3 CIRCUITI SEQUENZIALI

Nei circuiti realizzati con **logica combinatoria** l'output del circuito dipende esclusivamente dai valori presenti in quel momento agli ingressi.

Nei circuiti realizzati con **logica sequenziale** l'output del circuito dipende sia dal valore corrente sia da valori precedenti dell'input. **In tal senso si dice che il sistema ha memoria**

I circuiti sequenziali sono caratterizzati da uno **stato interno** il quale rappresenta l'informazione che mantiene la **storia** di un circuito sequenziale.
La storia del circuito è necessaria per prevedere il suo comportamento futuro.

Lo **stato di un sistema** viene memorizzato in componenti detti "**state elements**" come i **latches e flip-flop**

Com'è realizzato un circuito sequenziale?
Un circuito sequenziale (**sincrono**) è composto da:
- una **logica combinatoria** che definisce l'**evoluzione del sistema**
- **banchi di flip-flop** che servono a memorizzare gli **stati del sistema**

Un aspetto peculiare dei sistemi sequenziali è quello della retroazione (feedback), ovvero i segnali di output vengono riportati in input

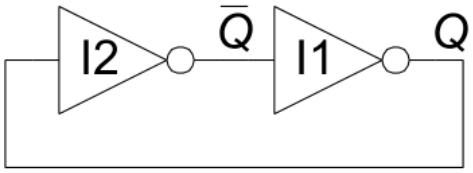
State elements
Nella logica sequenziale lo stato di un circuito influisce sull'evoluzione del sistema.
Gli **state elements** sono tutte quelle componenti circuitali fondamentali che vengono adoperate per memorizzare lo stato di un circuito.
- Circuiti bistabili
- SR Latch
- D Latch
- D Flip-flop

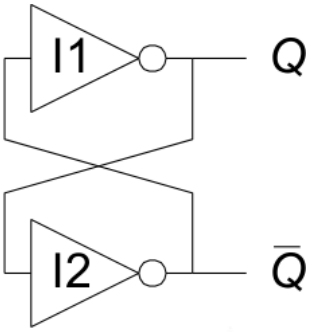
Circuito bistabile
E' un blocco costitutivo fondamentale di memoria, fa da building block per altri state elements.

E' un circuito che ha due stati stabili ed è composto da una **coppia di inverter (porte NOT) retroazionati** collegati ad **anello in modo sequenziale** oppure **in croce**.

L'ingresso del primo inverter I1 è l'uscita del secondo inverter I2 e viceversa
Il circuito non ha ingressi perchè sono gli output ad essere retroazionati verso l'input.

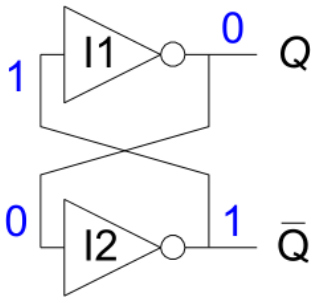
Ci sono due output: Q e Q*
Q dipende da Q*, e Q* dipende a sua volta da Q.
Per analizzare il circuito si considerano i due casi in cui Q è 0 oppure in cui Q è 1





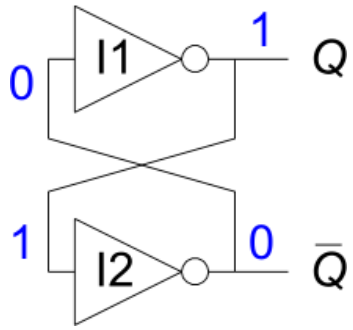
Caso I: Q = 0

I2 riceve in ingresso il valore di Q ossia Q=0 e di conseguenza produce un'uscita su Q*=1.
I1 riceve in ingresso Q*=1, e produce a sua volta un'uscita Q=0.
Dal momento che Q = 0 di nuovo come all'inizio, questo caso viene detto **stabile o consistente**



.

Caso II: Q = 1
I2 riceve in ingresso il valore di Q ossia Q=1 e di conseguenza produce un'uscita su Q*=0.
I1 riceve in ingresso Q*=0, e produce a sua volta un'uscita Q=1.
Dal momento che Q = 1 di nuovo come all'inizio, anche questo caso viene detto **stabile o consistente**



Dal momento che gli inverter collegati hanno due stati stabili, Q = 0 e Q = 1, il circuito viene chiamato **bistabile.**
Esso memorizza 1 bit di informazione nella variabile di stato Q (o Q negato)
Il valore di Q contiene tutte le informazioni sul passato necessarie a spiegare il comportamento futuro della rete. Nello specifico, se Q = 0, il valore rimarrà sempre zero, mentre se Q = 1, il valore rimarrà sempre 1

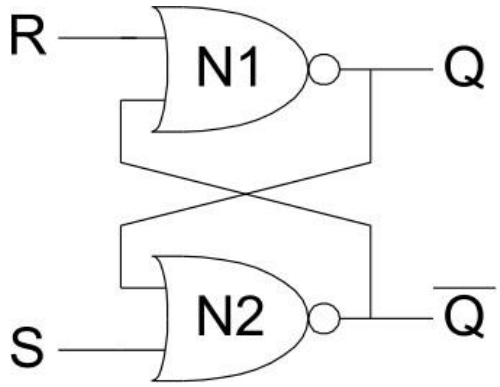
Il circuito all'inizio si troverà in uno dei due stati. Q=0 o Q=1 che dipende dalla configurazione iniziale del sistema. Quando una rete sequenziale viene accesa, lo stato iniziale è sconosciuto e solitamente imprevedibile, e può essere diverso a ogni nuova accensione della rete

L'utente non ha a disposizione un ingresso che gli permetta di controllare lo stato per questo si utilizzano i latch e i flip flop

SR (Set/Reset) Latch
Il latch SR rappresenta uno dei circuiti sequenziali più semplici ed è composto da **2 porte NOR collegate a croce**. Il latch ha due ingressi, **S e R**, e due uscite, **Q e Q***.

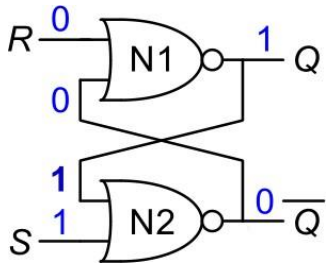
Il latch SR è simile al circuito bistabile ma **il suo stato può essere controllato mediante gli ingressi, S e R, che attivano (set) e disattivano (reset) l'uscita Q.**

Consideriamo i **4 possibili stati** , combinazione degli ingressi S e R:
S = 1, R = 0 SET
S = 0, R = 1 RESET
S = 0, R = 0 MEMORIA
S = 1, R = 1 NULL STATE



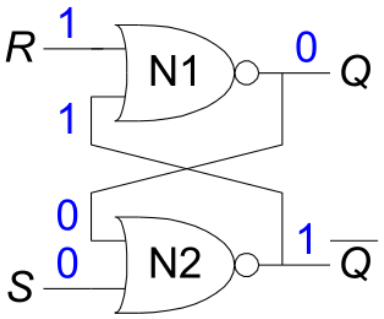
SET (S = 1, R = 0)

La porta NOR N1 riceve come ingressi R=0 e Q*.
Dal momento che il valore di Q* a questo punto è ancora sconosciuto, non è possibile determinare che valore assume Q.
La porta NOR N2 riceve in ingresso S=1, quindi produce un'uscita **Q*=0** (per la tavola di verità del NOR).
A questo punto è possibile tornare a N1 e, sapendo che entrambi gli ingressi S e R sono 0, il valore dell'uscita **Q= 1**, quindi abbiamo un SET



RESET (S = 0, R = 1)

La porta NOR N1 ha come input R=1 quindi produce un'uscita **Q=0**.
La porta N2 ha come ingressi quindi Q=0 e S=0, quindi produce un'uscita **Q*=1**



MEMORIA (S = 0, R = 0)

N1 riceve gli ingressi 0 e Q*. Dal momento che il valore di Q* è sconosciuto, non è possibile determinare il valore dell'uscita.
N2 riceve a sua volta gli ingressi 0 e Q, ma visto che anche il valore di Q è sconosciuto, anche in questo caso non è possibile determinare il valore dell'uscita.

Quindi consideriamo l'ipotesi Q abbia assunto un valore precedentemente noto, **Qprev** che può ovviamente essere 0 o 1 e rappresenta lo stato del sistema

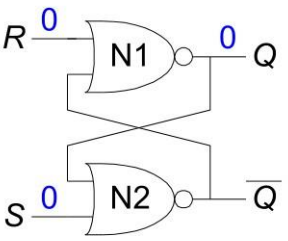
• **Qprev = 0**
S e Qprev sono entrambi 0, N2 produce un'uscita Q*=1.
A questo punto N1 riceve in ingresso Q*=1 e ha R=0 quindi la sua uscita **Q = 0 = Qprev**

•**Qprev = 1**
Se Qprev=1 e S=0 abbiamo che N2 produce un uscita Q*=0
Ora N1 riceve due ingressi R =0 e Q* = 0, quindi la sua uscita **Q = 1 = Qprev**

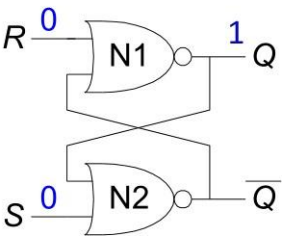
Quando R e S assumono valore 0, Q tiene memoria di tale valore precedente Qprev quindi si dice che il circuito mantiene memoria

$S = 0,\ R = 0$:
allora $Q = Q_{prev}$
Memoria

$Q_{prev} = 0$



$Q_{prev} = 1$



NULL STATE (S=1, R=1)

N1 e N2 hanno entrambi almeno un ingresso VERO (R oppure S), quindi entrambi producono un uscita pari a 0. Di conseguenza, sia Q sia Q* sono 0.
Questo è uno stato NON VALIDO perchè non può essere Q e Q* siano 0 perchè devono essere una il complemento dell'altra

$S = 1, R = 1$:
allora $Q = 0, \bar{Q} = 0$
Stato non valido
$Q \neq \text{NOT } \bar{Q}$

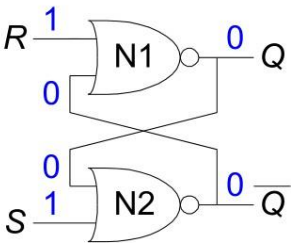


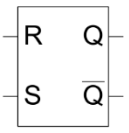
TABELLA DI VERITA'

Caso	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prec}	$\bar{Q}_{prec}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

Se dalla condizione S=R=1 si passa alla condizione S=R=0 allora
- Se i tempi di propagazione sono uguali allora il circuito va in oscillazione
- Nell'ipotesi, più realistica che le porte abbiano ritardi anche lievemente differenti, il circuito si mette in uno dei due stati possibili. Anche in questo caso, però, lo stato finale non è predicibile

- SR sta per Set/Reset
  - Memorizza un bit (Q)
- Set: Pone l'output a 1
  ($S = 1, R = 0, Q = 1$)
- Reset: Pone l'output a 0
  ($S = 0, R = 1, Q = 0$)
- Memoria: mantiene memoria dell'output
  ($S = 0, R = 0, Q = Q_{prev}$)

SR Latch Symbol



Occorre evitare lo stato non valido S = R = 1

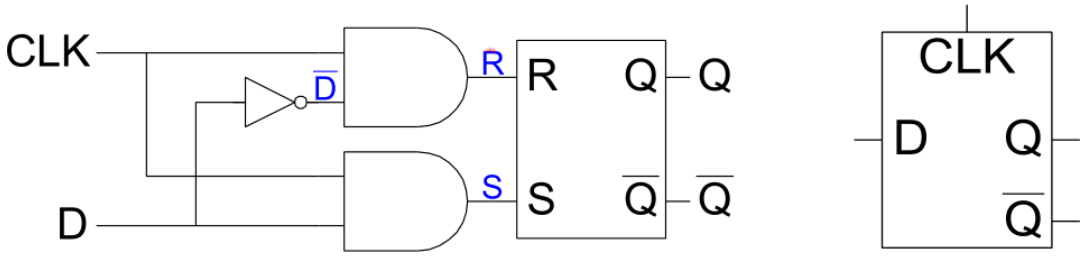
Come i negatori collegati a croce, anche il latch SR è un elemento bistabile con un bit di stato immagazzinato in Q. In questo caso però lo stato può essere controllato attraverso gli ingressi S e R. Quando viene attivato R, lo stato viene resettato a 0. Quando viene attivato S, lo stato viene settato a 1. Quando nessuno degli ingressi è attivato, lo stato mantiene il suo valore precedente

D LATCH
E' un estensione del latch SR
Il D latch ha **2 ingressi:**
- **un data input, D, che controlla in che cosa l'output cambia** al prossimo stato

- **un ingresso clock, CLK, che controlla invece quando l'output cambia** ossia in quale momento avviene il cambio di stato.

In particolare:
Se **CLK = 1**, D passa fino a Q (**stato transparente**) in output abbiamo quindi il valore di D
Se **CLK = 0** , Q mantiene il suo valore precedente (**stato opaco**) in output abbiamo quindi il valore Qprev

Il clock fa si che si eviti lo stato non valido



CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

Quando **CLK = 0,** sia S sia R sono 0, indipendentemente dal valore assunto da D.
Q ricorda il suo valore precedente, Qprev. STATO OPACO

Se invece **CLK = 1**, allora una porta AND produce un valore VERO e l'altra un valore FALSO, a seconda del valore di D che viene negato.
In questo caso avremo che i dati scorrono da D verso Q, D=Q. STATO TRASPARENTE

D Flip Flop

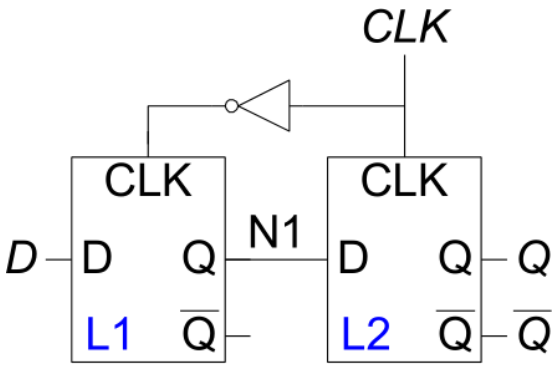
E' una variante del D-Latch
Presenta sempre 2 input D e CLK

La peculiarità del D Flip Flop è che:
quando il CLK passa da 0 a 1, ossia nel fronte di salita del clock, D passa fino a Q
altrimenti, Q mantiene il suo valore precedente Qprev.
Quindi Q cambia solo durante la transizione di CLK da 0 a 1.

Queste tipologie di componenti sono dette **edge-triggered** perché sono pilotate dalla transizione del clock

E' possibile ottenere un D flip flop mediante la combinazione di 2 D-Latch (L1 e L2) controllati da segnali di clock complementari.

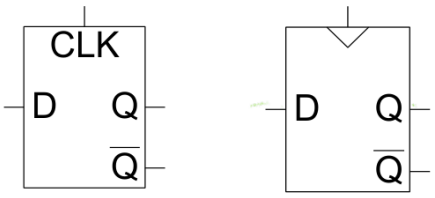
Il primo latch, **L1, viene detto master,** mentre il secondo latch, **L2, viene detto slave.**
Il nodo che unisce i due latch prende il nome N1.

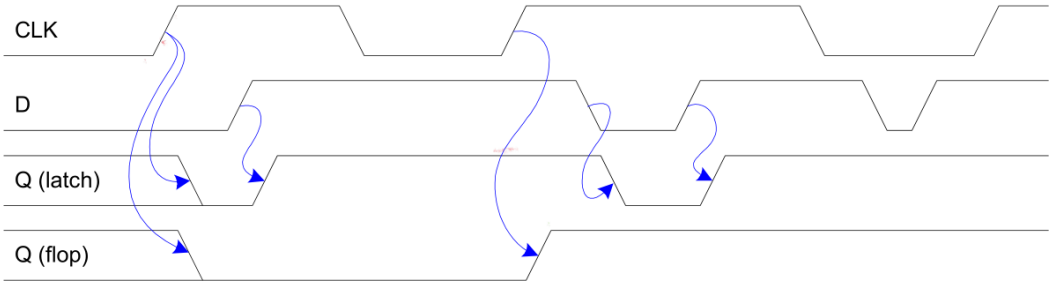


Quando CLK = 0,
il latch **L1 master è trasparente**
il latch **L2 slave è opaco**
Qualsiasi valore di D viene portato a N1

Quando CLK = 1,
il latch **L1 master è opaco**
il latch **L2 slave è trasparente**
il valore di N1 viene trasmesso a Q, ma N1 resta isolato da D.

Quindi, D passa fino a Q sulla transizione di CLK da 0 a 1
In tutti gli altri casi, ad esempio se D cambia quando CLK=1 , D non passa a Q e
Q mantiene il suo valore precedente, dal momento che c'è sempre il latch L1 opaco che blocca il passaggio di dati tra D e Q.



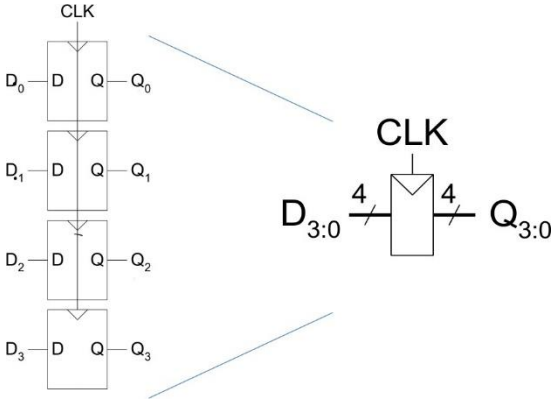


Il cambiamento di Q nel D flip flop avviene solo sul fronte di salita del clock, è più sincronizzato con esso
Nel D latch il cambiamento di Q dipende sia dal valore del clock che dal valore di D

La freccia indica la causa di un cambiamento nell'uscita. Il valore iniziale di Q è sconosciuto e può quindi essere 0 o 1, come indica la coppia di linee orizzontali. Per prima cosa si consideri il latch. Sul primo fronte di salita di CLK, D = 0 quindi Q diventa 0. Ogni volta che D cambia valore, mentre CLK = 1, Q si comporta nello stesso modo. Quando invece D cambia mentre CLK = 0, il suo cambiamento viene ignorato. Si consideri ora il flip-flop: per ogni fronte di salita di CLK, D viene copiato in Q. In tutti gli altri casi, Q mantiene il suo stato precedente.

Registri: Multi-bit Flip-Flop
Un registro a N bit è un banco di N flip-flop D messi in parallelo che condividono un ingresso CLK comune, in modo che tutti i bit vengano aggiornati allo stesso tempo.
I registri costituiscono i blocchi costitutivi chiave per la maggior parte dei circuiti sequenziali.

un registro a quattro possiede un ingresso D3:0 e un'uscita Q3:0. Sia D3:0 sia Q3:0 sono bus a quattro bit.



Flip-Flops “enabled”
Un flip-flop enable ha 3 input:
-CLK che controlla quando lo stato deve cambiare
- D input data
- EN , ENABLE, abilitatore che serve per determinare se memorizzare o no il dato sul fronte alto del clock.

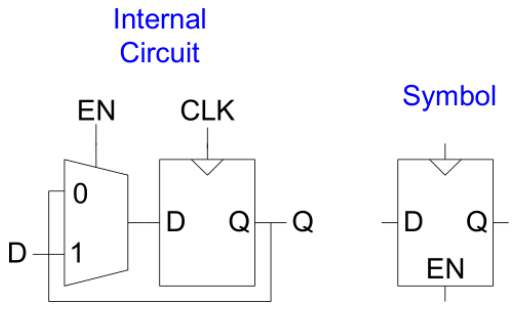
Quando EN=1
il flip-flop si comporta come un normale flip-flop D e memorizza il dato, quindi D passa fino a Q sul fronte alto del clock

Quando EN=0
il flip-flop indipendentemente dal valore del clock mantiene il suo stato precedente.

I flip-flop con abilitazione sono utili quando si desidera inserire un nuovo valore in un flip-flop esclusivamente in alcuni precisi momenti, piuttosto che a ogni cambio del clock.

Può essere realizzato con un multiplexer che ha come selettore EN

EN=1 trasmette il valore dell’ingresso D e poi viene trasportato a Q sul fronte alto del clock
EN=0, l’output sarà il valore precedente di Q memorizzato, Qprev



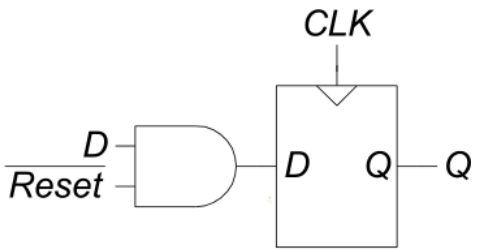
Flip-Flops “settabili” **semplice circuito sequenziale sincrono**
Flip-Flops “resettabili”
Un flip-flop resettabile ha 3 input:
- Un flip-flop resettabile ha 3 input
-CLK che controlla quando lo stato deve cambiare
- CLK che controlla quando lo stato deve cambiare
- D input data
- D input data
- RESET, che serve per resettare il flip flop

Quando l’ingresso **RESET = 0**, il flip-flop resettabile si comporta come un normale flip-flop D.

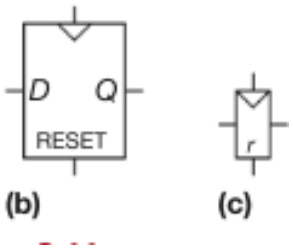
Quando invece **RESET = 1,** il flip-flop resettabile ignora D e, appunto, resetta l’uscita **Q= 0**.

Questa tipologia di flip-flop è utile nel caso in cui si desideri forzare uno stato 0 in tutti i flip-flop della rete quando viene accesa.

Vi sono due tipi di flip-flop resettabili:
- **Sincroni:** il reset è pilotato dal clock, quindi sul fronte alto del clock, se reset=1 allora Q=0
- **Asincroni**: il reset avviene non appena Reset = 1



RESET è un segnale attivo basso, il che significa che il segnale di reset esegue la propria funzione quando è 0 e non 1, forzando a 0 l’uscita della porta and.
Con l’aggiunta di un negatore, la rete avrebbe invece un segnale di reset attivo alto.

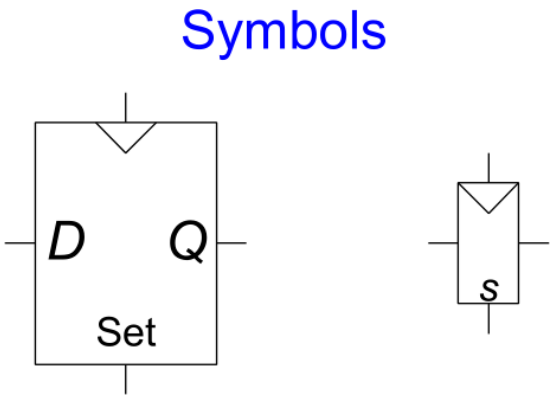


Flip-Flops “settabili”
Un flip-flop resettabile ha 3 input:
-CLK che controlla quando lo stato deve cambiare
- D input data
- SET, che serve per settare il flip flop

Un flip-flop D è il più semplice circuito sequenziale sincrono

Quando l'ingresso **SET = 0**, il flip-flop resettabile si comporta come un normale flip-flop D.

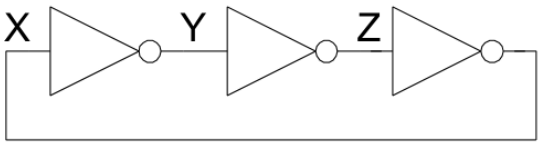
Quando invece **SET = 1,** il flip-flop resettabile ignora D e, appunto, setta l'uscita **Q=1**.

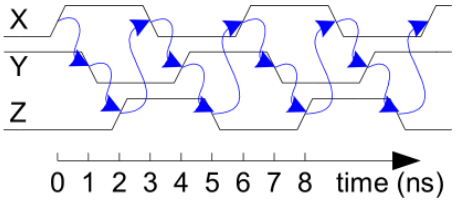


Criticità nella logica sequenziale
Alcuni circuiti sequenziali sono alquanto problematici perchè presentano delle criticità, dovute appunto alla logica sequenziale.

Un esempio è un circuito formato da 3 porte not in serie collegate ad anello.
Innanzitutto notiamo che il circuito è asincrono perchè l'output del terzo inverter è retroazionato in maniera diretta verso l'ingresso del primo inverter.
Se analizziamo il circuito notiamo che esso non ha stati stabili e dunque viene detto circuito **astabile**. Ogni nodo, quindi, oscilla tra 0 e 1 in un periodo (cioè in un tempo di ripetizione) pari a 6 ns. Questa rete è chiamata oscillatore ad anello.
Il periodo dell'oscillatore ad anello dipende dal ritardo di propagazione di ogni singolo negatore

Ogni ingresso prima è 0 e poi dopo è 1, poi di nuovo 0 e poi 1 e così via..nessun stato stabile





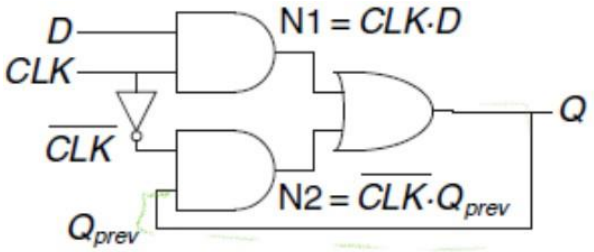
Adesso si prenda in analisi questo **latch d'asincrono,** esso presenta dei malfunzionamenti dovuti all'uso di più porte AND, NOT, OR che accumulano ritardi sui singoli cammini.

Si supponga che CLK = D = 1: in questo caso il latch è trasparente e trasmette D per portare Q = 1. Se a questo punto CLK scende a 0, il latch dovrebbe ricordarsi dell'ultimo valore, mantenendo Q = 1.
Si supponga però che ci sia un forte ritardo attraverso il negatore che va da CLK a CLK* maggiore di quello delle porte AND e OR: i nodi N1 e Q nel frattempo potrebbero entrambi abbassarsi prima che CLK abbia il tempo di alzarsi.

In questo caso, N2 non si alzerebbe mai, e Q rimarrebbe bloccato al valore 0.

Questo è un esempio di progetto di rete asincrona, nella quale le uscite sono direttamente collegate in retroazione agli ingressi

Un flip-flop D è il più semplice circuito sequenziale sincrono



$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

CLK 1→0 Q oscilla

Logiche sequenziali sincrone

I circuiti sequenziali asincroni (nei quali l'output è retroazionato in maniera diretta verso l'input) presentano delle criticità a volte difficilmente analizzabili che possono dipendere anche dalla struttura fisica dei componenti e dai ritardi accumulati, che possono causare dei malfunzionamenti nel circuito e portati a circuiti astabili oppure in stati di oscillamento

Per evitare questi problemi, si cerca di evitare di retroazionare l'output in maniera diretta e si interpone un registro nel ciclo di retroazione.
Questa operazione trasforma il circuito in un insieme di logica combinatoria e registri.
I registri contengono lo stato del sistema, che cambia solo in corrispondenza dei fronti di salita del clock, motivo per cui lo stato viene detto sincronizzato con il clock.

Il clock è abbastanza lento da far sì che tutti gli ingressi dei registri abbiano il tempo di adeguare il proprio valore prima del fronte di clock successivo

Quindi i circuiti sequenziali sincroni sono regolati dal clock

In generale un circuito sequenziale sincrono ha un insieme finito di stati {S0,…, Sk-1}.
L'output del sistema è funzione sia dell'input che dello stato corrente del sistema.
Il prossimo stato è funzione sia dell'input che dello stato corrente del sistema.

- Logica sequenziale sincrona:

$out = f(in, s_c)$
$s_n = g(in, s_c)$

Design di logiche sequenziali sincrone
- Inserire registri nei cammini ciclici
- I registri determinano lo stato S0,…, Sk-1 del sistema
- I cambiamenti di stato sono determinati dalle transizioni del clock: il sistema è sincronizzato con il clock

- **Regole di composizione:**

- Ogni componente è un **registro** o un **circuito combinatorio**

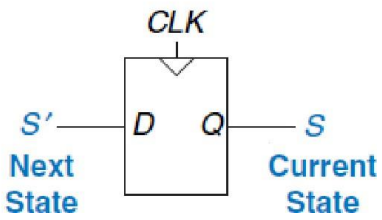
Un flip-flop D è il più semplice circuito sequenziale sincrono
- Almeno **un componente è un registro**
- Tutti i registri sono **sincronizzati** con un unico **clock**
- Ogni **ciclo** contiene **almeno un registro**

Due tipici circuiti sequenziali sincroni sono
Finite State Machines (FSMs)
Pipelines

Un flip-flop D è il più semplice circuito sequenziale sincrono

possiede un ingresso, D, un clock, CLK, un'uscita, Q, e due stati, {0, 1}.
La specifica funzionale di un flip-flop definisce che il next state è D e che l'uscita, Q, è il current state

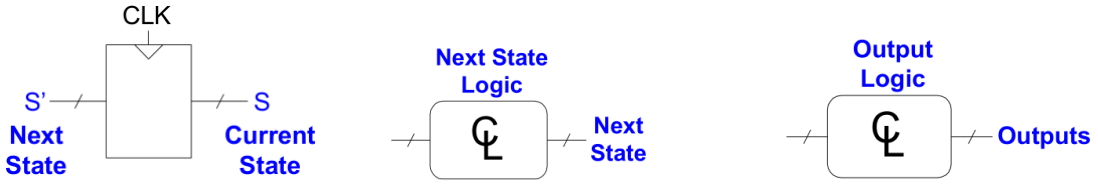
- $Q = s_c$
- $D = s_n$



Finite State Machines
Una FSM è una macchina a stati finiti, il nome deriva dal fatto che un circuito con k registri può trovarsi in uno di un numero finito (2 alla k) di stati diversi.

Una FSM possiede M ingressi, N uscite e k bit di stato. Inoltre, riceve un segnale di clock e, a volte, anche un segnale di reset.

Una FSM è composta da:
-almeno un registro, detto **STATE REGISTER**, che memorizza lo stato corrente e carica il prossimo stato al battere del clock
-da due blocchi di **logica combinatoria**: **la logica di next state** e **la logica di output**
La logica di next state "computa" il prossimo stato del sistema in funzione degli input e dello stato corrente **sn = g (in, sc)**
La logica di output "computa" l'output del sistema (f)



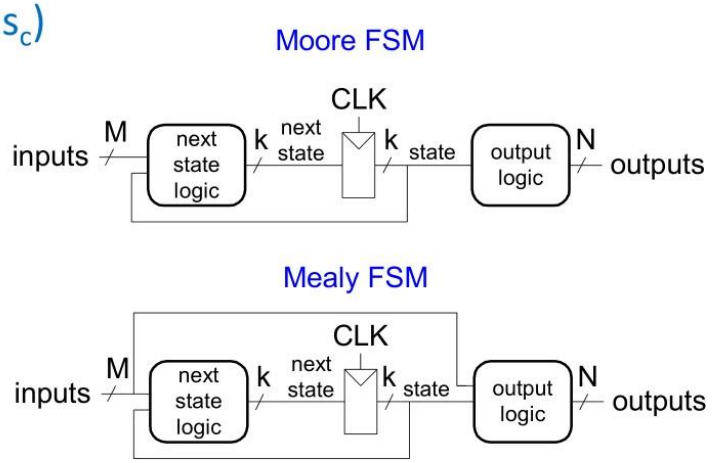
Ci sono 2 tipologie di FSM a seconda della logica di output:
- **MOORE FSM**: l'output è funzione del solo stato corrente **out = f (sc)**
- **MEALY FSM** l'output è funzione sia degli input che dello stato corrente **out=f (in, sc)**

In entrambe la logica di next state è funzione sia dell'input che dello stato corrente
Mealy è più generale.

- s_n dipende sia dall'input che da s_c

  $s_n = g(in, s_c)$

- 2 tipi di FSM a seconda della logica di output:
  - **Moore FSM:** $out = f(s_c)$
  - **Mealy FSM:** $out = f(in, s_c)$



Come si progetta una FSM?

1- Si identificano **gli input e output** del sistema

2- Si **disegna il diagramma degli stati**, che indica tutti i possibili stati del sistema e le transizioni tra di essi. Nel diagramma degli stati, i cerchi rappresentano gli stati e gli archi rappresentano le transizioni tra di essi. Le transizioni avvengono al fronte di salita del clock

3- Si riporta il diagramma degli stati nella **tabella di transizione degli stati** che indica, per ogni stato presente e valori di ingresso, il next state S' del sistema.

4- Selezionare un **encoding** degli stati: one hot o binario

5- Macchina di Moore/Mealy: **si riscrive la tabella di transizione degli stati con l'encoding** degli stati

6- Si scrive in maniera analoga la **tabella degli output** che indica, per ogni stato, quale deve essere il valore assunto dall'uscita.

7- Scrivere le **espressioni booleane** relative alla logica di prossimo stato e alla logica di output in forma SOP

8- Minimizzare le espressioni ottenute

9- Realizzare **schema circuitale** mediante le espressioni minimizzate di logica next state e logica di output

- Inputs: *CLK*, *Reset*, T_A, T_B
- Outputs: L_A, L_B

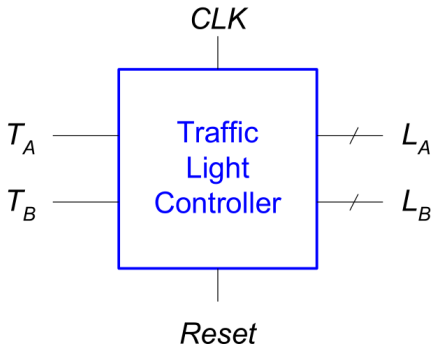


Diagramma di transizione: Moore FSM

- **Stati:** labellati con gli outputs
- **Transizioni:** labellate con gli inputs

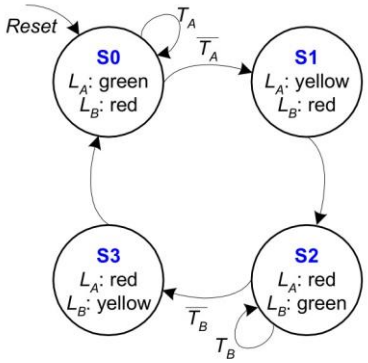


Tabella di transizione

Current State	Inputs		Next State
S	T_A	T_B	S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

State	Encoding
S0	00
S1	01
S2	10
S3	11

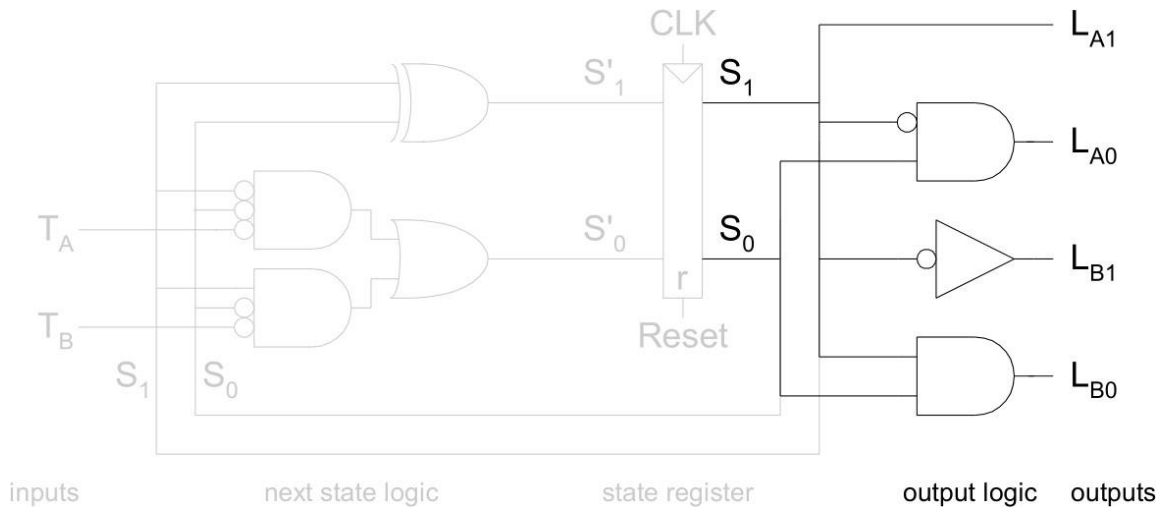
$$S'_1 = S_1 \oplus S_0$$
$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

Tabella dell'output

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

Output	Encoding
green	00
yellow	01
red	10

$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1}S_0$$
$$L_{B1} = \overline{S_1}$$
$$L_{B0} = S_1S_0$$

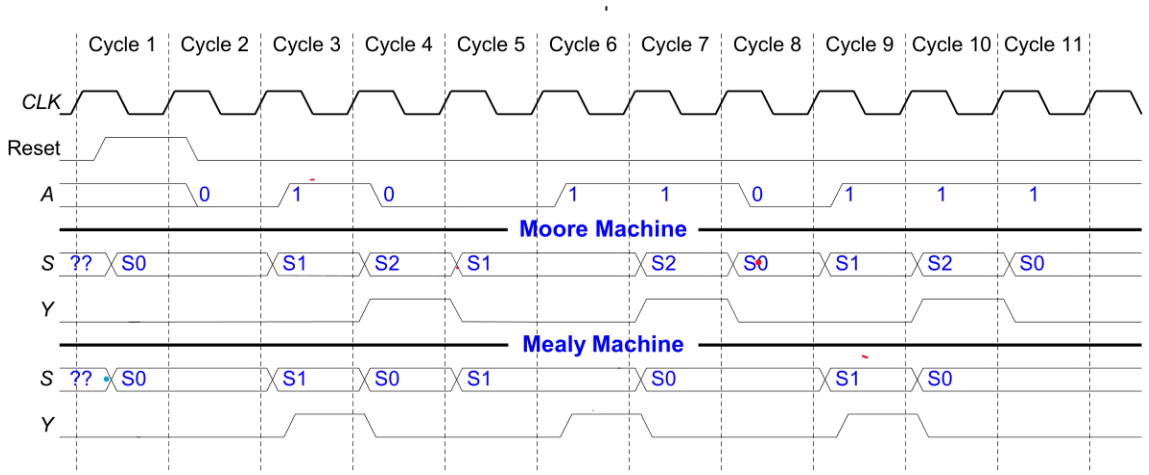


Encoding degli stati

Binario: ogni stato viene rappresentato da un numero binario.
Dal momento che K numeri binari possono essere rappresentati da log2K bit,
un sistema con K stati avrà bisogno solo di log2K bit di stato.

One hot: viene utilizzato un bit di stato per ognuno degli stati.
ogni bit è associato ad uno stato quindi solo un bit alla volta sarà alto
es per 4 stati, 0001, 0010, 0100, 1000
Richiede più flip-flops perchè ogni bit di stato viene immagazzinato in un flip-flop,
ma spesso la logica combinatoria associata è più semplice

Moore vs Mealy FSM. Timing diagram. Chi è più sincrono?



Le due macchine seguono due diverse sequenze di stati.
Nella macchina alla Mealy l'uscita passa a 1 in anticipo di un ciclo perché risponde all'ingresso invece di attendere il cambiamento dello stato. Se l'uscita della macchina alla Mealy fosse ritardata da un flip-flop, la sua temporizzazione corrisponderebbe a quella della macchina alla Moore. Nella scelta dello stile di progettazione di una FSM, serve quindi decidere quando si vuole che le uscite rispondano.

Nell' automa di Moore l'output è quasi sincronizzato rispetto al battere del clock, perchè esso dipende dal solo stato corrente e lo stato, essendo memorizzato nei registri è sincronizzato col clock e cambia sul fronte di salita.
Moore è più sincrono perché è in funzione del solo stato corrente che è sincronizzato col clock

Negli automi di Mealy l'output dipende sia dal registro che memorizza lo stato corrente che è sincronizzato col clock, ma anche dall'input e se l'input non è sincronizzato col clock il valore dell'output può cambiare in maniera non sincrona rispetto al clock. L'output viene influenzato da un certo ritardo rispetto al battere del clock dovuto al fatto che esso dipende anche dagli input. Il tempo per leggere l'output è ridotto in prossimità del battere del nuovo clock

Fattorizzazione di FSM
Spesso è più semplice progettare FSM complesse se queste possono essere decomposte in diverse macchine a stati più semplici che interagiscono tra loro, facendo sì che le uscite di alcune macchine siano gli ingressi di altre. Questa applicazione dei principi di gerarchia e modularità alle macchine viene chiamata fattorizzazione delle macchine a stati

Fattorizzare consiste quindi nel suddividere una FSM complessa in FSM più piccole che interagiscono fra loro

Parallelismo
Parallelismo vuol dire eseguire in contemporanea più task/compiti
La velocità di un sistema è caratterizzata dalla **latenza** e dal **throughput**
il parallelismo incrementa il throughput

Token: Gruppo di input da processare per ottenere un output significativo.
Pacchetto di informazioni

latenza: Tempo che occorre ad un token per essere processato e produrre un output

throughput: Numero di output prodotti per unità di tempo

2 tipi di parallelismo:
▪ **Spaziale:** duplicare l'hardware per eseguire più task contemporaneamente es. processori che lavorano in parallelo

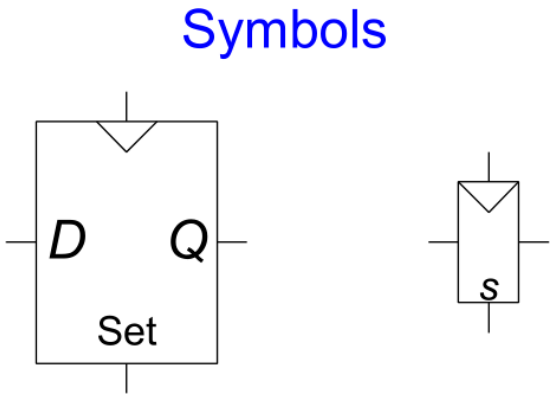
▪ **Temporale:** l'hardware è sempre 1 ma il task da eseguire viene suddiviso in più fasi e queste fasi sono eseguite in **pipelining** ossia in sequenza . Quando si finisce la fase di un task già inizia un'altra fase di un task diverso

il parallelismo temporale migliora il throughput a scapito della latenza.

- SET, che serve per settare il flip flop

Quando l'ingresso **SET = 0**, il flip-flop resettabile si comporta come un normale flip-

flop D. Quando invece **SET = 1,** il flip-flop resettabile ignora D e, appunto, setta



l'uscita **Q=1**.

Criticità nella logica sequenziale
Alcuni circuiti sequenziali sono alquanto problematici perchè presentano delle criticità, dovute appunto alla logica sequenziale.

Un esempio è un circuito formato da 3 porte not in serie collegate ad anello. Innanzitutto notiamo che il circuito è asincrono perchè l'output del terzo inverter è retroazionato in maniera diretta verso l'ingresso del primo inverter.
Se analizziamo il circuito notiamo che esso non ha stati stabili e dunque viene detto circuito **astabile**. Ogni nodo, quindi, oscilla tra 0 e 1 in un periodo (cioè in un tempo di ripetizione) pari a 6 ns. Questa rete è chiamata oscillatore ad anello.
Il periodo dell'oscillatore ad anello dipende dal ritardo di propagazione di ogni singolo negatore