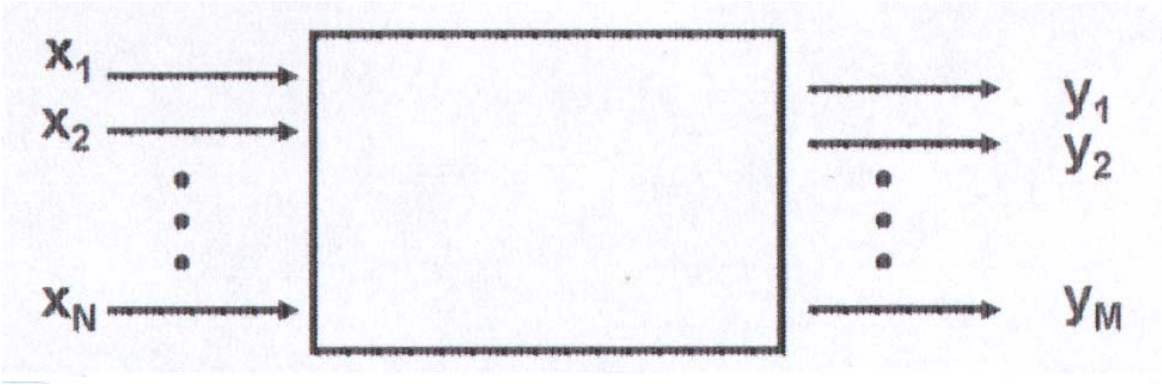
LOGICA SEQUENZIALE

Logica combinatoria

Un blocco di logica puramente combinatoria è un blocco con N variabili di ingresso e M variabili di uscita che sono funzione (booleana) degli ingressi in un certo istante.



Ad ogni istante le uscite dipendono solo dagli ingressi in quell'istante e non da quelli presenti in istanti precedenti.

Ricordiamo alcuni esempi di logica combinatoria che sono stati trattati in precedenza:

Encoder (codificatore): N bit in ingresso, M<N bit in uscita. L'uscita rappresenta quale dei bit in ingresso è attivo secondo un preciso codice.

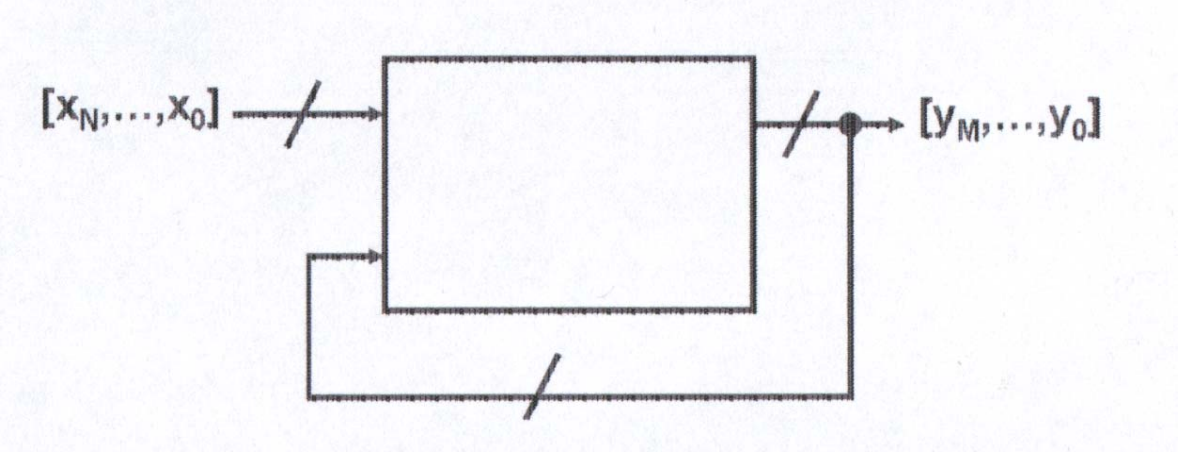
Decoder (decodificatore): N bit in ingresso, M>n bit in uscita. Viene attivata una delle uscite a seconda del codice in ingresso.

Sommatore: coppia di numeri in ingresso, (su N bit), l'uscita è la somma binaria dei due numeri (con riporto).

Multiplexer: N linee in ingresso, $\log_2 N$ bit di selezione, 1 uscita. L'uscita è uguale alla linea di ingresso selezionata dai bit di selezione.

Logica sequenziale

Un blocco di logica sequenziale è un blocco logico le cui uscite dipendono non solo dagli ingressi attuali ma anche dagli ingressi in istanti precedenti (oppure dallo stato che il sistema aveva assunto nell'istante precedente) cioè sono circuiti dotati di memoria.



Un esempio per distinguere un comportamento combinatorio da uno sequenziale si rende opportuno per chiarire le definizioni appena introdotte.

Esempio:

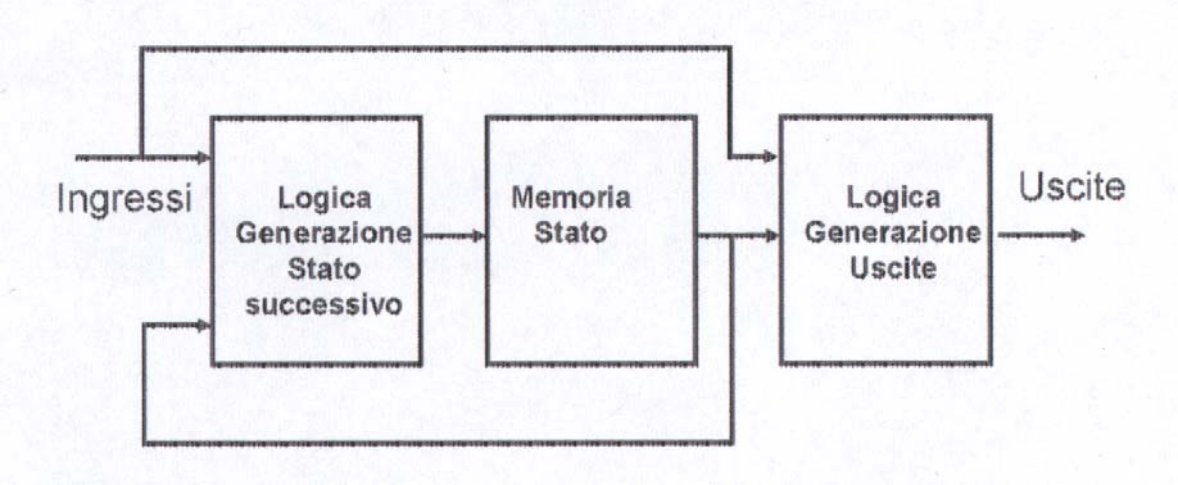
Supponiamo di avere una macchinetta distributrice di bevande automatica che funziona con l'inserimento di monete da 10 centesimi. Per avere una bevanda occorrono 60 centesimi. La macchinetta vede, attraverso la chiusura di un interruttore, il passaggio della moneta. Quando noi inseriamo le monete, si ha un comportamento sequenziale perché l'importo visualizzato dipende da quante monete sono state introdotte fino a quel momento.

Quando si preme un tasto per scegliere una bevanda si ha invece un comportamento combinatorio in quanto la scelta della bevanda è indipendente da quella fatta in precedenza.

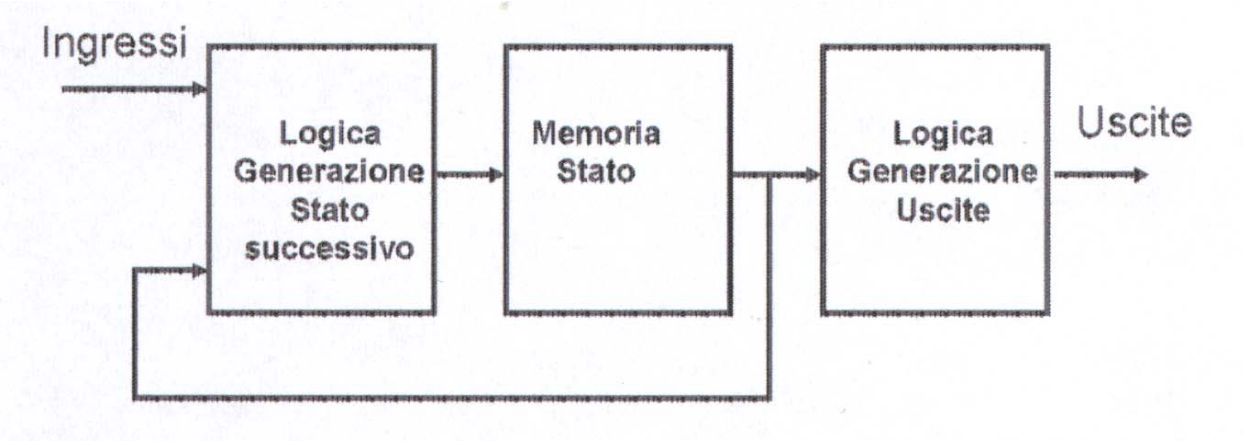
Macchine a Stati Finiti

Con circuiti dotati di memoria si realizzano le macchine a stati finiti. Le macchine a stati finiti rappresentano tutta la "storia passata" degli ingressi per mezzo di uno stato interno (rappresentato da N variabili binarie per un totale di 2^N stati possibili).

Ad ogni istante, se l'uscita di una macchina a stati è funzione degli ingressi attuali e dello stato assunto dalla macchina nell'istante precedente, allora, la macchina è detta "**Macchina a stati finiti di Mealy**" o "**Automa di Mealy**".



Se viceversa l'uscita dipende solo dalle variabili interne la macchina si dice " **Macchina a stati finiti di Moore**" o " **Automa di Moore**".

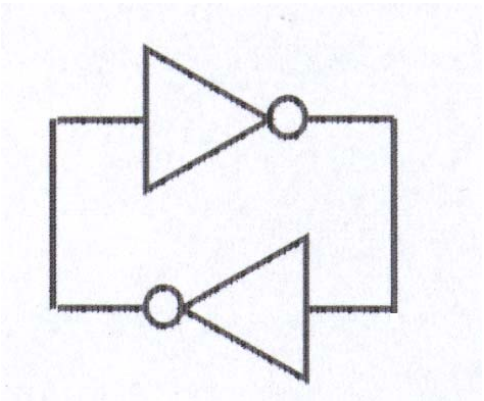


IL Latch

Come si è visto, l'elemento importante di distinzione tra logica combinatoria e logica sequenziale è la MEMORIA.

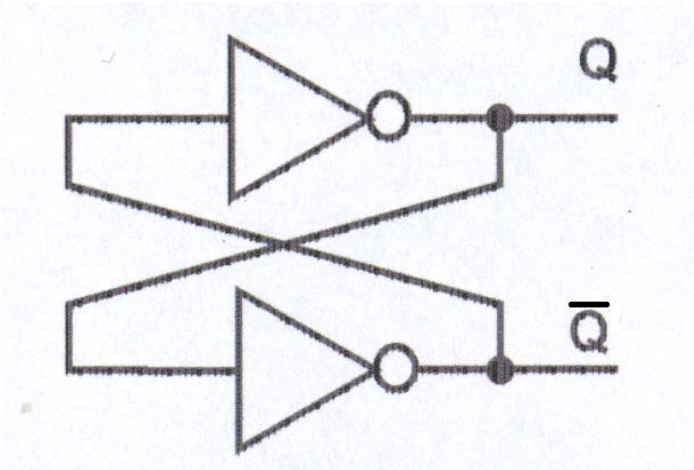
L'elemento base per l'implementazione di un elemento di memoria è il bistabile.

La connessione ad anello di due inverter realizza una contro-reazione positiva che è la base della memorizzazione.



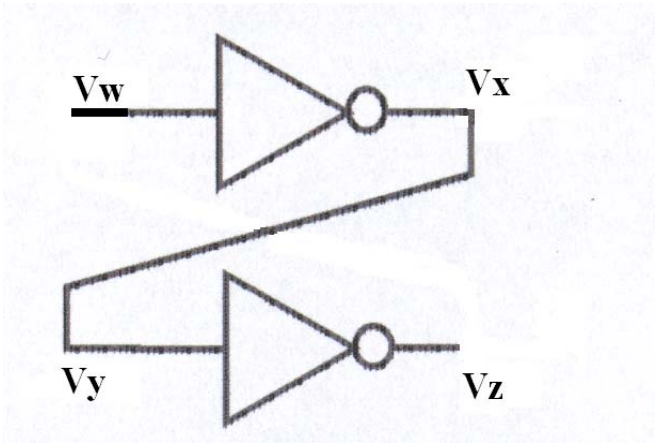
L'uscita del primo inverter è l'ingresso del secondo la cui uscita è l'ingresso del primo.

Girando, nella figura appena vista, uno dei due NOT si ottiene la classica disposizione in cui sono evidenziate le uscite Q e Q.



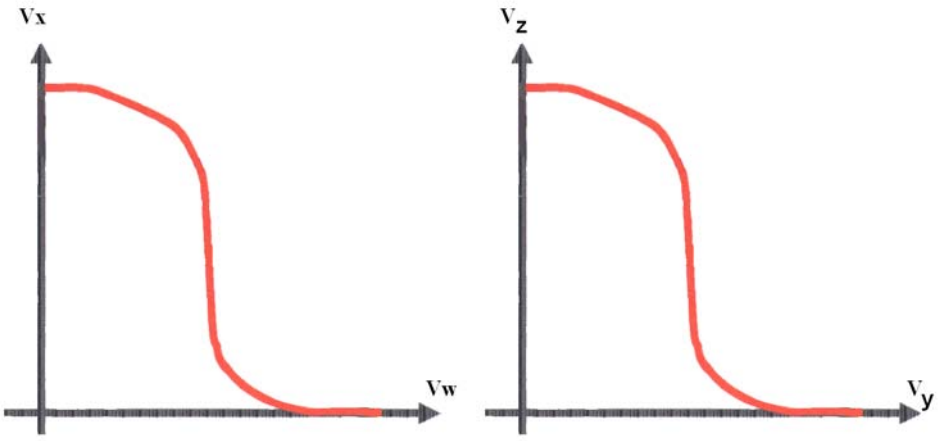
Un elemento di questo tipo è in grado di memorizzare un bit di informazione.

Per spiegare il funzionamento del latch si apre l'anello e si alimenta con un generatore di tensione l'ingresso del primo NOT. Si vede quando vale la tensione sull'uscita del secondo NOT.



Usiamo un procedimento grafico:

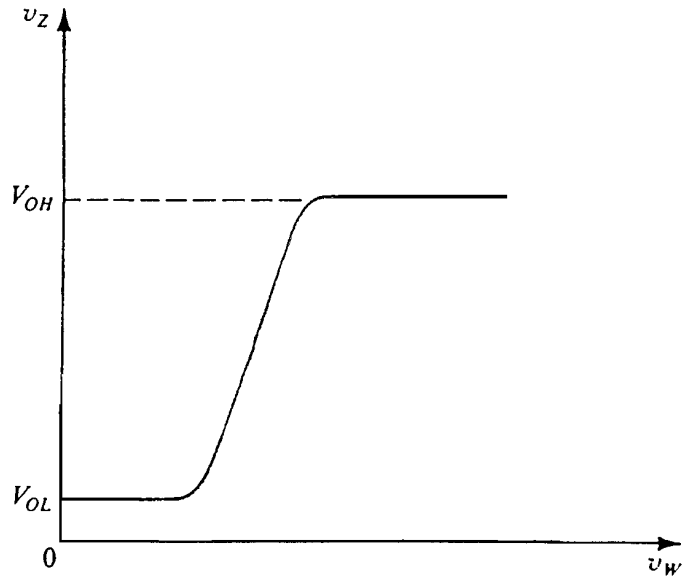
disegniamo le caratteristiche dei due NOT che sono uguali. Ricaviamo (per punti) da queste la caratteristica dei due NOT posti in cascata (dopo L'apertura dell'anello).



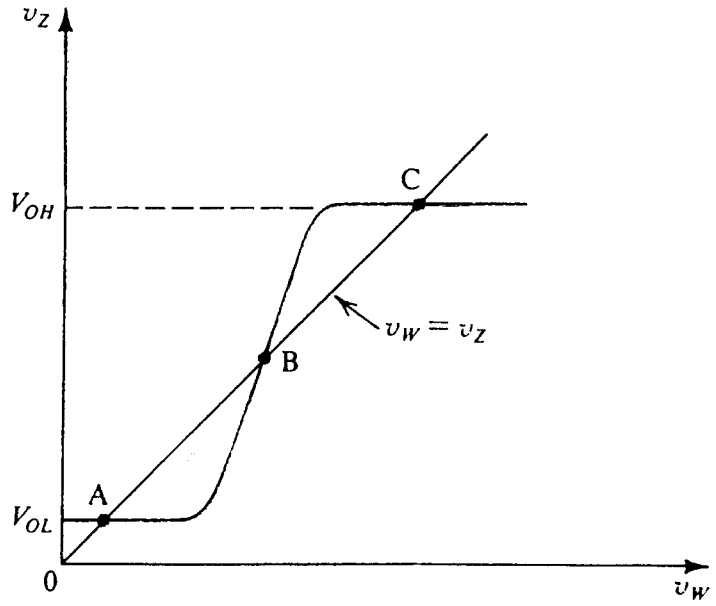
Supponiamo $V_W = 0 \rightarrow V_X = V_{OH} \rightarrow V_Y = V_X \rightarrow V_Z = V_{OL}$

E così via…… la costruzione può essere effettuata con l'ausilio di un compasso.

In questo modo si è giunti a disegnare il seguente grafico:



Ora dobbiamo chiudere l'anello cioè imporre che $V_W=V_Z$ questo significa - tradotto sul grafico - che i punti di funzionamento del bistabile sono i punti d'intersezione della curva ottenuta con la bisettrice $V_z=V_w$. Si individuano 3 punti A,B,C. di questi A e C sono punti di funzionamento stabili, mentre il punto B è instabile.



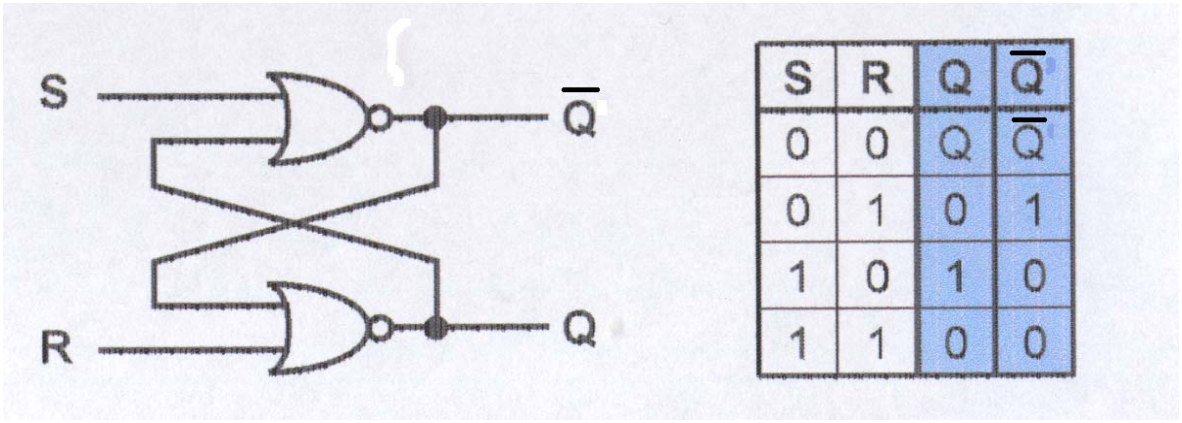
Infatti in *A* e *C* il guadagno d'anello è minore di 1 quindi variazioni di V_W dovuto ai disturbi (Rumore) non fa variare la tensione di uscita (vedi grafico), mentre in B il guadagno d'anello è molto maggiore di 1 e quindi piccole variazioni di V_W portano grandi variazioni dell'uscita riconducendola in una dei due punti stabili.

Per memorizzare un ' 1 ' si porta il bistabile nello stato in cui V_X è alto e V_Z è basso (Punto *A* del grafico) mentre per memorizzare uno ' 0 ' si porta il circuito nell'altro stato stabile (Punto *C* del grafico).

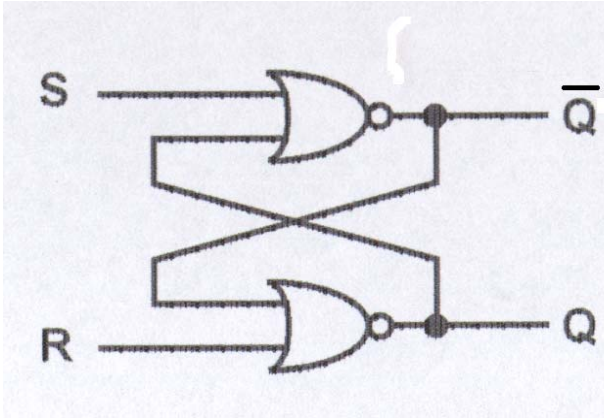
Flip Flop SR o Latch SR

Ora è necessario capire come fare per forzare il bistabile in uno dei 2 stati (cioè scrivere un valore).

Il Flip Flop SR è la più semplice implementazione di un elemento di memoria con segnali di scrittura (**S**et) e cancellazione (**R**eset). Un modo per realizzare tali funzionalità è fare uso di due porte NOR controreazionate come in figura nella quale è disegnata anche la tavola della verità.



S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0



S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	0	0

Osserviamo subito che quando gli ingressi S ed R sono entrambi ' 0 ' si riduce ad un bistabile perché una NOR con un ingresso a ' 0 ' equivale ad un inverter dell'altro ingresso (basta osservare la tabella della verità della NOR)

Viene memorizzato un ' 1 ' logico quando Q=1 e Q=0, uno ' 0 ' logico quando Q=0 e Q=1.

La combinazione S=1 e R=0 memorizza un ' 1 '

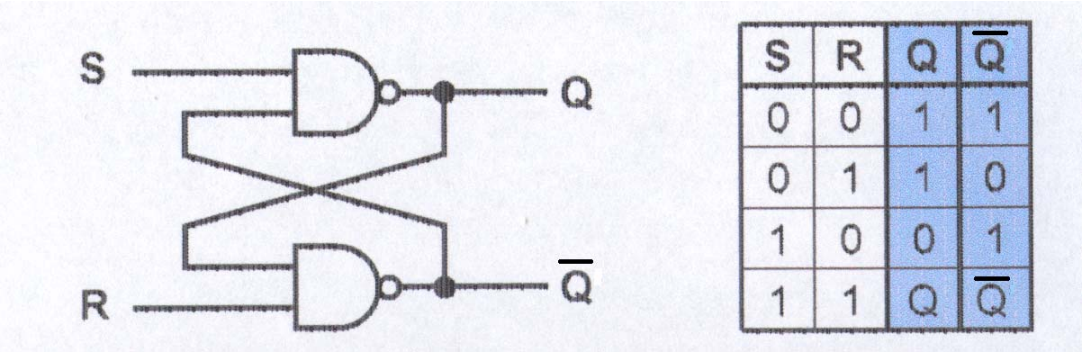
La combinazione S=0 e R=1 memorizza uno ' 0 '

La combinazione S=0 e R=0 non altera il valore

La combinazione S=1 e R=1 è proibita perché porta il latch in uno stato non consentito in cui Q=Q=0.

Il latch può essere realizzato anche con porte NAND.

L'implementazione è identica ma i segnali sono attivi bassi (S=0, R=1 per il set e S=1, R=0 per il reset)

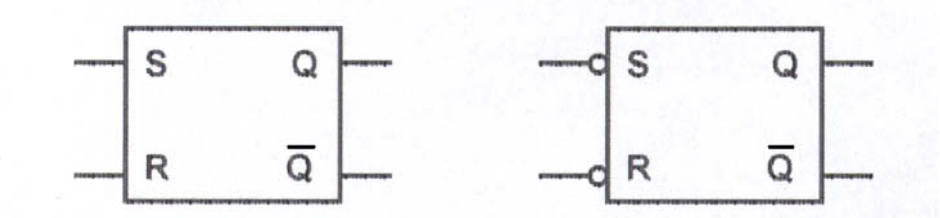


S	R	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$

osserviamo subito che quando S=1 e R=1 si riduce ad un bistabile perché una porta NAND con un ingresso a 1 equivale ad un inverter dell'altro ingresso (basta osservare la tavola della verità della porta NAND).

La condizione S=0 e R=0 è quella non usata perché porta il latch in uno stato non consentito Q=Q=1

Sotto sono raffigurati i simboli grafici del SR con porte NOR e del SR con porte NAND rispettivamente.

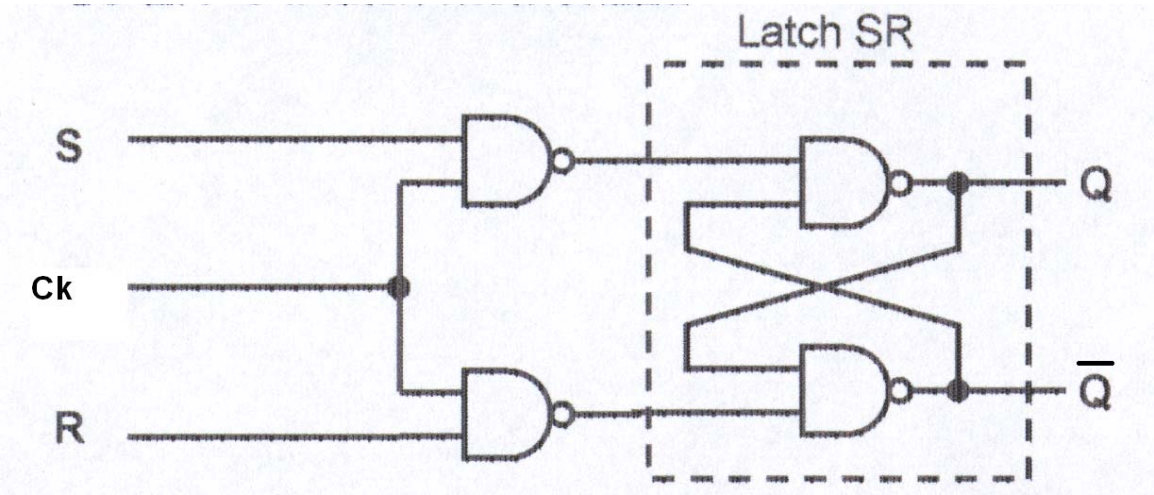


Simbolo grafico del latch SR con porte NOR

Simbolo grafico del latch SR con porte NAND

Latch SR sincronizzato

L'implementazione di un latch SR con segnale di Clock è la seguente.



Il clock serve per sincronizzare gli ingressi tra di loro e per sincronizzare l'uscita con l'ingresso.

Osserviamo che quando il clock è '0' le due porte NAND danno in uscita entrambe '1' , se guardiamo la tabella della verità del latch sr realizzato con porte NAND ci rendiamo conto che il latch memorizza.

Quando il clock diventa '1', in uscita dalle prime due NAND ritroviamo gli ingressi s ed r negati per cui ci ritroviamo nella situazione seguente:

La combinazione S=1 e R=0 memorizza un ' 1 '

La combinazione S=0 e R=1 memorizza uno ' 0 '

La combinazione S=0 e R=0 non altera il valore

La combinazione S=1 e R=1 è proibita perché porta il latch in uno stato non consentito in cui Q=Q=0.

Ck	S	R	Q	Q
0	x	x	Q	Q
1	0	0	Q	Q
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

Tabella della verità del latch SR sincronizzato.