Corso di Laurea in
Informatica

Sintesi di Reti Logiche Sequenziali

Architettura dei Calcolatori

Prof. Andrea Marongiu
andrea.marongiu@unimore.it

Anno accademico 2018/19

Reti sequenziali

Reti sequenziali: reti logiche in cui in ogni istante le uscite (e il comportamento interno) dipendono non solo dalla configurazione degli ingressi in quell'istante, ma anche dalle configurazioni degli ingressi negli istanti precedenti.

- Nelle reti sequenziali il comportamento dipende dalla storia passata; devono conservare **memoria** degli eventi passati nel proprio **stato** interno.
- Variazioni delle configurazioni di ingresso modificano, oltre che le uscite, anche lo stato interno. Lo stato interno attuale si dice **stato presente.** In seguito alla variazione degli ingressi il sistema può calcolare in ogni istante quello che sarà lo **stato futuro.**
- Quando avviene l'aggiornamento dello stato presente allo stato futuro appena calcolato?
- Le Reti sequenziali possono essere asincrone o sincrone:
  - **asincrone**, se le variazioni delle configurazioni di ingresso vengono sentite e modificano lo stato e le uscite in qualsiasi istante
  - **sincrone**, se le variazioni delle configurazioni di ingresso vengono sentite e modificano lo stato e le uscite solo in presenza di un opportuno evento di sincronizzazione
- L'evento di sincronizzazione è normalmente associato ad un segnale attivo (il **clock**) o al cambiamento dello stato del segnale di sincronizzazione (fronte del clock)

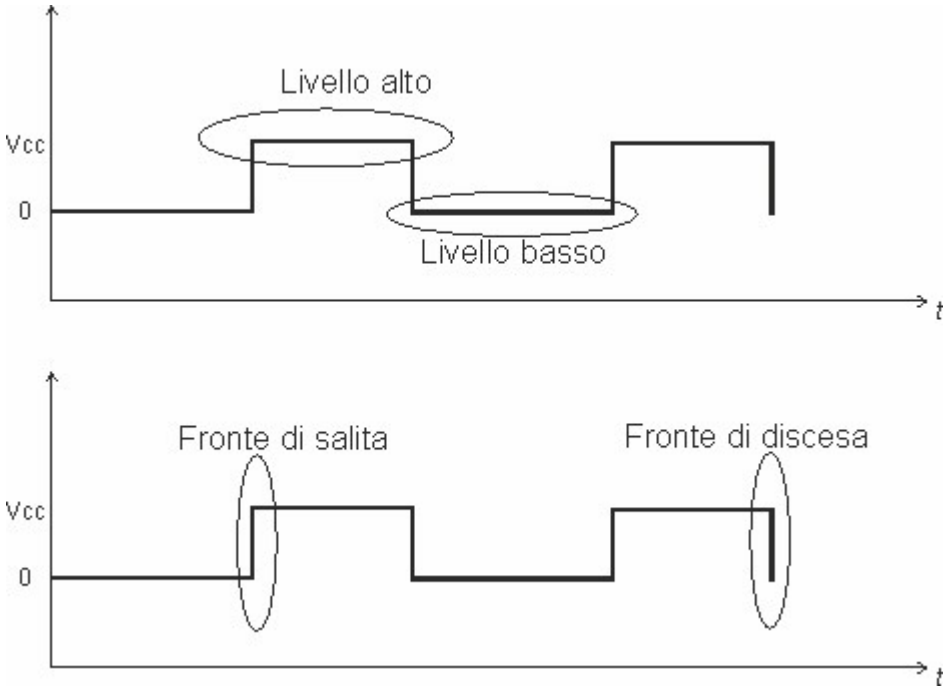
Il Clock

- Il segnale di *clock* è generato da un circuito (realizzato con un opportuno cristallo) che emette un segnale impulsivo periodico con una precisa durata (*pulse width*) e con un preciso intervallo tra due impulsi consecutivi.

- Il clock è un segnale free-running ossia che continua indefinitamente (almeno finche' il sistema e' alimentato), di tipo periodico, con un periodo detto **tempo di clock** Tck (clock cycle time); il suo reciproco è la **frequenza di clock** fck o f. Una rete che ha la frequenza di 100MHz ha un ciclo di clock di 10ns.

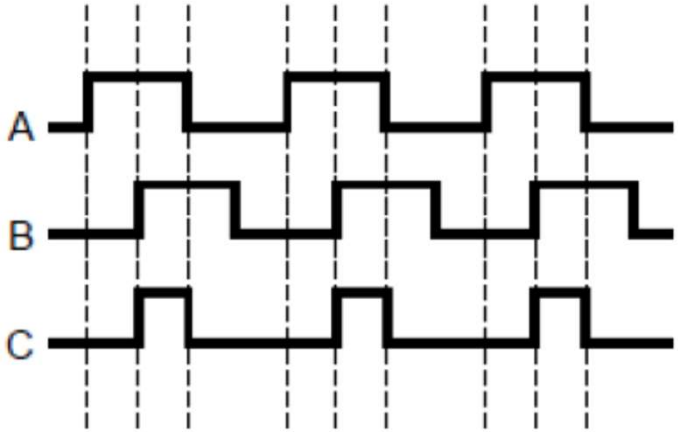
Il Clock (2)

- Si definiscono livelli (alto e basso) e fronti o edge (di salita e di discesa) le quattro parti della forma d'onda riportata in figura



Il Clock (3)

- Nei calcolatori il segnale di clock sequenzializza tutti gli eventi. Spesso nel calcolatore si usa oltre al clock primario dei clock secondari che sono sincroni ma che sono di dimensione minore (la metà) per eseguire più azioni nello stesso clock o maggiori (il doppio, il quadruplo) se alcune reti non sono sufficientemente veloci. Per questo si parla del clock della **CPU, del clock di sistema, o di clock multipli** (di frequenza).



- Nelle reti logiche ogni evento elementare si verifica in un ciclo di clock. Se l'evento si verifica mentre il clock è attivo (di solito alto) si dice che la logica lavora "a **livello**", se ogni evento, ogni transizione di stato e di uscite si verifica al cambiamento del clock si dice che l'evento è "**edge-triggered**" o "a **fronte**". Di solito si usa il fronte di salita, ma in alcuni casi si usano entrambi

- Le reti logiche che studieremo sono di tipo sincrone e normalmente di tipo *edge-triggered*.

Memoria binaria bistabile

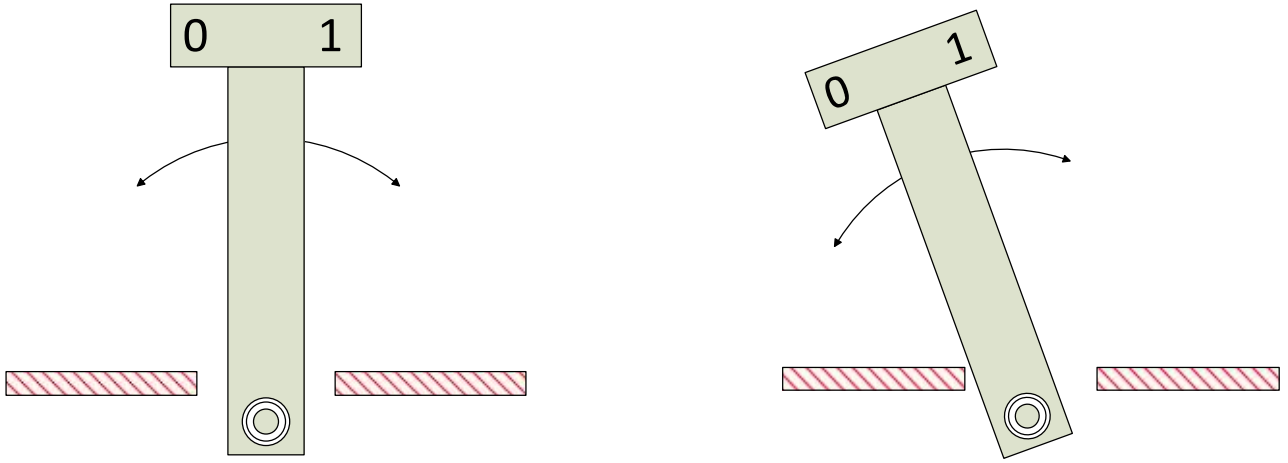
- Gli elementi di base delle reti sequenziali sono gli elementi di memoria chiamati **bistabili**, capaci di mantenere al loro interno il valore 0 o il valore 1. Questi elementi bistabili sono gli elementi di base capaci di mantenere 1 bit di memoria. E sono gli elementi di stato

Memoria binaria:
bistabile SR (Set-Reset)

- Come nelle reti combinatorie si definiscono i gate elementari, così per le reti sequenziali esistono blocchi elementari per memorizzare lo stato attuale

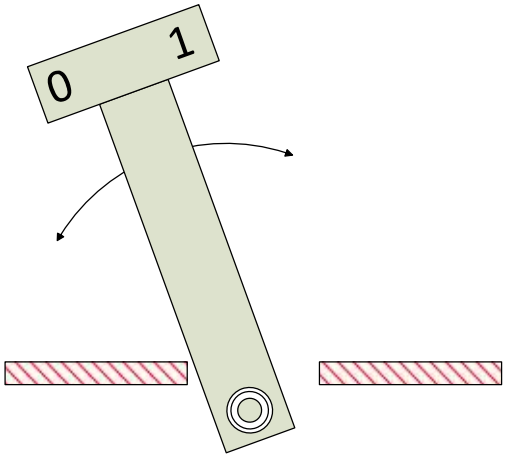
Memoria binaria (**bistabile** asincrono): elemento capace di memorizzare il valore di una variabile di stato binaria e di commutare alla presenza di un opportuna configurazione di ingresso.

Per capire meglio il funzionamento di un bistabile, usiamo una esempio reale.
Nel disegno è stilizzato un bistabile meccanico.
La parte fatta a T è libera di ruotare attorno al perno in basso.
I due blocchi tratteggiati in rosso funzionano da fermi. Dopo un eventuale
fase di equilibrio instabile, la parte a T «cadrà» a destra o a sinistra.



Memoria binaria: bistabile SR (Set-Reset)

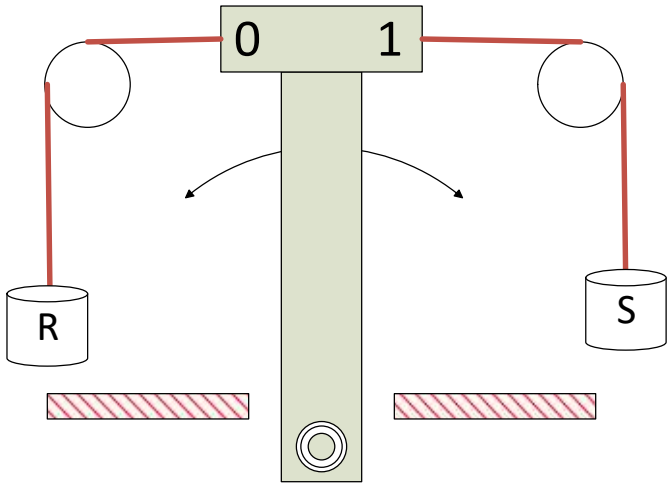
- Collocare l'oggetto a destra o a sinistra può servire per ricordarci qualcosa. Visto che la scelta è binaria, può servire per ricordarsi uno 0 o un 1. Una volta posizionato l'oggetto, starà fermo fino ad un prossimo comando.
- E' un banale esempio di memoria meccanica binaria.



- Supponiamo di avere collocato l'oggetto troppo in alto per poter essere manipolato. Cosa è possibile fare per modificare il suo stato?

Memoria binaria: bistabile SR (Set-Reset)

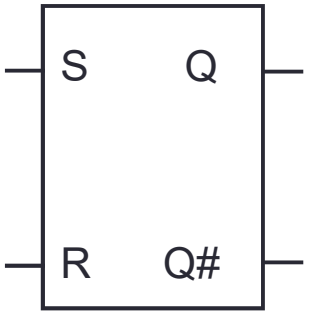
- Supponiamo di collegare, mediante appositi supporti, due corde. Chiameremo la corda a sinistra R e quella a destra S (vedi figura)



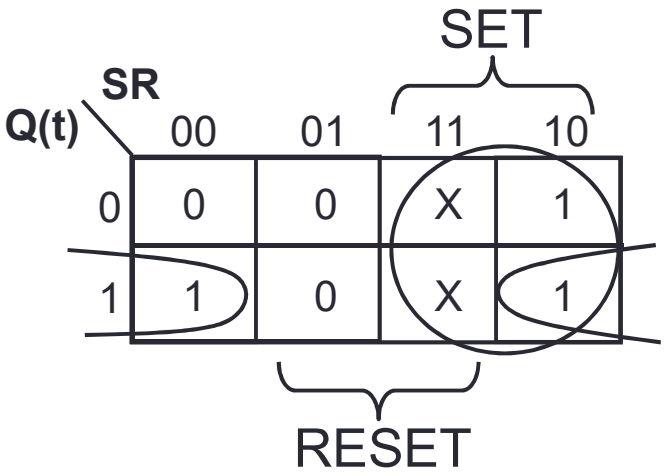
- Tirando la corda R si porta l'oggetto nella posizione 0, tirando la corda S nella posizione 1. Senza toccare le corde l'oggetto starà fermo nella posizione in cui è.
- Attenzione: tirando entrambe le corde, l'oggetto si potrebbe portare nella situazione al centro, di equilibrio instabile. Rilasciando le corde l'oggetto cadrà a sinistra o a destra in modo del tutto casuale e impredicibile. Meglio evitare…
- Chiameremo la corda di sinistra R per indicare l'operazione di Reset (porto a 0) e quella di destra S per indicare SET (porto a 1).

Memoria binaria: bistabile SR (Set-Reset)

SET-RESET: è una rete con due ingressi S e R e una uscita Q (ed una uscita complementata Q#). L'uscita Q assume il valore 1 quando S=1 e R=0 e il valore 0 quando S=0 e R=1. L'uscita rimane inalterata quando S=R=0. La combinazione di ingresso S=R=1 non si deve mai verificare (configurazione proibita).



S R	Q(t)	Q'(t)
0 0	Q(t-1)	Q' (t-1)
0 1	0	1
1 0	1	0
1 1	-	-

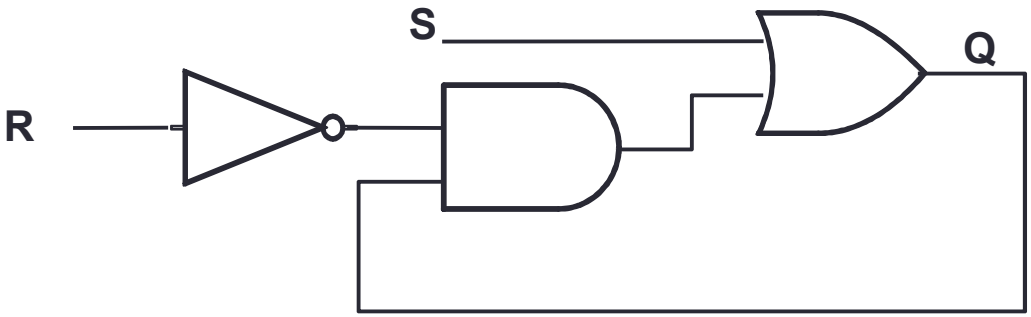


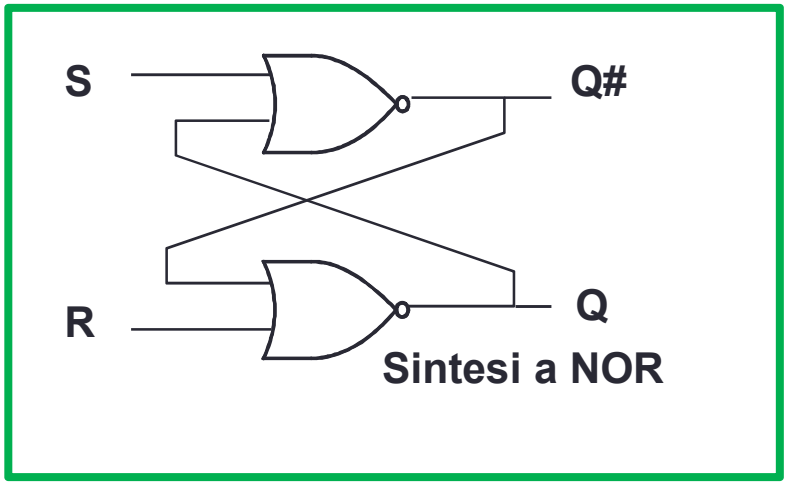
- Funzione di eccitazione del Set-Reset:

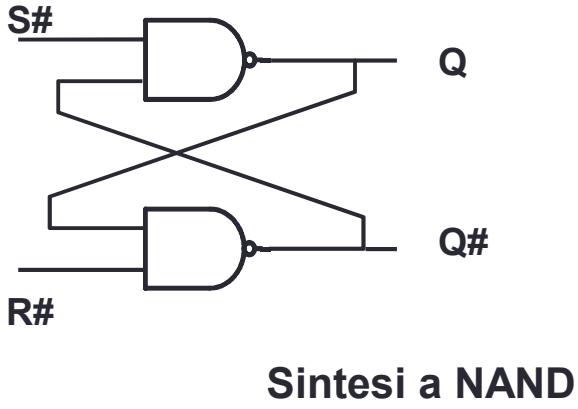
Q(t+1)=S+R'Q(t)

Bistabile Set-Reset

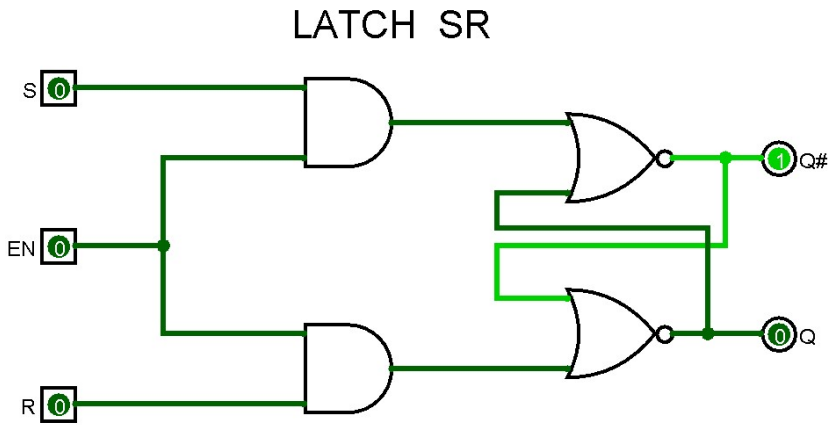
- **Q(t+1)=S+R'Q(t)**
- Q#(t+1)= (S+(R+Q')')'
- Esistono diverse realizzazioni:







Latch S-R sensibile a livello (del clock)

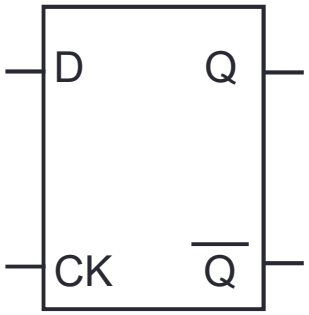


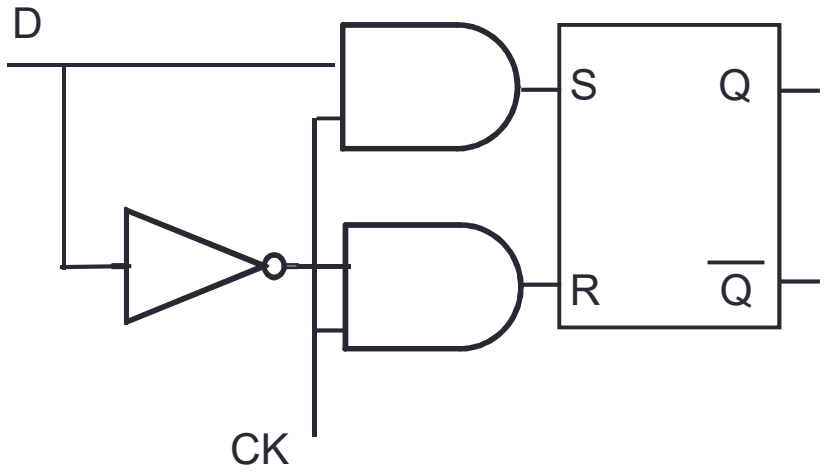
EN (**enable)**:
- se non è attivo (0) il secondo stadio ha la configurazione di ingresso (0,0) e rimane nello stato corrente (hold)

- se è attivo (1) la rete è sensibile al cambiamento degli ingressi:
  - se sono entrambi a 0 la rete rimane in hold,
  - altrimenti la rete cambia di stato (set o reset)

D Latch

- Memoria capace di mantenere l'uscita costante se il segnale di clock (o enable) non è attivo e di cambiare l'uscita campionando l'ingresso quando il segnale di clock /enable **è attivo** (74LS76)





D CK	Q Q#
- 0	Q Q#
0 1	0 1
1 1	1 0

Tabella di verità

RESET

SET

- Il ***flip-flop* D** (derivato da questo latch) è il flip flop più usato per memorizzare dei segnali il cui valore è significativo (e quindi deve essere campionato) solo in un dato istante (**sul fronte (normalmente di salita) del clock** es., per memorizzare dati ed indirizzi su bus multiplexati)

Flip flop e latch

- Si definisce **latch** un bistabile sincrono trasparente, capace di memorizzare o meno segnali di ingresso in funzione di un segnale di abilitazione (clock o enable).

- La transizione di stato avviene per tutto il tempo in cui il clock è attivo (alto) e si hanno tante transizioni di stato quanti cambiamenti di ingressi avvengono in tale periodo. Il latch è *trasparente* agli ingressi quando l'enable è attivo

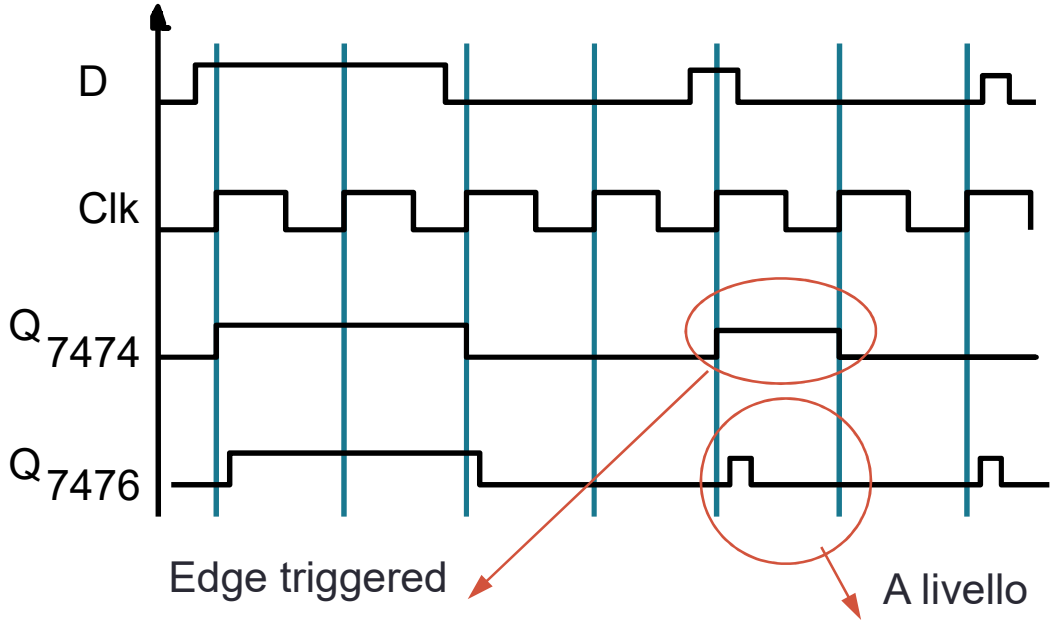
- **Flip Flop** è un dispositivo bistabile privo della proprietà di trasparenza

- Nel flip flop il cambiamento della uscita non è conseguenza del cambiamento dell'ingresso di dato ma è conseguenza del cambiamento (***edge-triggered***) di un ingresso di controllo sincrono (il clock) o asincrono (preset o clear)

- I flip flop si definiscono bistabili sincroni a commutazione sul fronte perché la transizione di stato avviene nell'istante in cui si ha l'evento significativo del clock (fronte di salita o di discesa) in base agli ingressi in quel momento

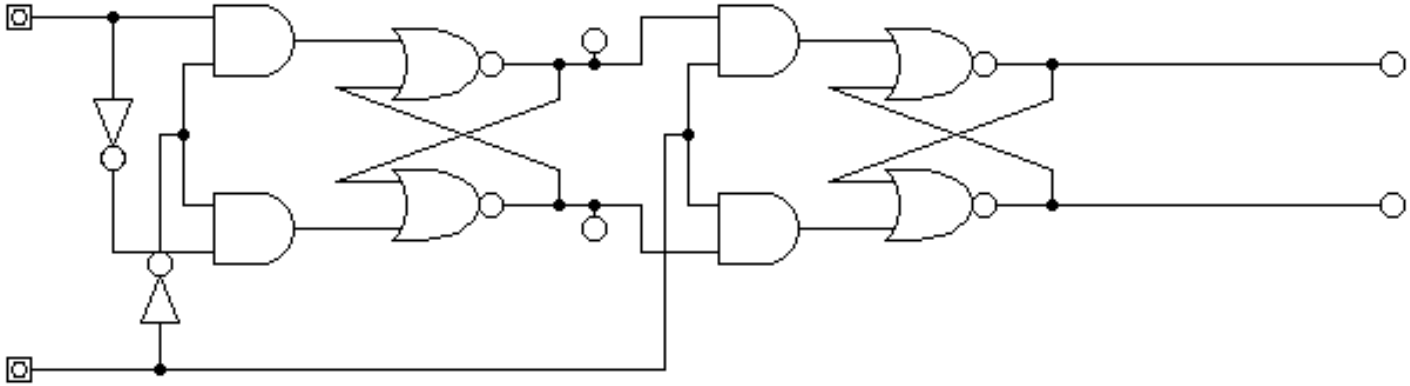
Flip flop e latch D - esempi

- Esempi di comportamenti di un Latch D (7476) e di un FF-D (7474)



Flip flop master slave

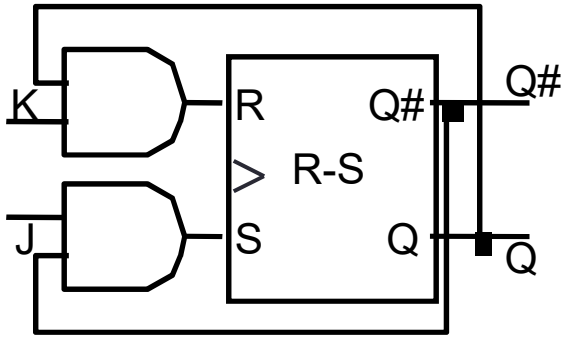
- Esempio di Flip-Flop D realizzato in modalità master slave.



Flip Flop JK

- Flip flop JK progettato come estensione del FF-SR
- Il problema dell'ingresso proibito (11) del flip flop SR non c'è più → toggle

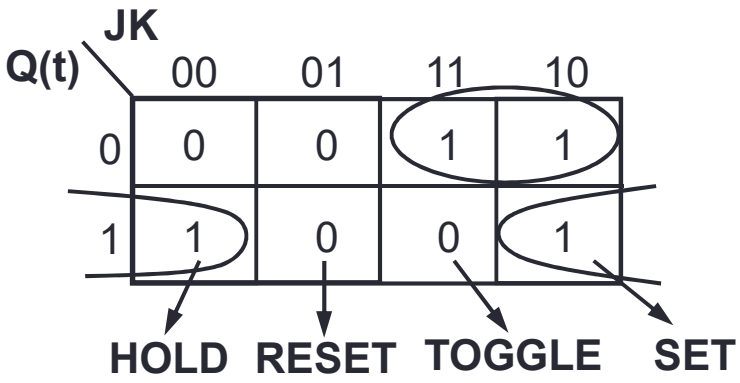
- **Q(t+1)=Q(t)K'+Q(t)'J**



- Vengono usati per il campionamento dei dati:
  - per memorizzare dato= 0 JK=01
  - per memorizzare dato=1 JK=10
- basta collegare il dato a J e collegare K a J negato

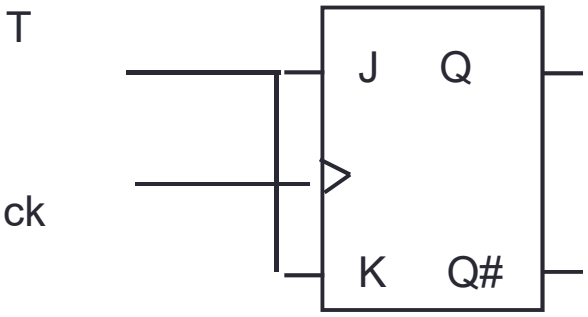
Tabella di verità

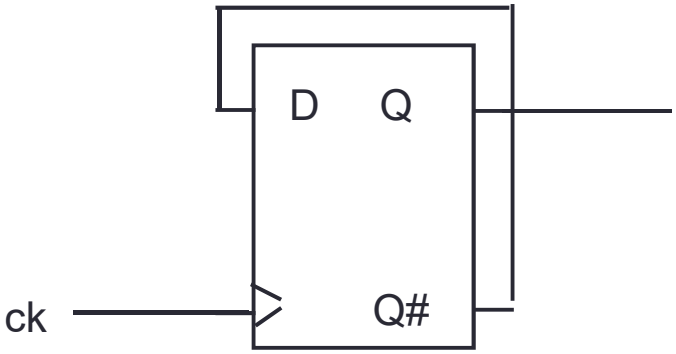
J	K	Q+	Qn+	Descrizione
0	0	Q	Qn	Memoria (nessun cambiamento)
0	1	0	0	Reset
1	0	1	1	Set
1	1	Qn	Q	Toggle (complemento)



Flip Flop T - Toggle

- Molto usati per fare commutare lo stato di uscita

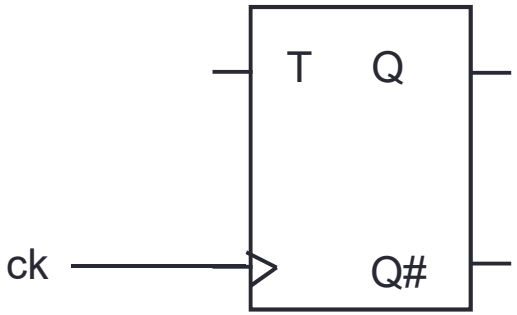




- **Q(t+1)=Q(t)T'+Q(t)'T**

Tabella di verità:

T	Q+	Descrizione
0	Q	Memoria (nessun cambiamento)
1	Qn	Toggle (complemento)



Proprietà: Se T=1 l'uscita Q ha frequenza dimezzata rispetto al clock.

Applicazioni: È il componente base dei contatori, infatti collegando a cascata vari flip-flop T ad ogni uscita si ottiene un clock dimezzato rispetto al clock precedente.

Flip Flop

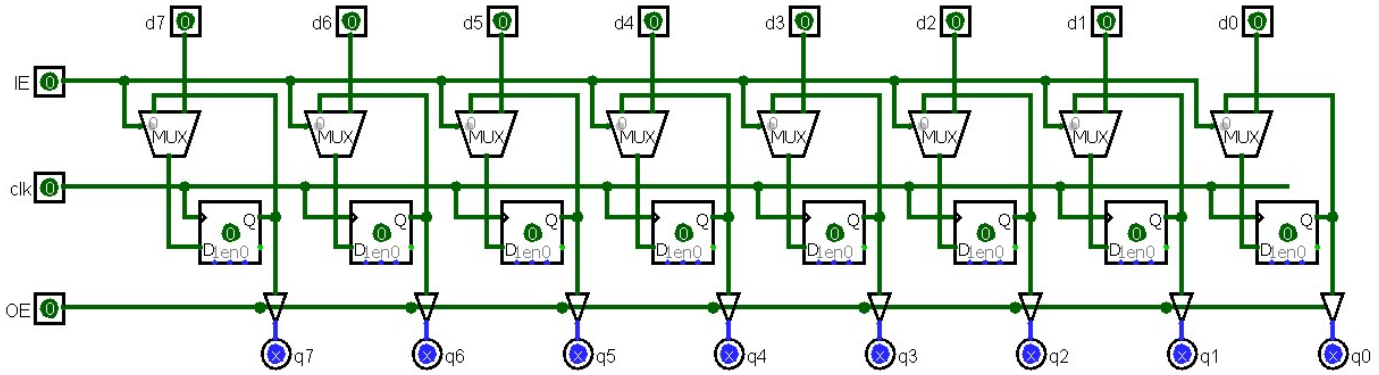
- Flip Flop e latch sono alla base dei circuiti sequenziali; ma quando usarli?
- **S-R latch** sono poco usati come blocchi funzionali (e comunque all'interno dei JK e D).
- **Flip Flop T** sono molto usati (realizzati con JK o D) all'interno dei contatori o per ricordarsi l'evoluzione di un contesto interno al sistema di elaborazione in due stati possibili
- **Flip Flop JK e D** sono entrambi i più usati: con JK si realizzano funzioni più complesse con meno logica esterna, ma richiedono più pin. In VLSI si usano più i D (componenti base della memoria)

SR:	**Q(t+1)=S+R'Q(t)**
D:	**Q(t+1)=D**
J-K:	**Q(t+1)=JQ'(t)+K'Q(t)**
T:	**Q(t+1)=TQ'(t)+T'Q(t)**

Q(t)	Q(t+1)	S	R	D	J	K	T
0	0	0	-	0	0	-	0
0	1	1	0	1	1	-	1
1	0	0	1	0	-	1	1
1	1	-	0	1	-	0	0

Registri

- Un registro è un elemento di memoria in cui *n* flip-flop vengono controllati dallo stesso clock, formando sostanzialmente una unità in grado di memorizzare parole composte da *n* bit.
- Tipicamente sono presenti un segnale di *Input Enable (o Chip Select CS)*, cioè una linea che consente di attivare la fase di memorizzazione e un segnale di *Output Enable* che rende visibile in uscita la parola memorizzata.



Memorie

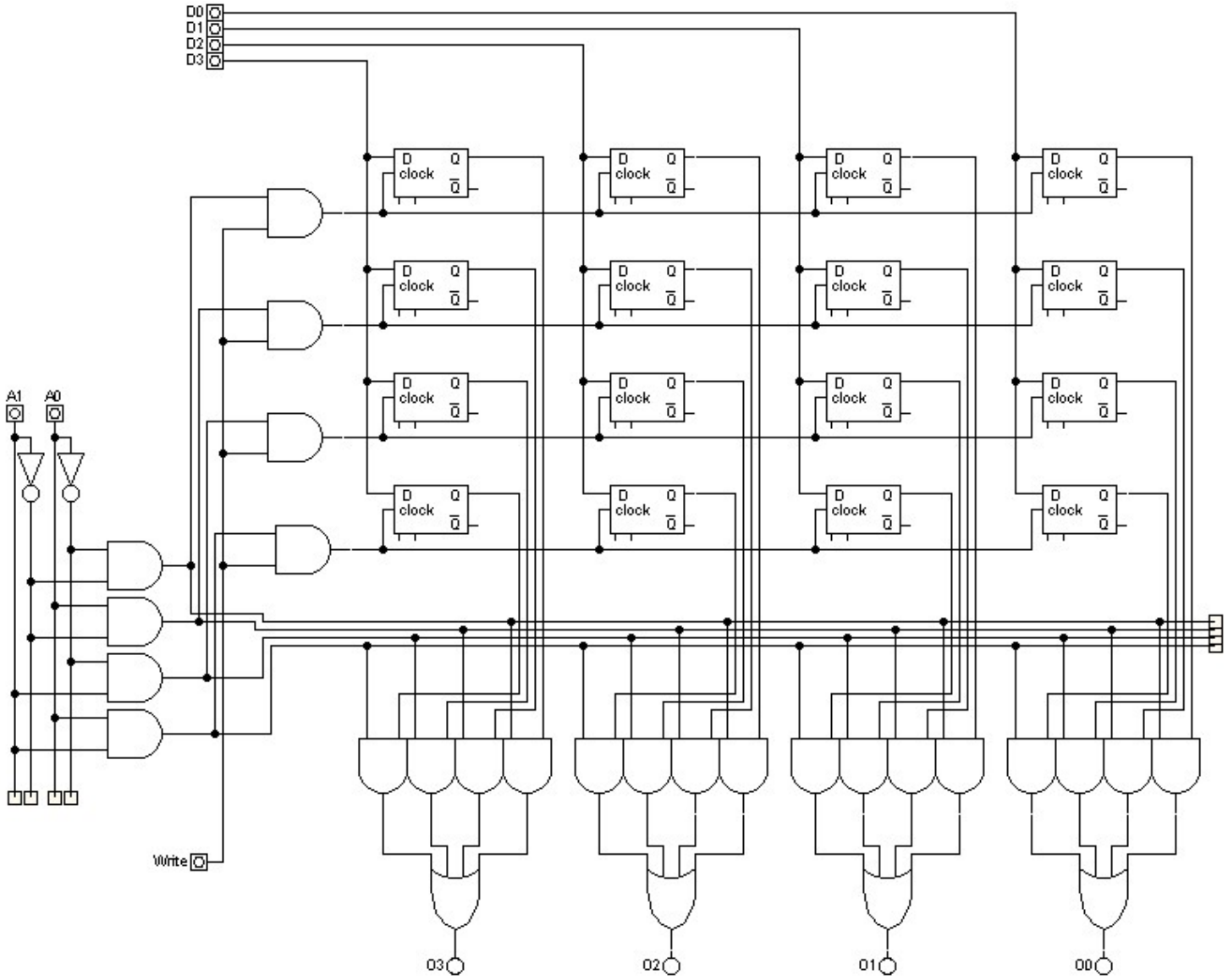
- Le memorie sono dispositivi di memorizzazione logicamente assimilabili a banchi di registri, anche se dal punto di vista architetturale se ne discostano profondamente.

- Ogni unità di memorizzazione viene detta **cella** di memoria.

- La presenza di più di un registro introduce la ovvia necessità di *selezionare* a quale registro vogliamo accedere. Dal momento che stiamo lavorando con circuiti binari, la scelta più ovvia è quella di codificare in *n* bit il numero e utilizzare un decoder per produrre i segnali di abilitazione della cella in questione.

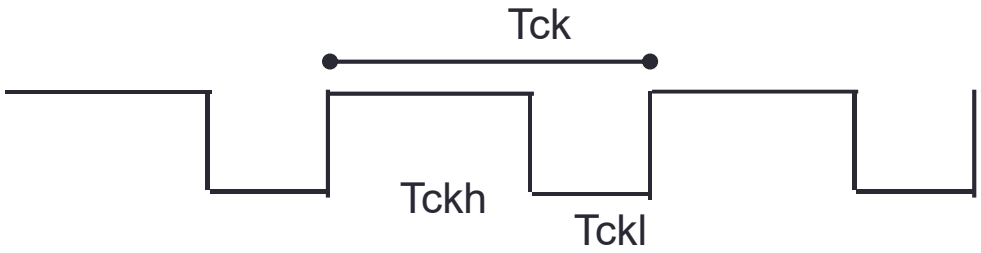
- Il numero così codificato viene detto **indirizzo** della cella e il numero di bit per l'indirizzamento verrà indicato con n_a dove la *a* indica la parola *address* (indirizzo). Il numero di bit contenuti in ogni cella viene indicato con n_d dove *d* indica la parola *data* (dati).

Memorie

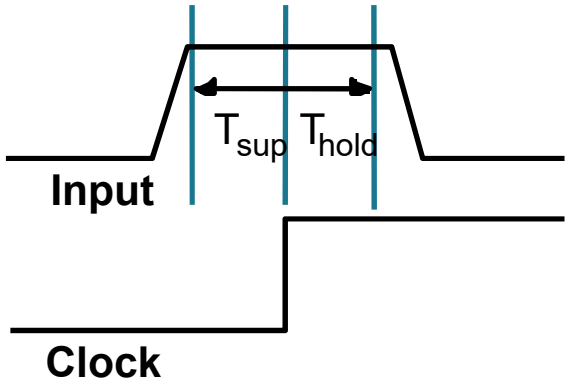


Il Clock (4)

- Si dice **periodo di clock** la lunghezza del ciclo di clock e **frequenza di clock** il suo inverso
- Il **duty cycle** è la percentuale del tempo in cui il clock rimane a livello alto

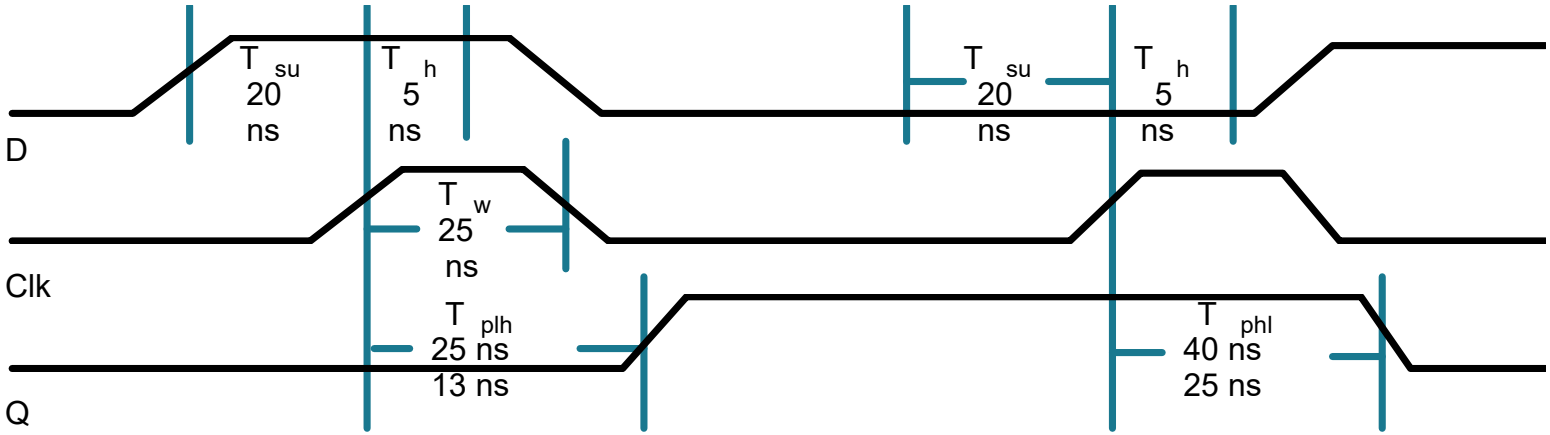


- Tsup (**tempo di setup**) è il periodo in cui gli ingressi devono rimanere stabili prima del fronte del clock per poter essere campionati correttamente
- Th (**tempo di hold**) è il periodo in cui gli ingressi devono rimanere stabili dopo l'evento del clock



Flip Flop edge triggered

- Tutte le misure sono fatte rispetto al fronte positivo del clock



- requisiti temporali del 74LS74:
- I tempi che si riportano sono
  - Setup time
  - Hold time
  - Minimum clock width
  - Propagation delays (low to high T_{plh}, high to low T_{phl}, max e typical)

Reti asincrone e sincrone

- Nel progetto di reti logiche si predilige l'impiego di reti sincrone
- Reti asincrone sono alla base delle reti sincrone, ma con un segnale di riferimento, il clock.
- Anche nelle reti sincrone (come i FF-D) esistono segnali asincroni (clear e pre-set)

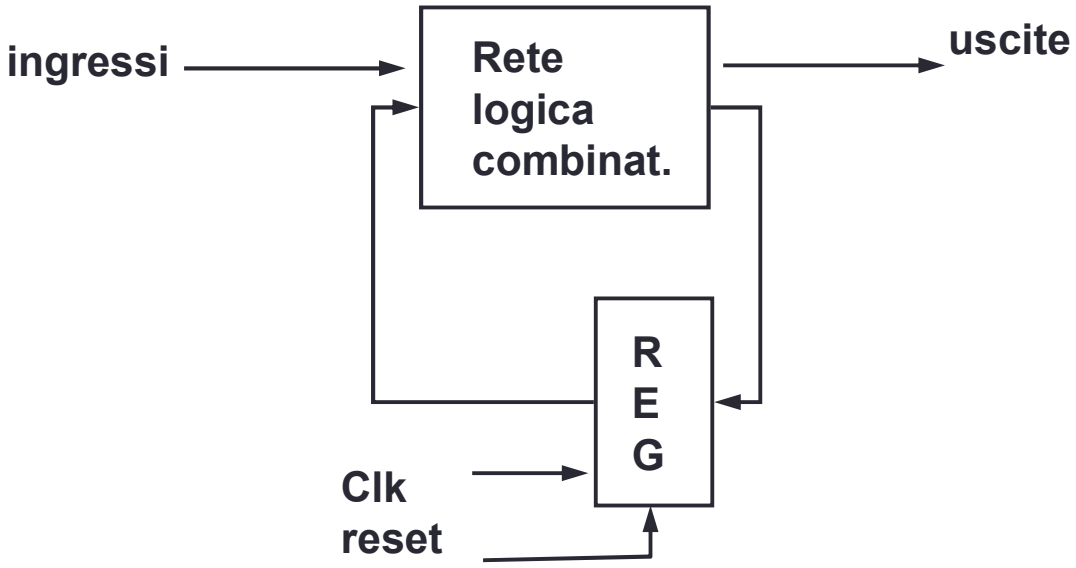
- è meglio evitare reti asincrone (soprattutto ad alta frequenza) perché sono sensibili ad alee (corse critiche).

In alcuni casi, le reti asincrone sono inevitabili

- circuiti di reset
- segnali esterni
- segnali di handshake
- segnali di wait nelle memorie

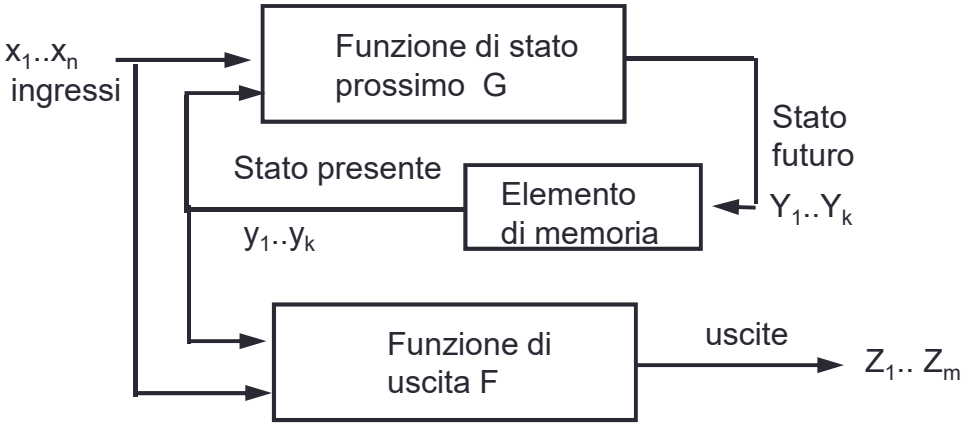
Reti sequenziali (1/2)

- Una rete sequenziale memorizza le informazioni sulle configurazioni di ingresso che si verificano nel tempo; la memorizzazione avviene in *stati interni*
- Le variabili di stato che definiscono lo stato interno in cui si trova la rete sono memorizzate in elementi di retroazione
- Tra le reti sequenziali, importanza fondamentale hanno le macchine a stati finiti (**FSM, Finite state machine**) in cui gli elementi di retroazione sono Flip Flop con un unico segnale di clock
- L'insieme dei FF è detto **registro di stato** e memorizza lo stato futuro presentando a valle lo stato presente

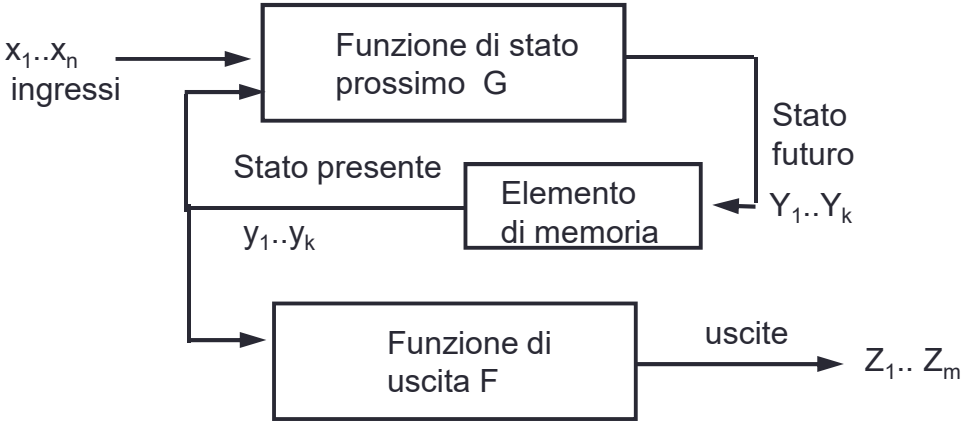


Automa a stati finiti

- Modello generale (**automa di Mealy**): il valore delle uscite dipende dallo stato presente e dagli ingressi in quell'istante



- Modello equivalente (**automa di Moore**): il valore delle uscite dipende solo dallo stato presente e non dagli ingressi in quell'istante

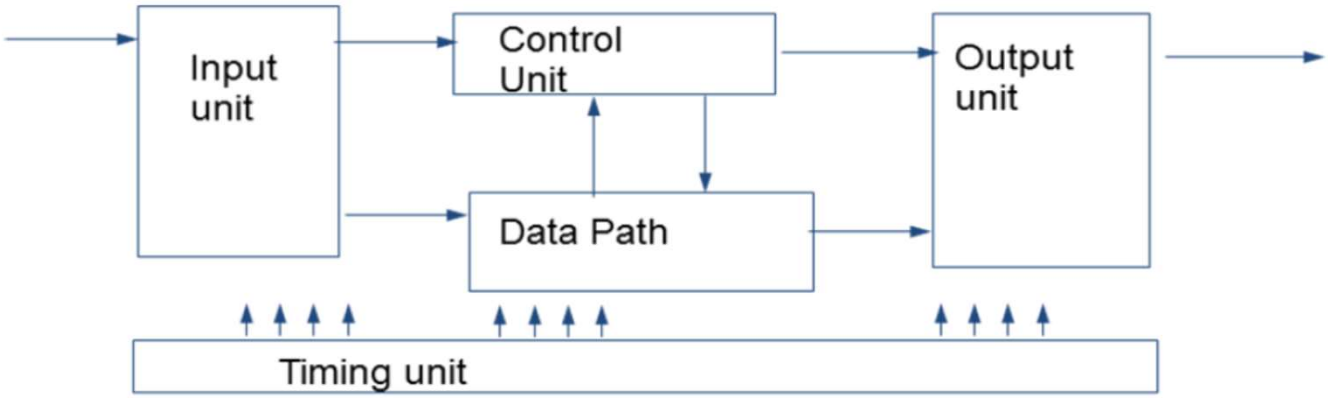


E' sempre possibile passare da un modello all'altro
Il modello di Moore ha più stati ma funzioni di uscita più semplici

Reti sequenziali

- La maggior parte delle reti sequenziali sincrone sono descrivibili come FSM, più o meno complesse.

- Anche la **CPU** è descrivibile come una FSM avente la parte sequenziale composta dalla **Control Unit** che passa attraverso diversi stati interni (lettura delle istruzioni, decodifica, esecuzione) in base ai segnali esterni (istruzioni e dati) allo stato interno (flag, stato di esecuzione attuale, ...), per fornire le uscite, i dati elaborati e i segnali esterni al calcolatore).



Reti sequenziali

- Una generica rete sequenziale è pertanto definita dalla tupla <X,Z,S,F,G> e richiede in pratica la realizzazione di due funzioni combinatorie (F,G) che dipendono dai due insiemi di valori (X e S). Inoltre, sono necessari dispositivi in **grado di memorizzare lo stato prossimo e presentarlo come stato presente** nell'intervallo di lavoro successivo della rete sequenziale.

- A seconda del progetto e della descrizione a parole si può decidere di realizzare un automa di **Moore** o di **Mealy**. Di solito l'automa di **Mealy** ha meno stati (quindi meno elementi di memoria) ma ha le reti combinatorie ed in particolare la rete combinatoria delle uscite più complessa e quindi potenzialmente più lenta. Spesso si realizza l'automa di **Moore** perché è concettualmente più semplice.

Sintesi di reti sequenziali

1) Si prepara una **descrizione comportamentale a parole** o con un linguaggio di descrizione dell'hardware. (*specifiche di progetto*)

2) Si definisce il **diagramma degli stati** per definire le transizioni che si traduce nella tabella di flusso. Questa e' la fase piu' importante che corrisponde in software alla creazione dell'algoritmo perche' si definiscono gli stati interni e le transizioni

3) Si impiegano metodi manuali o automatici per la **minimizzazione degli stati.** Spesso il diagramma degli stati può essere minimizzato con un numero minore di stati (esistono algoritmi appositi).

4) Dal diagramma minimizzato e tabella di flusso corrispondente si crea **la tabella delle transizioni e delle uscite** con l'assegnamento degli stati (indicando quale numero binario corrisponde ad ogni stato, date le variabili di stato presente e futuro).

5) Infine si ottiene la **implementazione** (avendo scelto i componenti bistabili elementari e i gate elementari per le reti combinatorie).

Diagramma degli stati

- Una rete sequenziale può essere rappresentata da un **diagramma degli stati:**
  - il diagramma degli stati è un grafo con tanti nodi quanti gli stati e tanti archi quante le transizioni da uno stato all'altro dovute a cambiamenti degli ingressi
- nel diagramma degli stati vengono rappresentati inoltre i valori delle uscite per ogni stato:
  - negli archi se il modello è di Mealy
  - nei nodi se il modello è di Moore

Esempio di sintesi

- **Esercizio**: Progettare una rete sequenziale in grado di riconoscere, in presenza di una sequenza di cifre binarie, quando si sono presentati successivamente due 0 e un 1.

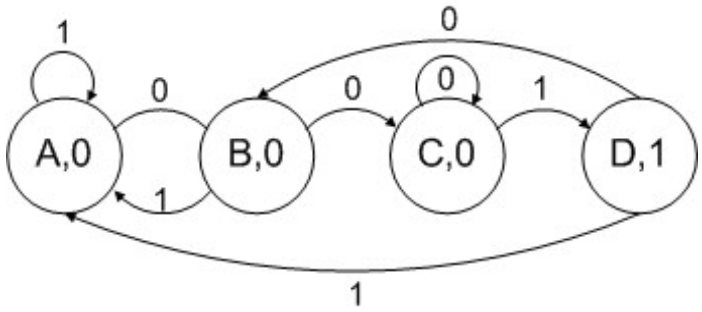


Diagramma di Moore

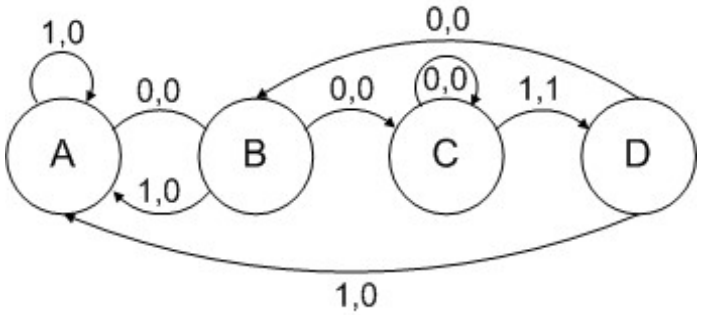


Diagramma di Mealy

- Codifica degli stati e sintesi delle uscite (Moore)

s1s0 - stati presenti
S1S0 - stati futuri

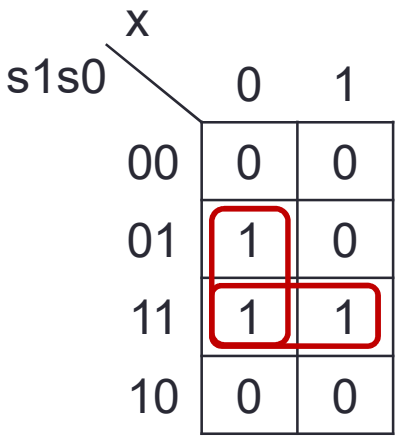
s	s1	s0	z
A	0	0	0
B	0	1	0
C	1	1	0
D	1	0	1

z = s1s0'

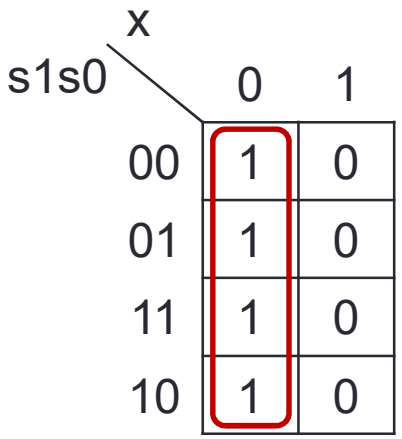
Esempio di sintesi

- Tabella di Transizioni degli stati (Moore)

Stato presente			Ingresso	Stato Futuro		
s	s_1	s_0	x	S	S_1	S_0
A	0	0	0	B	0	1
	0	0	1	A	0	0
B	0	1	0	C	1	1
	0	1	1	A	0	0
C	1	1	0	C	1	1
	1	1	1	D	1	0
D	1	0	0	B	0	1
	1	0	1	A	0	0



S1 = s0 x' + s1s0



S0 = x'

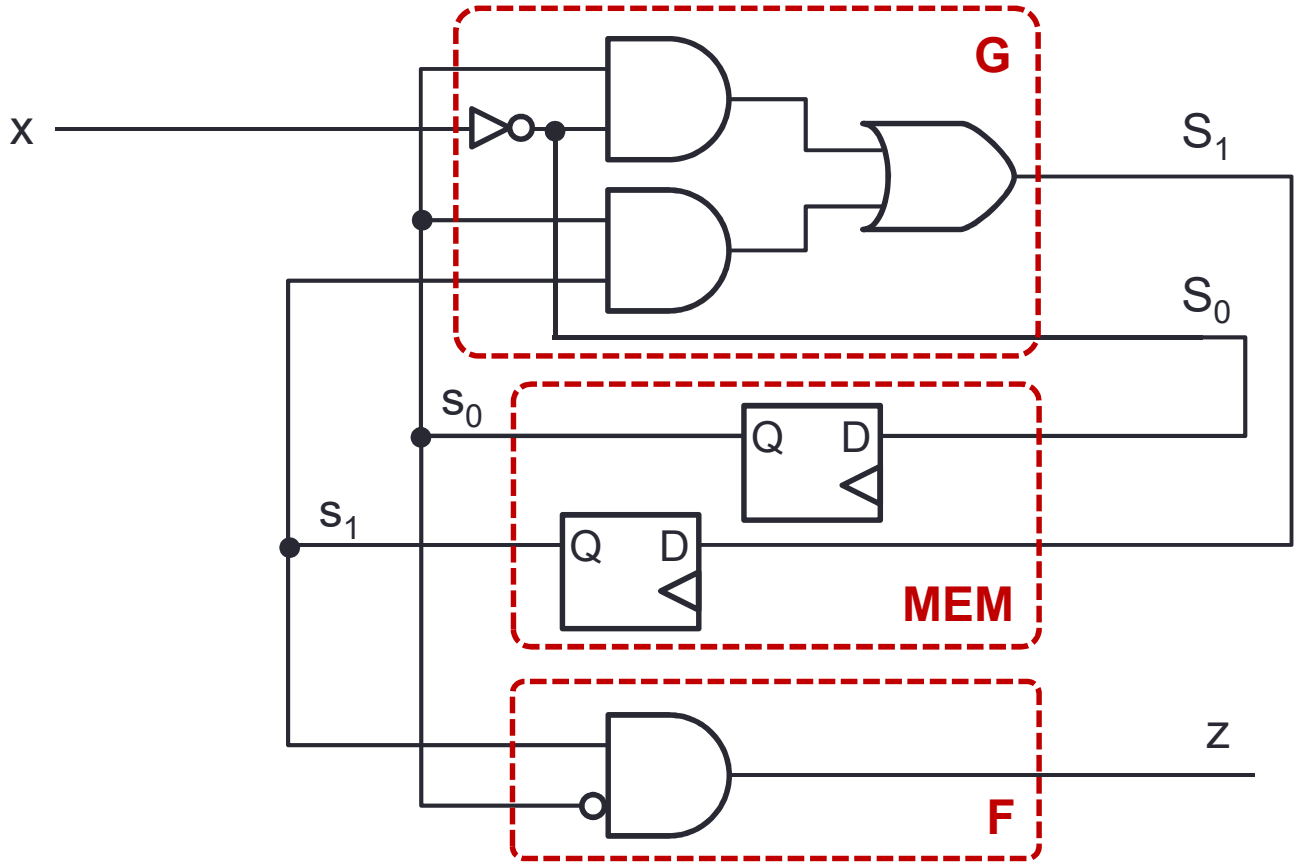
Sintesi con Flip Flop D

- Flip Flop D: **Q(t+1)= D(t)**

- l'uscita è uguale all'ingresso
- un flip flop per ogni bit di stato da memorizzare
- D1=S1
- D0=S0

S1 = s0 x' + s1s0
S0 = x'

z = s1s0'



Esempio – ventilatore digitale

- **Testo:** Effettuare la sintesi di un automa a stati finiti sincrono che controlla un ventilatore “digitale”. Il ventilatore, oltre allo stato spento, può funzionare a 3 velocità differenti (V0, V1, V2). Per controllare la velocità, esistono due ingressi, rispettivamente Più (P) e Meno (M) che incrementano e decrementano di una unità la velocità del ventilatore. Inizialmente il ventilatore risulta essere spento. Premendo il tasto P si porta alla velocità V0, quindi alle altre velocità. Dalla velocità V0 premendo il tasto M è possibile spegnere il ventilatore. La politica da usare nel caso della pressione contemporanea dei tasti più e meno è a discrezione dello studente e deve essere riportata e commentata. Le uscite della rete devono essere la velocità del ventilatore e il suo stato (acceso/spento).
- *Realizzare l’automa a stati e sintetizzare le funzioni di transizione di stato e di uscita. Indicare se nella sintesi si è utilizzato un automa di Mealy o di Moore. Disegnare il circuito logico corrispondente utilizzando Flip Flop D.*

Esempio – *identificazione I,O,S*

- Come prima cosa è necessario identificare bene le componenti dell'automa, ovvero ingressi, stato, uscite.
- Il numero di ingressi è pari a due (P, M). Comportamento del sistema:
- P=0 e M=0 -> rimango nello stato attuale
- P=1 e M=0 -> incremento la velocità
- P=0 e M=1 -> calo la velocità
- P=1 e M=1 -> calo la velocità
- Nel caso di pressione contemporanea di due tasti, imponiamo al sistema di considerare solo il tasto M che ha la precedenza sul P.
-
- Il numero totale di stati è 4: Sp, V0, V1, V2.
- Quindi servono due bit per la sua codifica (S_0S_1).

Esempio – *codifica degli stati*

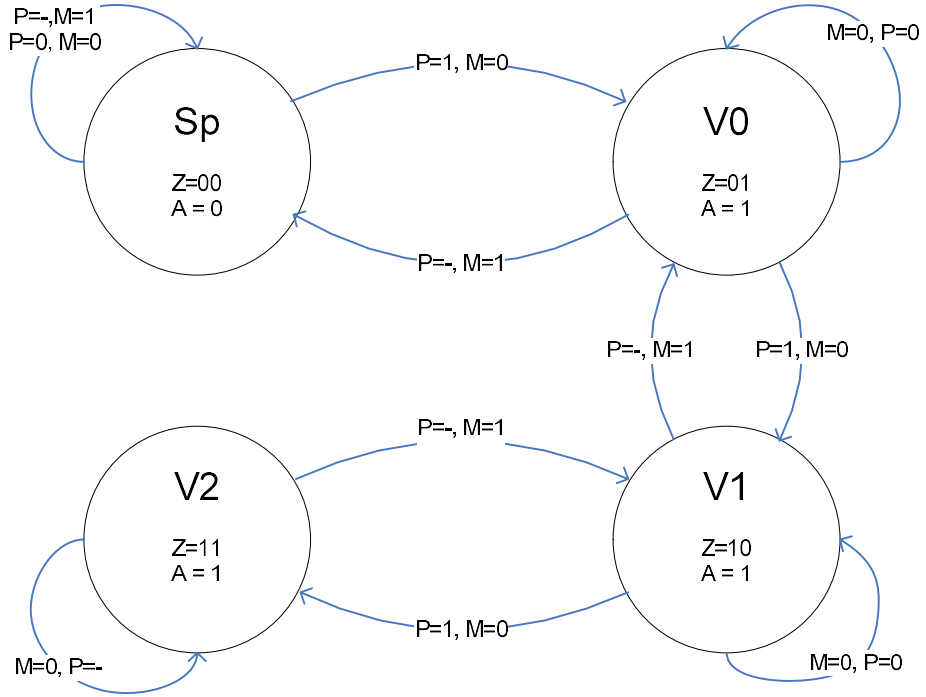
Introduciamo una codifica degli stati:

Stato	**S_0S_1**
Sp	00
V_0	01
V_1	10
V_2	11

Le uscite infine sono due: la velocità, che chiamiamo Z, la quale dovendo variare da 1 a 3 deve essere composta da due bit: Z_1Z_0. L'altra uscita, invece, può essere chiamata "acceso" (A), che varrà 1 se e solo se il ventilatore è acceso

Esempio – *diagramma degli stati*

- Visto che entrambe le uscite dipendono solo dallo stato in cui si trova il ventilatore e non dall'ingresso che ha portato a quello stato, effettuiamo la sintesi dell'automa di Moore corrispondente.
- Il diagramma degli stati è ricavabile direttamente dal testo del problema:



Nota: è necessario inserire anche gli auto-anelli corrispondenti alla combinazione di ingresso M=0 e P=0!!

Esempio – *tabella di transizione degli stati*

- La corrispondente tabella di transizione degli stati che mi permette di ricavare lo stato futuro a partire da ogni combinazione di ingresso/stato corrente è la seguente (usando *s* minuscolo per lo stato presente, *S* maiuscolo per quello futuro)

Stato presente			Ingresso		Stato Futuro		
s	s_1	s_0	M	P	S	S_1	S_0
Sp	0	0	0	0	Sp	0	0
Sp	0	0	0	1	V0	0	1
Sp	0	0	1	0	Sp	0	0
Sp	0	0	1	1	Sp	0	0
V_0	0	1	0	0	V0	0	1
V_0	0	1	0	1	V1	1	0
V_0	0	1	1	0	Sp	0	0
V_0	0	1	1	1	Sp	0	0
V_1	1	0	0	0	V1	1	0
V_1	1	0	0	1	V2	1	1
V_1	1	0	1	0	V0	0	1
V_1	1	0	1	1	V0	0	1
V_2	1	1	0	0	V2	1	1
V_2	1	1	0	1	V2	1	1
V_2	1	1	1	0	V1	1	0
V_2	1	1	1	1	V1	1	0

Esempio – *sintesi minima SP stato futuro*

- Effettuiamo la sintesi di S_1 e S_0 mediante mappe di Karnaugh. S_1 e S_0 sono due funzioni che dipendono stato attuale e dagli ingressi.

S_1		M,P			
		00	01	11	10
s_1,s_0	00	0	0	0	0
	01	0	1	0	0
	11	1	1	1	1
	10	1	1	0	0

$$S_1 = s_1 s_0 + s_1 \overline{M} + \overline{M} P s_0$$

S_0		M,P			
		00	01	11	10
s_1,s_0	00	0	1	0	0
	01	1	0	0	0
	11	1	1	0	0
	10	0	1	1	1

$$S_0 = s_0 \overline{M}\,\overline{P} + \overline{s_0}\,\overline{M} P + s_1 \overline{M} P + s_1 \overline{s_0} M$$

Esempio – *sintesi minima uscite*

- La sintesi delle uscite invece è molto semplice e si può evitare di passare tramite le mappe. Infatti la velocità corrisponde allo stato in cui ci si trova, e la seconda uscita vale 0 solo nello stato Spento Sp.

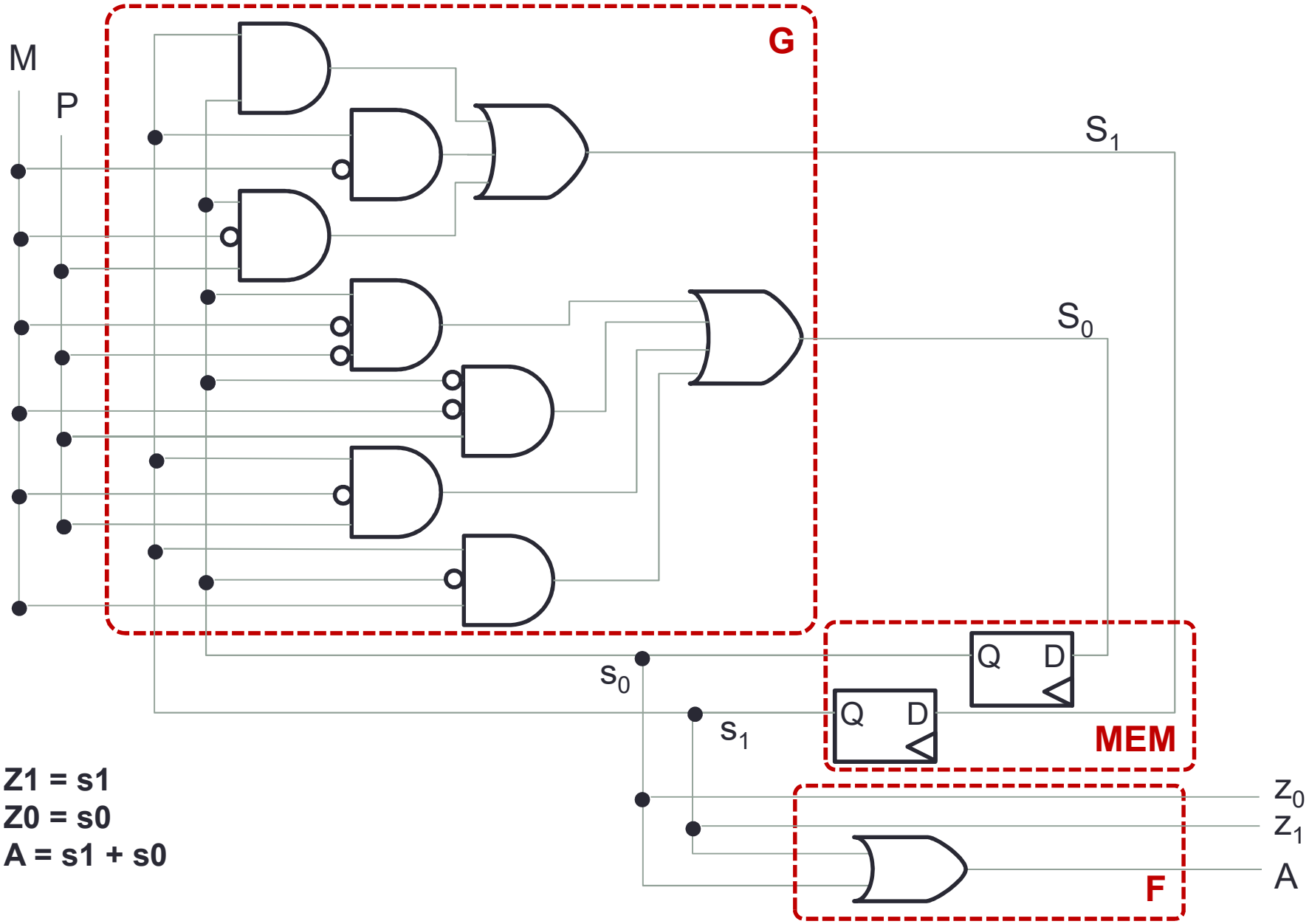
Stato			Uscite		
s	s1	s0	Z1	Z0	A
Sp	0	0	0	0	0
V0	0	1	0	1	1
V1	1	0	1	0	1
V2	1	1	1	1	1

Da cui:

$Z_1=s_1$
$Z_0=s_0$
$A=s_1+s_0$

Sintesi con Flip Flop D

S1 = s1s0 + s1M'+s0M'P
S0 = s0M'P' + s0'M'P + s1M'P + s1s0'M



Z1 = s1
Z0 = s0
A = s1 + s0

Esercizio 1

- **Esercizio di progetto**. Si *vuole progettare un DISTRIBUTORE di bibite. Ogni bibita costa 30centesimi e accetta monete da 10 e da 20 cent. Non da' resto. Deve memorizzare nel suo stato interno l'ammontare ricevuto e dare una uscita che vale ad 1 solo quando deve azionare il meccanismo di distribuzione.*

- Provare a progettare la rete sia con automa di Moore che di Mealy.

Esercizio 2

- **Esercizio** si vuole realizzare una FSM capace di riconoscere la sequenza 101. La macchina ha un ingresso X ed una uscita Z . La macchina e' sequenziale con un clock e l'uscita diventa 1 solo per l'intervallo di tempo in cui arriva l'ultimo valore corretto della sequenza
  Es :
  X= 0,1,1,1,1,1,0,0,0,1,0,1,0,0 e
  Z= 0,0,0,0,0,0,0,0,0,0,0,1,0,0,