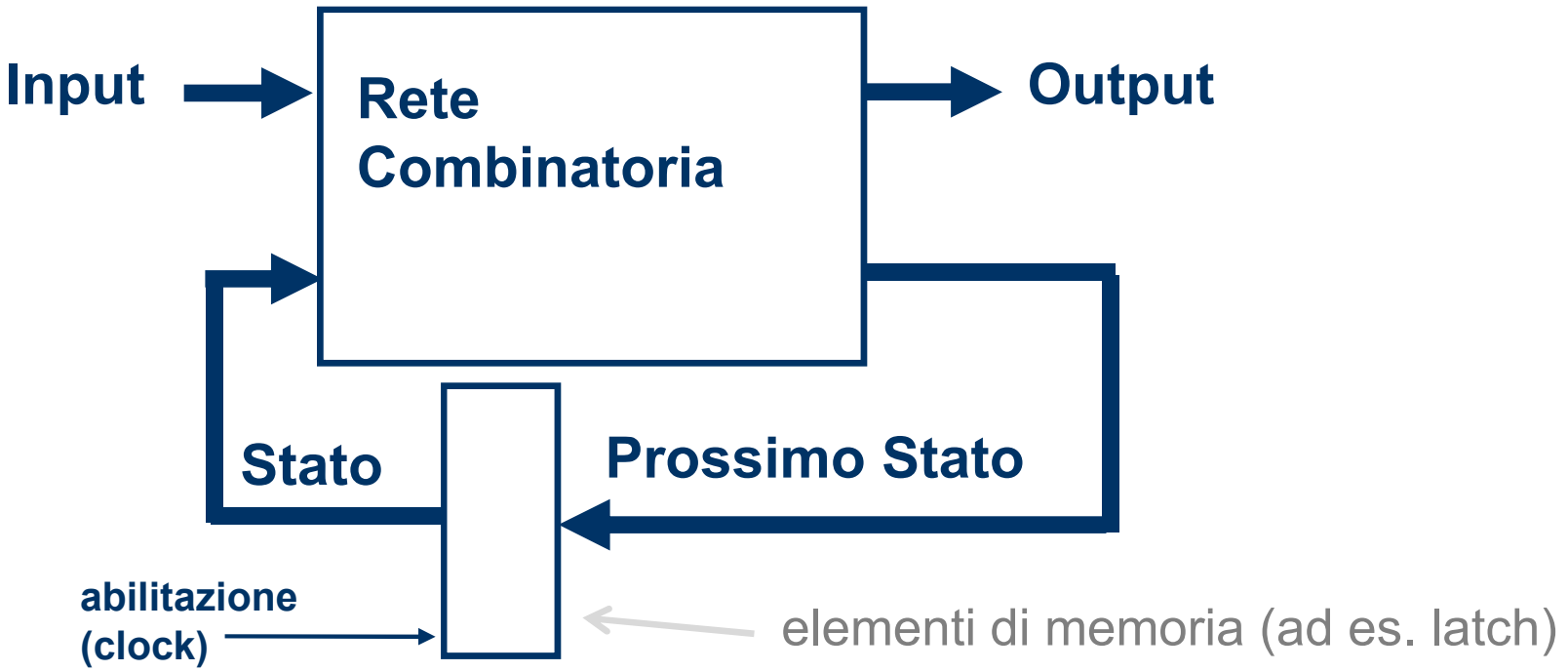
- In molte situazioni è necessario progettare reti logiche **sequenziali**, ovvero *dotate di memoria*:
  - l'uscita in un dato istante non è funzione soltanto del valore degli ingressi applicati in quell'istante, ma anche di tutti i valori applicati negli istanti precendenti

- Dobbiamo individuare una metodologia per comporre reti combinatorie e circuiti elementari dotati di memoria allo scopo di realizzare il comportamento di reti sequenziali complesse

Elementi di memoria nelle macchine sequenziali

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

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



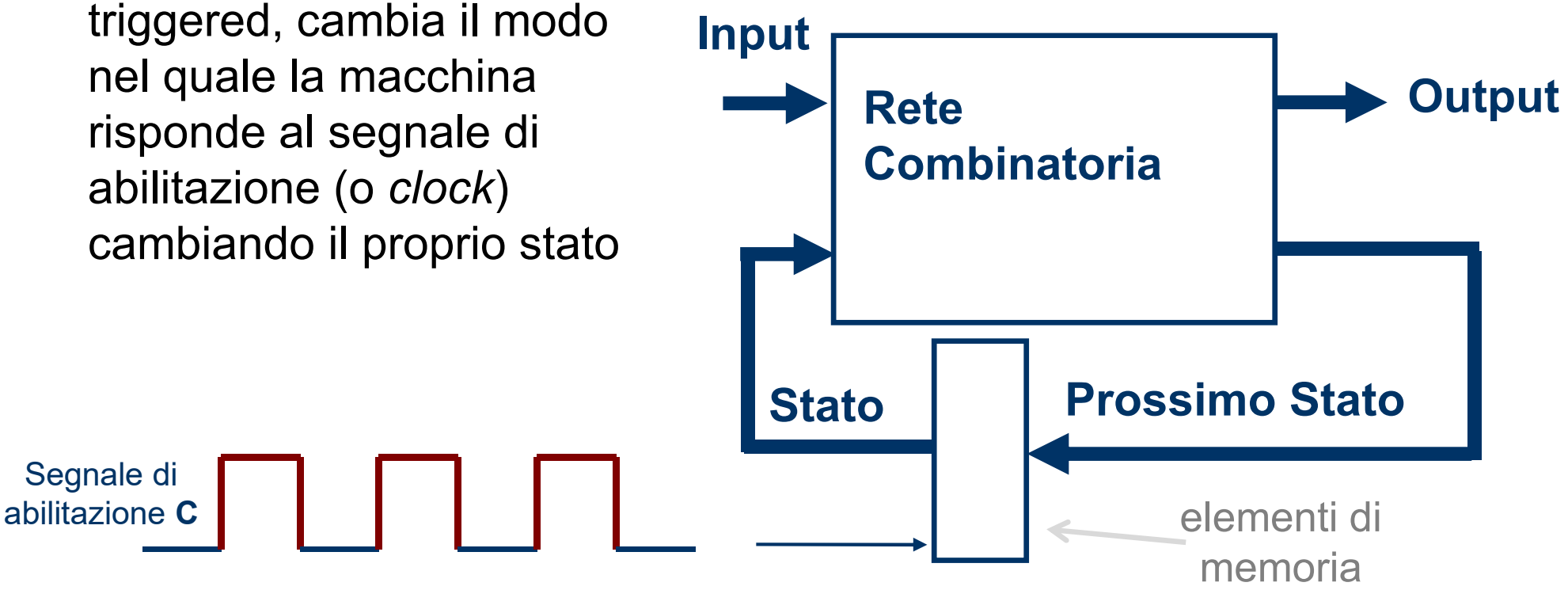
Elementi di memoria: Riepilogo

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

  (ognuno ha la propria tabella funzionale/di eccitazione)

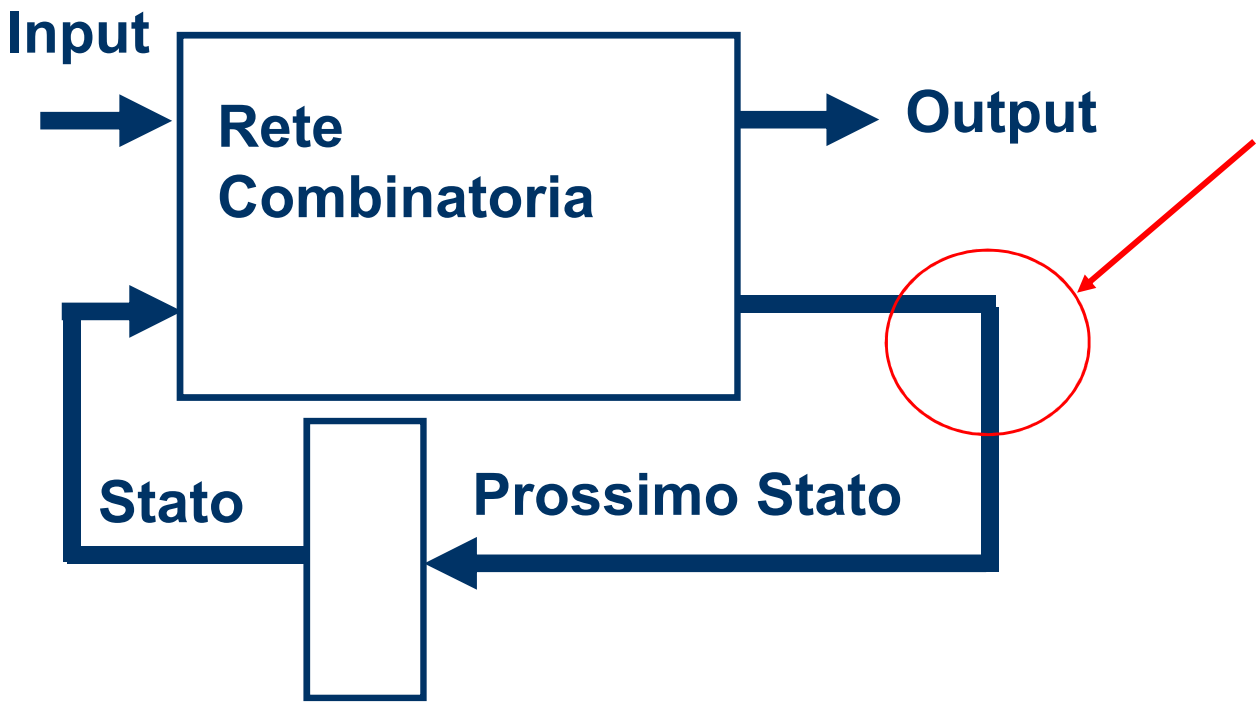
Tempificazione degli elementi di memoria

- La scelta del tipo di tempificazione influenza il progetto della macchina sequenziale
- A seconda che gli elementi di memora siano latch o flip-flop master-slave o edge-triggered, cambia il modo nel quale la macchina risponde al segnale di abilitazione (o *clock*) cambiando il proprio stato



Tipo di ingressi degli elementi di memoria

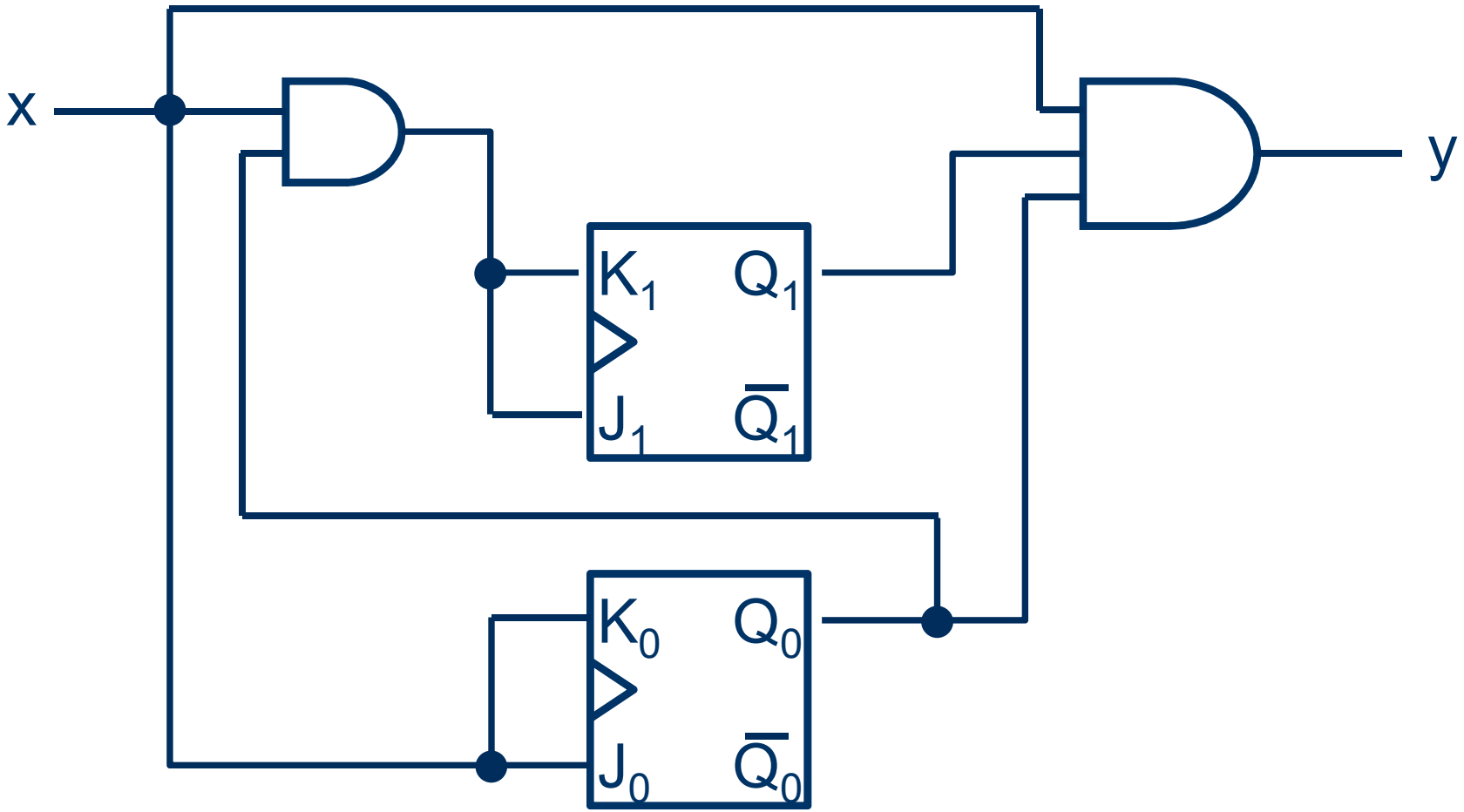
- Inoltre, in base al tipo di elemento usato, cambiano i segnali che la rete combinatoria deve generare (detti *segnali di posizionamento*) per determinare la transizione ad un nuovo stato prossimo



- segnali **R** ed **S** ?
- segnale **D** ?
- segnali **K** e **J** ?
- segnale **T** ?

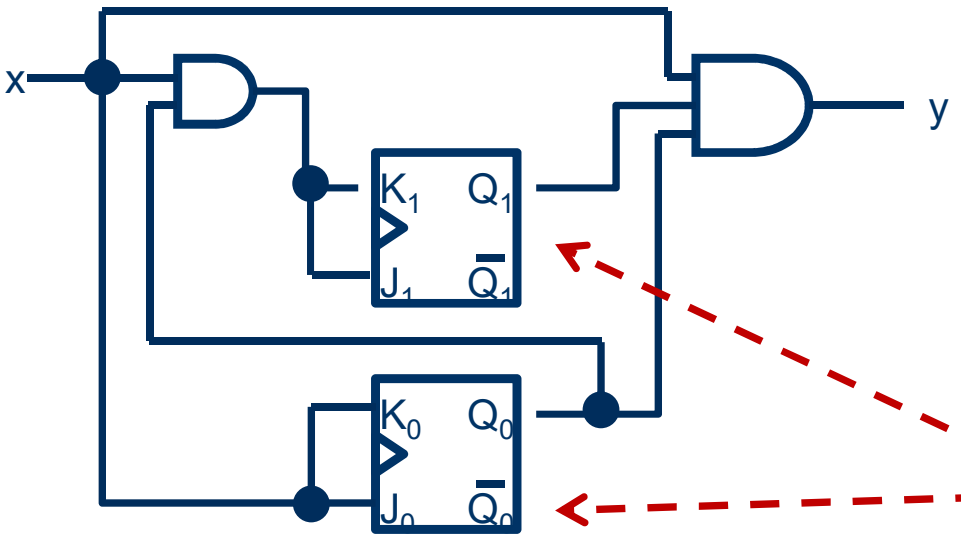
Per ogni elemento di memoria utilizzato, gli ingressi da generare cambiano in base al tipo di elemento scelto

Esempio di analisi di una rete sequenziale



Esempio di analisi di una rete sequenziale

- Teniamo presente la tabella funzionale del flip-flop JK

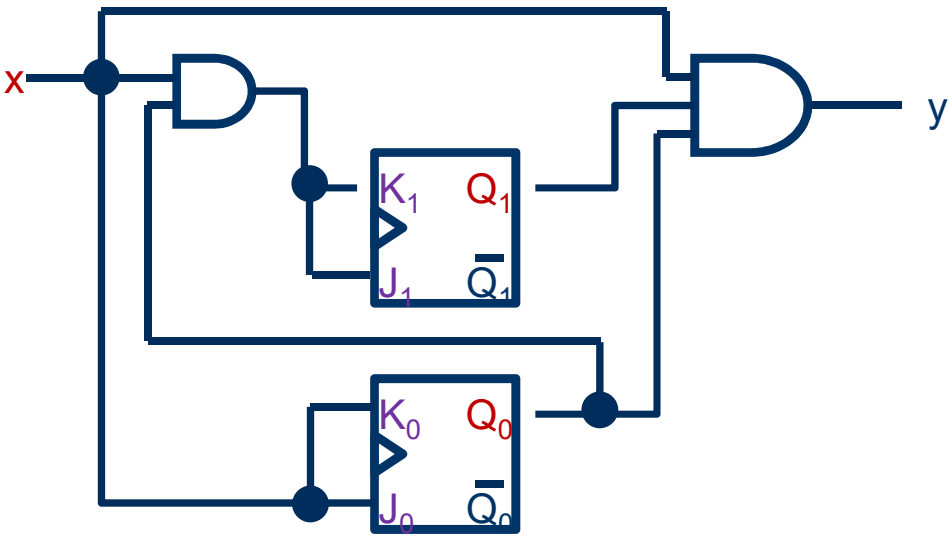


Q(t)	J	K	Q(t+1)	Commento
0	0	0	0	inalterato
0	0	1	0	resetta Q
0	1	0	1	setta Q
0	1	1	1	*toggle* (inverte Q)
1	0	0	1	inalterato
1	0	1	0	resetta Q
1	1	0	1	Setta Q
1	1	1	0	*toggle* (inverte Q)

tabella funzionale del flip-flop JK

Esempio di analisi di una rete sequenziale

- Per ogni possibile combinazione di ingresso **x** e stato corrente $\mathbf{Q_1}$, $\mathbf{Q_0}$, deriviamo i valori dei segnali *di posizionamento* $\mathbf{K_1}$,$\mathbf{J_1}$, $\mathbf{K_0}$,$\mathbf{J_0}$ e quindi dello stato prossimo $\mathbf{Q_1}$', $\mathbf{Q_0}$'

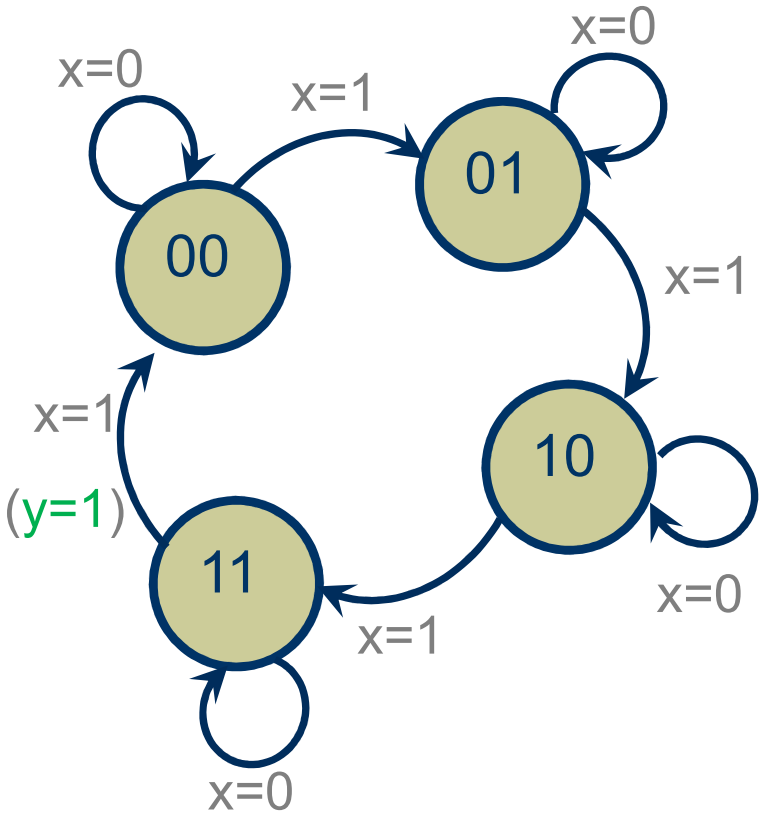


prossimo valore dello stato

x	Q_1	Q_0	K_1	J_1	K_0	J_0	Q_1'	Q_0'	y
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0
0	1	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	1	0
1	0	0	0	0	1	1	0	1	0
1	0	1	1	1	1	1	1	0	0
1	1	0	0	0	1	1	1	1	0
1	1	1	1	1	1	1	0	0	1

Esempio di analisi di una rete sequenziale

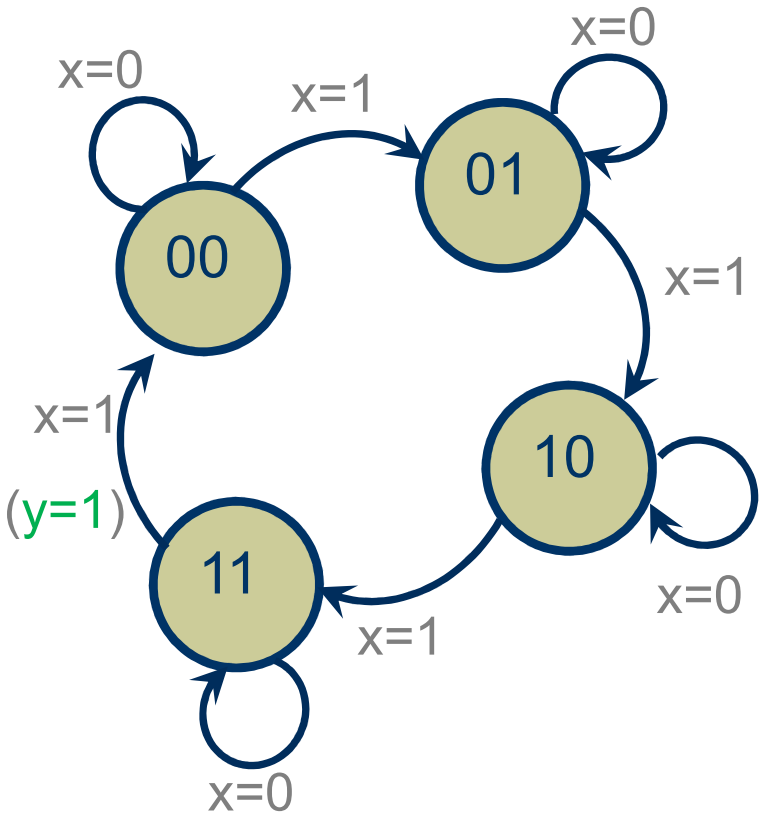
- Per ogni possibile combinazione di ingresso **x** e stato corrente $\mathbf{Q_1}$, $\mathbf{Q_0}$, deriviamo i valori dei segnali *di posizionamento* $\mathbf{K_1}$,$\mathbf{J_1}$, $\mathbf{K_0}$,$\mathbf{J_0}$ e quindi dello stato prossimo $\mathbf{Q_1'}$, $\mathbf{Q_0'}$



prossimo valore dello stato

x	Q_1	Q_0	K_1	J_1	K_0	J_0	Q_1'	Q_0'	y
0	**0**	**0**	0	0	0	0	**0**	**0**	0
0	**0**	**1**	0	0	0	0	**0**	**1**	0
0	**1**	**0**	0	0	0	0	**1**	**0**	0
0	**1**	**1**	0	0	0	0	**1**	**1**	0
1	**0**	**0**	0	0	1	1	**0**	**1**	0
1	**0**	**1**	1	1	1	1	**1**	**0**	0
1	**1**	**0**	0	0	1	1	**1**	**1**	0
1	**1**	**1**	1	1	1	1	**0**	**0**	1

Esempio di analisi di una rete sequenziale



Grafo di transizione di stato

- **Nodi**:

  tutti i possibili stati in cui può trovarsi la macchina sequenziale
- **Archi**:

  tutte le possibili transizioni da uno stato all'altro, causate da ciascun valore dell'ingresso applicato quando la macchina si trova in ciascun differente stato
- Se le uscite dipendono da stato e ingresso, gli archi riporteranno anche i valori delle corrispondenti uscite
- Le uscite possono dipendere solo dallo stato (si indicano allora all'interno del nodo)

Modello Formale: Macchine a Stati Finiti

- Il grafo e la tabella mostrati prima costituiscono due modi per rappresentare una *Macchina* (o *Automa*) *a Stati Finiti*, o *Finite State Machine* (**FSM**)
- Le **FSM** sono un modello formale per la rappresentazione di sistemi sequenziali estremamente utile e comunemente impiegato in molti ambiti dell'ingegneria dell'informazione
- Due varianti
  - macchine di Mealy
  - macchine di Moore

Modello Formale: Macchine a Stati Finiti

Definizione di *Macchina di Mealy:*

Una macchina sequenziale (a stati finiti) è una quintupla ordinata di enti M(Q,I,U,t,w), dove:

- Q è un insieme finito di “Stati Interni”
- I è un insieme finito di “Stati di Ingresso”
- U è un insieme finito di “Stati di Uscita”
- t è una funzione $Q \times I \rightarrow Q$
  - funzione stato prossimo
- w è una funzione $Q \times I \rightarrow U$
  - funzione di uscita: dipende dallo stato e dall’ingresso applicato alla macchina

Modello Formale: Macchine a Stati Finiti

Definizione di *Macchina di Moore:*

Una macchina sequenziale (a stati finiti) è una quintupla ordinata di enti M(Q,I,U,t,w), dove:

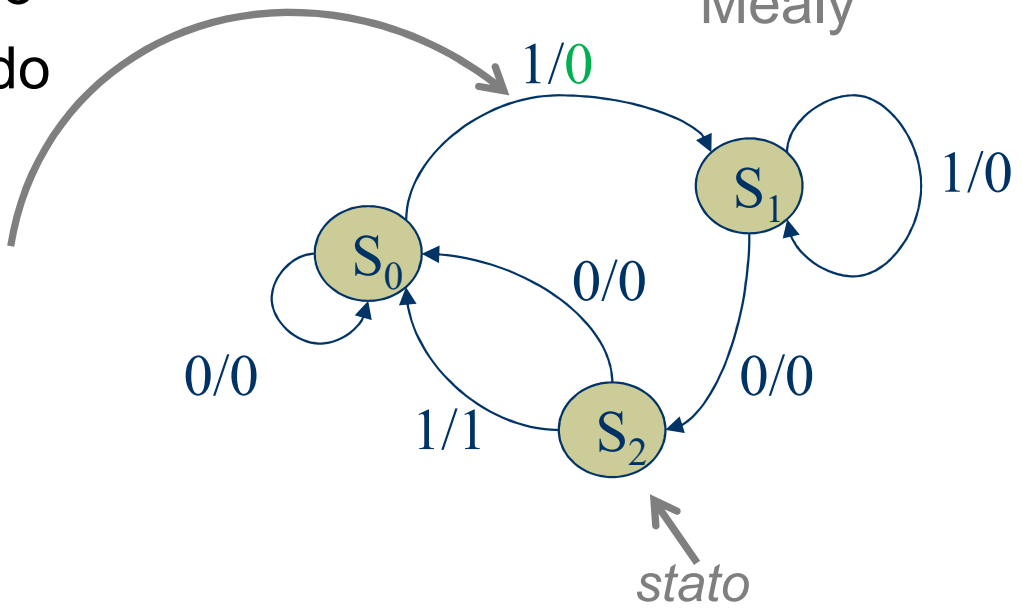
- Q è un insieme finito di “Stati Interni”
- I è un insieme finito di “Stati di Ingresso”
- U è un insieme finito di “Stati di Uscita”
- t è una funzione $Q \times I \rightarrow Q$
  - funzione stato prossimo
- w è una funzione $Q \rightarrow U$
  - funzione di uscita: dipende solo dallo stato e non dall’ingresso applicato alla macchina

Descrizione della macchina

- Il comportamento della macchina può essere descritto come una tabella o come un grafo:
- Tabella Stato/Ingresso → prossimo stato
- Grafo Passaggi di stato
  - Mealy: uscita associata all'arco
  - Moore: uscita associata al nodo

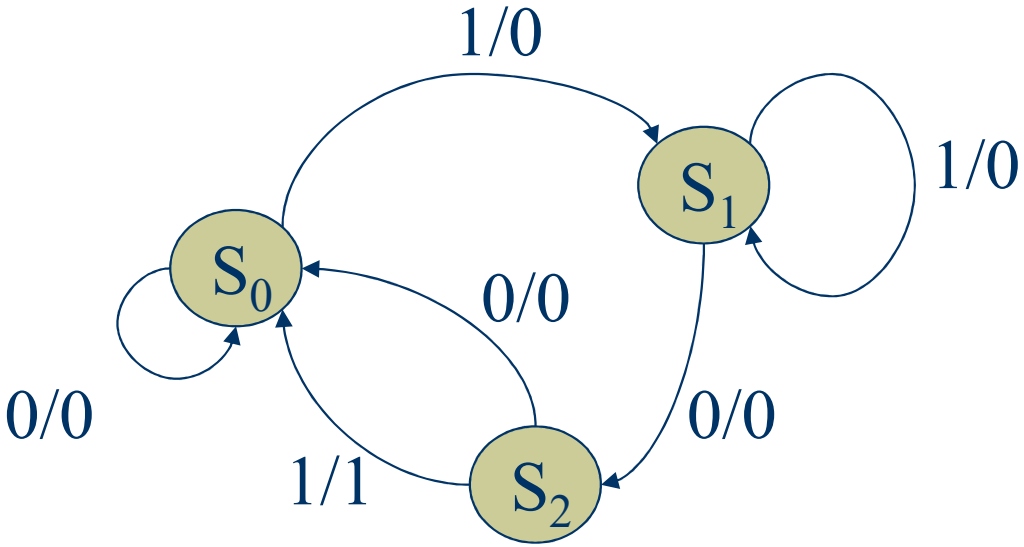
transizione di stato: **1**/**0** sull'arco indica che la transizione sa S_0 a S_1 avviene quando si applica l'ingresso **1**, producendo nel contempo l'uscita **0**. Nelle macchine di Moore l'uscita dipende solo dallo stato corrente e si indica pertanto direttamente all'interno dei nodi dl grafo

Esempio di Macchina di Mealy



Esempio

Automa in grado di riconoscere la sequenza 101



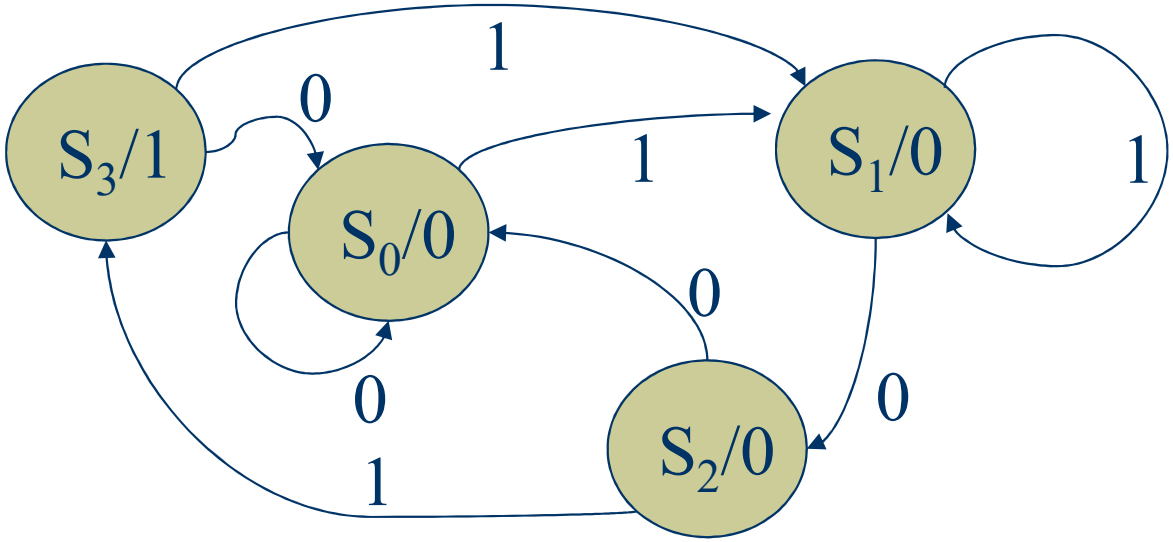
	0	1
S_0	$S_0/0$	$S_1/0$
S_1	$S_2/0$	$S_1/0$
S_2	$S_0/0$	$S_0/1$

E’ una Macchina di Mealy

Le uscite dipendono non solo dallo stato corrente ma anche dall’ingresso applicato alla macchina

Esempio

Automa in grado di riconoscere la sequenza 101



	0	1
$S_0/0$	S_0	S_1
$S_1/0$	S_2	S_1
$S_2/0$	S_0	S_3
$S_3/1$	S_0	S_1

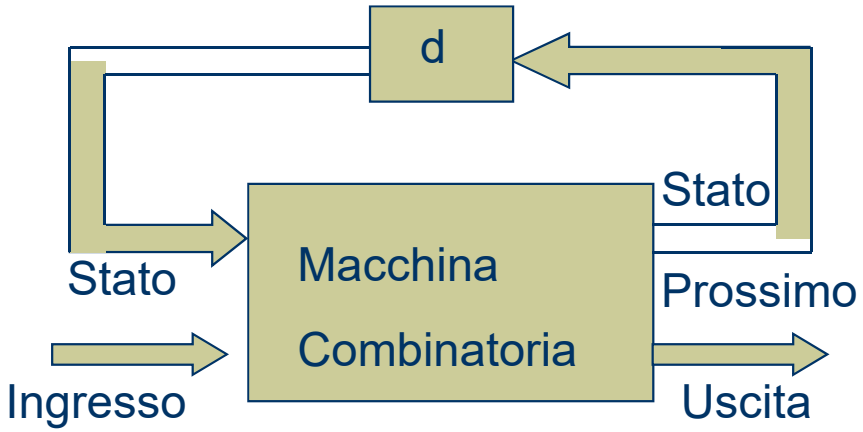
E' una Macchina di Moore

Le uscite dipendono solo dallo stato corrente e non dall'ingresso applicato alla macchina. (Si noti che per ottenere un comportamento equivalente alla macchina di Mealy abbiamo dovuto aggiungere uno stato)

Modello realizzativo generale

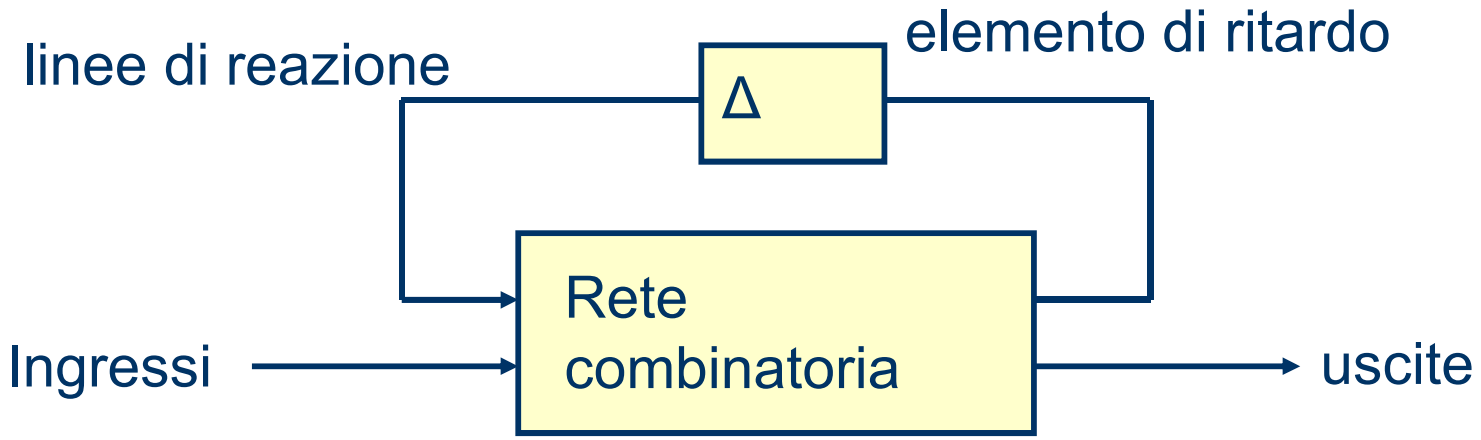
Modello di Huffman: Una Macchina Sequenziale può essere realizzata con:

- Una Macchina Combinatoria
  - riceve l'ingresso esterno e lo stato corrente
  - calcola l'uscita esterna e lo stato prossimo
- Un Elemento di Ritardo
  - ritarda fino al prossimo passo la transizione dallo stato corrente a quello prossimo



Progetto di reti sequenziali: Modello generale

- Nel modello generale astratto il funzionamento dell'elemento di ritardo non è specificato
- Il modello non ha ancora una diretta corrispondenza fisica



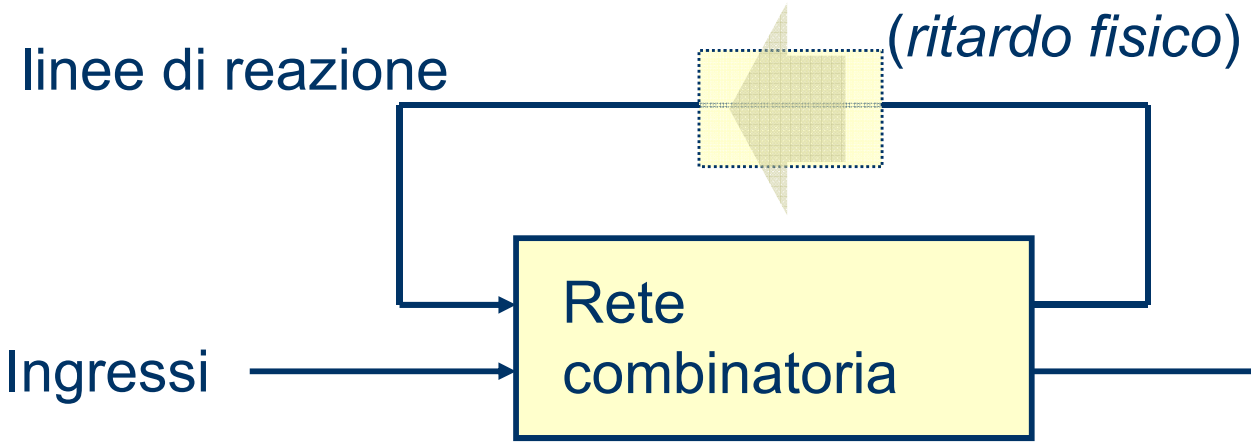
Progetto di reti sequenziali

Sulla base della tipologia di elemento di ritardo utilizzato è possibile tracciare la seguente classificazione:

- Reti asincrone
- Reti sincrone
  - ad ingressi impulsivi (autosincronizzate)
  - a sincronizzazione esterna

Macchine asincrone

- Nella macchine **asincrone**, l'elemento di ritardo è trasparente all'ingresso, che in uscita ricompare semplicemente traslato in avanti nel tempo
- L'elemento può essere realizzato anche come semplice ritardo di propagazione su un collegamento diretto tra stato prossimo e stato corrente
  - non occorrono Latch né Flip-Flop



Lo stato prossimo si ripresenta direttamente in ingresso alla rete combinatoria attraverso la linea di reazione. La sua propagazione lungo la linea di reazione non è interrotta, ma semplicemente ritardata

Macchine asincrone: Problemi di tempificazione

- Occorre evitare che il nuovo stato, propagato alla macchina combinatoria, causi una nuova transizione di stato mentre l'ingresso precedente è ancora applicato
- Il corretto funzionamento della macchina asincrona impone il controllo preciso della durata degli ingressi
  - normalmente non è possibile controllare con precisione la durata dei segnali nella realizzazione fisica dei circuiti
- Questo vincolo si può rilassare (rendendo fisicamente realizzabile la macchina asincrona) se la macchina a stati finiti rispetta alcune proprietà

Macchine asincrone

- Una delle condizioni necessarie per realizzare una macchina asincrona, è che la tabella deve evolvere verso uno **stato stabile** per qualsiasi ingresso applicato costantemente

	i_1	i_2	i_3
q1	q1	q2	q3
q2	q4	q2	q3
q3	q1	q4	q3
q4	q4	q4	q3

uno stato ***q*** è stabile sotto un determinato ingresso ***i*** se, applicando l'ingresso ***i*** nello stato ***q***, la macchina permane in maniera indefinita in ***q***

Si indicano nella tabella con un cerchio (vedi esempi a sinistra)

Macchine asincrone

Materiale facoltativo

- Un'altra condizione necessaria per realizzare una macchina asincrona, è che la tabella non deve contenere **alee essenziali**

	i_1	i_2	i_3
q1	q1	q2	q3
q2	q4	q2	q3
q3	q1	q4	q3
q4	q4	q4	q3

$$\tau(q_1, i_1) = q_1$$
$$\tau(q_1, i_2) = q_2$$
$$\tau(q_2, i_1) = q_4$$
$$\tau(q_4, i_2) = q_4$$

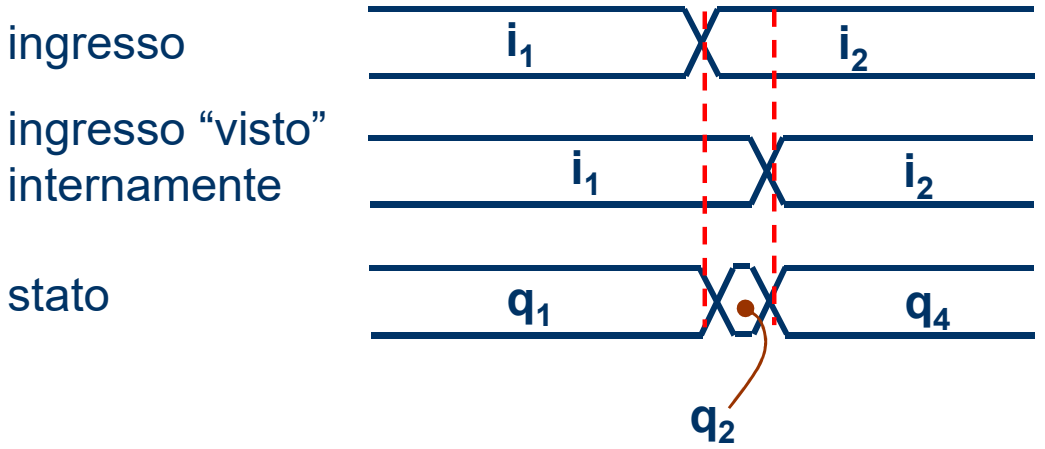
Rappresenta una situazione in cui, a causa dei diversi ritardi presenti nel circuito, una parte della rete "vede" il nuovo stato (ad esempio $\mathbf{q_2}$) in presenza dell'ingresso precedente (ad es. $\mathbf{i_1}$), causando una transizione verso uno stato prossimo errato (ad es. $\mathbf{q_4}$)

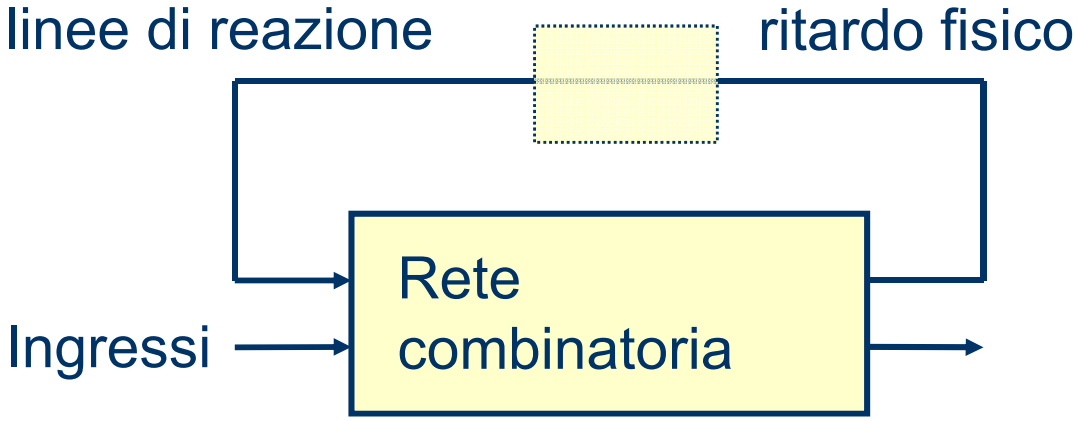
Macchine asincrone

Materiale facoltativo

Esempio di *alea essenziale*

	i_1	i_2	i_3
q1	q1	q2	q3
q2	q4	q2	q3
q3	q1	q4	q3
q4	q4	q4	q3





Progetto di reti asincrone

Materiale facoltativo

- Alee introdotte dal progetto combinatorio
  - alee statiche
  - alee dinamiche
- Alee intrinseche nella specifica della macchina
  - alee essenziali
- Codifica degli stati
  - corse critiche

Progetto di reti asincrone

Materiale facoltativo

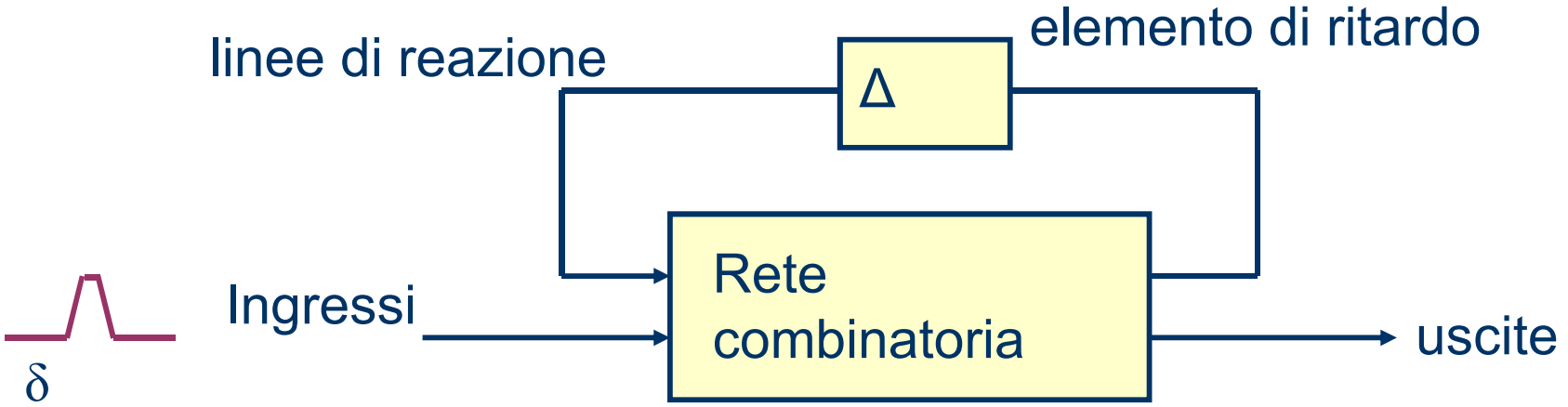
- definizione della macchina (tabella o grafo) a partire da una specifica informale
  - eventualmente ridondante (sarà poi minimizzata)
  - ogni stato è dotato di un significato ben definito
- minimizzazione della macchina
  - la tabella deve essere priva di alee essenziali
- codifica di stati ed ingressi
  - non più di 2N+1 linee per evitare transizioni non adiacenti
- progetto della rete combinatoria
  - stati e uscite: evitare alee sulle linee di uscita
- definizione delle linee di reazione
  - non serve ritardo in assenza di alee essenziali

Reti sincrone

- Il progetto delle reti asincrone deve tenere in conto numerosi fattori ed è spesso estremamente complesso
- Nella maggior parte dei casi si ricorre pertanto al progetto di **reti sincrone**
- Nelle reti sincrone il ritardo è realizzato attraverso un elemento di memoria (latch o flip-flop) che *interrompe la propagazione dello stato prossimo* lungo la linea di reazione
  - L'istante in cui avviene la transizione di stato può essere controllato attraverso i segnali che pilotano gli elementi di memoria
  - questo permette di risolvere buona parte dei problemi di tempificazione

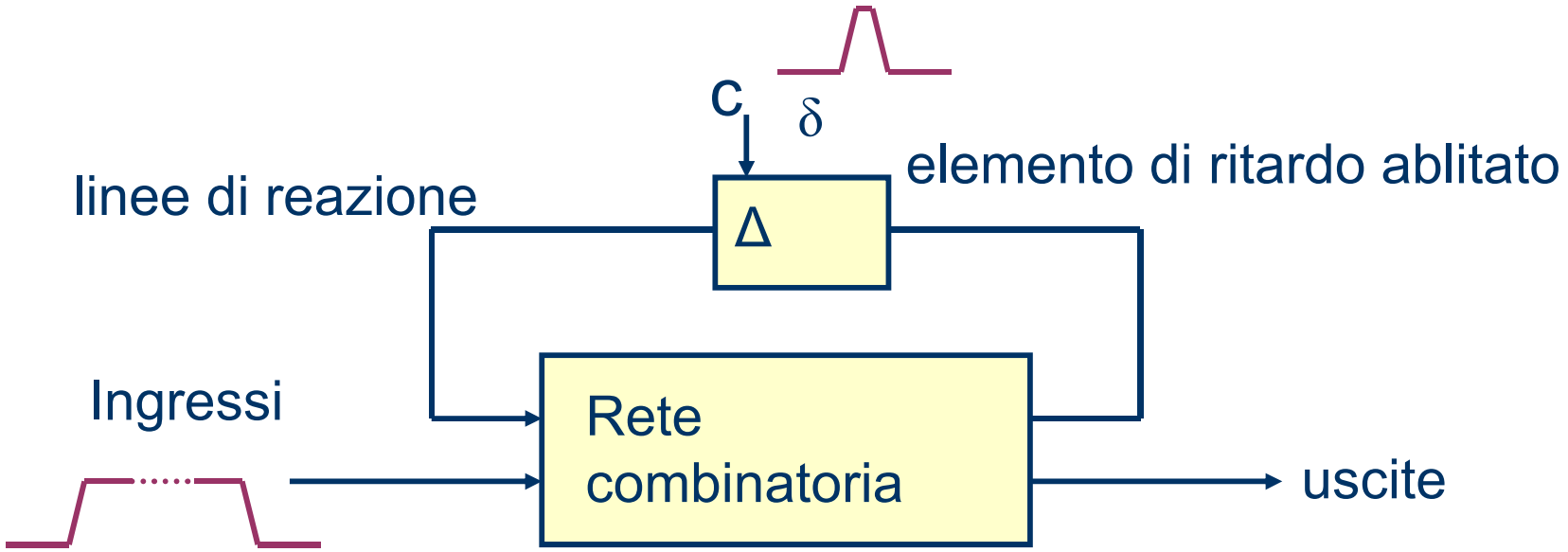
Reti sincrone autosincronizzate

- L'elemento di memoria è tipicamente un Latch
- Gli ingressi sono *impulsivi*
  - assumo la configurazione che innesca il cambiamento di stato solo per il breve tempo δ richiesto dalla transizione
  - In corrispondenza dell'ingresso impulsivo, la rete combinatoria genera gli opportuni segnali (ad es. **R** ed **S**) necessari per pilotare gli elementi di ritardo



Reti sincrone a sincronizzazione esterna

- usa dei flip-flop edge-triggered
- I momenti di transizione di stato sono dettati dai fronti del segnale di clock
- Tra un fronte e l'altro, la rete combinatoria può anche produrre uscite temporaneamente non corrette: queste non avranno effetto sui flip-flop



Progetto di reti sincrone

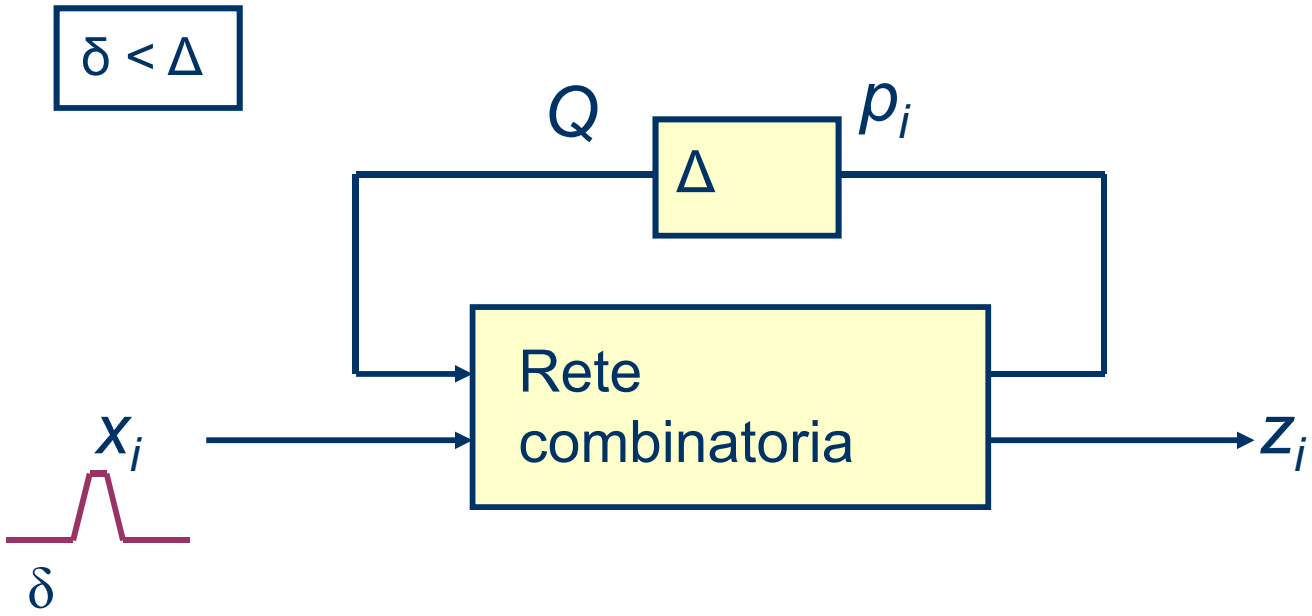
- Elementi di ritardo realizzati come latch o flip-flop:
  - l'unico vincolo di progetto è che la durata δ dell'impulso che abilita la transizione sia inferiore al ritardo Δ con cui gli elementi di memoria registrano il nuovo stato
  - nel caso di reti autosicronizzate (che usano latch) δ è la durata dell'impulso applicato sugli ingressi
    - ocorre ancora controllare la durata dei segnali, sebbene in questo caso sia più facile garantire il rispetto del vincolo
  - nel caso di reti a sincronizzazione esterna (che usano flip-flop edge-triggered), l'impulso corrisponde ad un fronte del clock, quindi δ è pressocché istantaneo

Reti autosicronizzate

Ingressi impulsivi di durata δ

Stati *Q* a "livelli" (stabili tra un impulso e l'altro)

Funzioni di posizionamento p_i impulsive



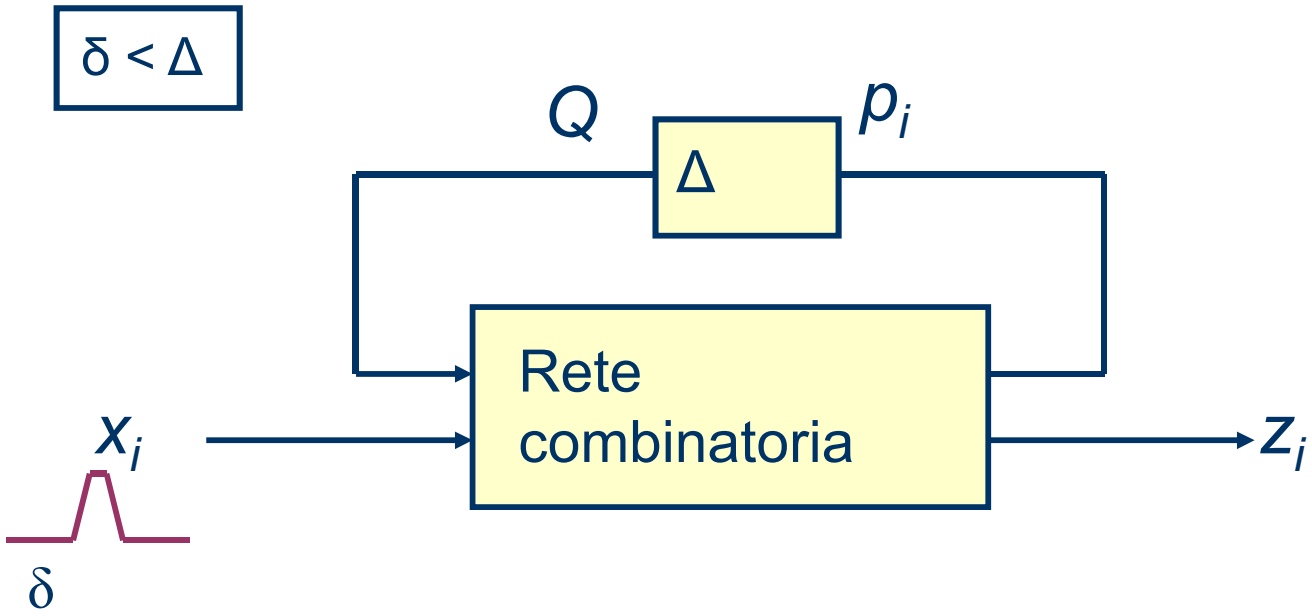
$$p_i = \sum x_j f_{ij}(Y)$$

Reti autosicronizzate

Le uscite possono essere:

- a livelli (se funzione solo dello stato)
- o impulsive (funzione di stato e ingressi)

$$\begin{cases} z_i = z_i(Y) \\ \text{oppure} \\ z_i = \sum x_j z_{ij}(Y) \end{cases}$$



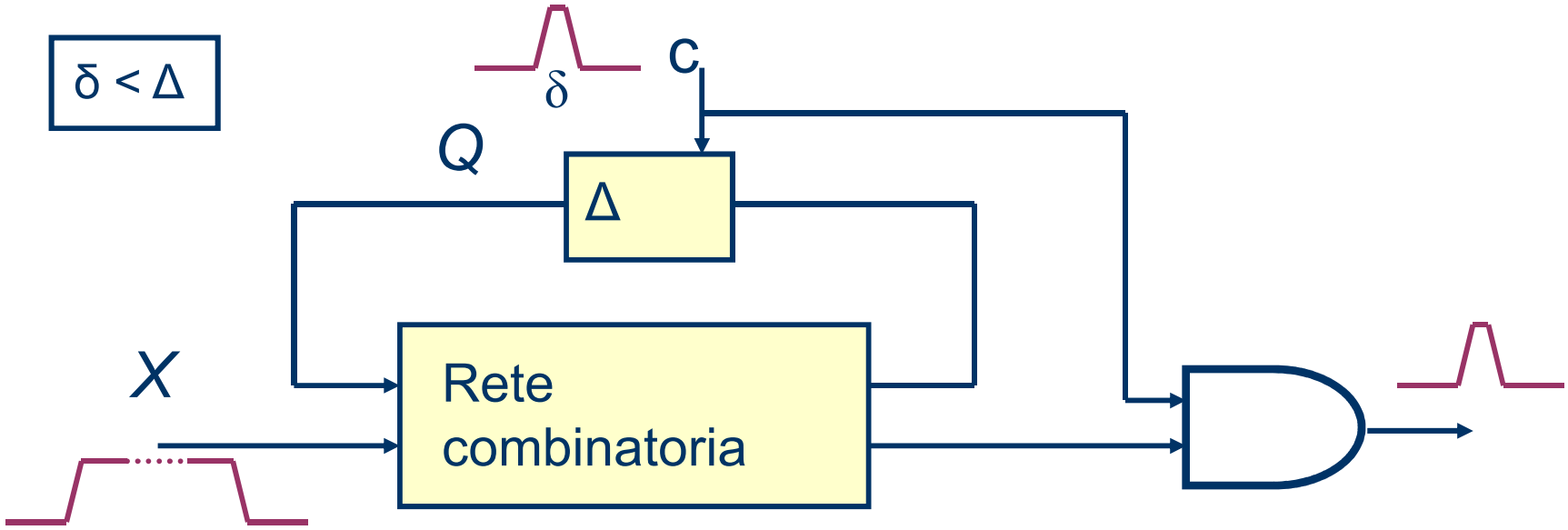
Reti a sincronizzazione esterna

Ingressi, stati, funzioni di posizionamento a livelli
(stabili tra un fronte e l'altro del clock)

Le uscite possono essere:

- a livelli (funzione di stato e ingressi)
- o impulsive (anche abilitate dal clock)

$$\begin{cases} z_i = z_i(X,Y) \\ z_i = c \cdot z_i(X,Y) \end{cases}$$



Progetto di reti sincrone

1. ricavare il **diagramma/tabella di stato** a partire dalla descrizione del problema
   - non servono necessariamente stati stabili
2. **minimizzare gli stati**
3. **codificare gli stati**
   - non occorre considerare corse critiche
4. ricavare le **equazioni di ingresso** (o **posizionamento**) dei latch/flip-flop a partire da stato corrente e ingressi
   - questo passo dipende dai flip-flop usati (uso delle tabelle di eccitazione)
5. ricavare le **equazioni di uscita**
6. **minimizzare le espressioni delle equazioni**
   - la tempificazione sincrona evita problemi legati alle alee

   non occorre controllare i ritardi delle linee di reazione

Progetto di reti sincrone

- Per le reti a **sincronizzazione esterna**:

  esiste un unico ingresso impulsivo, definito dai fronti del clock
  - è implicito nella definizione della macchina
  - agisce solo sugli elementi con memoria e, al più, sulle uscite
- Per le reti **autosincronizzate**:

  gli ingressi sono tutti impulsivi
  - mai attivi insieme (altrimenti: alee impulsive!)
  - si può sfruttare la relazione

    $$p_i = \sum x_j f_{ij}(Y)$$

    per il progetto
  - si possono progettare le f_{ij} separatamene

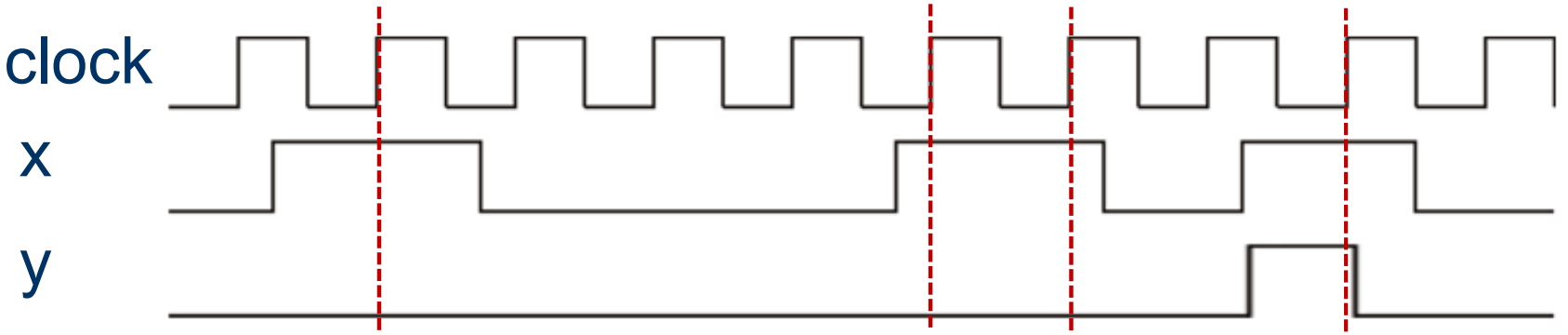
Esempio di progetto (o *sintesi*)

- Immaginiamo di voler progettare una macchina sequenziale, specificata attraverso il grafo già visto nell'esercizio di analisi
- Questa volta il grafo (o equivalentemente la tabella) **rappresenta l'ingresso** del problema da risolvere, e non la soluzione come invece nell'esercizio di analisi della rete

Esempio di progetto (o *sintesi*)

- Traccia: *progettare, come macchina sequenziale a sincronizzazione esterna, un divisore per* **4**, *ovvero una macchina che riceve un segnale di ingresso* ***x*** *e produce un'uscita* ***y*** *tale che, ogni* **4** *letture del valore* **1** *sull'ingresso* ***x*** *da parte della macchina, l'uscita* ***y*** *deve diventare alta*

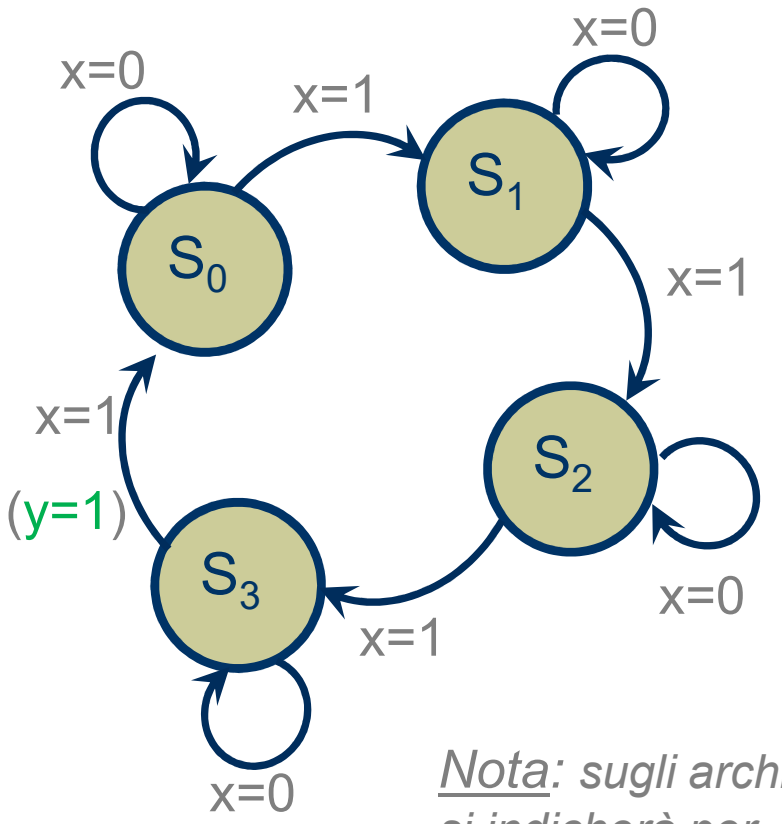
  (Non è richiesto l'uso di uno specifico tipo di flip-flop)



esempio di funzionamento (con elementi di memoria edge-triggered su fronte di salita)

Esempio – Definizione del grafo

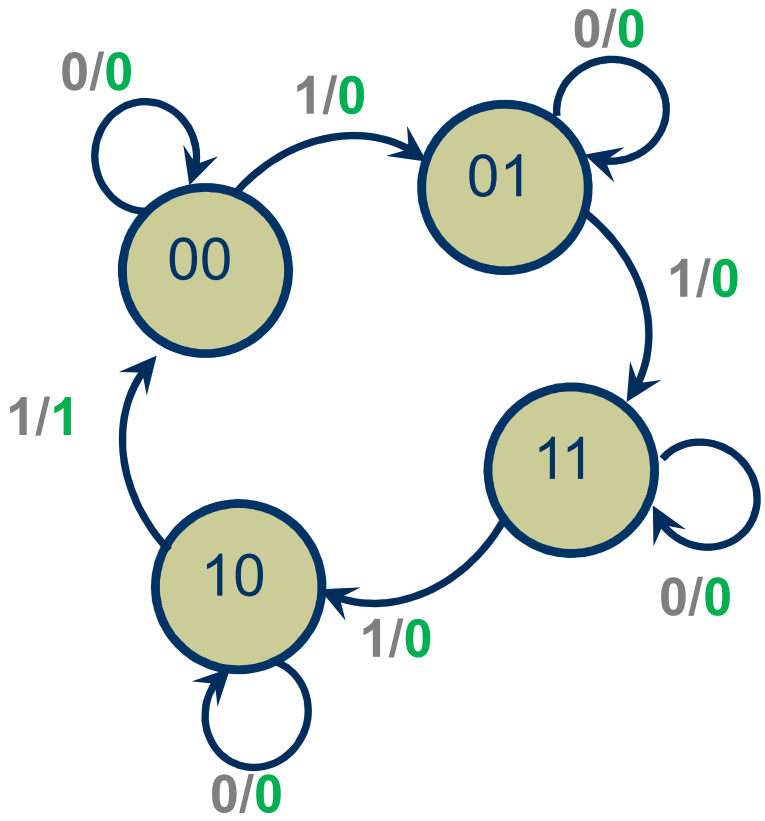
- Definizione del grafo a partire dalla specifica
  - partendo da uno stato iniziale $\mathbf{S_0}$, aggiungiamo un nuovo stato per ogni diversa circostanza in cui la macchina potrà trovarsi
  - per ogni stato, chiediamoci cosa accade quando si presenta ciascuno dei diversi valori possibili in ingresso
  - se necessario, aggiungiamo quindi nuovi stati, fino a quando sono state considerate tutte le possibili transizioni
  - annotiamo i valori delle uscite
- Nell'esempio, sono necessari 4 stati



Nota: sugli archi si indicherà per brevità ***ingresso/uscita***

Esempio – Codifica degli Stati

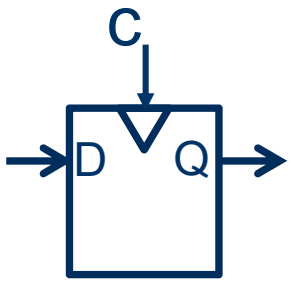
- Associamo ad ogni stato, finora simbolicamente indicato con $\mathbf{S_i}$, una stringa di bit
  → definiamo la ***codifica*** degli stati
- Se gli stati sono **n**, sono necessari *almeno* **[log2 n]** bit
- Diversi obiettivi di progetto (velocità, affidabilità) possono richiedere diverse codifiche, anche ridondanti:
  - binaria standard a lunghezza minima
  - *one-hot* (stati decodificati su **n** bit)
  - stati adiacenti, etc…



- Nell'esempio, scegliamo una codifica con stati adiacenti (lungo ogni transizione, cambia un solo bit della codifica dello stato)

Esempio – Scelta degli elementi di memoria

- E' richiesta una macchina a sincronizzazione esterna
  → servono dei flip-flop edge-triggered
- Scegliamo (qui arbitrariamente) di usare flip-flop edge-triggered attivi su fronti di salita di tipo **D**
- Ne serviranno due (uno per ciascun bit della codifica dello stato)
- Ricordiamo la tabella di eccitazione del Flip-Flop **D**

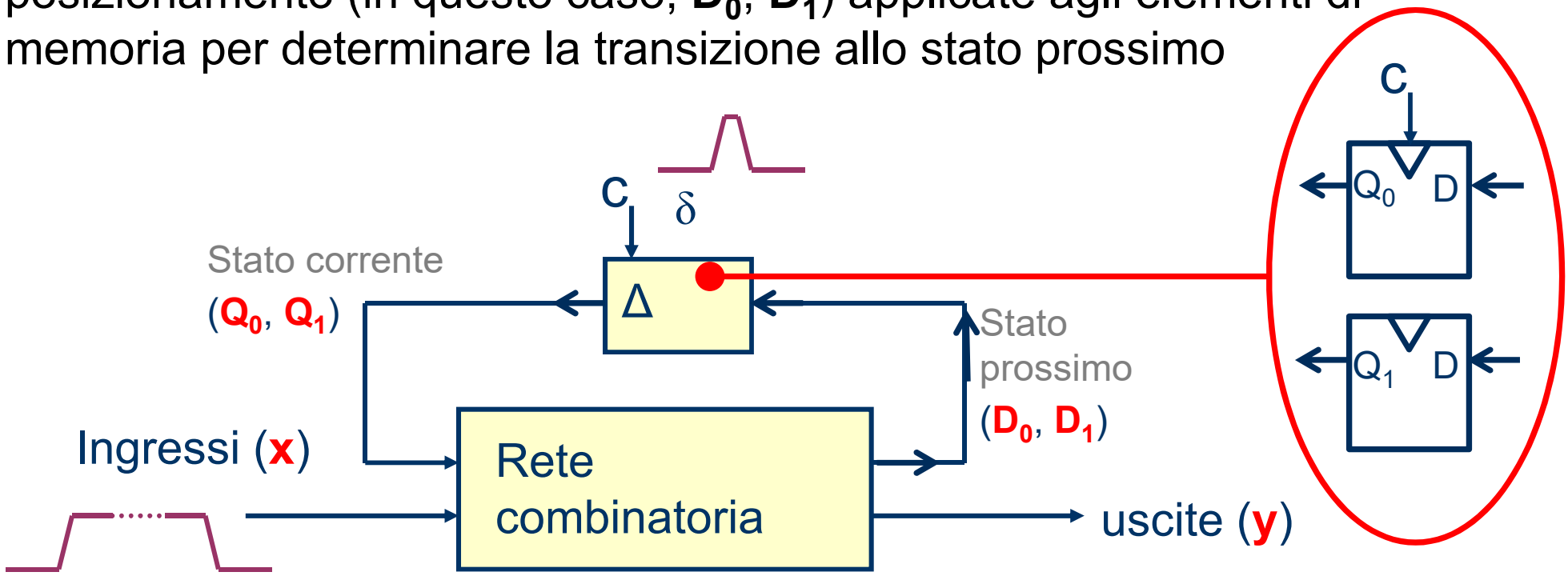


Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Flip-Flop D: tabella di eccitazione

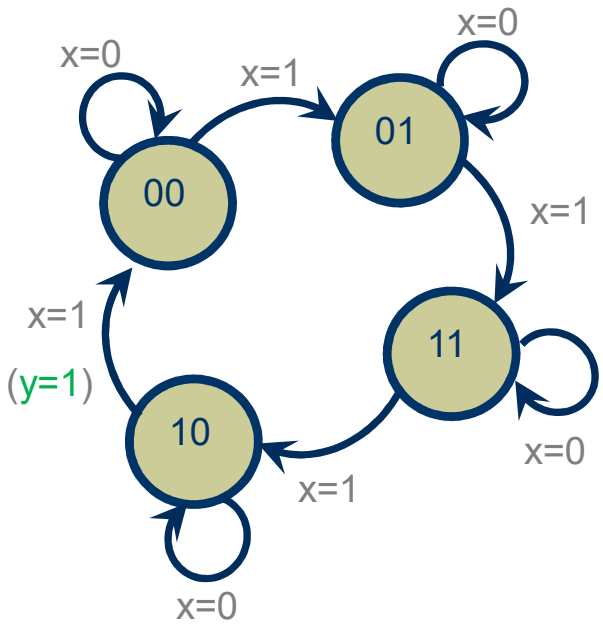
Esempio – Scelta degli elementi di memoria

- La figura in basso riporta lo schema realizzativo della macchina da realizzare
- Individuati gli elementi di memoria, occorre ora progettare le reti combinatorie che realizzano l'uscita **y** e le funzioni di posizionamento (in questo caso, $\mathbf{D_0}$, $\mathbf{D_1}$) applicate agli elementi di memoria per determinare la transizione allo stato prossimo



Esempio – Funzioni di Posizionamento

- Direttamente dal grafo, deriviamo la Tabella di Transizione che riporta Stato Prossimo (**Q_1'**, **Q_0'**) e Uscita (**y**) richiesti
- Deriviamo poi le funzioni di posizionamento **D_1**, **D_0**



prossimo valore dello stato

x	Q_1	Q_0	Q_1'	Q_0'	y	D_1	D_0
0	0	0	0	0	0		
0	0	1	0	1	0		
0	1	0	1	0	0		
0	1	1	1	1	0		
1	0	0	0	1	0		
1	0	1	1	1	0		
1	1	0	0	0	1		
1	1	1	1	0	0		

Esempio – Funzioni di Posizionamento

- Per ogni transizione $Q_i \rightarrow Q_i'$, la tabella di eccitazione indica direttamente i valori delle funzioni di posizionamento da applicare

Ad esempio, nella prima riga dobbiamo avere la transizione $Q_1 \rightarrow Q_1'$ *pari a* $0 \rightarrow 0$*. La tabella di eccitazione ci dice che in questo caso occorre applicare* $D_1=0$

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

Flip-Flop D: tabella di eccitazione

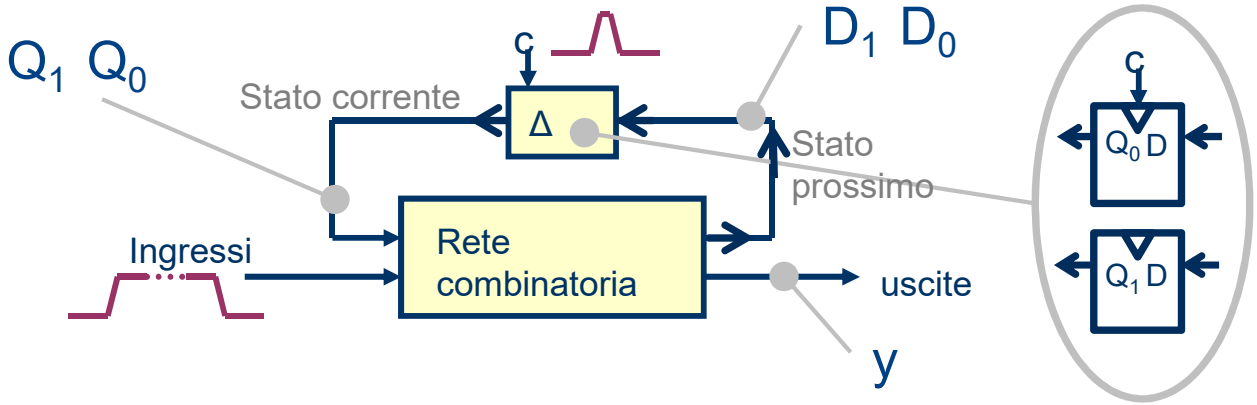
prossimo valore dello stato

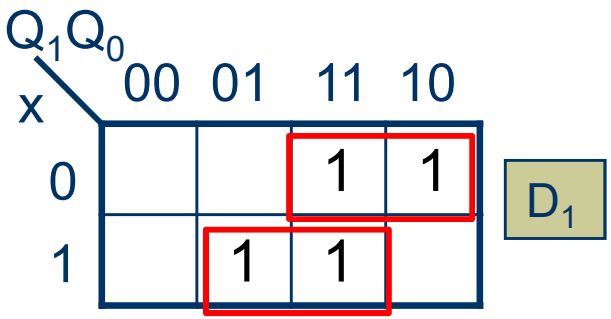
x	Q_1	Q_0	Q_1'	Q_0'	y	D_1	D_0
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	1	1	1	0	1	1
1	1	0	0	0	1	0	0
1	1	1	1	0	0	1	0

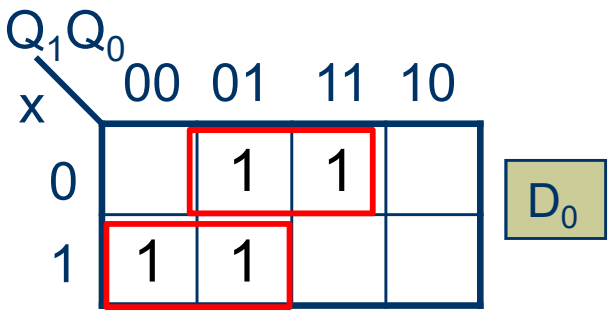
Esempio – Funzioni di Posizionamento

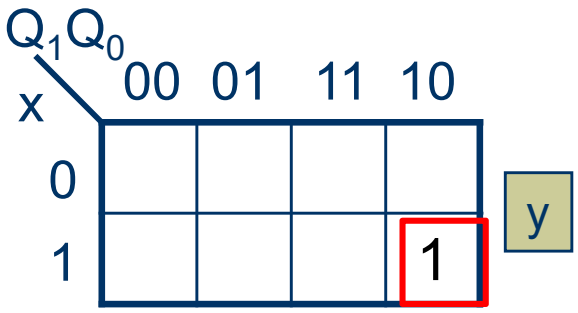
- Non resta che minimizzare le funzioni uscita (**y**) e posizionamento ($\mathbf{D_1}$, $\mathbf{D_0}$)

x	Q_1	Q_0	y	D_1	D_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	0	0	1	0





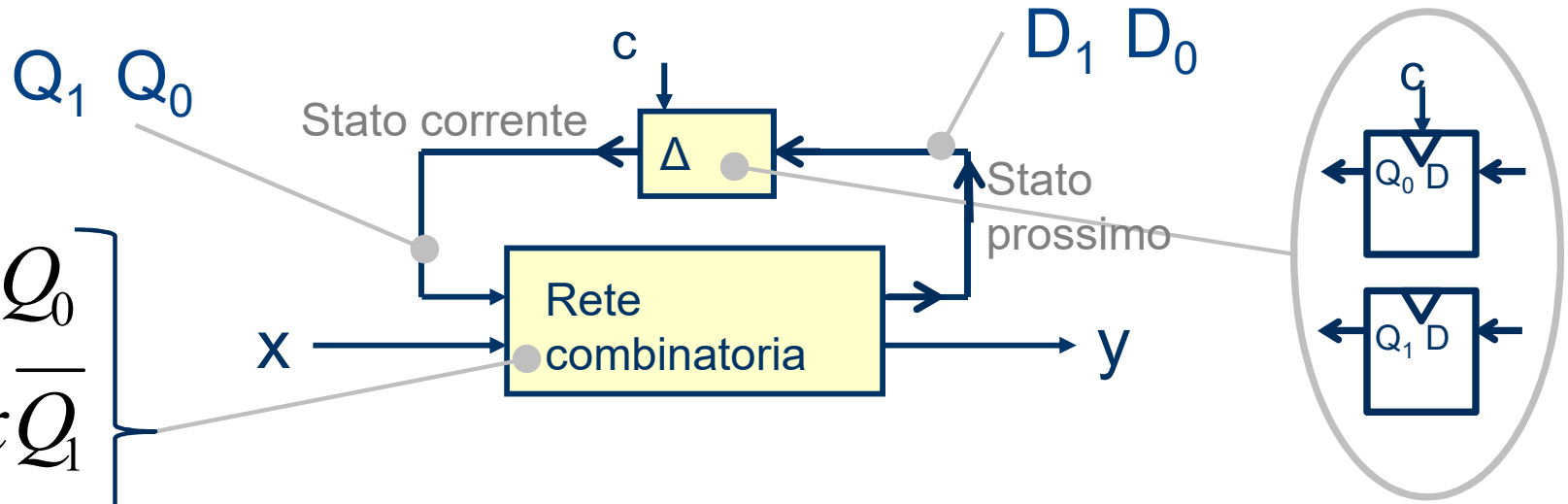




Esempio

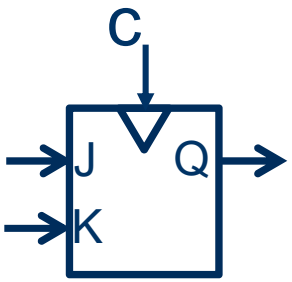
- Una volta minimizzate le funzioni combinatorie di posizionamento/uscita e disegnato il corrispondente circuito, il progetto è terminato

$$D_1 = \bar{x}Q_1 + xQ_0$$
$$D_0 = \bar{x}Q_0 + x\overline{Q_1}$$
$$y = xQ_1\overline{Q_0}$$



Esempio – Scelta degli elementi di memoria

- Se avessimo scelto (o se ci fosse stato chiesto) di usare dei flip-flop **JK**?



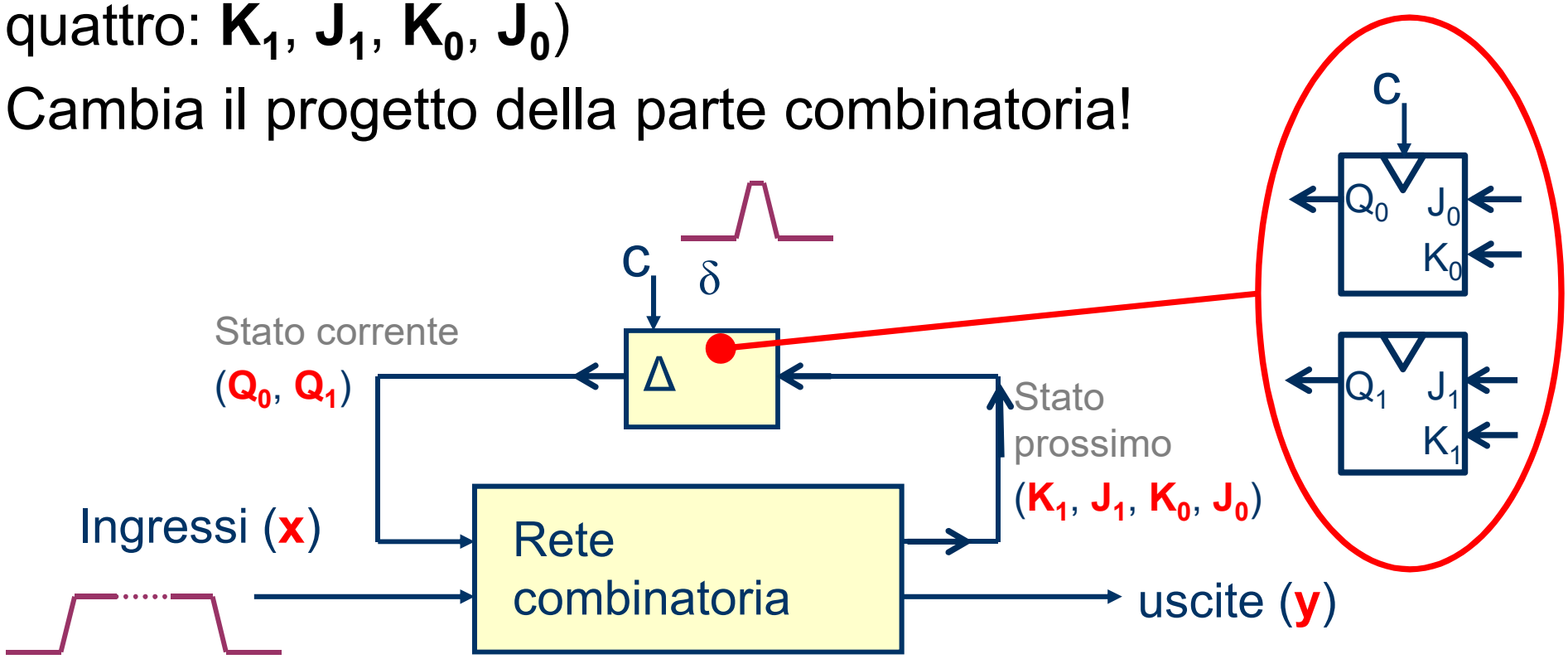
- Teniamo presente il fatto che il flip-flop **JK** richiede *due* ingressi di posizionamento
- La tabella di eccitazione, ovviamente, è diversa

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

Flip-Flop JK: tabella di eccitazione

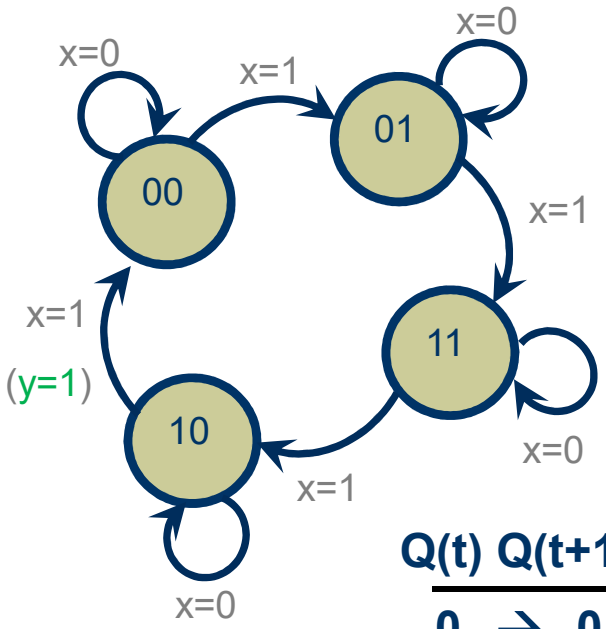
Esempio – Scelta degli elementi di memoria

- Lo schema realizzativo cambia nella parte che contiene gli elementi di memoria, e quindi nel numero e tipo di funzioni di posizionamento da progettare (qui sono quattro: $\mathbf{K_1}$, $\mathbf{J_1}$, $\mathbf{K_0}$, $\mathbf{J_0}$)
- Cambia il progetto della parte combinatoria!



Esempio – Funzioni di Posizionamento

- Guardiamo ciascuna transizione e teniamo presente la tabella di eccitazione



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

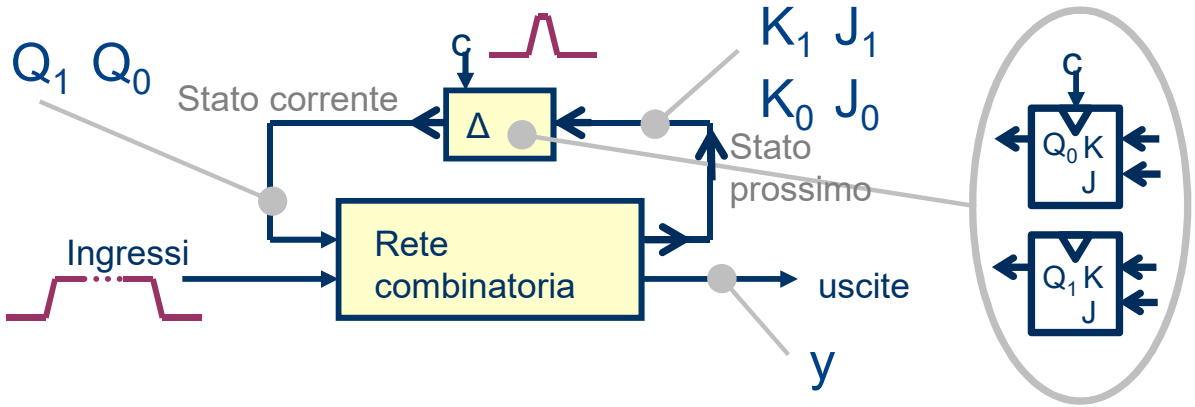
Flip-Flop JK: tabella di eccitazione

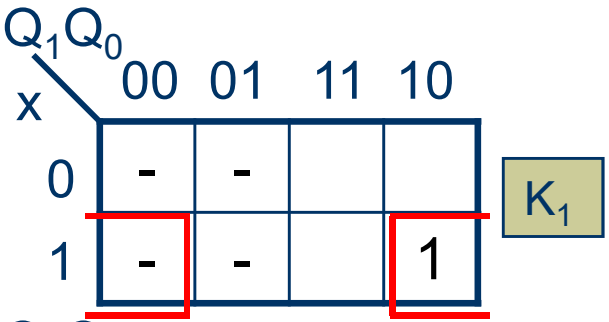
prossimo valore dello stato

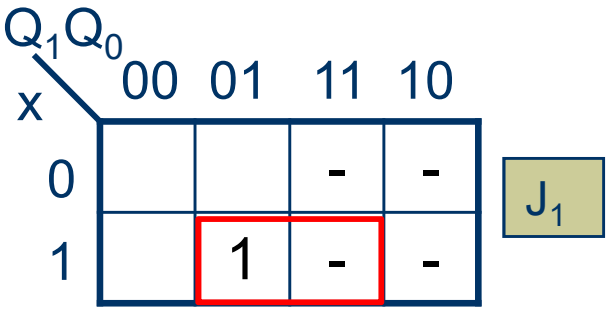
x	Q_1	Q_0	Q_1'	Q_0'	y	K_1	J_1	K_0	J_0
0	0	0	0	0	0	-	0	-	0
0	0	1	0	1	0	-	0	0	-
0	1	0	1	0	0	0	-	-	0
0	1	1	1	1	0	0	-	0	-
1	0	0	0	1	0	-	0	-	1
1	0	1	1	1	0	-	1	0	-
1	1	0	0	0	1	1	-	-	0
1	1	1	1	0	0	0	-	1	-

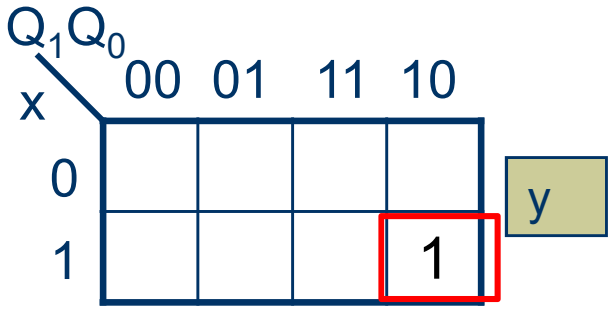
Esempio – Funzioni di Posizionamento

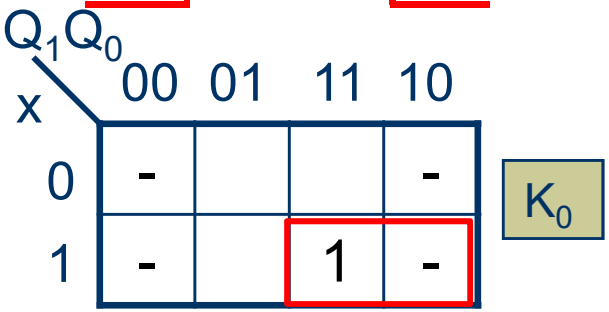
x	Q_1	Q_0	y	K_1	J_1	K_0	J_0
0	0	0	0	-	0	-	0
0	0	1	0	-	0	0	-
0	1	0	0	0	-	-	0
0	1	1	0	0	-	0	-
1	0	0	0	-	0	-	1
1	0	1	0	-	1	0	-
1	1	0	1	1	-	-	0
1	1	1	0	0	-	1	-

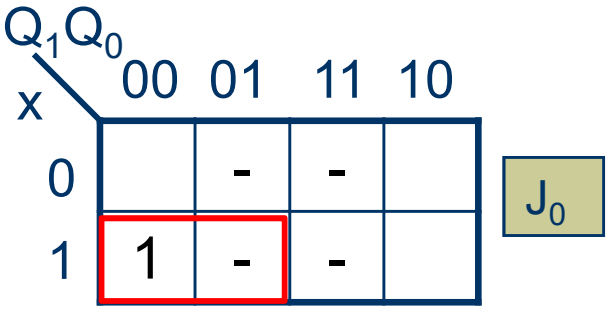








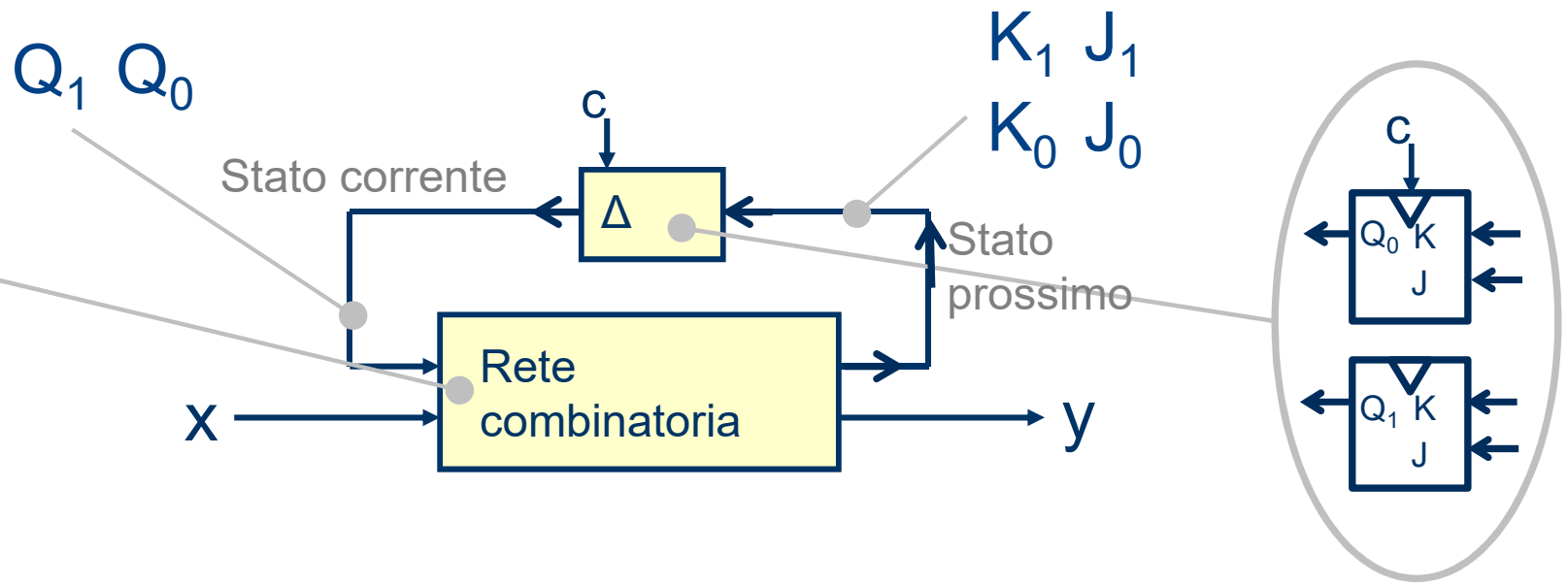




Esempio

- Una volta minimizzate le funzioni combinatorie di posizionamento/uscita e disegnato il corrispondente circuito, il progetto è terminato

$$K_1 = x\overline{Q_0}$$
$$J_1 = xQ_0$$
$$K_0 = xQ_1$$
$$J_0 = x\overline{Q_1}$$
$$y = xQ_1\overline{Q_0}$$



Riepilogo delle Tabelle di Eccitazione

A seconda dell'elemento di memoria usato, occorrerà tenere presente, per il progetto della parte combinatoria, la corrispondente tabella di eccitazione:

flip-flop D

Q(t+1)	D	Operazione
0	0	Reset
1	1	Set

flip-flop T

Q(t+1)	T	Operazione
$Q(t)$	0	Invariato
$\bar{Q}(t)$	1	toggle

flip-flop SR

Q(t)	Q(t+1)	S	R	Operazione
0	0	0	-	Invariato
0	1	1	0	Set
1	0	0	1	Reset
1	1	-	0	Invariato

flip-flop JK

Q(t)	Q(t+1)	J	K	Operazione
0	0	0	-	Invariato
0	1	1	-	Set
1	0	-	1	Reset
1	1	-	0	Invariato

Esercizio proposto

1) *Analizzare il circuito derivandone grafo e tabella di transizione*

2) *Riprogettare il circuito usando flip-flop edge-triggered di **tipo T***