# Reti sequenziali

Seppure le reti combinatorie permettono di realizzare funzioni aritmetico-logiche, esse non sono in grado di realizzare una delle funzionalità più importanti per un calcolatore elettronico, cioè la possibilità di memorizzare l'informazione e di usare tale memoria per elaborare in modo opportuno quest'ultima. Le reti sequenziali costituiscono lo strumento concreto per la realizzazione di tale possibilità.
In particolare, esse permettono l'implementazione di funzioni **con stato**, ossia di **automi a stati finiti**, che in un sistema di elaborazione sono utilizzati per la realizzazione sia della **parte controllo** che della **parte operativa**. Un automa a stati finiti può intuitivamente essere compreso come una macchina che ad intervalli di tempo regolari riceve informazioni in ingresso ed in base allo stato in cui essa si trova, provvede sia a generare un output che a (eventualmente) cambiare lo stato in cui si trova. In termini di poco più formali si può dire che:

un automa a stati finiti (con output) è una macchina caratterizzata da:

- **n** variabili logiche di **ingresso** ($2^n$ stati di ingresso $X_i$);
- **m** variabili logiche di **uscita** ($2^m$ stati di uscita $Z_i$);
- **k** variabili logiche di **stato interno** ($2^k$ stati interni $S_i$);
- una **funzione σ di transizione dello stato interno**
  $\sigma : X \times S \rightarrow S$ ;
- una **funzione ϖ delle uscite**
  $\varpi : X \times S \rightarrow Z$ ;

In un modello **ideale** di rete sequenziale di tipo **sincrono**, le variazioni di stato ($S_i \rightarrow S_{i+1}$) avvengono in corrispondenza degli istanti di una sequenza temporale **discreta** ($t_1,t_2,\ldots,t_n,\ldots$) di intervallo **costante** ($\Delta=t_n-t_{n-1}$).

A seconda di come le funzioni di transizione dello stato interno e di uscita sono definite, si ottiene un modello matematico di automa. I due modelli a cui si fa tipicamente riferimento sono il **Modello di Mealy** e il **Modello di Moore**.

Nel **Modello di Mealy** sia lo stato interno **successivo** $S(t_{i+1})$ che lo stato di uscita successivo $Z(t_{i+1})$ dipendono tanto dallo stato di ingresso **presente** $X(t_i)$ che dallo stato interno presente $S(t_i)$:

- $S(t_{i+1}) = \sigma(X(t_i), S(t_i))$
- $Z(t_{i+1}) = \varpi(X(t_i), S(t_i))$

Nel **Modello di Moore** lo stato interno **successivo** $S(t_{i+1})$ dipende tanto dallo stato di ingresso **presente** $X(t_i)$ che dallo stato interno presente $S(t_i)$, mentre lo stato di uscita successivo $Z(t_{i+1})$ dipende **solo** che dallo stato interno presente $S(t_i)$:

- $S(t_{i+1}) = \sigma(X(t_i), S(t_i))$
- $Z(t_{i+1}) = \varpi(S(t_i))$

In particolare, si noti che la sequenza di uscita è ritardata di un intervallo $\Delta$ rispetto a quella di un modello di Mealy poiché $Z(t_{i+1}) = \varpi(S(t_i)) = \varpi(\sigma(X(t_{i-1}), S(t_{i-1}))) = \varpi`(X(t_{i-1}), S(t_{i-1}))$, dove $\varpi`(\ ) = \varpi(\sigma(\ ))$ rappresenta una "nuova" funzione delle uscite.
Le funzioni σ ed ϖ sono tipicamente realizzate mediante reti combinatorie, che danno luogo alla parte combinatoria delle reti sequenziali.
Il modo classico di rappresentare (specificare) una rete sequenziale è mediante tabelle di verità. Ad esempio, consideriamo il seguente esempio di tabella, dove $y = S(t_i)$ ; $Y = S(t_{i+1})$ ; $x = X(t_i)$ ; $z = Z(t_{i+1})$

**Tabella di verità**

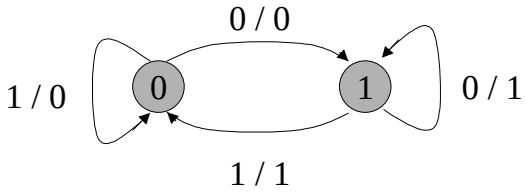
| | | ϖ | σ |
|---|---|---|---|
| **v** | **x** | **z** | **Y** |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Questa tabella rappresenta un automa che ha due stati interni (prima colonna: y=0 o y=1), e due possibili ingressi (seconda colonna: x=0 o x=1). Le ultime due colonne della tabella specificano rispettivamente la funzione di uscita (terza colonna) e la funzione di transizione dello stato interno (quarta colonna), al variare delle possibili configurazioni dell'ingresso e dello stato corrente. Come si vede, tutte le possibilità sono coperte dalla tabella, che quindi specifica completamente il funzionamento dell'automa.

Vediamo un esempio di funzionamento dell'automa specificato dalla tabella di sopra. Supponiamo che l'automa si trovi, ad un certo istante di tempo t, nello stato interno 1 (y(t)=1), e che in ingresso si presenti 0 (x(t)=0). Questi due valori individuano la seconda riga della tabella, determinano quindi un valore di output 1 e come stato interno successivo 1. Quindi y(t+1) = 1. Se ora in ingresso abbiamo 1 (x(t+1)=1), viene individuata la quarta ed ultima riga della tabella che determinerà un valore di uscita di 1 e come nuovo stato interno il valore 0 (y(t+2)=0).
L'automa descritto dalla tabella di verità può essere rappresentato graficamente:

- ad ogni stato interno corrisponderà un elemento grafico (ad esempio un cerchio);
- ad ogni riga della tabella di verità corrisponderà una freccia che parte dal cerchio associato allo stato individuato dalla prima colonna ed arriva al cerchio associato allo stato individuato dalla quarta colonna; tale freccia sarà annotata con il valore di ingresso (prima annotazione) che compare nella seconda colonna seguito dal valore di output che compare nella terza colonna (seconda annotazione).

La rappresentazione grafica dell'automa descritto sopra è la seguente:

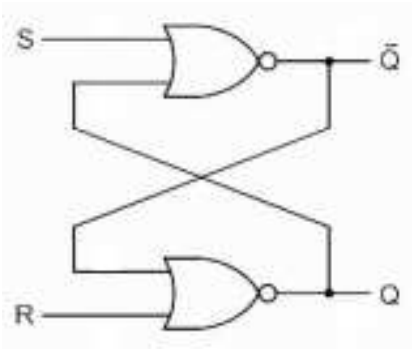



Il funzionamento dall'automa può essere "tracciato" graficamente nel seguente modo:

- lo stato corrente è individuato all'interno della rappresentazione grafica;
- fra le frecce uscenti dallo stato corrente si seleziona quella che ha come prima annotazione il valore di ingresso corrente (si noti che per definizione ne esiste una sola) ;
- viene generato in output la seconda annotazione della freccia selezionata e il nuovo stato corrente diventa lo stato di arrivo della freccia;
- si prosegue come al primo punto.

Come detto in precedenza, la possibilità di realizzare automi tramite reti sequenziali è molto importante sia per memorizzare le informazioni che per realizzare funzioni con stato. Un primo esempio di circuito sequenziale che permette la memorizzazione di un singolo bit è il Latch SR:

**Latch SR**




Permette di memorizzare un singolo bit;
Ha due stati stabili (bit a 0, e bit a 1);

**Ingressi**: S e R

**Stato**: Q e $\bar{Q}$

**Uscite**: Q e $\bar{Q}$

Da un punto di vista logico, la rete implementa le seguenti funzioni di transizione dello stato e di uscita:

$S(t_{i+1}) = \boldsymbol{\sigma}(X(t_i), S(t_i)) = \boldsymbol{\sigma}(St_i,Rt_i,Qt_i,Qt_i)$

$= \{Qt_{i+1} = nor(St_i,Qt_i), Qt_{i+1}= nor(Qt_i,Rt_i)\}$ (notare che lo stato è costituito da una coppia di valori)

$Z(t_{i+1}) = \boldsymbol{\varpi}(X(t_i), S(t_i)) = \boldsymbol{\varpi}(St_i,Rt_i,Qt_i,Qt_i) = \boldsymbol{\sigma}(St_i,Rt_i,Qt_i,Qt_i)$

Vediamo di seguito la tabella di verità che corrisponde a questo circuito:

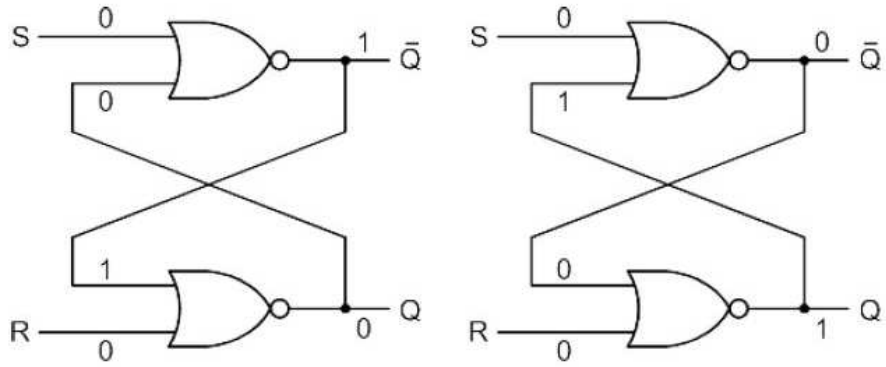
| input t | | stato t | | stato t+1 | | |
|---|---|---|---|---|---|---|
| S | R | $\bar{Q}$ | Q | $\bar{Q}$ | Q | |
| 0 | 0 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 0 | 1 | 0 | 1 | ← stato stabile |
| 0 | 0 | 1 | 0 | 1 | 0 | ← stato stabile |
| 0 | 0 | 1 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | 1 | 0 | ← stato stabile |
| 0 | 1 | 1 | 1 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 0 | 1 | ← stato stabile |
| 1 | 0 | 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 0 | 0 | ← stato stabile |
| 1 | 1 | 0 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | 0 | 0 | |
| 1 | 1 | 1 | 1 | 0 | 0 | |

Esistono 3 "stati stabili" per alcune configurazioni in ingresso:

00 01 01 (stato stabile "1")
00 10 10 (stato stabile "0")
01 10 10
10 01 01
11 00 00 (stato stabile da evitare)

A noi interessa interpretrare Q come il complemento di Q, cioè Q = not(Q)
Quindi gli stati 00 e 11 li consideriamo non coerenti (output non coerente).

Di seguito esplicitiamo i due stati che rappresentano rispettivamente la memorizzazione del valore 0 e del valore 1:



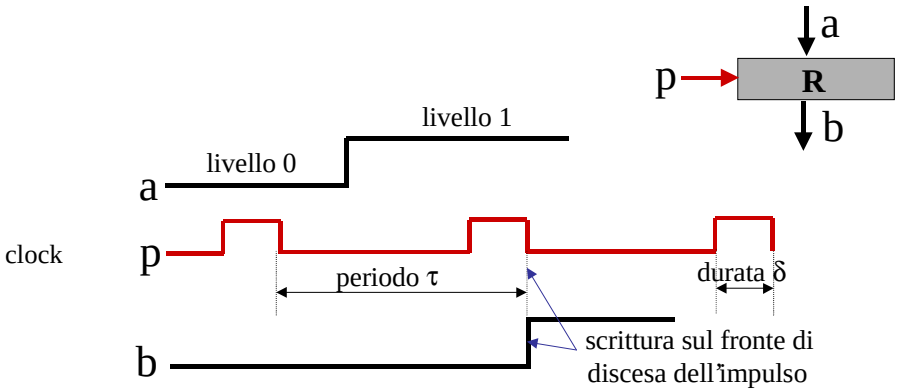

Si noti che S (setting) imposta lo stato 1, mentre R (reset) imposta lo stato 0.

Nel modello **ideale** le funzioni **σ** ed **ϖ** hanno un tempo di stabilizzazione **nullo**. Nel modello **reale** occorre un ritardo **non nullo** per la stabilizzazione delle uscite, a partire da quando gli ingressi sono stabili. Gli ingressi di tipo $\{y\} = S(t_i)$ variano, in principio, in tempi **diversi** rispetto agli ingressi di tipo $\{x\} = X(t_i)$.
Occorre dunque ripristinare l'ipotesi che le variabili di ingresso alla rete varino tutte simultaneamente, ad intervalli temporali discreti $\Delta$.. Ciò si ottiene facendo ricorso a reti sequenziali impulsate

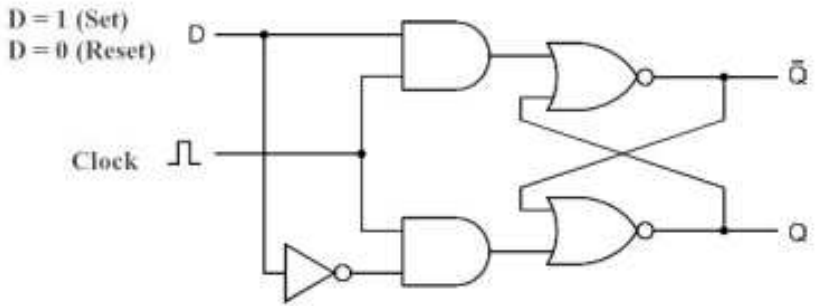
Una rete sequenziale impulsata **R** con ingresso ed uscita a livelli, e che non trasforma il segnale di ingresso, può essere descritta in generale nel seguente modo:

**R** ::= **when** p **do** b := a

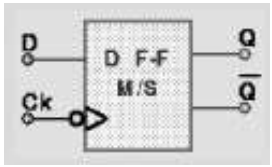
Dove facciamo riferimento alla figura sottostante per comprendere la semantica della regola descritta sopra. In sostanza, la rete copia il valore sulla linea di ingresso a sulla linea di uscita b, quando il segnale di clock si trova sul fronte di discesa. E' anche possibile definire una rete sequenziale simile, dove la "copiatura" in output avviene sul fronte di salita invece che sul fronte di discesa.

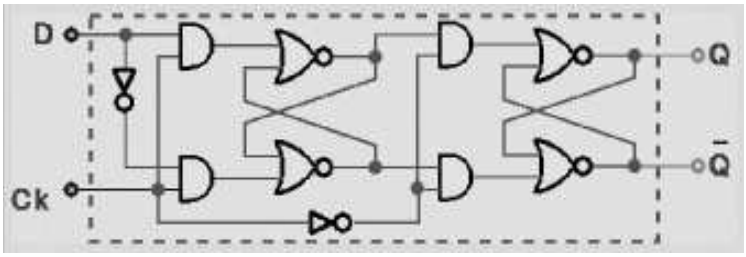



Il vantaggio di questo tipo di dispositivo è che l'output del dispositivo cambia (copiando il segnale in ingresso) solo quando il segnale di clock è sul fronte di discesa, che in teoria è il momento in cui il segnale in ingresso dovrebbe essersi stabilizzato. In particolare, il periodo di clock è scelto in modo da garantire tale stabilità. Si noti che, ovviamente, in generale il segnale di ingresso sarà trasformato tramite una porzione di rete combinatoria e che quindi l'uscita del circuito sarà diversa dal segnale di ingresso.
Questo concetto di sincronizzazione, unito a quanto visto precedentemente per quanto riguarda la memorizzazione di un singolo bit, porta alla definizione di un circuito detto Latch D sincronizzato, che oltre ad essere sincronizzato, ha il pregio di evitare che S e R siano settati ad uno stesso valore:



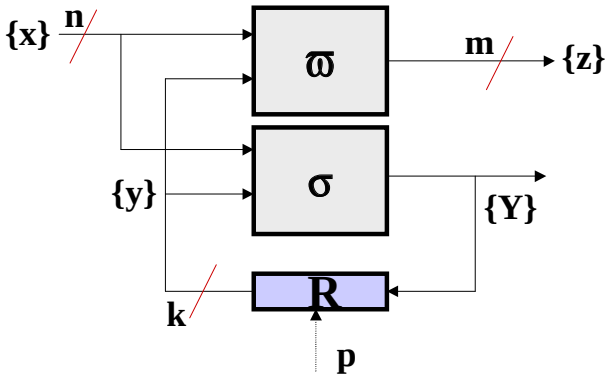

Si noti che in questo circuito c'è bisogno di un solo segnale D di set/reset equivalente al segnale S e che il segnale R è praticamente ottenuto negando (vedi porta NOT in basso) il segnale D.
Una versione più sofisticata di tale circuito, che evita che l'output sia sensibile al livello delle linee di ingresso, è costituito dal circuito Master/Slave Edge-triggered. In tale circuito la variazione dell'output si ha quando cambia la variabile di controllo (edge-triggered) e non direttamente dal valore (level-triggered), come succedeva nel circuito precedente:



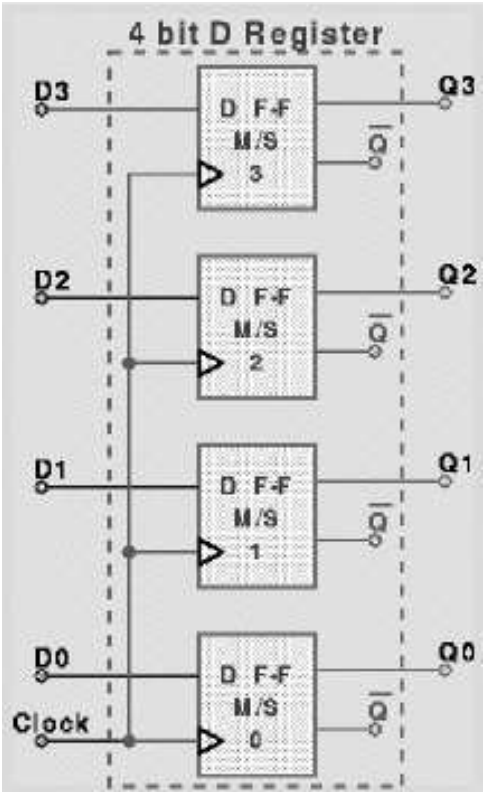

In questo circuito (in cui riportiamo in alto a destra una icona che utilizzeremo in futuro per riferirlo), quando l'ingresso Ck (clock) assume stabilmente il valore 0, il primo flip-flop a sinistra, denominato Master, risulta essere bloccato sulla memorizzazione del valore precedente, mentre il secondo flip-flop a destra, denominato Slave, riproduce tale valore in uscita. Ciò è ottenuto grazie all'inversione del segnale di controllo in ingresso alle porte AND dei flip-flop. Quando il segnale Ck commuta al valore 1, lo Slave rimane bloccato al valore assunto, mentre il Master diviene "sensibile" al segnale di ingresso D, cioè assumerà il valore di D, ma senza che ciò abbia conseguenze immediate in uscita al circuito, in quanto lo Slave è bloccato. Solo nel momento in cui si ha la commutazione di Ck da 1 a 0, il valore corrente dell'ingresso D verrà memorizzato sul Master e riprodotto in uscita dallo Slave (si noti che fino a quando Ck assume valore 1, il master "segue" il valore di D, che variando, determina una uguale variazione del Master.)
Si può quindi constatare come il circuito cambi i propri segnali non in dipendenza dei livelli degli ingressi, ma dalla transizione del segnale di controllo dal valore 1 al valore 0. E' possibile definire un circuito equivalente, dove la memorizzazione avviene quando la variabile di controllo Ck passa dal livello 0 al livello 1. Nella icona che rappresenta il circuito questa caratteristica (edge-triggered) è rappresentata dal triangolino che segue la linea di controllo Ck. Il pallino (di negazione) anteposto al triangolino sta' invece ad indicare che la memorizzazione avviene al momento della transizione del segnale di controllo dal livello 1 al livello 0. Se non ci fosse stato il pallino, allora significava che il circuito indicato memorizzava l'informazione al momento della transizione del segnale di controllo dal livello 0 al livello 1.

L'importanza della sincronia in questo tipi di circuiti risiede principalmente nel cercare di gestire al meglio il problema delle alee di commutazione. Infatti, in un circuito non sincronizzato (asincrono) si possono creare problemi di stabilizzazione del segnale, o meglio fenomeni transitori, detti alee di communtazione, che possono determinare un comportamento non atteso della rete sequenziale. In particolare, una alea di commutazione è determinata dal ritardo di propagazione della variazione dei segnali all'interno del circuito dovuto al fatto che nell'attraversare una porta logica, il segnale subisce un certo ritardo. Poiché uno stesso segnale può percorrere cammini con lunghezze diverse nel fluire da uno o più ingressi di un circuito alla sua uscita, per essere sicuri che in un certo istante di tempo il segnale che si legge in output sia quello corretto, bisogna sincerarsi che il segnale cambiato in ingresso, abbia avuto tutto il tempo necessario per fluire attraverso tutti i percorsi interni del circuito e che quindi si sia stabilizzato. Per un esempio di alea di commutazione si guardino i lucidi delle lezioni. Ovviamente, in un circuito combinatorio tali alee di commutazione possono presentarsi sono in numero finito in quanto non esistono cicli nel circuito. In una rete sequenziale, pero il segnale si instrada in cicli e quindi può accadere che le alee non terminino mai. Per questo motivo si provvede a evitare che ciò accada sincronizzando i circuiti, cioè permettendo al segnale di fluire attraverso i cicli del circuito sono negli istanti di impulso del clock. Ovviamente, il periodo del clock viene deciso in modo da permettere la stabilizzazione di tutte le parti combinatorie delle reti sequenziali. Ciò è fatto semplicemente andando a calcolare quale è il percorso più lungo che il segnale deve percorrere all'interno delle componenti combinatorie e facendo in modo che il periodo di clock permetta al segnale di percorrere completamente tale (o tali) percorsi più lunghi.
Queste considerazioni portano al seguente schema generale di rete sequenziale sincrona:
Rete sequenziale sincrona '*Level input Level output Clocked*' (**LLC**), modello di Mealy; {**x**} e {**y**} sono a **livelli**; **p** (clock) è **impulsivo**



Si noti come il segnale di stato fluisca indietro in ingresso alle funzioni di transizione dello stato e di uscita solo quando arriva il segnale di clock p.

Vediamo ora alcuni esempi di reti sequenziali di interesse. Ad esempio, mostriamo come si possano combinare molto semplicemente dei circuiti di tipo Master/Slave F-F D in modo da ottenere un registro (a 4 bit):



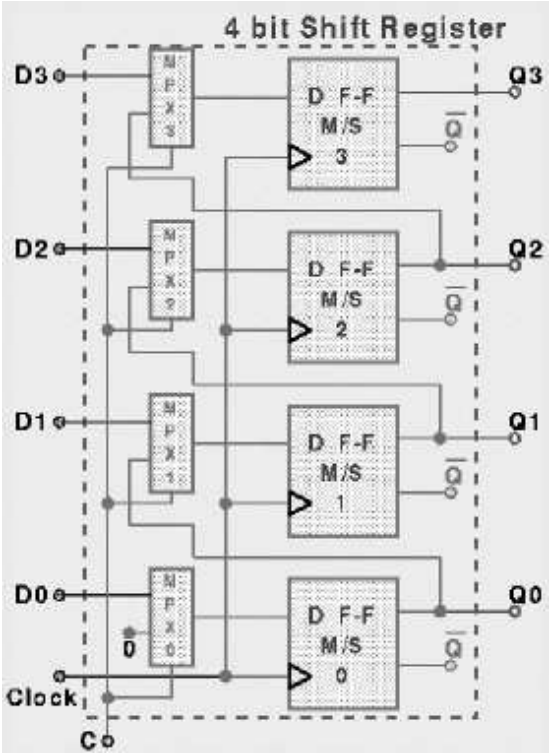

Questo è il circuito sincrono più semplice che realizza un registro. Brevemente, ecco descritte di seguito le sue modalità operative:

- memorizzazione (store): dati presentati in ingresso e clock da 0 a 1 (l'uscita riproduce l'ingresso);
- mantenimento (hold): clock da 1 a 0 (poi costante); l'uscita rimane invariata indipendentemente dal valore degli ingressi.

Una variante del registro presentato sopra è costituita dal seguente registro a scorrimento, che a seconda del valore della variabile di controllo C permette:

- (C=0) scorrimento verso l'alto dei bit, e valore "0" inserito sul bit meno significativo;
- (C=1) store degli ingressi.

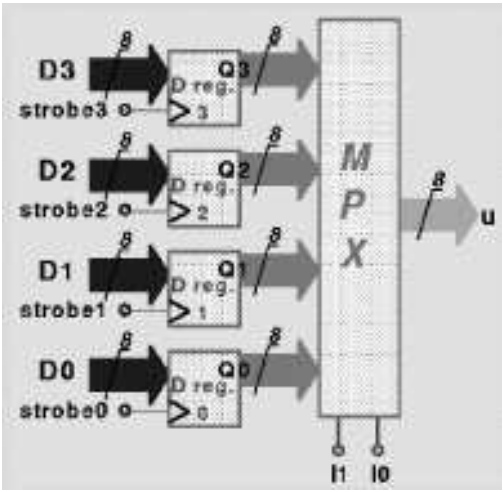
Lo scorrimento si ottiene tramite l'opportuno posizionamento di alcuni circuiti multiplexer (MPX) controllati simultaneamente dalla stessa linea di controllo C.
Tale registro a scorrimento permette la conversione "parallelo/seriale": si memorizzano i dati in parallelo e si "trasmettono" uno alla volta su $Q_3$.




Una classe di circuiti sequenziali molto importanti è costituita dalle Memorie (RAM) che permettono la memorizzazione ed un accesso veloce delle informazioni necessarie durante la elaborazione. Queste possono essere viste come un insieme di celle identificate da un numero (**indirizzo**); ogni cella è in grado di codificare n bit (es. n=8 [byte], n=16, …) ed ogni cella deve preservare nel tempo il suo contenuto (**memoria**) a meno di operazioni di scrittura. Inoltre, una cella deve poter essere letta fornendo l'indirizzo della stessa (**lettura**). La lettura di una cella non deve cancellare l'informazione letta. Infine, una cella deve poter essere scritta fornendo sia l'indirizzo della stessa che l'informazione da memorizzare (compatibile con il valore di n) (**scrittura**).

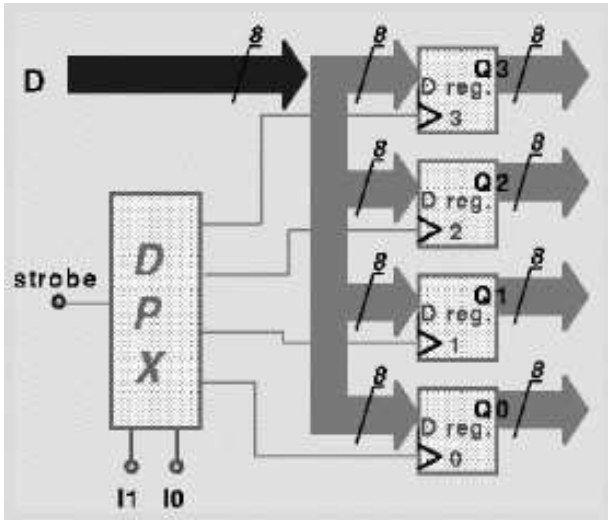
Un esempio di Memorie RAM è dato dalle RAM statiche: queste sono implementate tramite registri di tipo D. Tali registri, tuttavia, devono essere utilizzati in modo opportuno. Ad esempio è necessario un meccanismo per consentire l'accesso in scrittura e lettura di una singola cella di memoria alla volta (specificata dall'indirizo in ingresso alla memoria). Supponendo che gli indirizzi della memoria siano specificati come numeri interi progressivi rappresentati in base 2 (rappresentazione binaria; es. *ind* da 0 a 7, richiede 3 bit: *ind*=5 è rappresentato da 101, *ind*=6 è rappresentato da 110, ... ) vediamo come realizzare separatamente il meccanismo per implementare la **lettura** e quello per implementare la **scrittura.**

La lettura viene realizzata molto semplicemente tramite un circuito multiplexer (MPX), comandato dai bit che esprimono l'indirizzo della cella a cui si vuole accedere (vedi figura sotto).

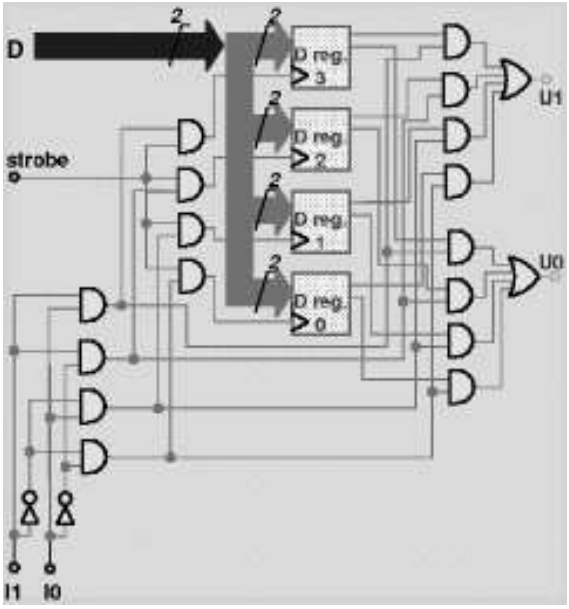


Nell'esempio di sopra, 4 celle di 8 bit sono realizzate tramite registri sincroni di tipo D. Per indirizzare le 4 celle si ha bisogno di 2 bit di indirizzo ($I_0$, $I_1$). Sempre con riferimento alla figura di sopra, strobe è il segnale di clock, mentre MPX è un multiplexer controllato dai bit di indirizzo, che provvedono a far fluire in uscita solo l'informazione contenuta nel registro selezionato dalla configurazione corrente dei bit di indirizzo.

Per quanto riguarda la scrittura, bisogna far riferimento ad un meccanismo inverso rispetto a quello utilizzato nella lettura, cioè abbiamo bisogno di un demultiplexer (DPX) comandato dai bit di indirizzo per selezionare quale dei registri dovrà memorizzare il dato in ingresso D che è distribuito in parallelo a tutti i registri. Tuttavia, solo il registro selezionato dal demultiplexer sarà arrivo e quindi provvederà a memorizzare il dato di ingresso D.




E' possibile combinare insieme i meccanismi di lettura e scrittura in modo da ottenere una RAM ottimizzata che combina il multiplexer con il demultiplexer. In questo caso la lettura è permessa sempre, mentre la scrittura avviene come già visto in precedenza.

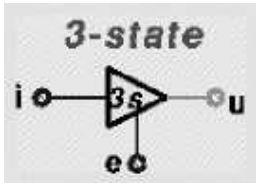


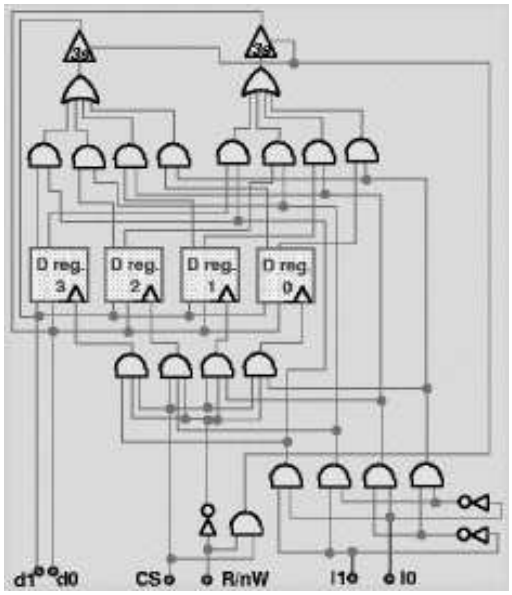

Un aspetto importante delle Memorie RAM è che queste possono essere costruite utilizzando più di un modulo di memoria dando luogo ad un sistema modulare flessibile e poco costoso. Per permettere, tuttavia, la composizione di più moduli di memoria occorre utilizzare un dispositivo particolare detto a tre stati che permette di collegare o scollegare un circuito ad una linea usando un segnale di controllo elettronico. Tale dispositivo pertanto permette di utilizzare la stessa linea dati condividendola fra più moduli di memoria.

Il dispositivo a 3 stati su menzionato è mostrato di seguito a sinistra. La sua funzionalità è descritta da seguente tabella:

| i | e | u |
|---|---|---|
| 0 | 0 | non connesso |
| 0 | 1 | 0 |
| 1 | 0 | non connesso |
| 1 | 1 | 1 |

Due o piu uscite a tre stati possono essere connesse sullo stesso filo a patto di poter garantire che al massimo uno dei dispositivi sia comandato col valore 1 per e. Questa condizione permette di evitare conflitti fra dispositivi che vogliono produrre valori possibilmente diversi sullo stesso filo. Come si vede dalla figura alla destra, che rappresenta un modulo componibile di memoria (con 4 celle di 2 bit), il dispositivo a 3 stati viene utilizzato per connettere le linee dati ($d_0$ e $d_1$) alla uscita del circuito stesso. Quindi le stesse linee dati sono utilizzate sia per i dati in ingresso che in uscita.




Inoltre, per garantire il giusto funzionamento del modulo, le linee di controllo del modulo sono attivate secondo un protocollo ben definito. In particolare CS = 0 indica che il modulo non deve essere attivato, ed in questo caso non è rilevante il valore assunto da R/nW. Se invece CS = 1, questo significa che il modulo deve essere attivato per una operazione di lettura o scrittura. La particolare operazione da eseguire in questo caso è indicata dal valore assunto dalla variabile R/nW: se R/nW = 1 si deve eseguire una lettura all'indirizzo specificato da $I_0$ e $I_1$, mentre se R/nW = 0, allora si deve operare una scrittura all'indirizzo specificato da $I_0$ e $I_1$ dei dati di ingresso $d_0$ e $d_1$.
Si noti che in caso di lettura, le linee $d_0$ e $d_1$ sono utilizzate per portare all'esterno del circuito i valori letti. Comunque, il circuito collegherà il suo output a tali linee solo in presenza della combinazione CS = 1 e R/nW = 1, mentre le stesse linee vengono usate in ingresso in caso di scrittura: CS = 1 e R/nW = 0.
Si noti che la possibilità di non attivare il modulo tramite la linea CS è particolarmente utile in un sistema modulare, dove solo il modulo (o i moduli) coinvolti nelle attività di calcolo potranno essere attivati.
Di seguito si mostra come due dei moduli appena descritti possono essere collegati in modo da funzionare insieme. Si può notare che i bit più significativi dell'indirizzo (in questo caso $I_2$ e $I_3$) sono confrontati con valori di riferimento assegnati ai moduli, in modo da determinare quale particolare modulo deve essere attivato (tramite la linea CS). I bit meno significativi dell'indirizzo vengono invece posti in ingresso a tutti i moduli in modo da poter essere usati direttamente dal modulo che risulterà effettivamente attivato (dai bit più significativi dell'indirizzo). Questa soluzione è resa fattibile grazie hai dispositivi a 3-stati che permettono di collegare la stessa linea dati a tutti i moduli.

**CS = 0  chip non attivo  CS = 1 chip attivo  R/nW=1 lettura  R/nW=0 scrittura**