Capitolo 8 Sintesi di Reti Logiche Sequenziali

Reti sequenziali

Sono 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.
Stato presente ⇒ stato interno attuale
Stato futuro ⇒ calcolato dal sistema, in ogni istante, in seguito alla variazione degli ingressi

Quando avviene l'aggiornamento dallo stato presente allo stato futuro?
Occorre introdurre il concetto di temporizzazione.

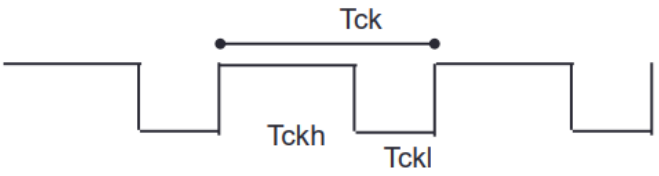
Le reti possono essere :
asincrone: se le variazioni delle configurazioni di ingresso vengono sentite e modificano lo stato e le uscite in qualsiasi istante
sincrone: (+ diffuse)se le variazioni delle configurazioni di ingresso vengono sentite e modificano lo stato e le uscite solo in presenza di un opportuno ***evento di sincronizzazione.***

L'evento di sincronizzazione è associato ad un segnale attivo (alto o basso) detto **clock**.
Tale evento può essere associato anche al cambiamento di stato del clock detto **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 oscilla indefinitamente, indipendentemente dal resto (almeno finche' il sistema e' alimentato), di tipo periodico.
Il suo periodo (lunghezza del ciclo) è detto ***tempo di clock*** Tck (clock cycle time); il suo inverso è la **frequenza di clock** fck o f.
Il **duty cycle** è la percentuale del tempo in cui il clock rimane 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

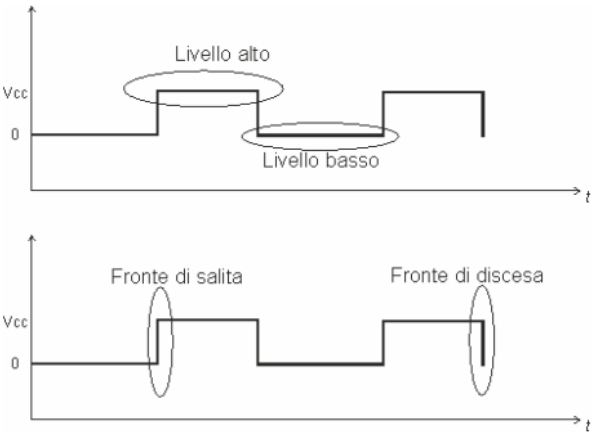
Th **(tempo di hold)** è il periodo in cui gli ingressi devono rimanere stabili dopo l'evento del clock

Una rete che ha la frequenza di 100MHz ha un ciclo di clock di 10ns.

La forma d'onda ha 4 parti:

- ***Livello:***
  - alto
  - basso
- ***Fronte/Edge:***
  - di salita
  - di discesa

Il segnale di clock sequenzializza tutti gli eventi.



Clock Tree: spesso nel calcolatore si usa oltre al ***clock primario*** dei ***clock secondari*** (derivanti dal primario) 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. Hanno quindi frequenze differenti.
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 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"**

Studieremo principalmente le reti sincrone edge-triggered, nello specifico nel fronte di salita.

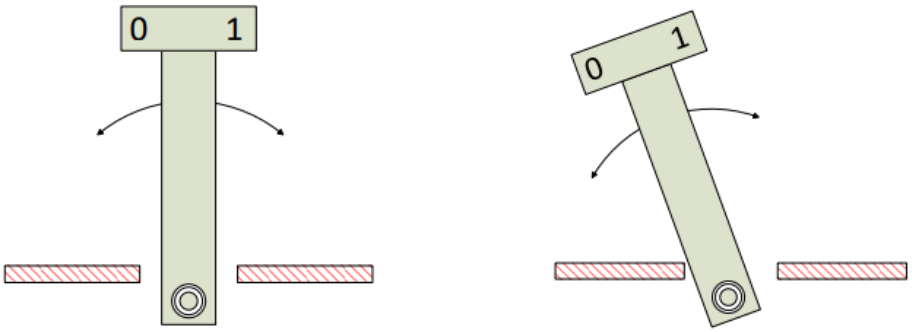
Memoria binaria bistabile (Set Reset)

Gli elementi base delle reti sequenziali per la memorizzazione sono chiamati **bistabili**.
Essi sono capaci di mantenere al loro interno il valore 0 o il valore 1 (1 bit di memoria).
Sono gli elementi di stato.

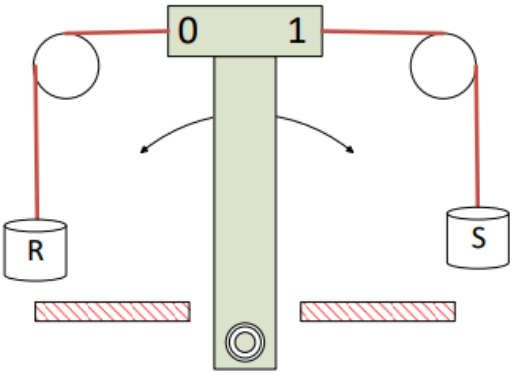
Anche per le reti sequenziali esistono blocchi elementari per memorizzare lo stato attuale.
Memoria binaria (bistabile asincrono): elemento capace di memorizzare il valore di una variabile di stato binaria e di commutare alla presenza di un'opportuna configurazione di ingresso.

Esempio:
Bistabile meccanico, la parte a T è libera di ruotare attorno al perno in basso.

Dopo un eventuale fase di equilibrio instabile (al centro), la parte a T «cadrà» a destra o a sinistra (stati stabili).
Collocare l'oggetto a destra o a sinistra può servire per ricordarsi uno 0 o un 1.
Una volta posizionato l'oggetto, starà fermo fino ad un prossimo comando.



Supponiamo di collegare, mediante appositi supporti, due corde. Chiameremo la corda a sinistra R e quella a destra S
Tirando la corda R si porta l'oggetto nella posizione 0, tirando la corda S nella posizione 1.



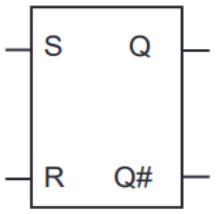
Attenzione: tirando entrambe le corde, l'oggetto si potrebbe portare nella situazione al centro, di equilibrio instabile. Rilasciando le corde l'oggetto cadrà a sinistra o a destra in modo del tutto casuale e impredicibile. Meglio evitare…

R sta per **RESET**(porta a 0) e S per indicare **SET** (porta a 1).

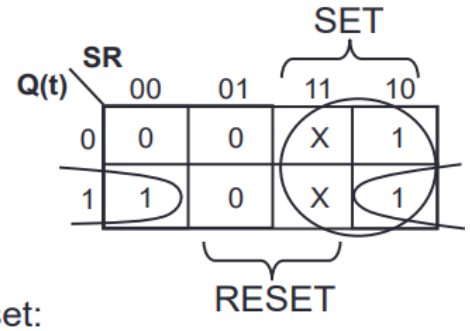
SET-RESET: è una rete con due ingressi S e R e una uscita Q (e la complementata Q#).
L'uscita Q assume il valore:

- 1 se S=1 e R=0
- 0 se S=0 e R=1
- H(inalterata) se S=R=0

La combinazione di ingresso S=R=1 non si deve mai verificare (configurazione proibita).

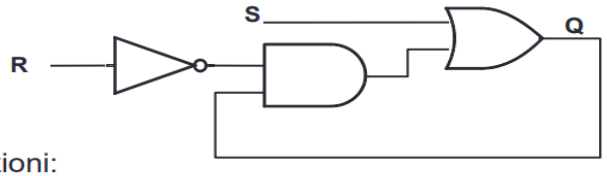


S R	Q(t)	Q'(t)
0 0	Q(t-1)	Q' (t-1)
0 1	0	1
1 0	1	0
1 1	-	-



Funzione di eccitazione del Set-Reset:

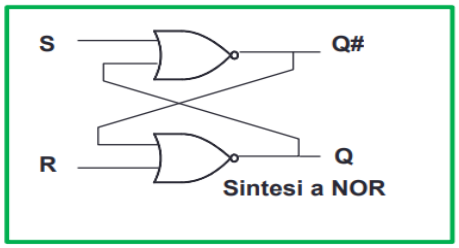
Q(t+1)=S+R'Q(t)

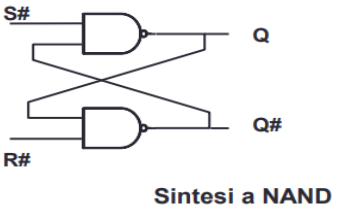


Q(t+1)=S+R'Q(t)

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

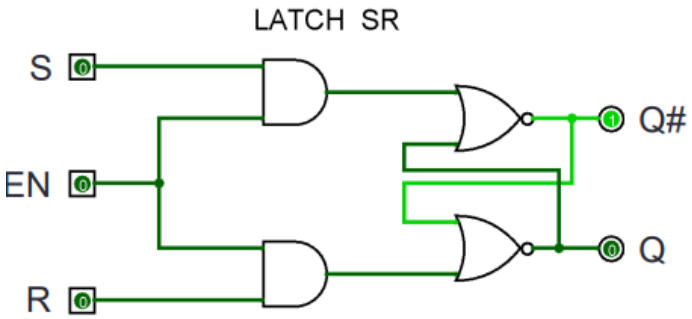
Esistono diverse realizzazioni:





In una rete sequenziale si possono notare delle ***retroazioni***: gli output ritornano in input.
I circuiti prima sono asincroni in quanto non comandati da clock.

Latch S-R sensibile a livello (del clock)



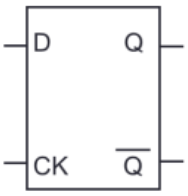
EN	S	R	Q_{n+1}	
0	X	X	Q_n	*memoria*
1	0	0	Q_n	*memoria*
1	0	1	0	*reset*
1	1	0	1	*set*
1	1	1	X	*non ammesso*

Si tratta di un circuito sincrono, attraverso l'ingresso di enable.
LATCH-SR ⇒ circuito attivo a livello, ovvero lavora nel periodo in cui enable è attivo, non solo nei fronti di salita/discesa. È sensibile a tutto il 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



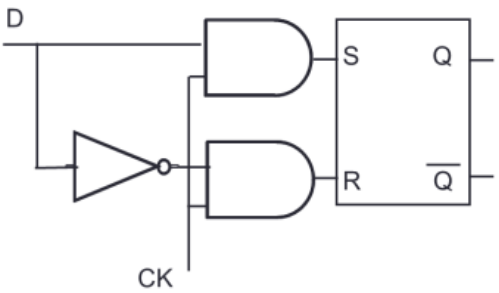


Tabella di verità

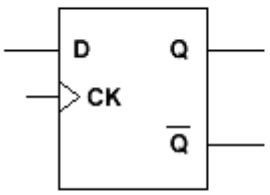
D	CK	Q	Q#	
-	0	Q	Q#	
0	1	0	1	RESET
1	1	1	0	SET

LATCH-D ⇒ Memoria che risolve il problema dello stato 11 vietato.
Segnale di clock o enable:

- se non è attivo (0) mantiene l'uscita costante (hold)
- se è attivo (1) cambia l'uscita campionando l'ingresso:
  se D è a 1 → S=1 e R=0 → SET
  se D è a 0 → S=0 e R=1 → RESET

Si definisce latch un bistabile sincrono ***trasparente***, capace di memorizzare o meno segnali di ingresso in funzione del clock; la transizione di stato avviene per tutto il tempo in cui il clock è attivo. 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.
Si dice che gode della proprietà di trasparenza.

Flip-Flop D



D	Q	$\overline{Q}$
0	0	1
1	1	0

Q(t+1)=D

Derivato da questo latch, è il flip flop più usato per memorizzare dei segnali il cui valore è significativo solo in un ***dato istante*** (**sul fronte di salita**, solitamente, del clock es., per memorizzare dati/indirizzi su bus multiplexati)

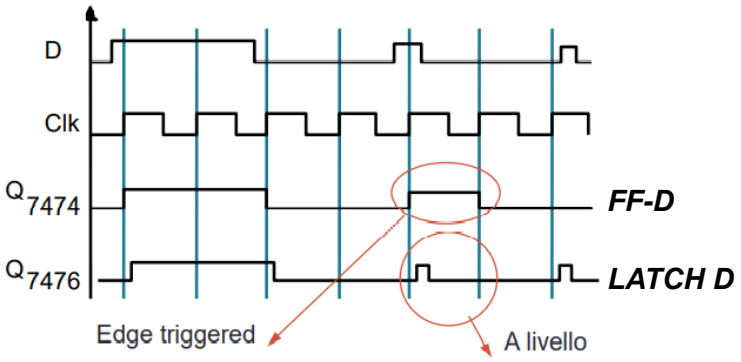
FLIP-FLOP ⇒ campiona il valore degli ingressi su un fronte, diventa sincrono solo sui fronti di salita o di discesa.

Flip Flop è un dispositivo bistabile privo della proprietà di trasparenza.
Gli ingressi sono visibili solo sul fronte d'onda, quindi si perdono i cambiamenti di ingressi che avvengono a livello.

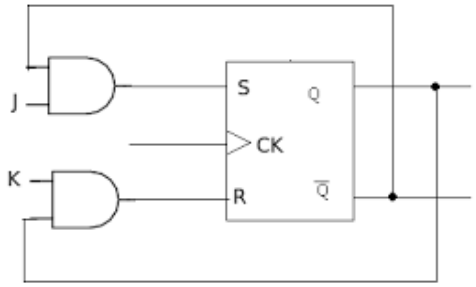
Nel flip flop il cambiamento della uscita non è conseguenza del cambiamento dell'ingresso di dato ma è conseguenza del cambiamento (***edge-triggered***) di un ingresso di controllo sincrono (il clock) o asincrono (preset o clear).

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.

Esempio:



Flip Flop JK



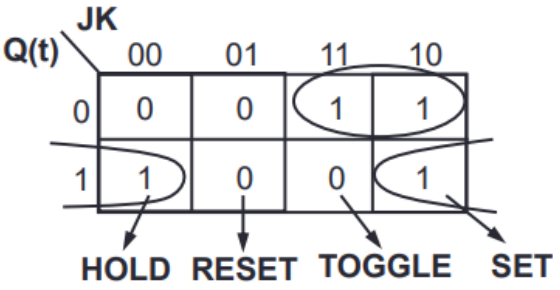


Tabella di verità

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)

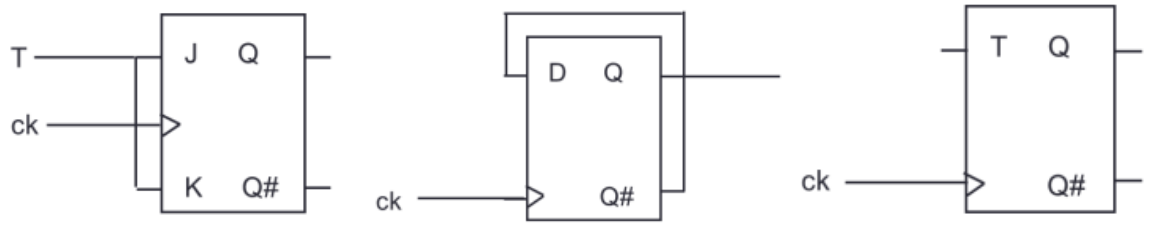
$$Q(t+1)=Q(t)K'+Q(t)'J$$

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

Vengono usati per il campionamento dei dati:

- per memorizzare dato = 0 J=0 K=1
- per memorizzare dato = 1 J=1 K=0

Flip Flop T - Toggle



$$Q(t+1)=Q(t)T'+Q(t)'T$$

Tabella di verità:

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

Flip Flop T ⇒ Molto usati per fare commutare lo stato di uscita
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

Quando usare latch e Flip Flop?

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:	$Q(t+1)=TQ'(t)+T'Q(t)$

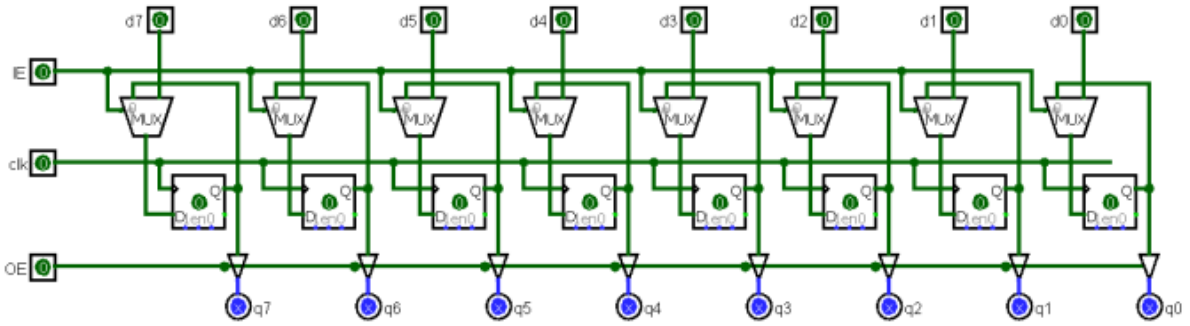
Q(t)	Q(t+1)	S	R	D	J	K	T
0	0	0	-	0	0	-	0
0	1	1	0	1	1	-	1
1	0	0	1	0	-	1	1
1	1	-	0	1	-	0	0

Registri

Sono elementi di memoria (a n bit) in cui n flip-flop vengono controllati dallo stesso clock, formando sostanzialmente una unità in grado di memorizzare parole composte da n bit.

Sono presenti:

- segnale di ***Input Enable (Chip Select, CS)***: linea che consente di attivare la fase di memorizzazione
- segnale di ***Output Enable***: rende visibile in uscita la parola memorizzata



Memorie

Le memorie sono dispositivi di memorizzazione logicamente assimilabili a banchi di registri, ma non dal punto di vista architetturale.

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.

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 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).

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 preset).
In alcuni casi le reti asincrone sono inevitabili:

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

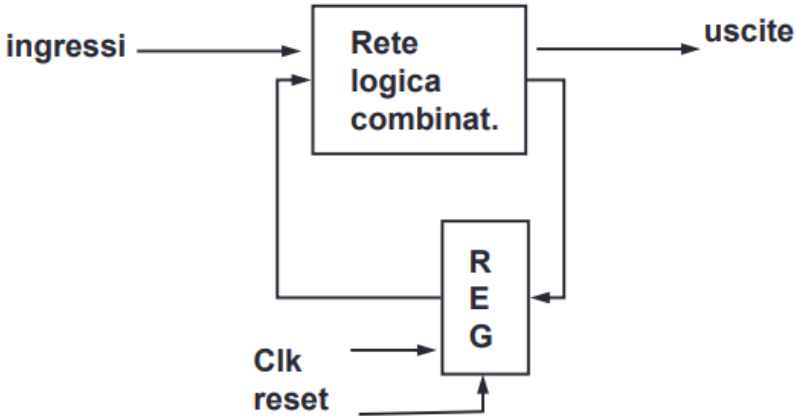
Reti sequenziali (2)

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**.

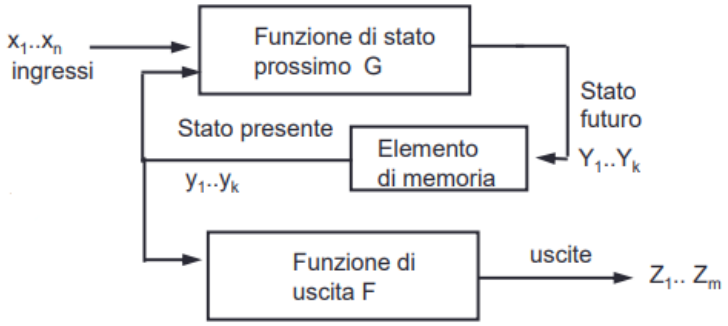
Sono di importanza fondamentale le **macchine a stati finiti (FSM)** in cui gli elementi di retroazione sono Flip Flop con un unico segnale di clock

L'insieme dei FF è detto **registro di stato** e memorizza lo stato futuro presentando a valle lo stato presente

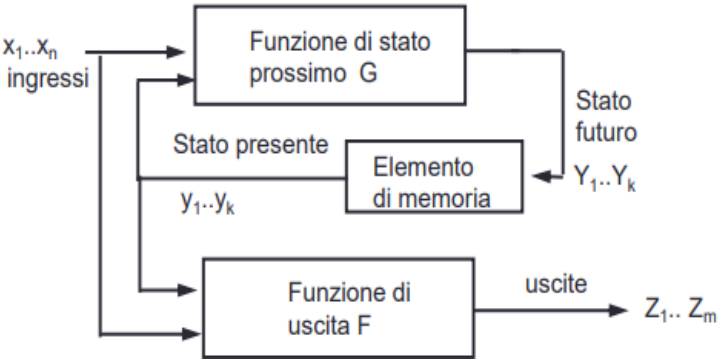


Automa a stati finiti

Automa di Moore: il valore delle uscite dipende solo dallo stato presente e non dagli ingressi in quell'istante



Automa di Mealy: il valore delle uscite dipende dallo stato presente e dagli ingressi in quell'istante



Essi sono equivalenti, quindi è sempre possibile passare da un modello all'altro.

Differenze:
Il modello di Moore:
- Contro**:** ha più stati, perché associa il valore dell'uscita allo stato
- Pro**:** concettualmente più semplice, ha funzioni di uscita più semplici

Il modello di Mealy:
- Contro**:** ha un fan-in maggiore, ovvero funzioni di uscita più complesse (rete + lenta)
- Pro**:** ha un minor numero di stati, meno bit necessari per la memorizzazione

Reti sequenziali (3)

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.

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.

Sintesi di reti sequenziali

1. **descrizione comportamentale a parole** o con un linguaggio di descrizione dell'hardware. (specifiche di progetto)
2. **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. **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. 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

Il diagramma degli stati è un ***grafo*** con
- tanti nodi quanti gli stati
- tanti archi quante le transizioni da uno stato all'altro dovute a cambiamenti degli ingressi

I valori delle uscite vengono rappresentati:

- negli archi se il modello è di Mealy
- nei nodi se il modello è di Moore

(vedi es a pagina 42)