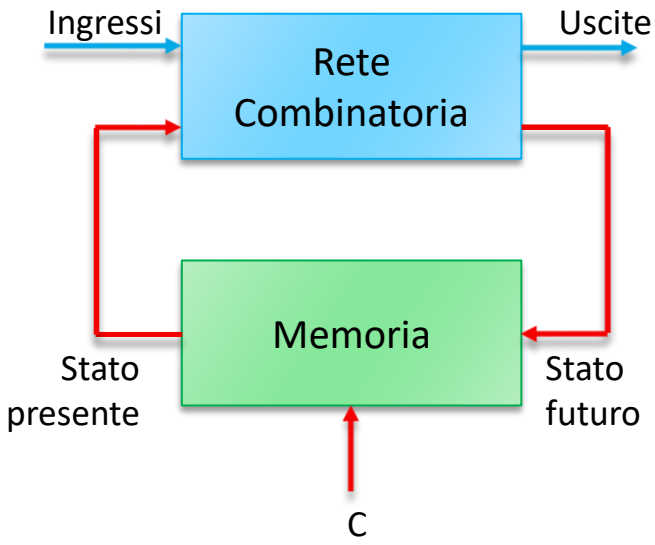
Macchine a stati finiti

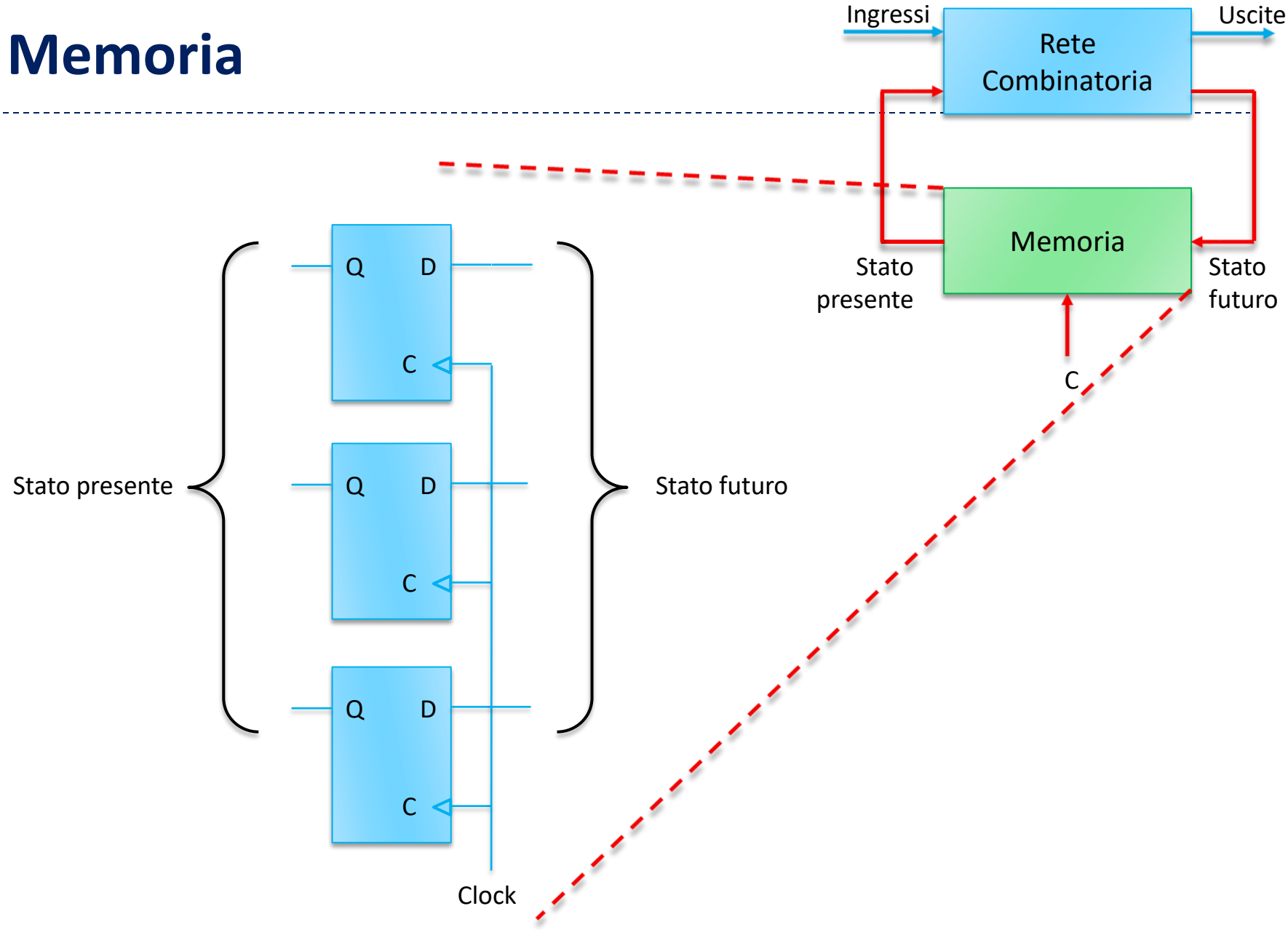
Struttura e rappresentazione di circuiti sequenziali

Stato

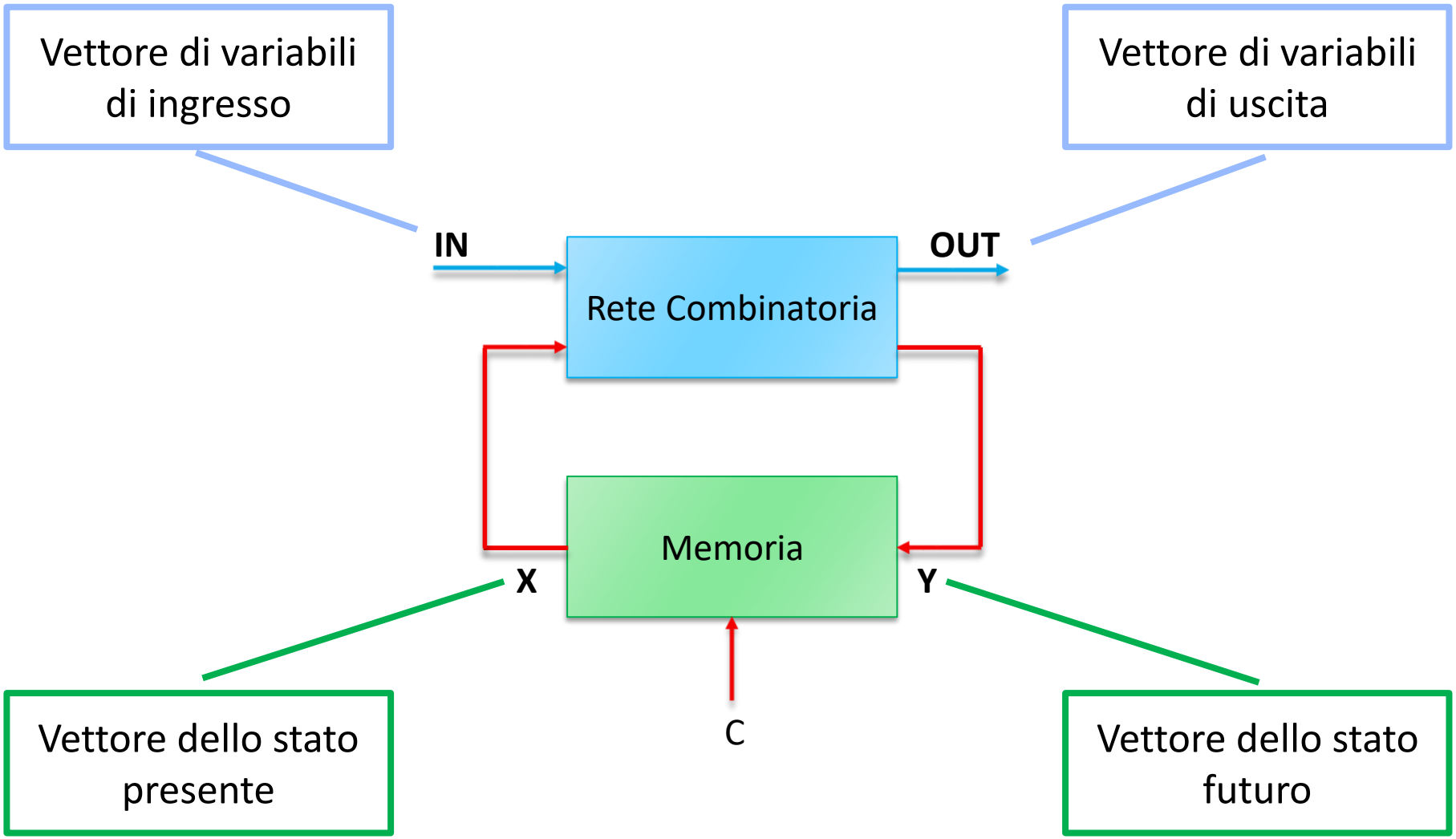
- **Indicheremo con il termine *stato* l'insieme dei valori contenuti nella memoria**
- **Lo stato è quindi, in generale, un vettore di variabili booleane**
  - Ogni variabile è memorizzata tramite un flip flop o registro
- **Lo stato presente e quello futuro sono congruenti**
  - Ogni variabile dello stato presente ha una corrispondente variabile nello stato futuro



Memoria

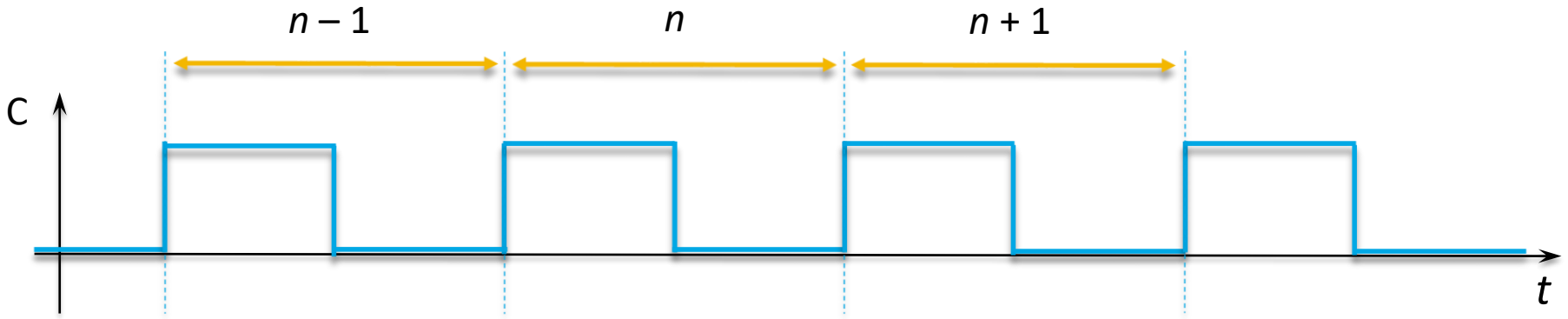

Ingressi
Uscite
Rete Combinatoria
Memoria
Stato presente
Stato futuro
C
Q
D
C
Q
D
C
Q
D
C
Stato presente
Stato futuro
Clock

Circuiti sequenziali



Cadenza

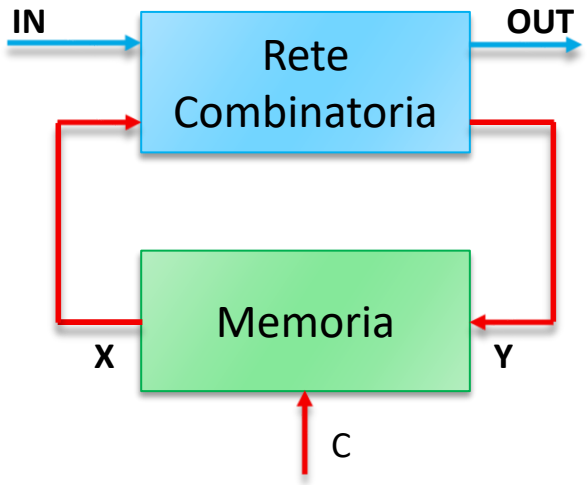
- **Il funzionamento del circuito è cadenzato**
  - Lo stato evolve solo al fronte del clock
  - Usiamo un indice *n* per contare i fronti del clock
  - Il periodo tra due fronti è detto *ciclo* di clock, o anche *colpo* di clock
  - Lo stato presente *non* cambia durante l'intero ciclo, mentre lo stato futuro in generale può cambiare
  - Con flip flop edge triggered, lo stato presente al ciclo *n* corrisponde allo stato futuro alla fine del ciclo *n* – 1



Equazioni di stato

- **Il circuito ha il seguente funzionamento**
  - Indichiamo lo stato corrispondente al ciclo *n*-esimo con **X**(*n*)
  - Lo stato futuro **Y**(*n*) dipende dal valore degli ingressi al ciclo *n* e dallo stato al ciclo *n* secondo una funzione combinatoria **Y**(*n*) = Δ(**IN**(*n*), **X**(*n*))
  - Lo stato presente al ciclo *n* + 1 è uguale a quello futuro al ciclo *n*

    X(*n* + 1) = **Y**(*n*)
  - Le uscite **OUT** al ciclo *n* dipendono dagli ingressi al ciclo *n* e dallo stato al ciclo *n* secondo una funzione combinatoria Λ
- **Equazioni**
  - **X**(*n* + 1) = Δ(**IN**(*n*), **X**(*n*))
  - **OUT**(*n*) = Λ(**IN**(*n*), **X**(*n*))
- **Equazioni non più implicite**
  - Lo stato futuro e quello presente sono chiaramente separati, **X**(*n*) non dipende da **X**(*n* + 1)
  - Non c'è problema di inesistenza o di troppe soluzioni



Procedimento di progetto

- **In base alla specifica, si determinano i possibili stati del circuito**
  - Per esempio caldaia accesa e spenta
  - Fase creativa, occorre capire i modi base del sistema
  - Un po' come definire le variabili di un programma
- **Si determina l'evoluzione dello stato e delle uscite in base agli ingressi**
  - Considerando quale deve essere lo stato futuro a partire da ogni possibile stato presente
  - Semplice se si è scelto lo stato con cura
- **Si scrive la tabella della verità relativa alle equazioni di stato e delle uscite**
  - Quindi si semplifica la relativa espressione
  - Si realizza il circuito utilizzando degli appositi registri per lo stato
  - Procedimento automatico, non c'è bisogno di pensarci troppo

Tabella degli stati: la caldaia

- **Lo stato *X* indica se la caldaia è accesa o spenta**
  - L'uscita *C* sarà dunque uguale allo stato presente
  - Lo stato futuro X_{n+1} dipende invece dagli ingressi e dallo stato presente X_n

TABELLA DEGLI STATI		Stato presente	Ingressi		Stato futuro Δ	Uscite Λ
		X_n	H	L	X_{n+1}	C
Spenta, $T_{min} < T < T_{max}$	Spenta	0	0	0	0	0
Spenta, $T < T_{min}$	Accendi	0	0	1	1	0
Spenta, $T > T_{max}$	Spenta	0	1	0	0	0
		0	1	1	-	-
Accesa, $T_{min} < T < T_{max}$	Accesa	1	0	0	1	1
Accesa, $T < T_{min}$	Accesa	1	0	1	1	1
Accesa, $T > T_{max}$	Spegni	1	1	0	0	1
		1	1	1	-	-

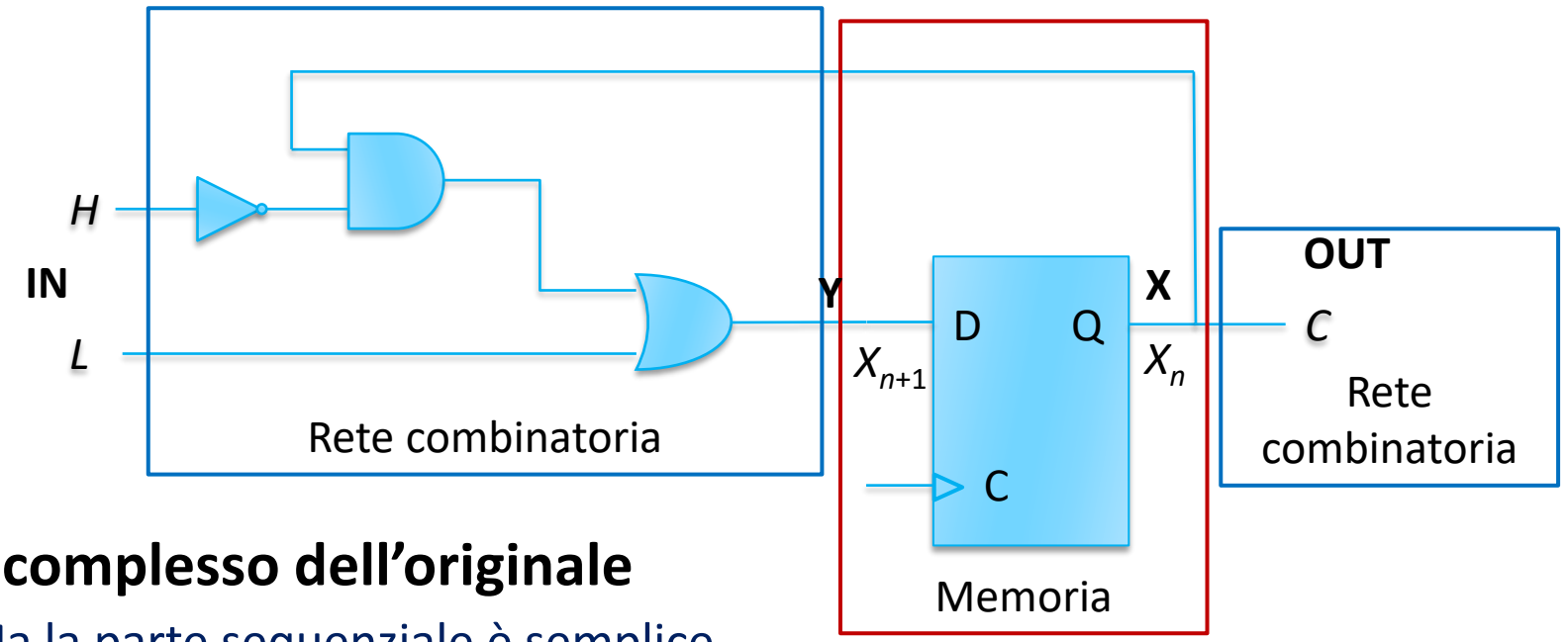
Realizzazione

X_n \ HL	00	01	11	10
0	0	1	-	0
1	1	1	-	0

X_n \ HL	00	01	11	10
0	0	0	-	0
1	1	1	-	1

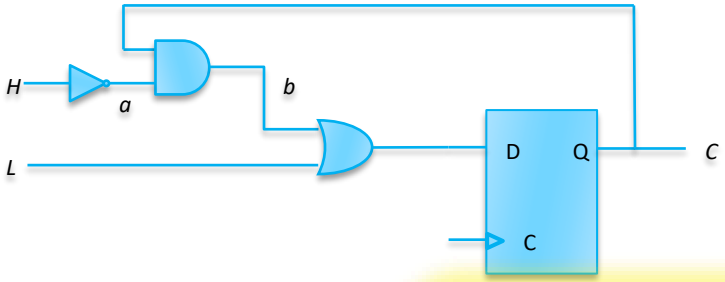
$X_{n+1} = L + X_n H'$

$C = X_n$



- **Più complesso dell'originale**
  - Ma la parte sequenziale è semplice
  - Circuito sincrono, il feedback è interrotto dalla presenza del registro

Diagramma temporale



- **Ingressi troppo veloci possono andare persi**
  - Risoluzione temporale finita

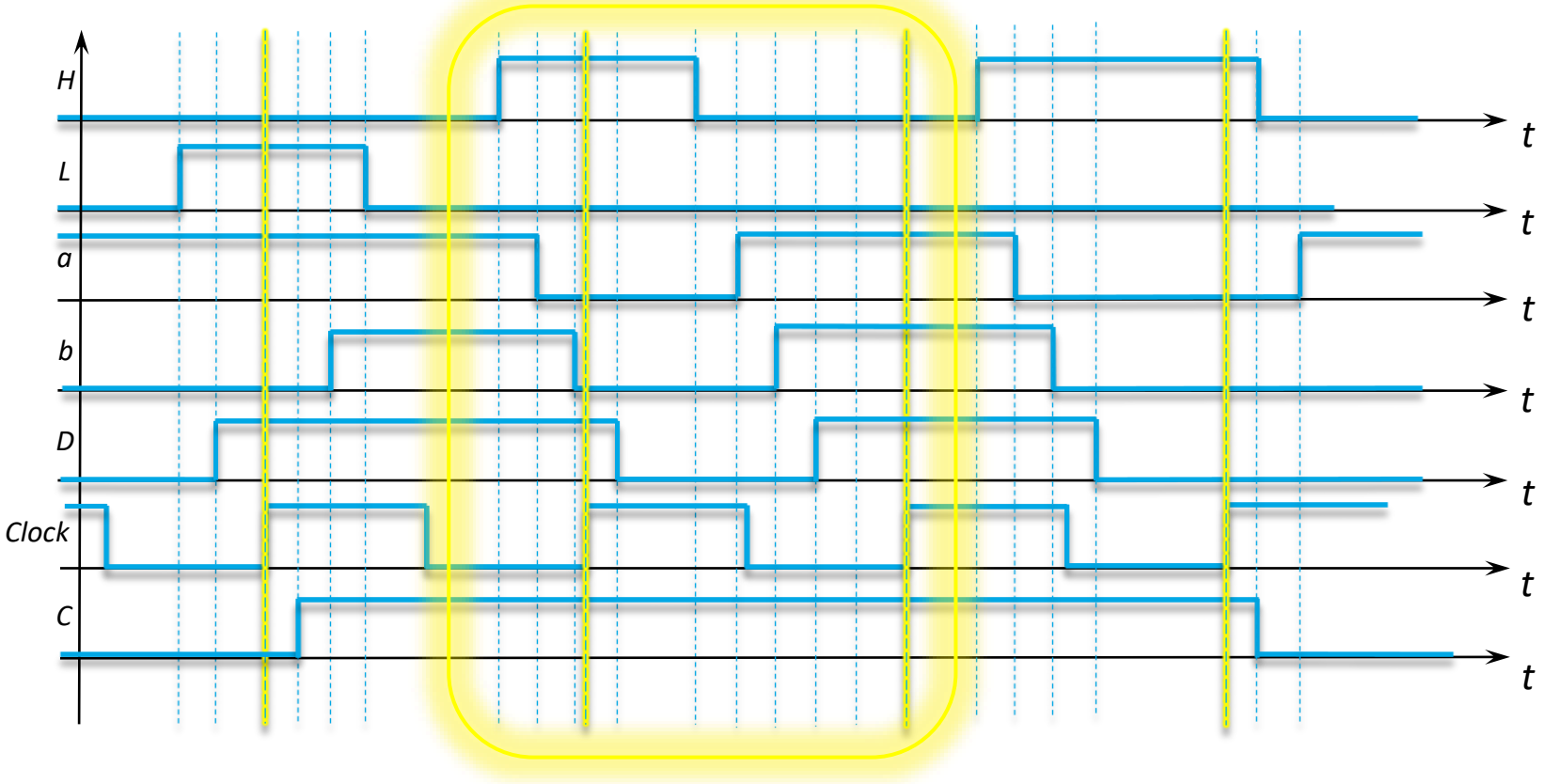


Diagramma degli stati

Come mostrare le funzioni in forma grafica

Diagramma degli stati

- **La tabella delle funzioni Δ e Λ mostrano come cambia lo stato e le uscite in funzione**
  - Dello stato corrente
  - Degli ingressi
- **Il passaggio da uno stato ad un altro si chiama transizione**
  - Ogni riga della tabella è quindi una possibile transizione
  - Lo stato di partenza e quello di arrivo sono quello presente e quello futuro
- **Possiamo far vedere la stessa cosa in forma grafica**
  - Ogni stato diventa un cerchio con il valore delle variabili di stato
  - Ogni transizione diventa una freccia che va dal cerchio con lo stato presente a quello con lo stato futuro
  - Sulla freccia si indica il valore degli ingressi per i quali si fa la transizione
  - E si indica il valore delle uscite corrispondenti allo stato di partenza e all'ingresso segnato sulla transizione

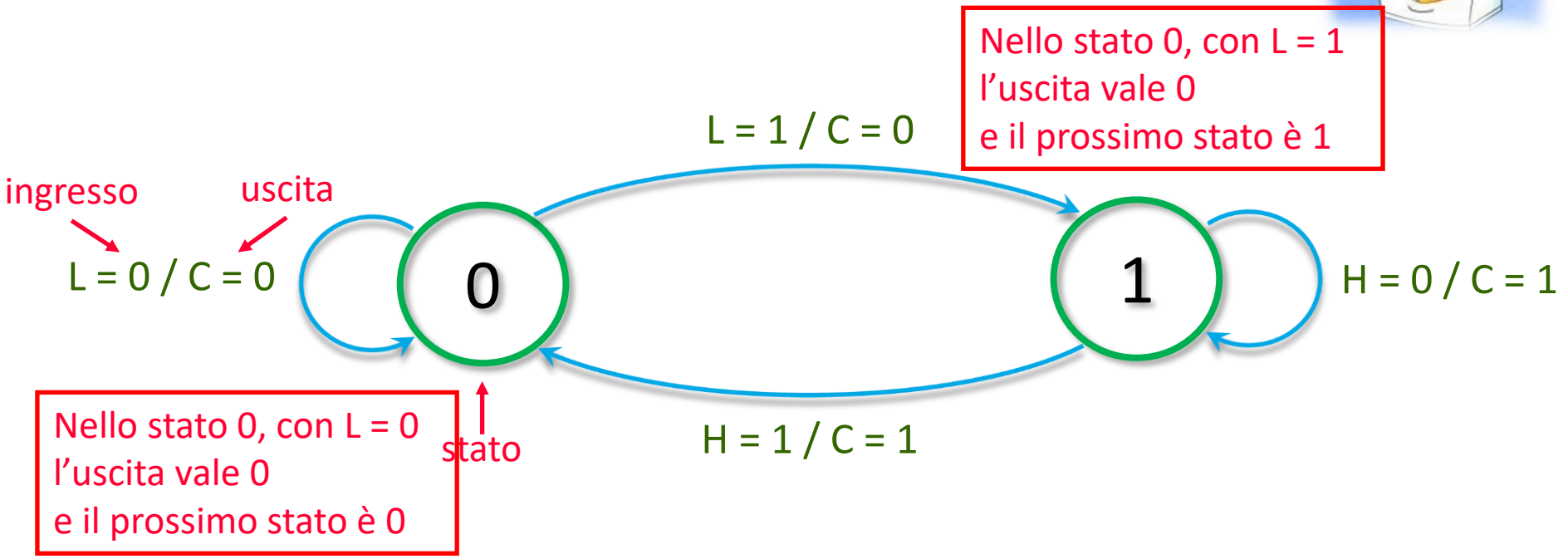
Non ne possiamo più della caldaia

- **Proviamo prima a semplificare, anche se non ce n'è bisogno**
  - Se due righe hanno lo stesso stato presente, stato futuro e uscita, le fondiamo (sono la stessa transizione!)
  - Se ci sono dei don't care, li usiamo per semplificare altre righe
  - E' un po' come fare le mappe di Karnaugh, si potrebbe fare anche con quelle, ma ora non è importante

Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	*H*	*L*	X_{n+1}	*C*
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	-	-
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	-	-

Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	*H*	*L*	X_{n+1}	*C*
0	-	0	0	0
0	-	1	1	0
1	0	-	1	1
1	1	-	0	1

Diagramma degli stati



Riassumendo

- **Un cerchio per ogni stato**
- **Una transizione per ogni riga**

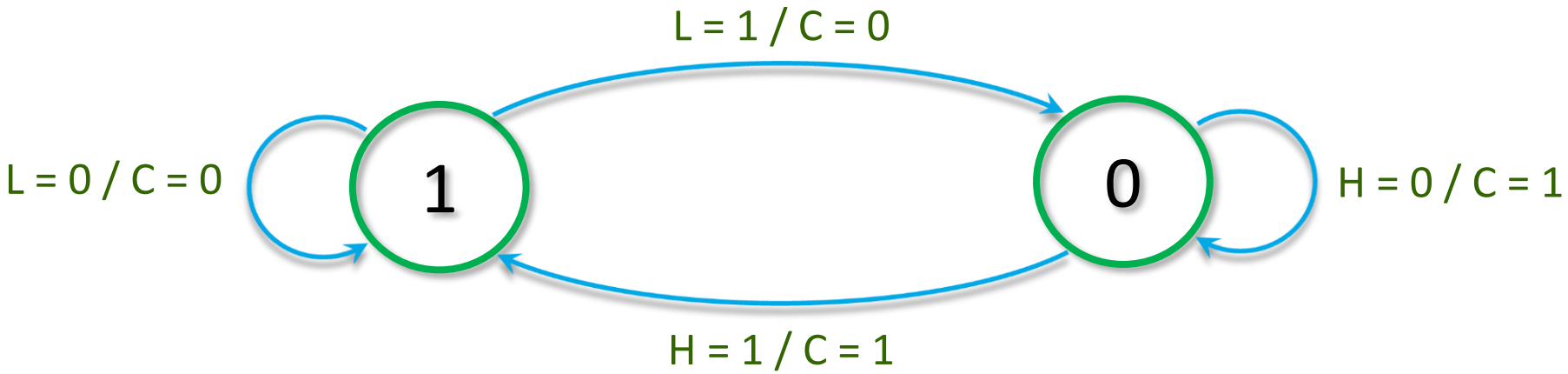
Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	*H*	*L*	X_{n+1}	*C*
0	-	0	0	0
0	-	1	1	0
1	0	-	1	1
1	1	-	0	1

Diagramma degli stati

- **Il diagramma degli stati fornisce le stesse informazioni della tabella della verità**
  - La forma grafica è più semplice da interpretare
  - La sua lettura si avvicina di più al nostro modo di pensare
- **Caratteristiche del diagramma degli stati**
  - Ha tanti stati quante sono le possibili combinazioni delle variabili di stato
    - Per n flip flop ci saranno al più 2^n stati
    - Non è detto però che ci siano tutti!
  - Ogni stato ha tante transizioni quante sono le possibili combinazioni degli ingressi
    - Per m ingressi ogni stato avrà 2^m transizioni
    - Nel caso precedente, ogni transizione vale per due!
  - Nella tabella degli stati ci saranno allora 2^{n+m} righe

Codifica degli stati

- **Il valore assegnato ad ogni stato è arbitrario**
  - Avremmo potuto codificare la condizione di caldaia spenta con il valore 1, e di caldaia accesa con il valore 0
  - L'uscita, invece, deve seguire quanto indicato dalle specifiche, e quindi deve essere a 0 per spegnere la caldaia e a 1 per accenderla
  - Il circuito ovviamente cambia!
  - Si può (si deve?) dunque cercare una codifica che porti al circuito migliore
  - Il problema è piuttosto complesso

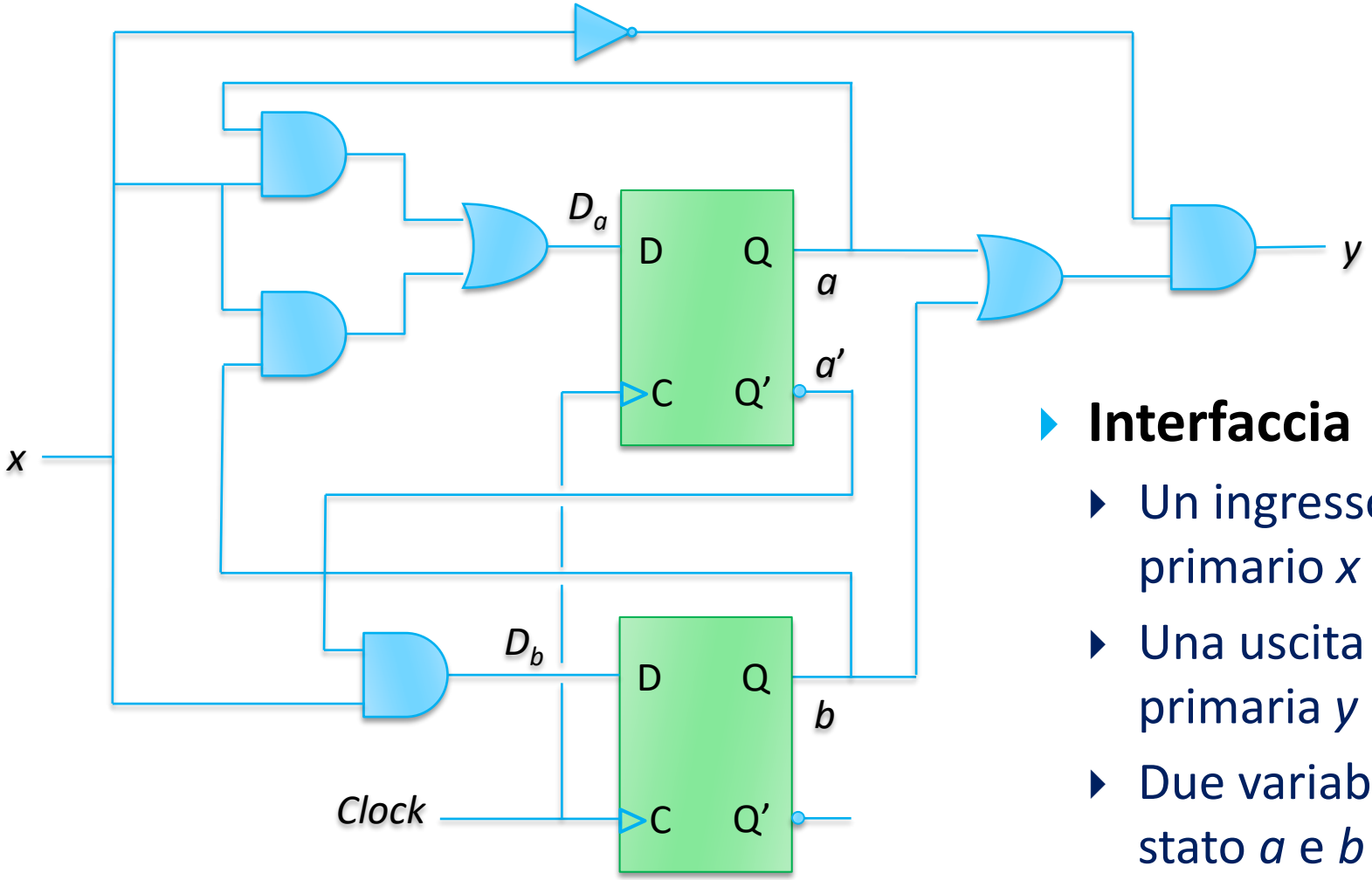


Analisi e Progetto di circuiti sequenziali con i diagrammi

Analisi di circuiti sequenziali

- **Dato un circuito sequenziale, è facile risalire al suo diagramma degli stati**
  - Dal numero di flip flop si risale ai possibili stati del sistema
  - Guardando la rete combinatoria possiamo determinare la funzione che calcola lo stato futuro e le uscite, e quindi le transizioni
  - A questo punto si costruisce il diagramma corrispondente

Esempio

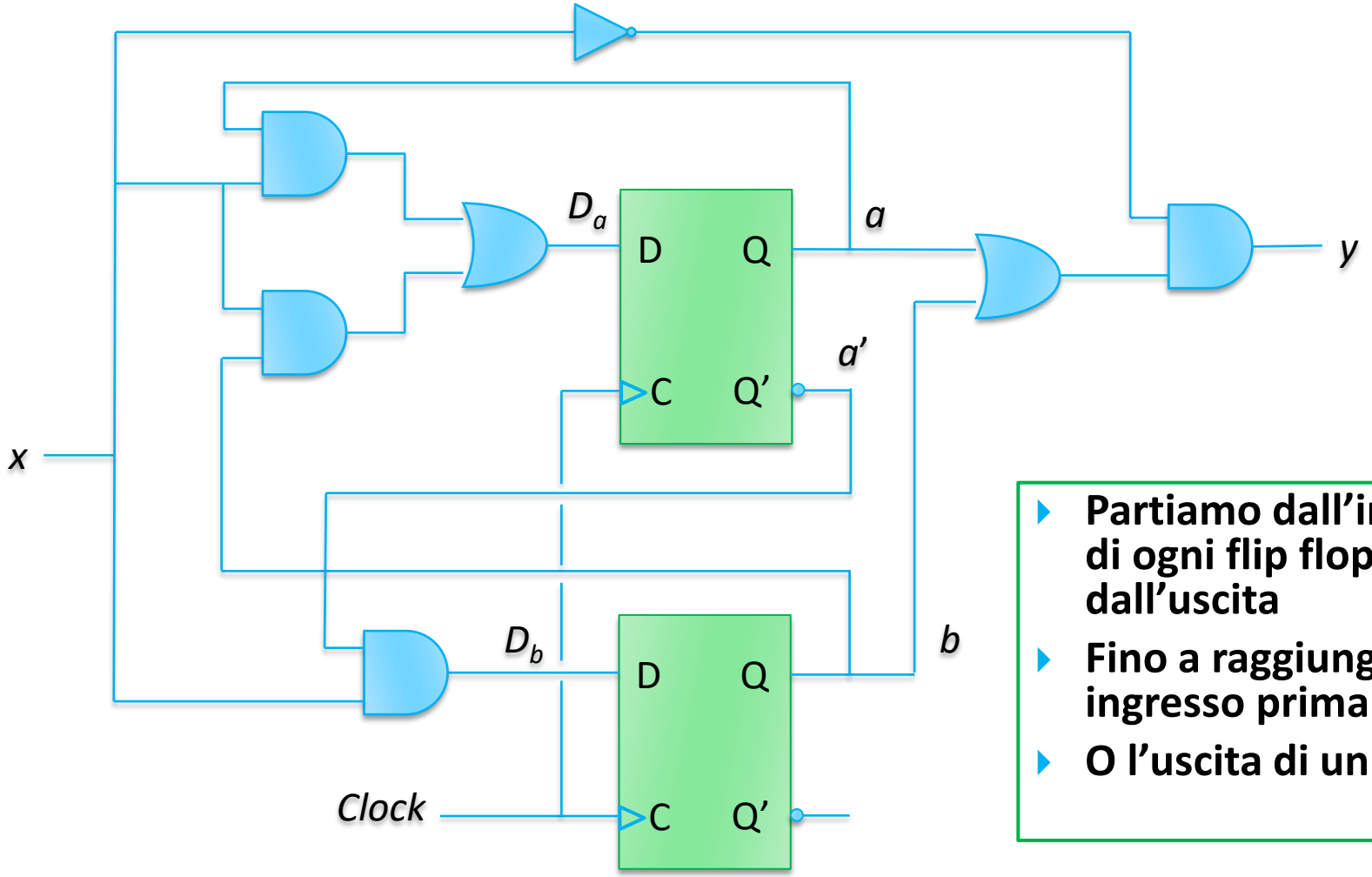


- **Interfaccia**
  - Un ingresso primario *x*
  - Una uscita primaria *y*
  - Due variabili di stato *a* e *b*

Procedura

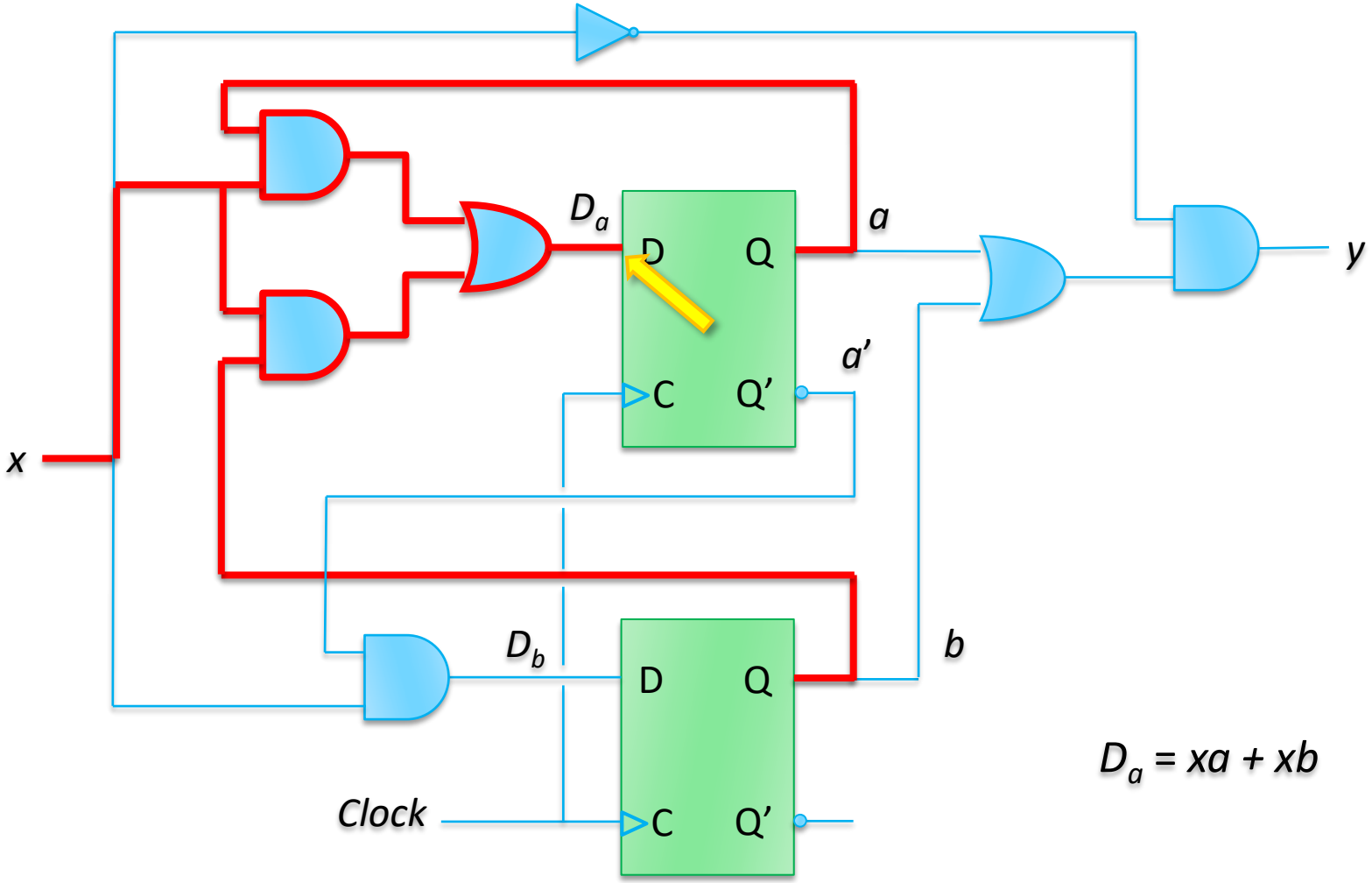
- **Deriviamo dal circuito l'espressione degli ingressi di ciascun elemento di memoria**
  - Le espressioni così ottenute costituiscono la funzione di stato futuro della macchina a stati
  - E' una funzione degli ingressi primari del circuito e dello stato presente, ossia l'uscita degli elementi di memoria
- **Deriviamo dal circuito l'espressione delle uscite**
  - Ottenendo la funzione di uscita
- **Costruiamo la tabella degli stati e il diagramma degli stati**
  - Osservando tabella e diagramma cercheremo di capire cosa fa il circuito in pratica

Esempio

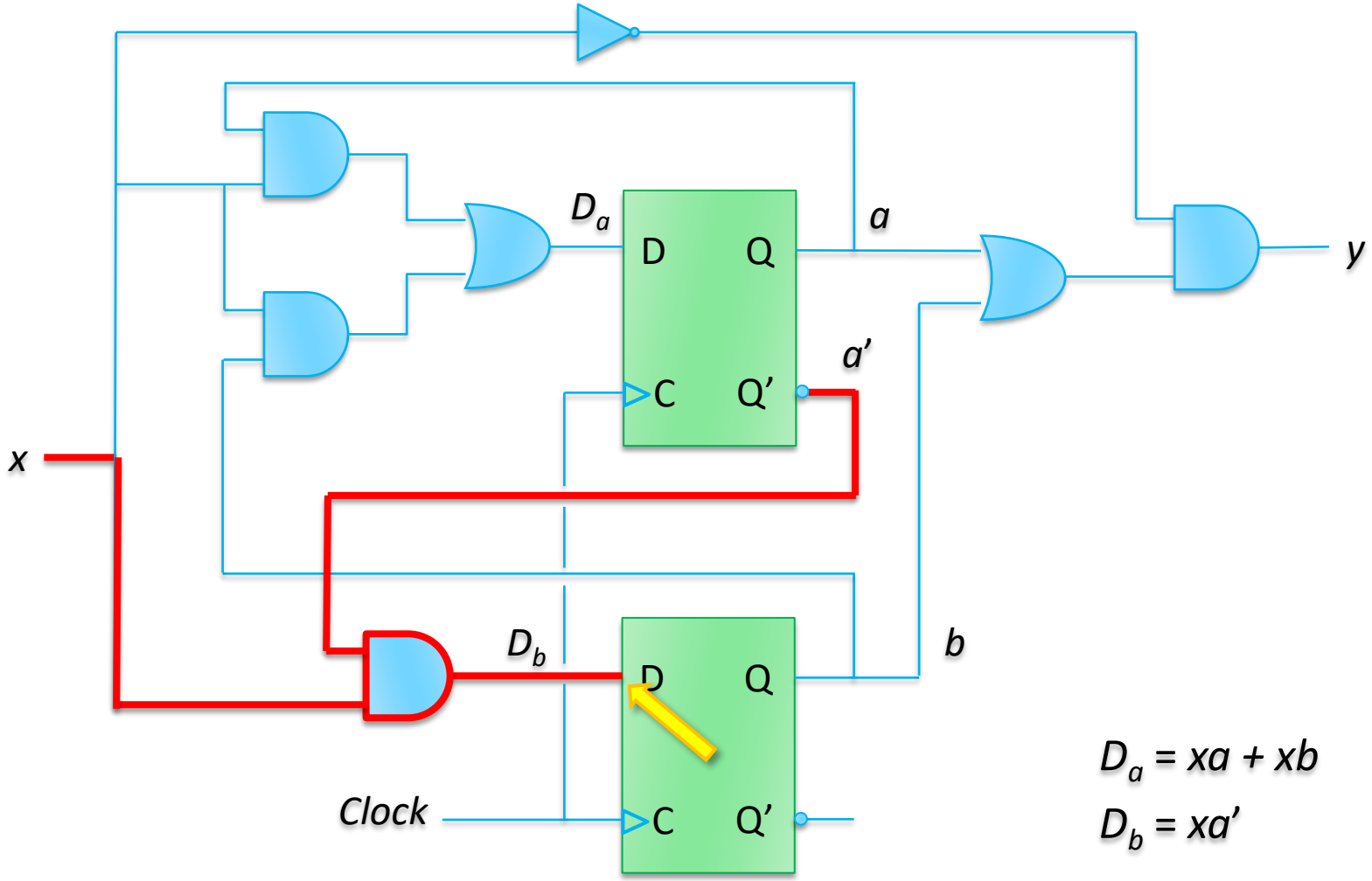


- **Partiamo dall'ingresso di ogni flip flop o dall'uscita**
- **Fino a raggiungere un ingresso primario**
- **O l'uscita di un flip flop**

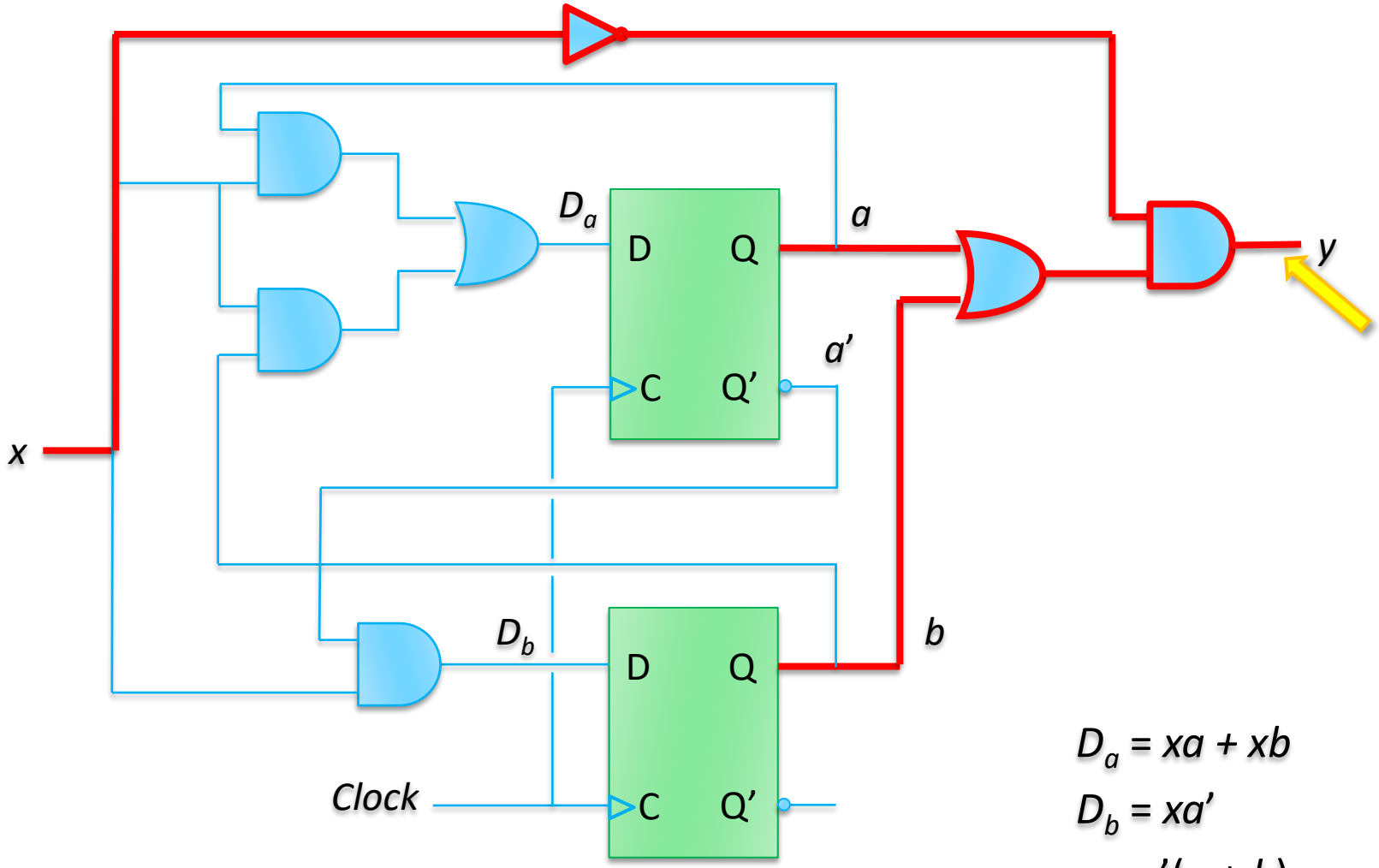
Esempio



Esempio



Esempio



$D_a = xa + xb$

$D_b = xa'$

$y = x'(a + b)$

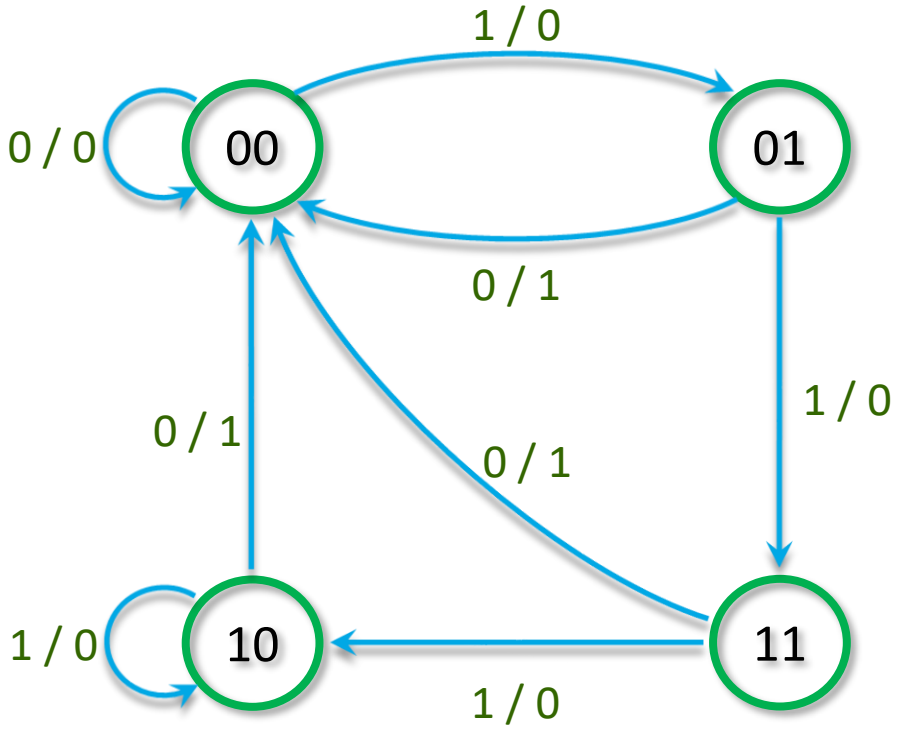
Tabella degli stati

$D_a = xa + xb$ $D_b = xa'$ $y = x'(a + b)$

Stato presente		Ingresso	Stato futuro		Uscite
a	*b*	*x*	*a*	*b*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Diagramma degli stati

- **Funzionamento**
  - Il circuito attende con l'uscita a 0 finché l'ingresso è 0
  - Quando l'ingresso diventa 1 cambia di stato e mette l'uscita a 1 quando l'ingresso torna nuovamente a 0

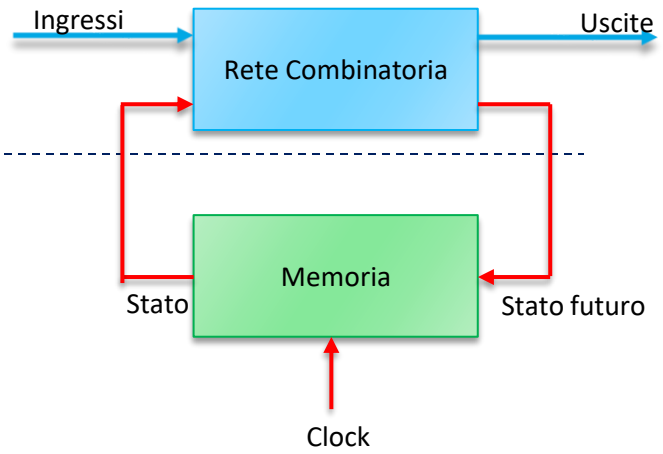


Stato presente		*In*	*Stato futuro*		*Out*
a	*b*	*x*	*a*	*b*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Progetto: considerazioni preliminari

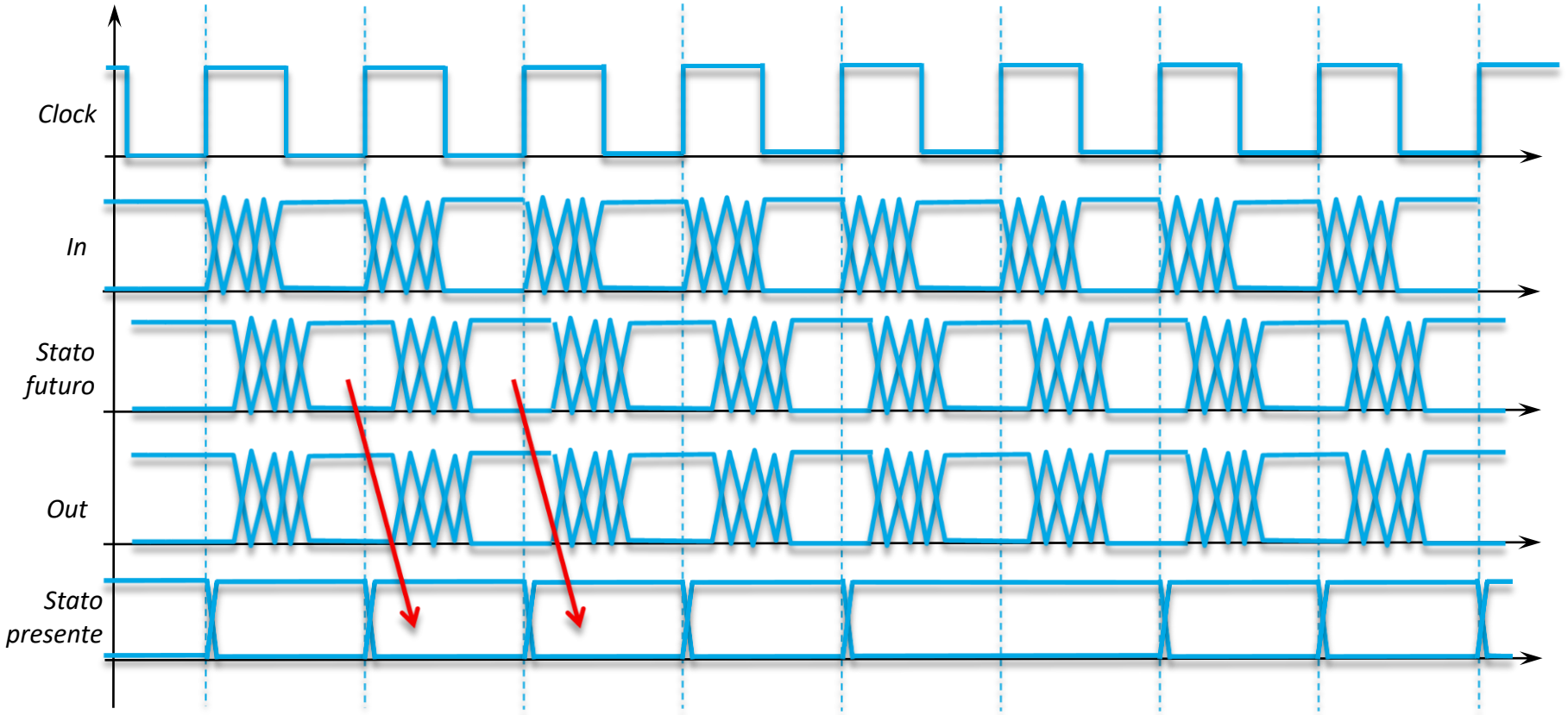
Cose utili da sapere

Se gli ingressi cambiano durante il ciclo di clock



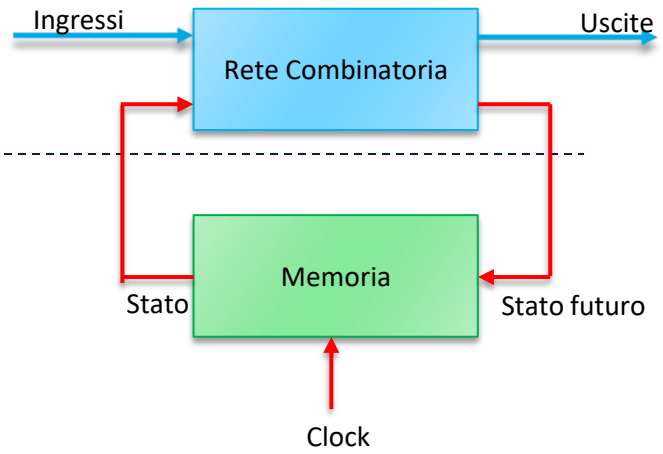
- **Per lo stato**
  - Quando il circuito è temporizzato da un clock, la transizione viene effettuata solo quando sopraggiunge il fronte attivo
  - Nel frattempo la rete combinatoria che calcola lo stato futuro segue l'andamento degli ingressi (ma lo stato ovviamente non cambia)
  - Se gli ingressi cambiano durante un ciclo di clock, cambierà anche lo stato futuro (ma non quello presente)
  - La transizione che viene presa è quella che viene selezionata dalla rete combinatoria al momento in cui sopraggiunge il fronte del clock (e si cambia stato presente)
- **Per le uscite**
  - Se gli ingressi cambiano durante un ciclo di clock, e le uscite dipendono dagli ingressi, allora le uscite cambiano!

Andamento temporale



- **Ingressi cambiano durante il ciclo di clock**
  - Uscite e stato futuro cambiano di conseguenza
  - Lo stato presente cambia solo sul fronte attivo

Velocità

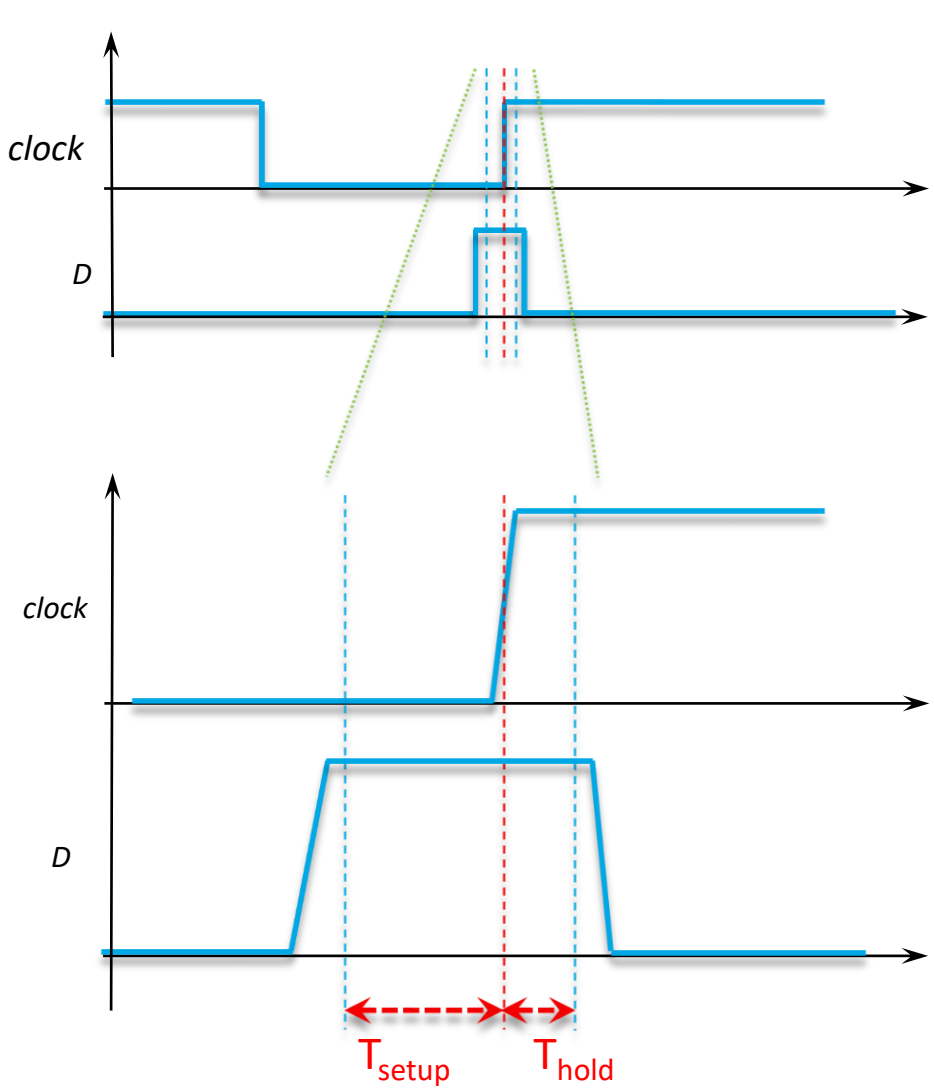


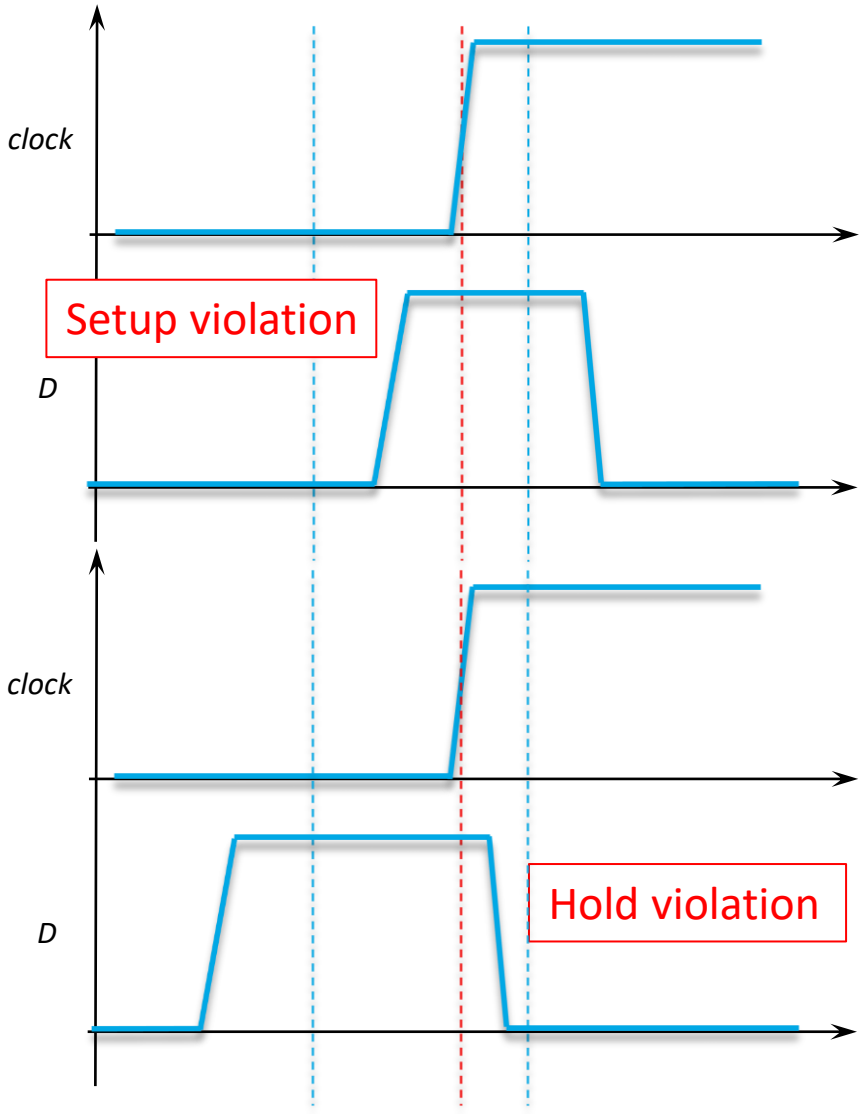
- **A che frequenza può andare il clock?**
  - I segnali in ingresso ai flip flop devono essere stabili entro il prossimo fronte attivo
  - L'informazione di stato deve propagarsi dall'uscita dei flip flop al loro ingresso in un tempo pari al periodo di clock
  - Il ritardo introdotto dalla rete combinatoria nel caso peggiore definisce quindi il periodo minimo del clock, e la sua frequenza massima
  - Per esempio, se voglio andare a 1 GHz, la rete combinatoria deve calcolare lo stato futuro in al più 1 ns

Tempistiche dei flip flop

- **Lo stato del flip flop dipende contemporaneamente dal clock e dal valore dell'ingresso D**
  - Il fronte del clock provoca il cambio di stato
  - Si deve instaurare un feedback che si autosostiene
  - Questo feedback impiega un certo tempo per essere stabile
- **Pertanto il segnale di ingresso D dovrà essere costante per un certo tempo *prima* e *dopo* il fronte attivo del clock**
  - Tempo di Setup: tempo prima del fronte durante il quale D deve essere stabile
  - Tempo di Hold: tempo dopo il fronte durante il quale D deve essere stabile
- **Se le condizioni non sono rispettate, il flip flop può finire in uno stato casuale (metastabilità)**

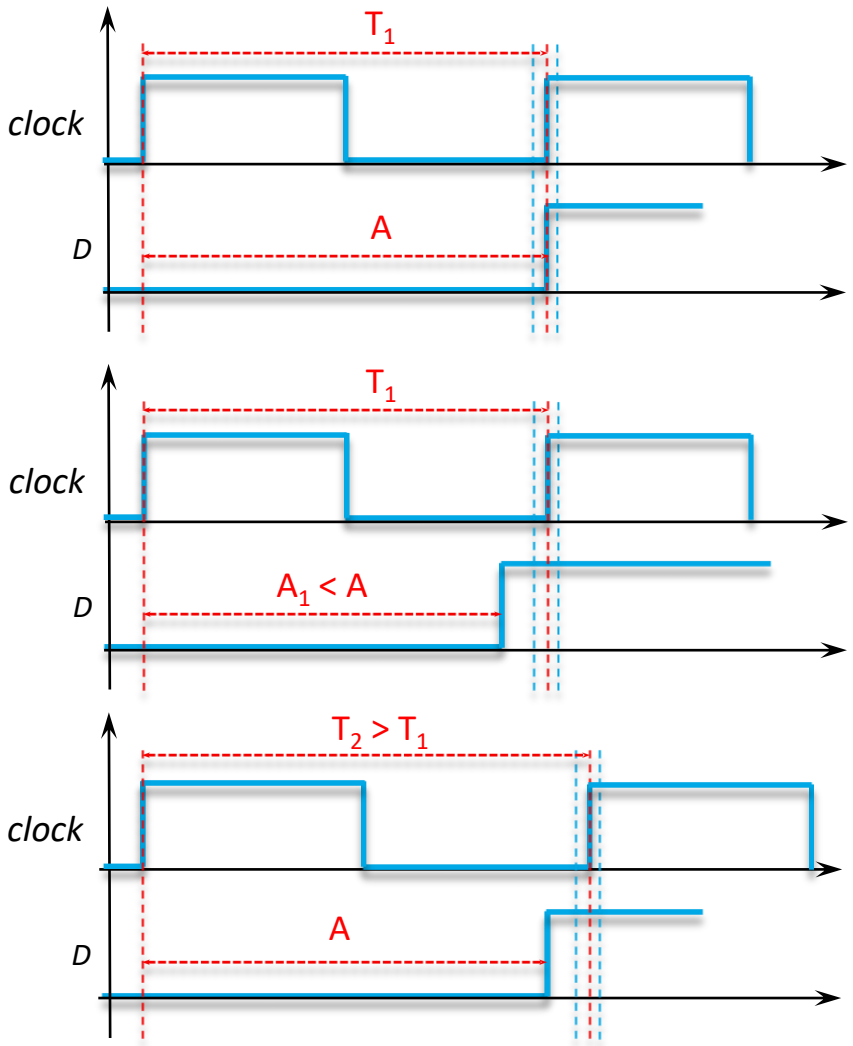
Tempi di setup e di hold





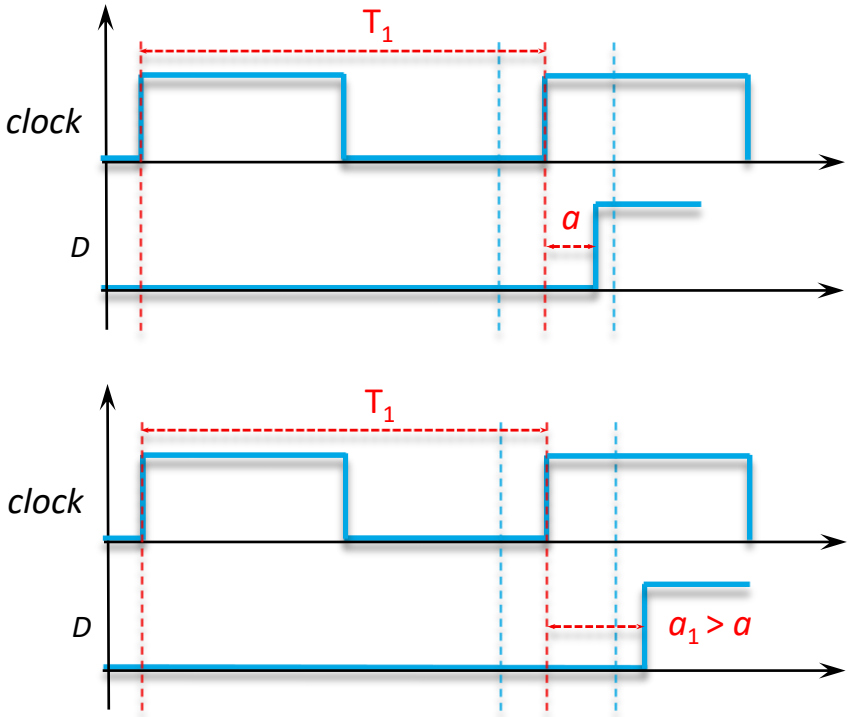
Rimedi per setup violation

- **Se vi sono violazioni del tempo di setup significa che il circuito è troppo lento**
  - Lo stato futuro non viene calcolato in tempo
  - Occorre ottimizzare la logica combinatoria in modo che vada più veloce
  - Oppure si può rallentare il clock
  - La massima frequenza del clock dipende dal ritardo massimo tra ogni coppia di registri, più il tempo di setup
  - Processo del binning



Rimedi per hold violation

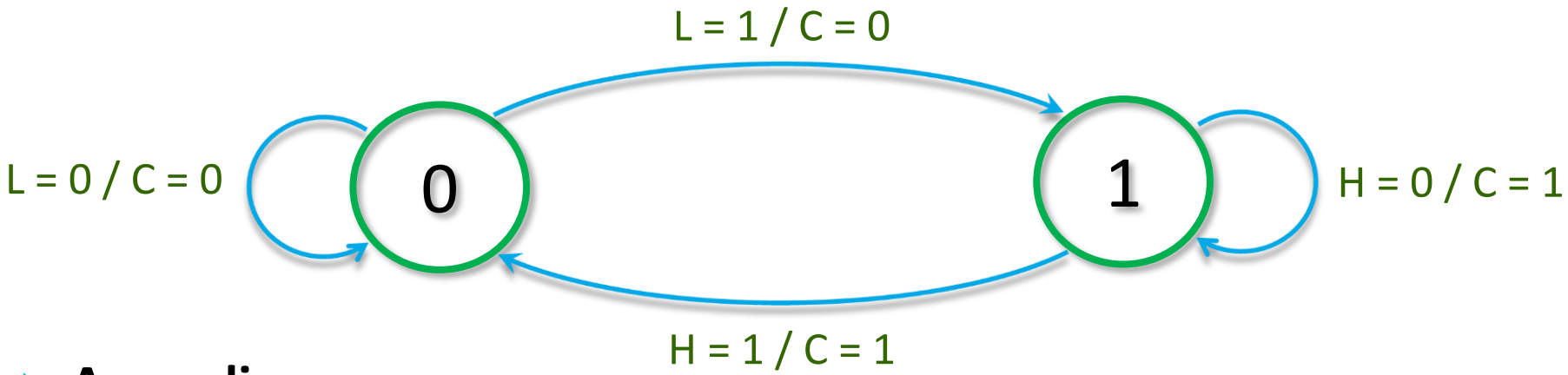
- **Significa che la logica combinatoria è troppo veloce**
  - L'uscita di un registro si propaga all'ingresso di un altro troppo in fretta (per esempio in uno shift register)
  - Si gioca sullo stesso fronte, non è possibile rimediare modificando il clock
  - Occorre rallentare la logica, aggiungendo buffer non invertenti
  - No binning, se un circuito presenta hold violation va buttato



Macchine di Mealy e di Moore

- **Mealy**
  - E’ il caso più generale in cui le uscite dipendono sia dallo stato corrente che dagli ingressi
- **Moore**
  - Caso specifico in cui le uscite dipendono solamente dallo stato corrente e non dagli ingressi
  - In questo caso le uscite non cambiano durante l’intero ciclo di clock (perché non cambia lo stato corrente)
  - Poichè dipendono solo dallo stato, si può scrivere il valore delle uscite direttamente nello stato corrispondente invece che nelle transizioni

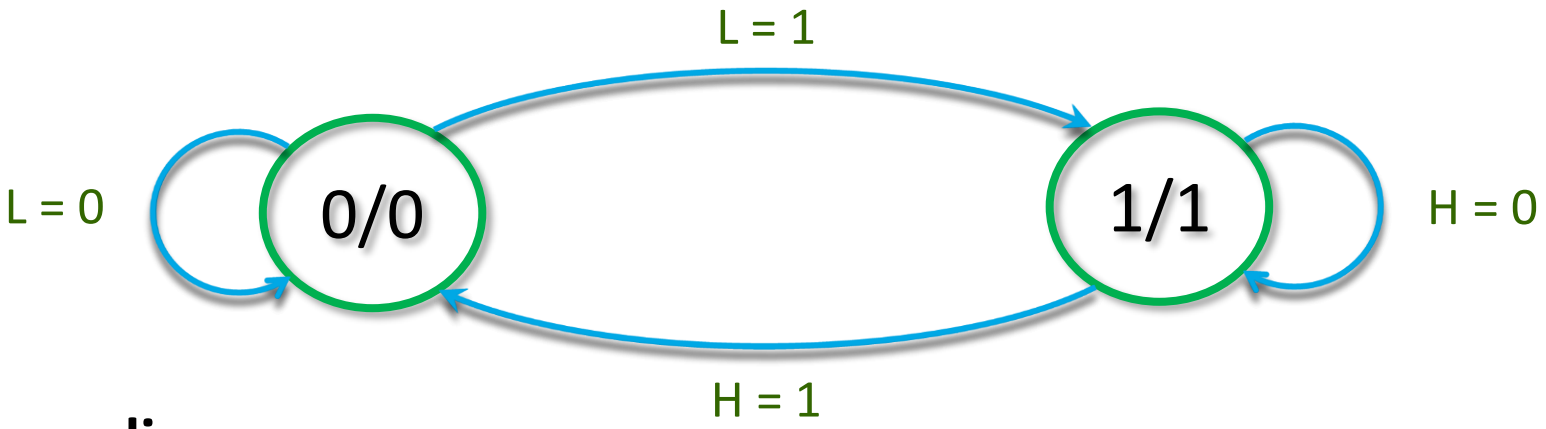
Macchina di Moore



- **Accendiamo o spegnamo la caldaia solo dopo aver fatto la transizione**
  - C'è potenzialmente un ritardo di un ciclo di clock dalla variazione degli ingressi a quella delle uscite

Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	*H*	*L*	X_{n+1}	*C*
0	-	0	0	0
0	-	1	1	0
1	0	-	1	1
1	1	-	0	1

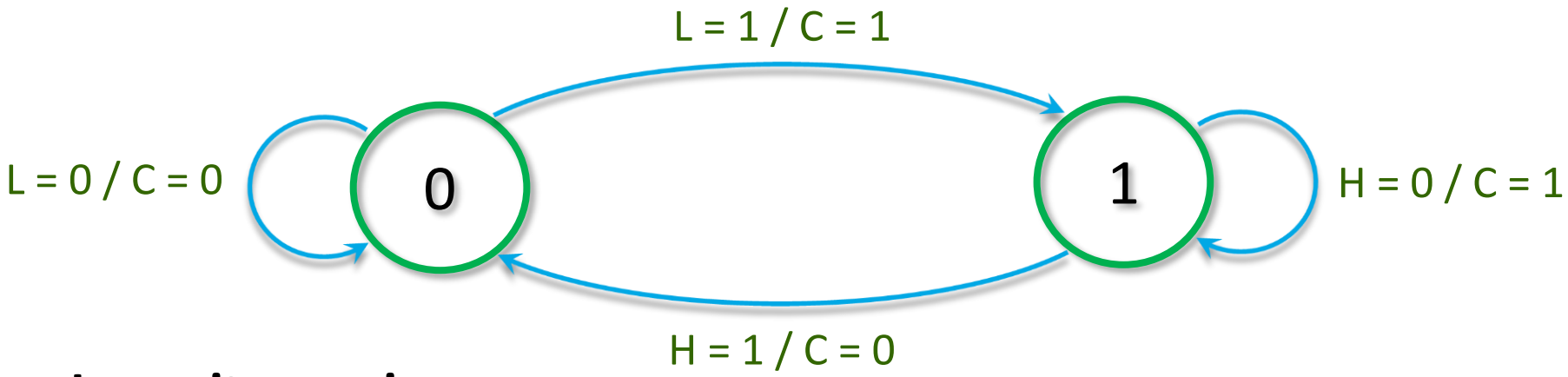
Macchina di Moore



- **Accendiamo o spegnamo la caldaia solo dopo aver fatto la transizione**
  - C'è potenzialmente un ritardo di un ciclo di clock dalla variazione degli ingressi a quella delle uscite

Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	*H*	*L*	X_{n+1}	*C*
0	-	0	0	0
0	-	1	1	0
1	0	-	1	1
1	1	-	0	1

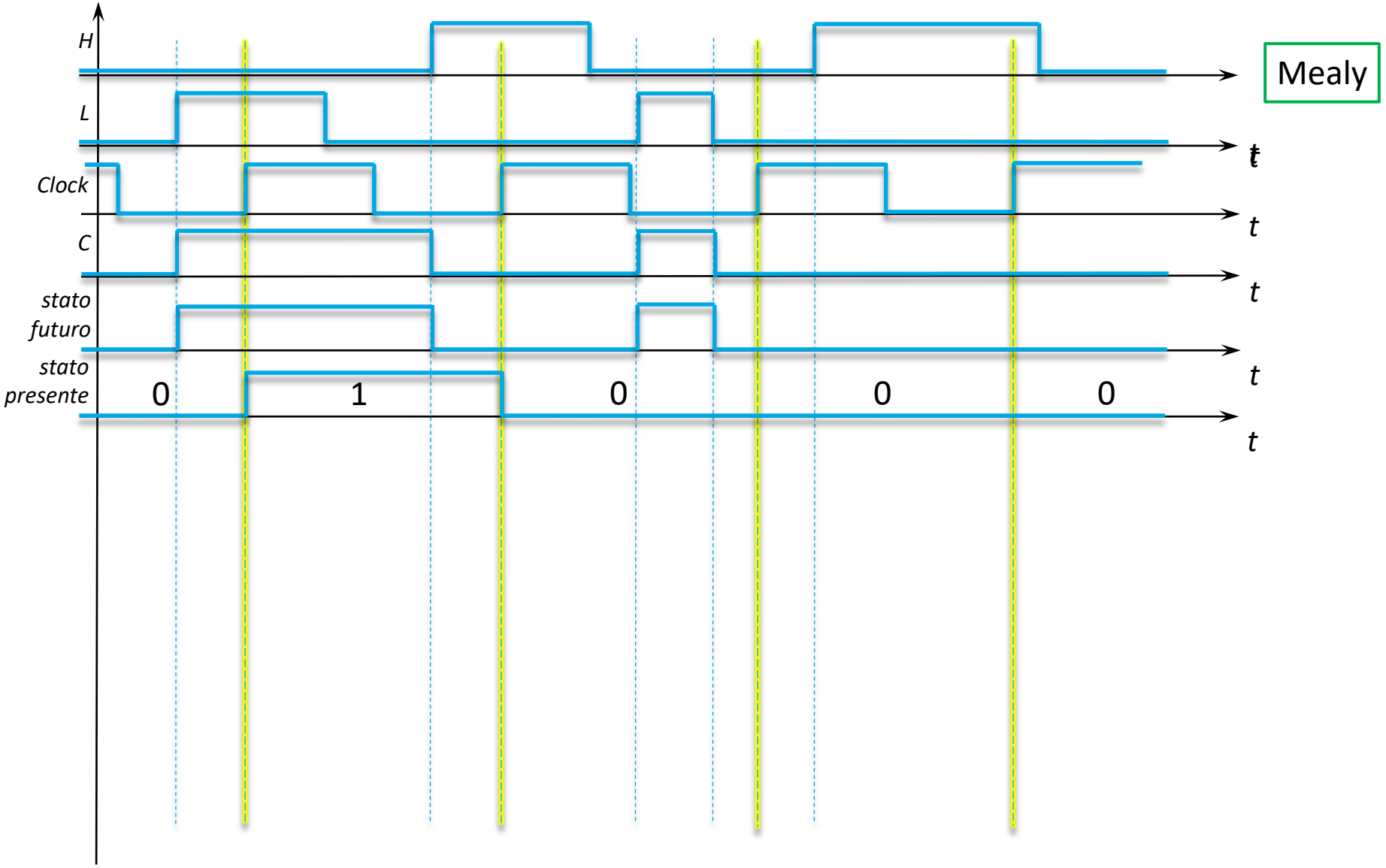
Macchina di Mealy



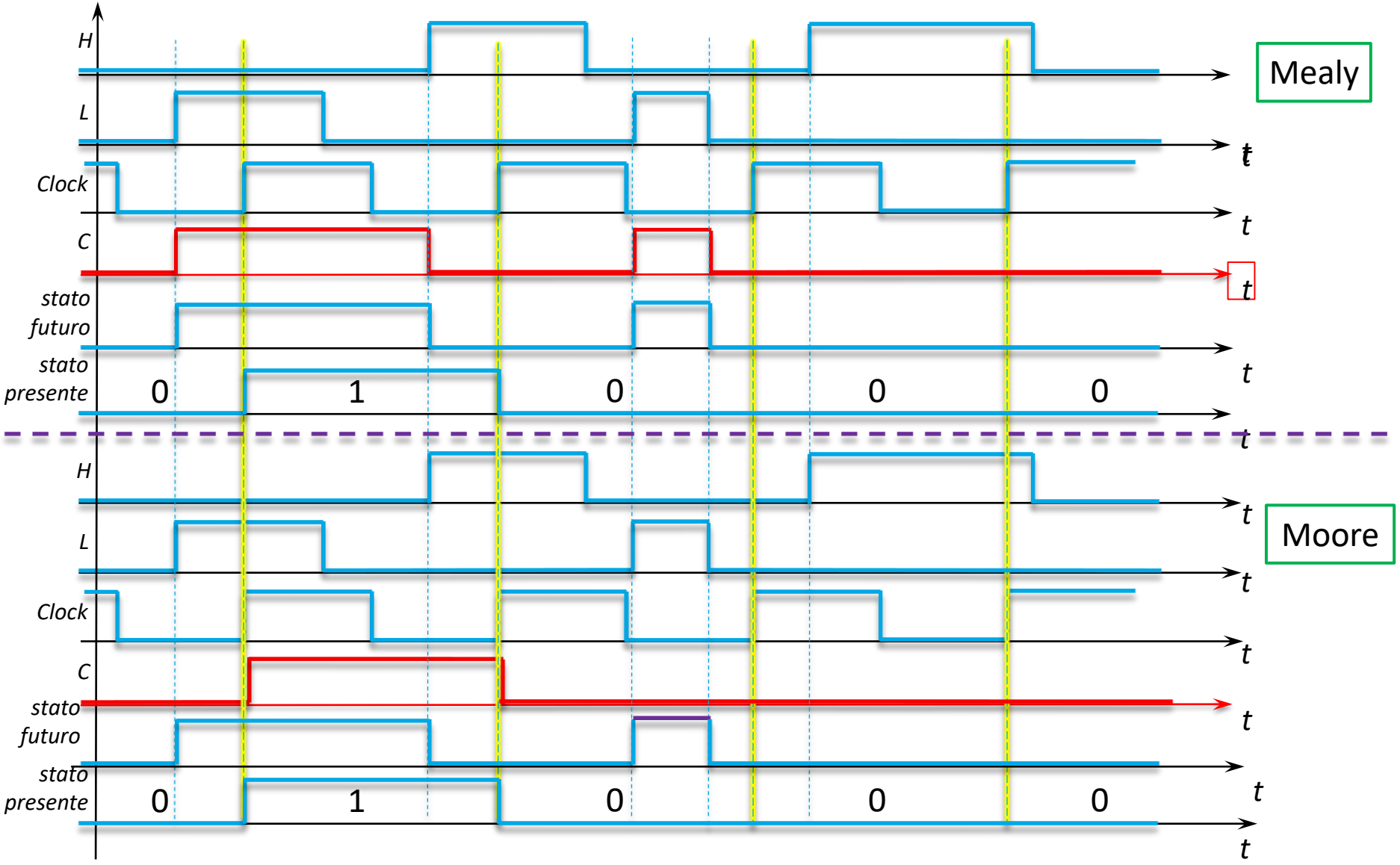
- **Le uscite reagiscono immediatamente alle variazioni degli ingressi**
  - Non c'è il ritardo
  - Il circuito ovviamente cambia
  - $C = L + XH'$

Stato presente	*Ingressi*		*Stato futuro*	*Uscite*
X_n	H	L	X_{n+1}	C
0	-	0	0	0
0	-	1	1	1
1	0	-	1	1
1	1	-	0	0

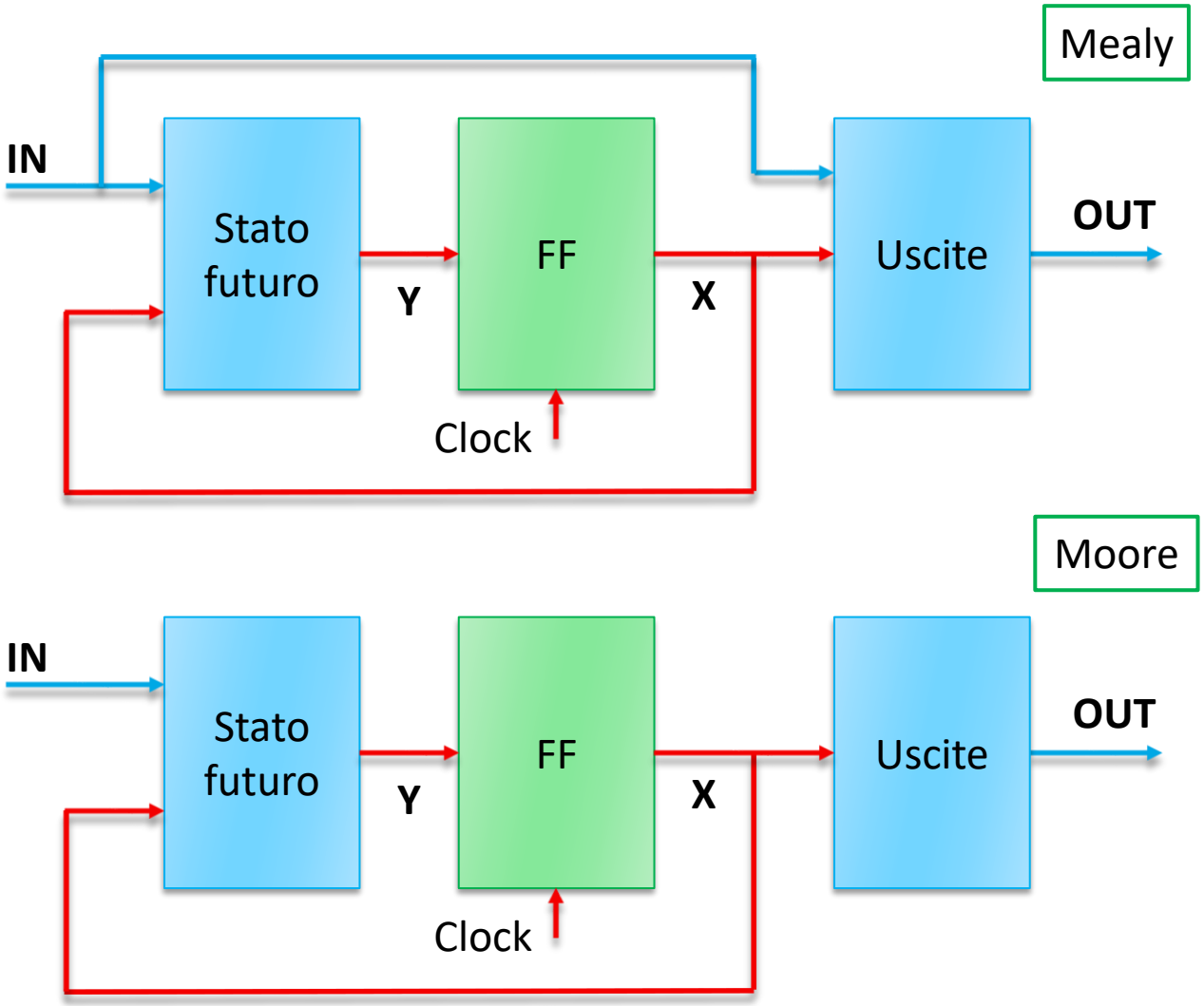
Diagrammi temporali (senza ritardi)



Diagrammi temporali (senza ritardi)

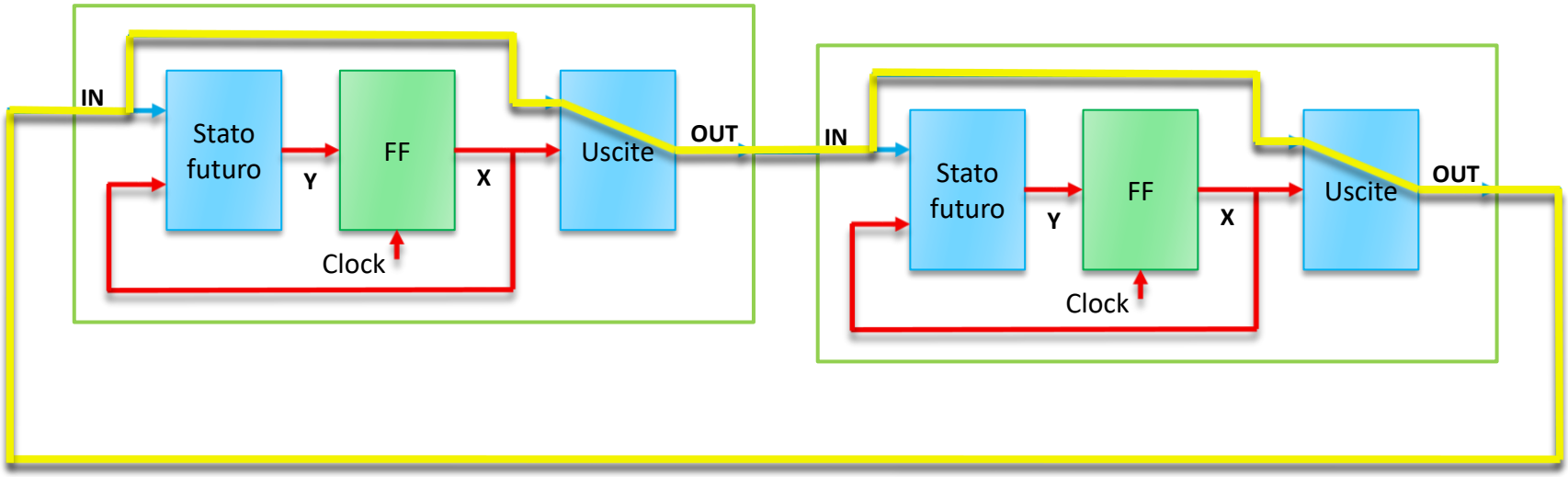


Struttura Mealy e Moore



Potenziali problemi con Mealy

- **Abbiamo visto per ora solo macchine a stati isolate**
  - Ma in generale più macchine a stati possono essere usate contemporaneamente
  - Le uscite di una diventano gli ingressi dell'altra
  - E spesso vice versa
- **Si possono formare degli anelli di feedback combinatori!**
  - Attenzione a possibili instabilità
- **Il problema non si può presentare usando macchine di Moore**



Metodo di progetto

Progetto di macchine a stati

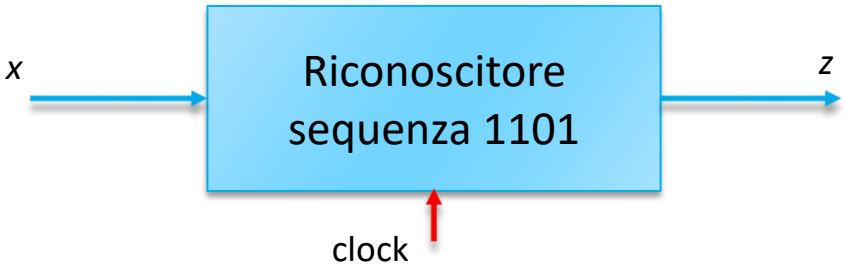
- **Abbiamo visto come analizzare un circuito sequenziale e derivare la corrispondente macchina a stati**
  - Procedimento abbastanza meccanico
  - Anche un computer lo saprebbe fare
- **Nel progetto si esegue invece il procedimento contrario**
  1. Si stabiliscono gli stati del sistema
  2. Si disegna una diagramma degli stati (le transizioni)
  3. Si codificano gli stati
  4. Si scelgono gli elementi di memoria da utilizzare
  5. Si deriva la relativa tabella degli stati
  6. Si fanno le mappe di Karnaugh
  7. Si sintetizza il circuito

Cos'è uno stato?

- **Identificare gli stati del sistema è la parte più complessa del progetto**
  - Uno stato rappresenta la storia passata degli ingressi
  - Si ricorda di cosa è successo
  - Non deve però necessariamente ricordare tutto
- **Esempi**
  - Si crea uno stato per ricordare che è stato premuto un bottone, ma non ricordiamo necessariamente quante volte
  - Uno stato può indicare che un bit di start di una trasmissione non è ancora stato ricevuto
  - Uno stato ricorda che è stato richiesto il bus di sistema, ma l'arbitro del bus non ha ancora concesso l'accesso
  - Due stati ricordano se un ascensore sta salendo o scendendo
  - Vari stati ricordano quante volte è stata eseguita una operazione
- **Lo stato è quindi una astrazione della storia passata**
  - Le scelte possono essere diverse, e tutte funzionalmente equivalenti
  - Macchine a stati con diverso numero di stati possono fare esattamente la stessa cosa

Esempio: riconoscitore di sequenze

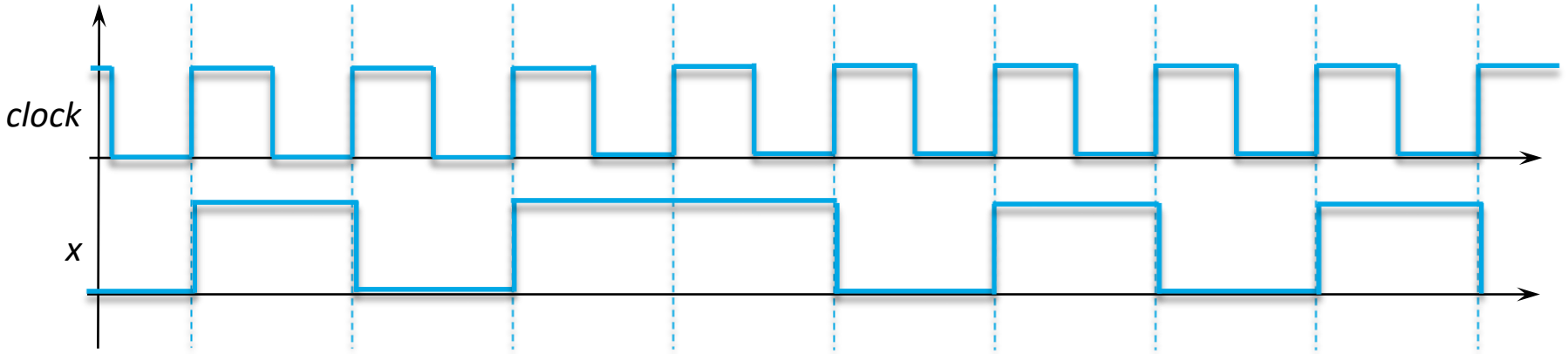
- **Si vuole realizzare un sistema con un ingresso *x* ed una uscita *z* che riconosca una sequenza di bit in ingresso**
  - Ad ogni ciclo di clock si presenta un nuovo bit (0 o 1) all'ingresso *x*
  - Il sistema deve riconoscere la sequenza di bit 1101 sull'ingresso
  - La sequenza va riconosciuta in una qualunque posizione
  - L'uscita deve essere normalmente a 0, e deve essere messa a 1 durante il ciclo di clock in cui si presenta l'ultimo bit a 1 della sequenza
- **A che serve?**
  - Per esempio per riconoscere un semplice codice di un apricancello



Esempio: riconoscitore di sequenze

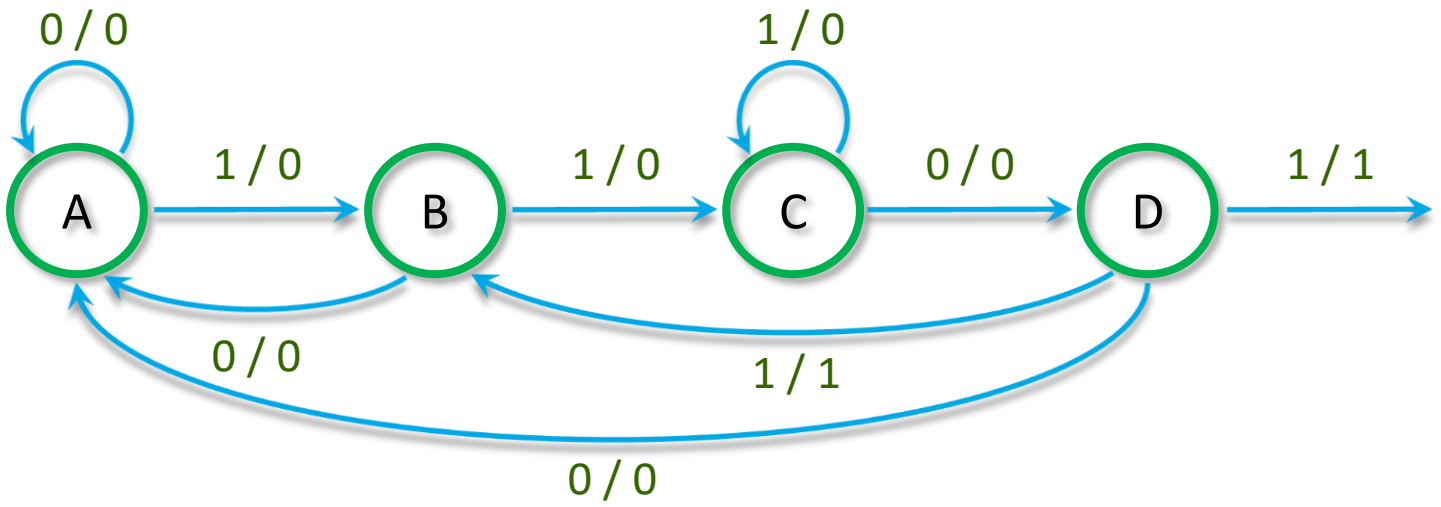
- **Per esempio**

10110101011101010010110110100011010101000011100



1. Identificazione degli stati
2. Diagramma degli stati

- **Dobbiamo ricordare i bit visti sull'ingresso**
  - Uno stato si ricorda che non si è ancora visto nessun bit della sequenza
  - Un secondo stato si ricorderà che è stato visto un 1
  - Un terzo stato che è stato visto un 1 seguito da un altro 1
  - Un quarto riconosce 110
  - Infine si termina la sequenza 1101
  - A questo punto si ricomincia



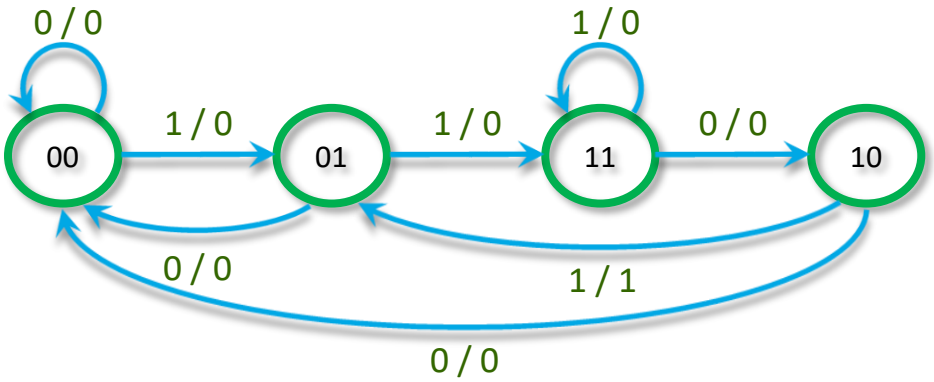
3. Codifica degli stati

- **Abbiamo dato agli stati dei nomi simbolici**
  - Il modo in cui codifichiamo lo stato non ha nessuna influenza sul funzionamento logico del circuito
  - Possiamo disegnare un diagramma a stati senza neanche indicare la codifica
  - La codifica avrà influenza invece sul modo in cui si realizza il circuito
- **Numero di bit di stato**
  - Per codificare *n* stati abbiamo bisogno di almeno $\lceil \log_2(n) \rceil$ flip flop
  - Potremmo in realtà usarne di più
  - Non tutti i codici sarebbero usati
- **Per il momento usiamo il numero minimo di bit**
  - Per esempio usiamo il codice Gray
  - A = 00, B = 01, C = 11, D = 10

4. Scelta degli elementi di memoria

- **Abbiamo a disposizione vari tipi di elementi di memoria**
  - A seconda del tipo il circuito sarà differente
  - Analizzeremo questo problema tra poco
  - Per il momento **scegliamo dei flip flop di tipo D edge triggered**
  - L'uscita del flip flop diventa uguale al suo ingresso al fronte attivo del clock

5. Derivazione della tabella degli stati

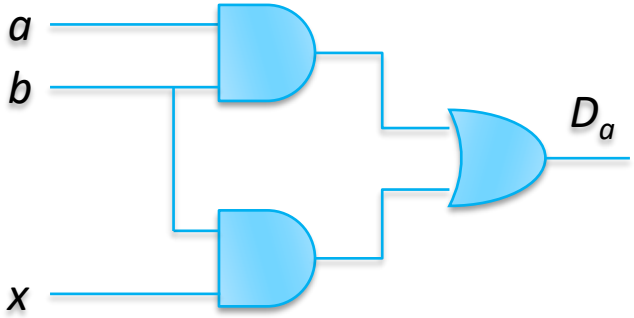


Stato presente		*Ingresso*	*Stato futuro*		*Uscita*
a	*b*	*x*	*a*	*b*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	0

6. Mappe di Karnaugh

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

$D_a = xb + ab$

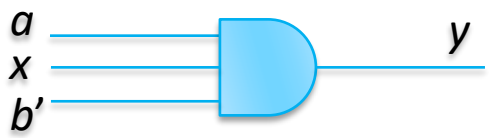


x \ ab	00	01	11	10
0	0	0	0	0
1	1	1	1	1

$D_b = x$

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

$y = xab'$



7. Circuito

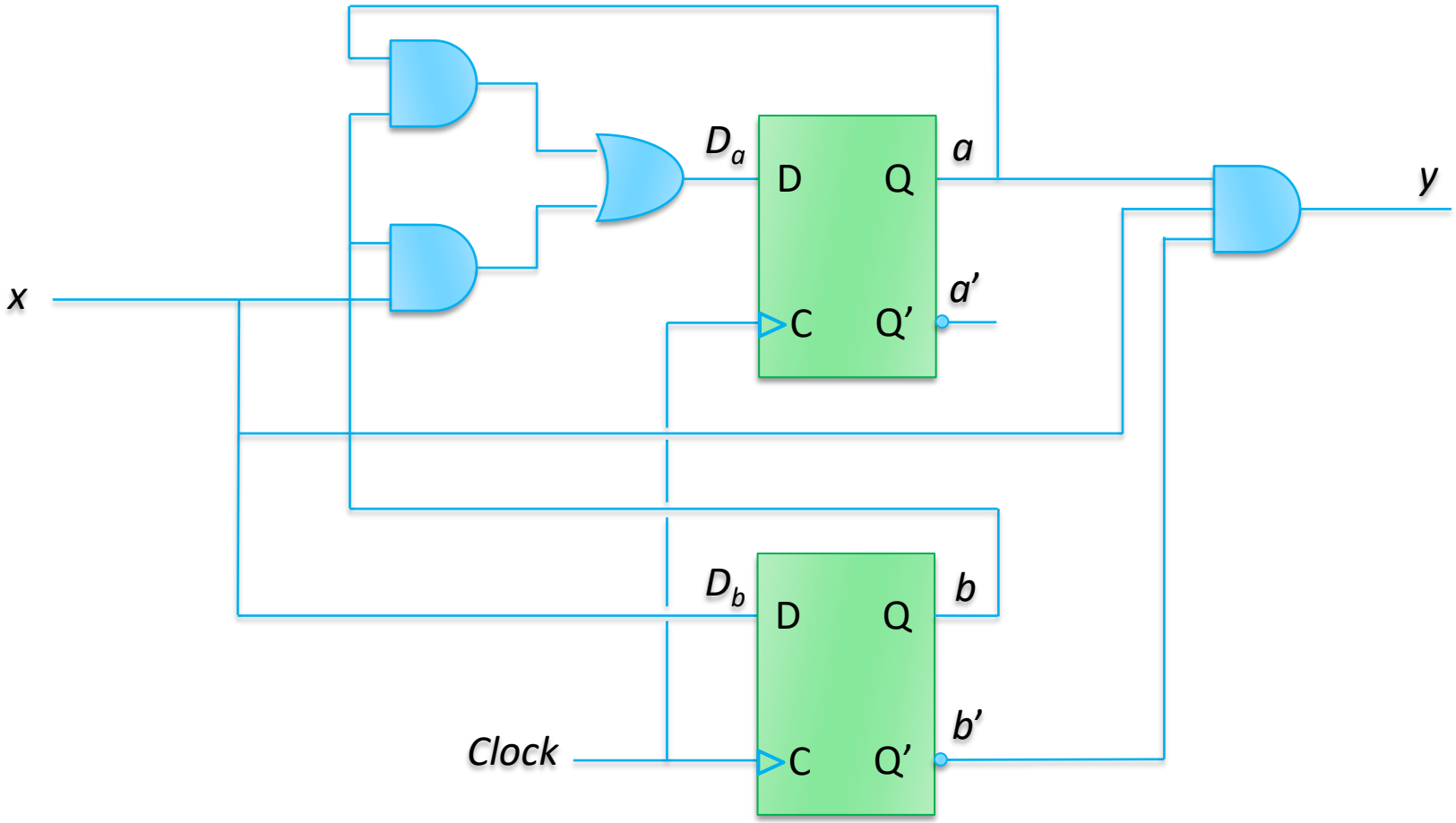
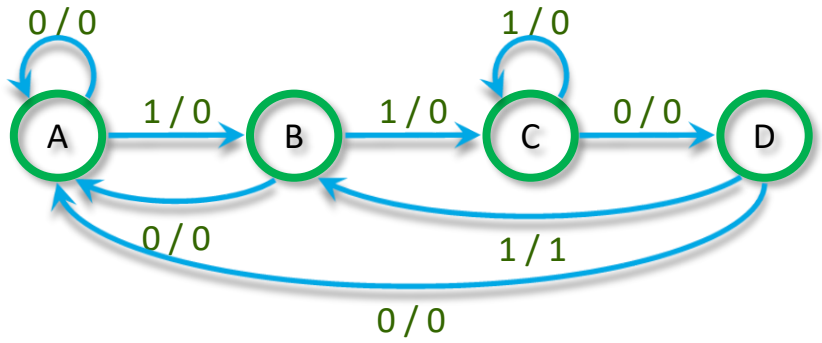
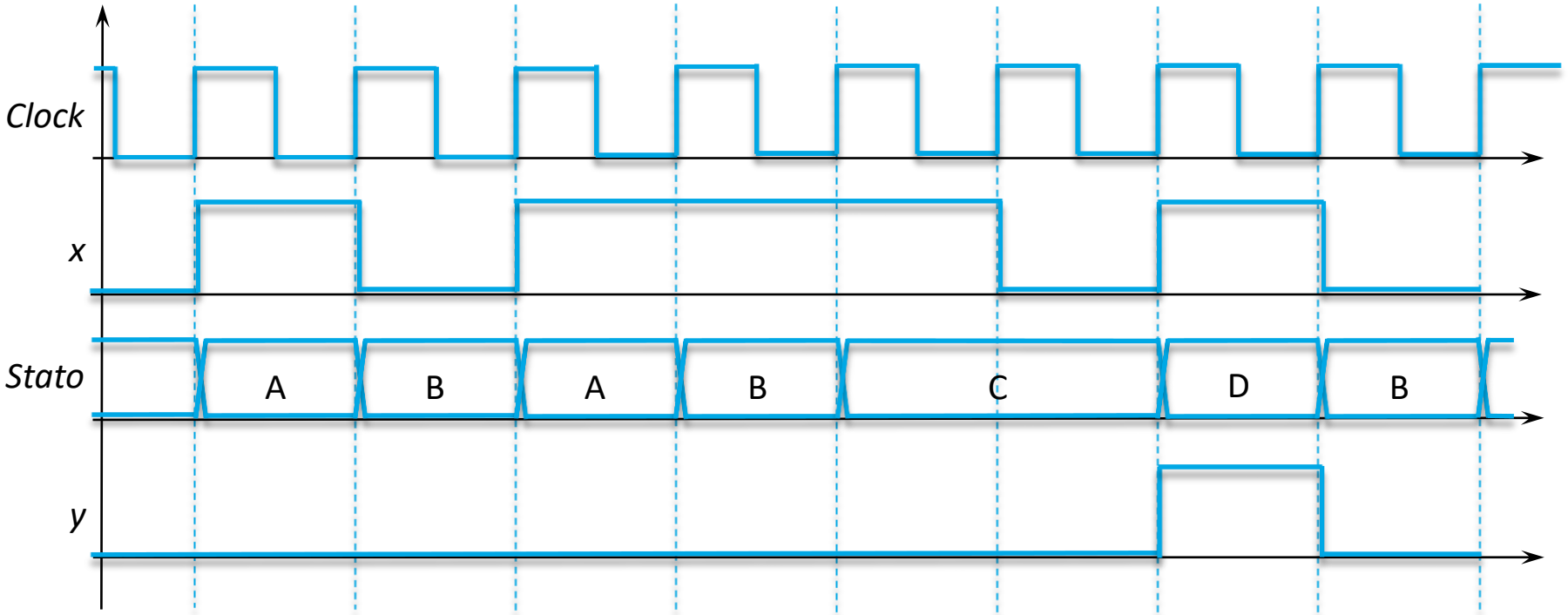


Diagramma temporale



- **Segnali correlati fusi in un diagramma**
  - Per esempio *a* e *b*
  - Si indica il cambiamento e il valore simbolico o numerico

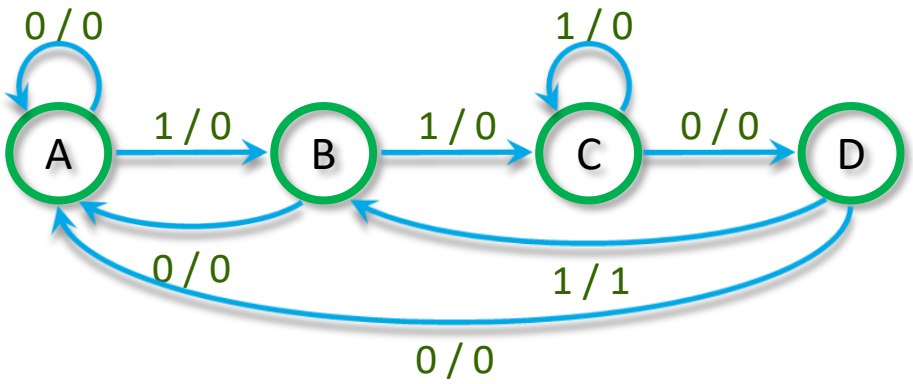


3. Codifica one-hot

- **Usiamo 4 flip flop**
  - Scegliamo la codifica in modo che ad ogni stato corrisponda uno ed un solo flip flop a 1
  - A = 1000, B = 0100, C = 0010, D = 0001
  - Indichiamo lo stato con le variabili *abcd*
  - Con 4 flip flop potremmo codificare 16 stati, ma ne usiamo solamente 4
- **Perché**
  - Può essere utile per avere uno stato già decodificato
  - Otteniamo le uscite molto più in fretta che a passare per una rete combinatoria di decodifica
  - Inoltre si può anche semplificare la rete di calcolo dello stato futuro (ma non necessariamente)

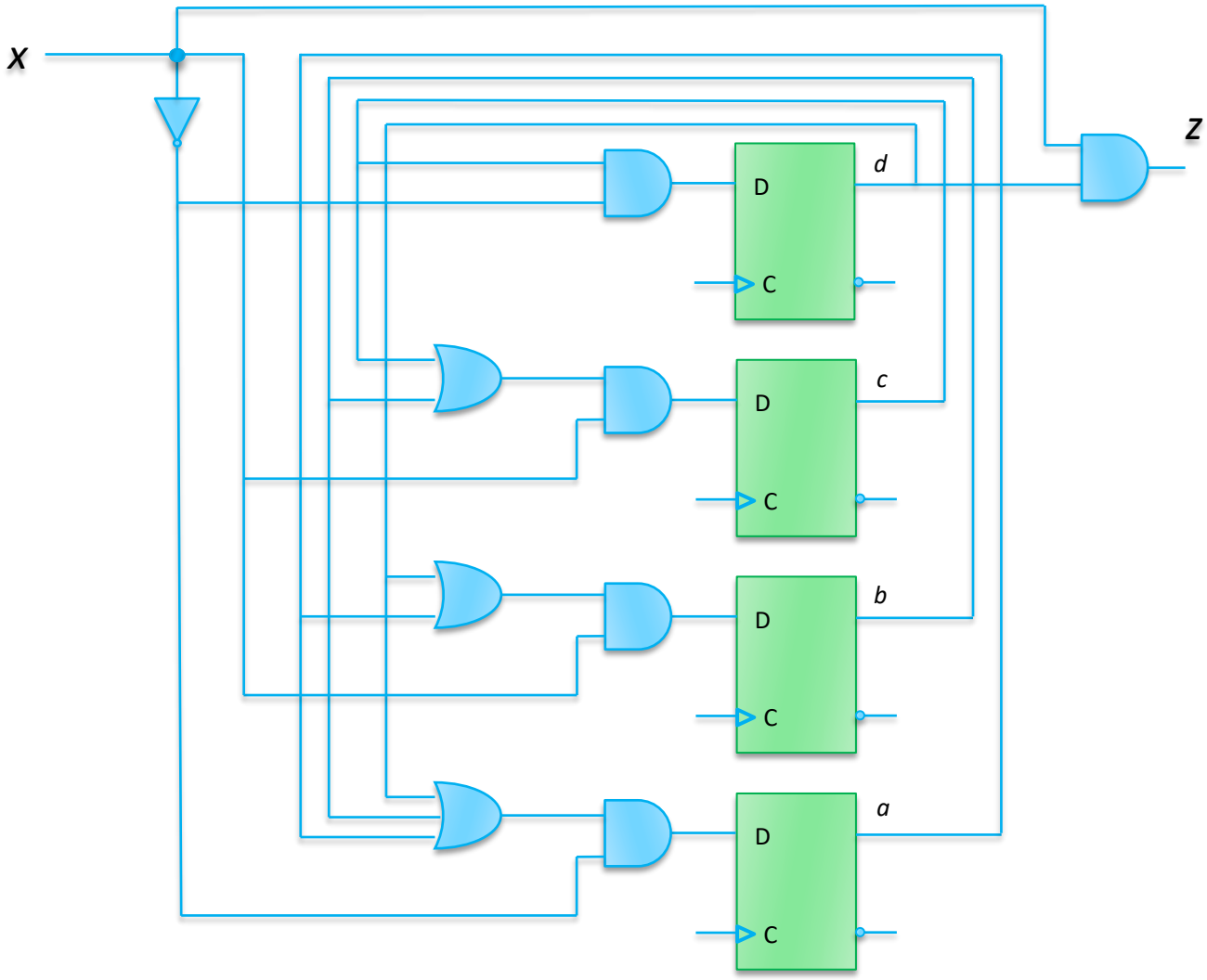
Tabella degli stati

- **Con 4 variabili di stato e 1 ingresso sono 32 righe**
  - Proviamo a derivare le espressioni a mano
- **Entriamo in A se e solo se**
  - Siamo in A e $x = 0$: ax'
  - Siamo in B e $x = 0$: bx'
  - Siamo in D e $x = 0$: dx'
- **Entriamo in B se e solo se**
  - Siamo in A e $x = 1$: ax
  - Siamo in D e $x = 1$: dx
- **Entriamo in C se e solo se**
  - Siamo in B e $x = 1$: bx
  - Siamo in C e $x = 1$: cx
- **Entriamo in D se e solo se**
  - Siamo in C e $x = 0$: cx'



Circuito

$D_a = x'(a + b + d)$
$D_b = x(a + d)$
$D_c = x(b + c)$
$D_d = cx'$



Take away

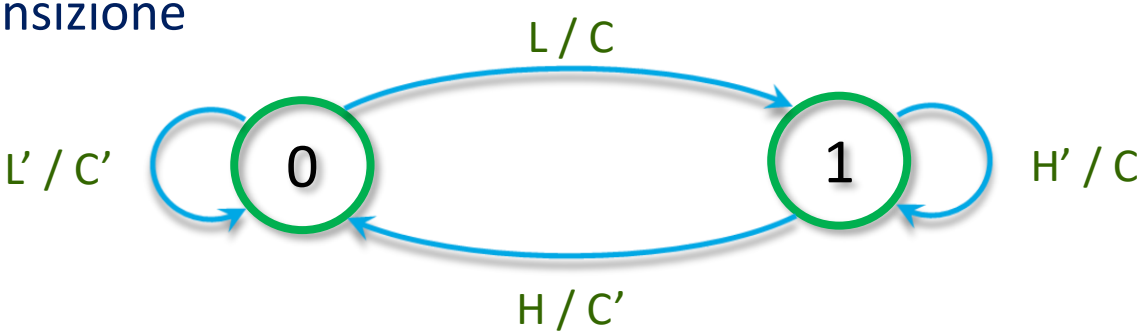
- **Il diagramma degli stati è una forma per rappresentare un circuito sequenziale**
  - Si avvicina al nostro modo di pensare, evidenziando gli stati del sistema e le sue possibili transizioni
  - Equivalente alla tabella degli stati, dalla quale è meccanico derivare il circuito corrispondente
- **La stessa funzione sequenziale può essere rappresentata da diagrammi differenti**
  - Esiste una macchina a stati con un numero minimo di stati
  - Spesso è più importante poter capire il diagramma
- **La codifica degli stati non influenza la funzione logica sequenziale**
  - Ha invece importanza per quanto riguarda l'implementazione
  - Vedremo che alcune codifiche si comportano meglio di altre

Algorithmic State Machines

Alla ricerca di strumenti di progetto efficienti

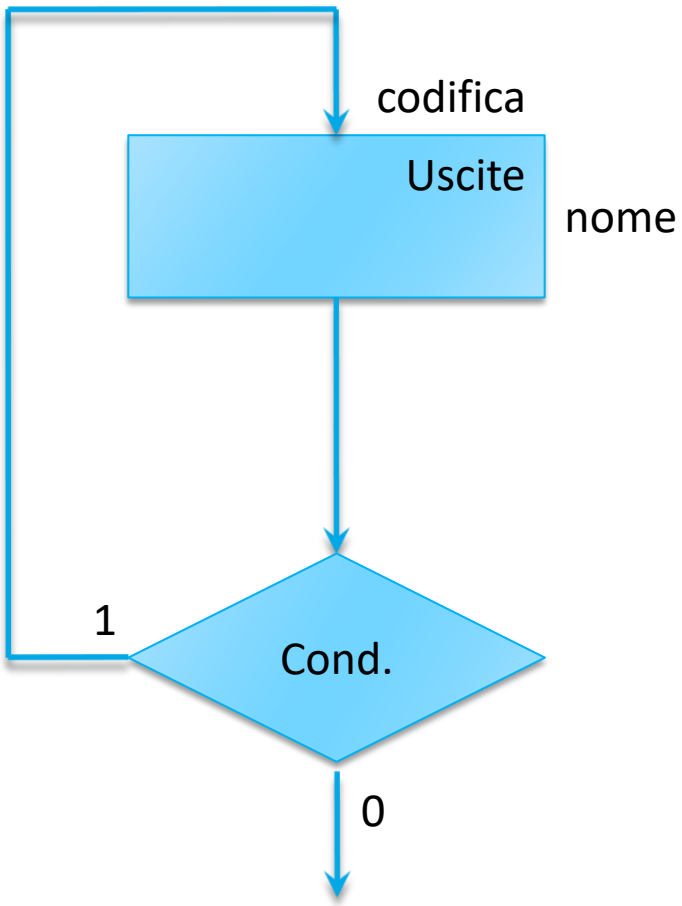
Varianti dei diagrammi

- **Il diagramma degli stati è uno strumento di progetto**
  - Lo si usa per rendere chiara la funzione di un sistema
  - Spesso viene personalizzato dal progettista per semplificarne la stesura e la lettura
  - Esistono quindi molteplici varianti
- **Utile cercare di semplificare il modo in cui sono espresse le transizioni**
  - Quando le variabili sono tante, l'enumerazione dei loro valori sulle transizioni può diventare problematico da interpretare
  - Invece dei valori, si può indicare una condizione sulle variabili di ingresso che attivino la transizione
  - If (condizione) then transizione



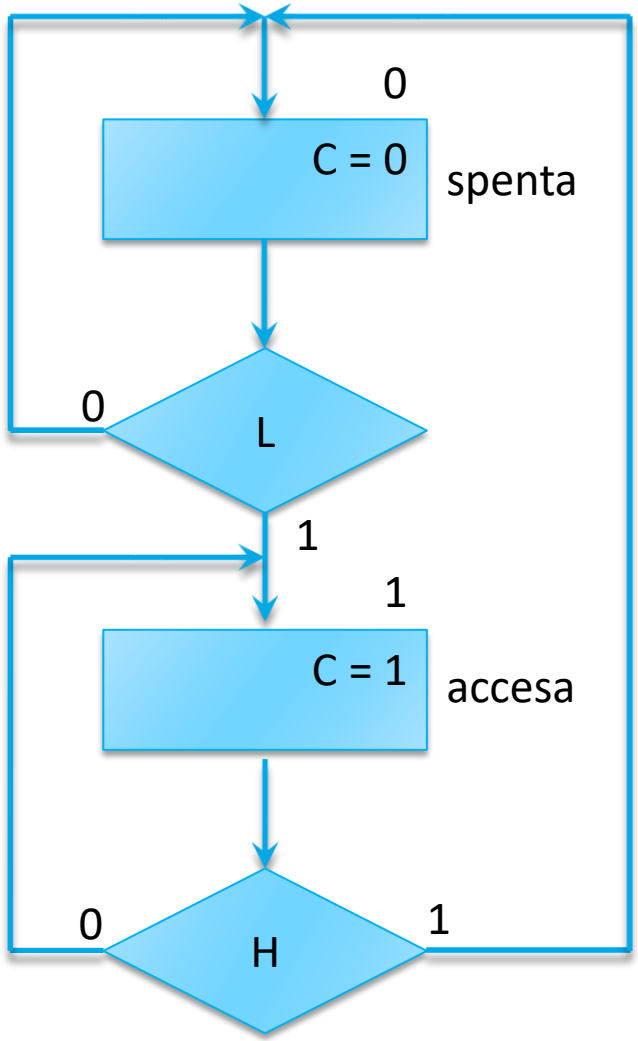
Algorithmic State Machines (ASM)

- **Una variante è il diagramma ASM**
  - Simile ad un diagramma di flusso
- **Stati**
  - Rettangoli con un nome
  - Eventuale codifica, anche simbolica
  - Valori delle **uscite** (Moore)
- **Condizioni**
  - Rombi con una condizione **sugli ingressi**
  - Due rami per condizione vera o falsa
- **Transizioni**
  - Frecce che da uno stato arrivano ad un altro, eventualmente passando per delle condizioni
  - Le transizioni entrano normalmente da sopra ed escono da sotto



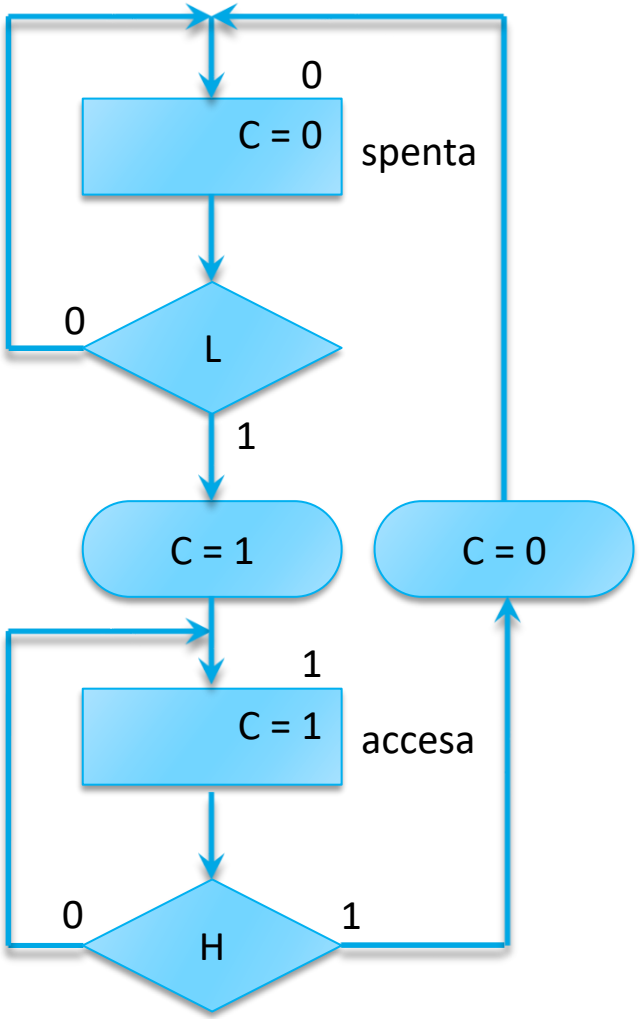
Tanto per fare un esempio...

- **Due stati per caldaia accesa e spenta**
  - Usiamo il modello di Moore
- **Due condizioni per verificare il superamento delle soglie**
  - Notare che due condizioni sono sufficienti per esprimere 4 transizioni
  - Ogni ramo in uscita da una condizione è una transizione diversa
- **Il diagramma fornisce le stesse informazioni di quello con i cerchi**
  - L'uso dell'uno o dell'altro è una questione di preferenze personali



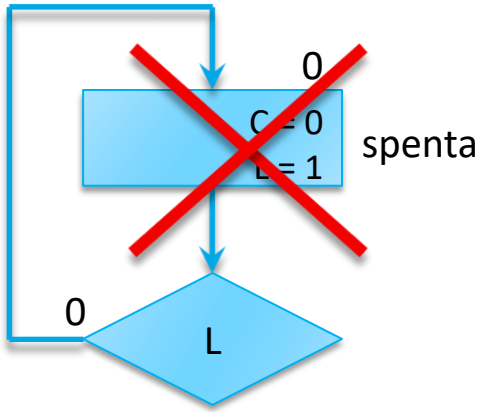
Uscite condizionate

- **Usate per rappresentare macchine di Mealy**
  - Un rettangolo ad angoli tondi specifica il valore delle uscite sulle transizioni
    - Ricordate che l'uscita assume il valore corrispondente mentre il circuito è nello **stato di partenza** della transizione
  - Il valore di default è dato nello stato
  - Il blocco di uscita condizionata lo si aggiunge quando si intende usare un valore diverso
  - L'uscita condizionata è attiva solo sulla base della condizione da cui dipende

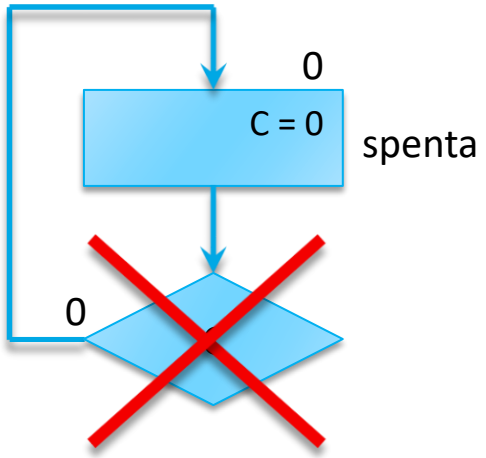


Regole

- **Non assegnate MAI un valore agli ingressi in uno stato o in una uscita condizionata**
  - Se è un ingresso, non ne potete decidere il valore

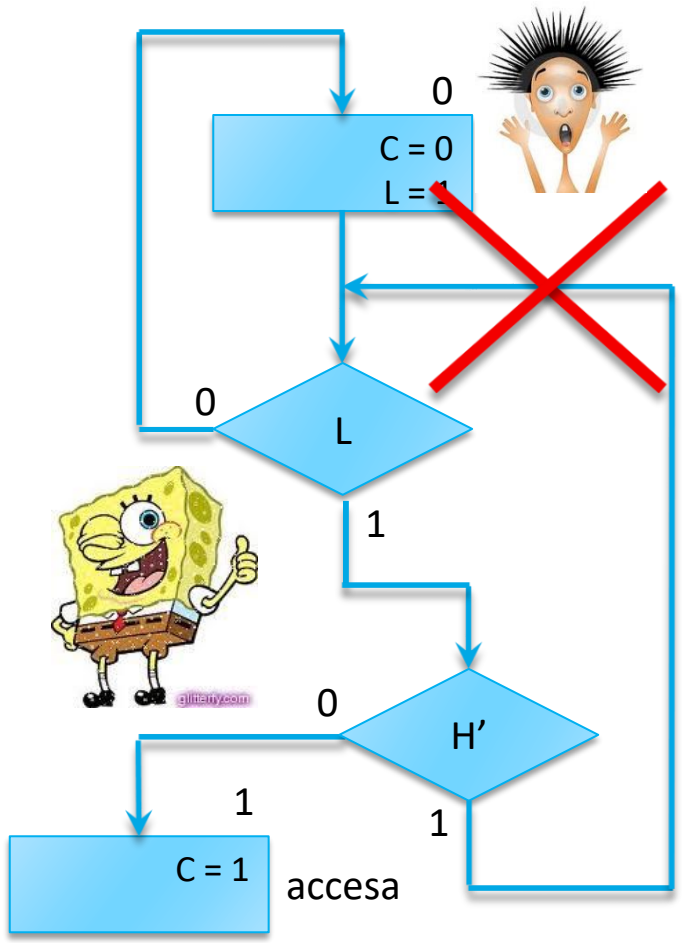


- **Non usate una uscita in una condizione di una transizione**
  - Se è una uscita sapete quanto vale, quindi è inutile guardarne il valore



Regole

- **Potete far seguire una condizione da un'altra condizione**
  - Molto utile, permette di fattorizzare le condizioni sulle transizioni e di scrivere meno espressioni
  - Rende quindi il diagramma più leggibile
- **Ma: le transizioni, alla fine, <u>devono</u> finire in uno stato**
  - Quindi non fate mai un ciclo in cui non compare uno stato
  - Una volta dentro non ci si può più uscire, perché i valori non cambiano fino al prossimo stato



Take away

- **Ci sono diverse varianti di diagrammi degli stati**
  - Usate quella con cui vi trovate meglio
  - L'importante è sapere precisamente cosa vogliono dire
- **Molti trucchi usati per semplificarne la scrittura**
  - Condizioni di default: se non scrivo niente, vuol dire che l'uscita è 0 (per esempio)
  - Espressioni booleane per esprimere condizioni
- **Importante è mettersi d'accordo**
  - Se fate il progetto con qualcun altro, dovete interpretare il diagramma allo stesso modo
  - Va bene fare cose strane, ma scrivetelo!!