Capitolo 6

RETI SEQUENZIALI MODULARI

6.1 - Registri

Con il nome di **registri** si indicano circuiti sequenziali, per lo più sincronizzati, la cui funzione è quella di memorizzare informazione. In generale un registro è costituito da un insieme di *n* elementi di memoria binari, riferiti con gli interi tra 0 ed *n*-1, ed è quindi in grado di mantenere l'informazione che può essere rappresentata con *n* bit. Gli *n* elementi di memoria costituiscono la **parola** del registro ed *n* è la sua **lunghezza**. In un registro, oltre a segnali di ingresso (**dati di ingresso**) e a segnali di uscita **(dati di uscita**), esistono un certo numero di segnali di controllo e di temporizzazione che vengono inviati a tutti gli elementi contemporaneamente; il numero e le funzioni dei segnali di controllo dipendono dal tipo di registro, come ne dipendono le operazioni che possono essere effettuate sui dati memorizzati; si possono così avere segnali di caricamento in parallelo (**load**) o in serie (**serial in**), di abilitazione (**enable** o **select**), di azzeramento del contenuto (**clear**); i segnali di temporizzazione si riducono in genere al **clock**. A seconda degli elementi costituenti, un registro potrà essere trasparente o non trasparente; la Fig. 6.1 riporta uno schema a blocchi generale:

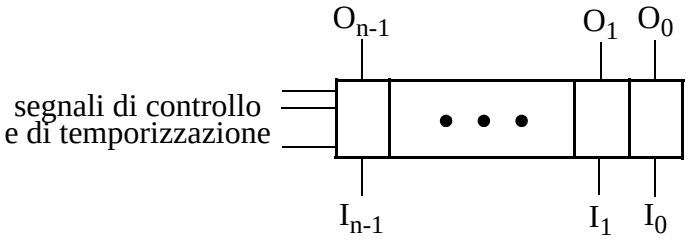


Fig. 6.1

In relazione al tipo di operazione che è possibile effettuare sui dati, si distinguono due

categorie fondamentali di registri: i registri **paralleli** ed i registri di **traslazione** o registri **di scorrimento** (shift registers).

6.1.1 - Registri paralleli

I registri di questo tipo sono caratterizzati dal fatto che tutti i bit sono sottoposti insieme alle stesse operazioni, indipendentemente dalla posizione che ciascuno occupa nel registro. Le operazioni possibili sono la **scrittura** nel registro e la **lettura** del contenuto; in una scrittura, i dati presenti sui terminali di ingresso vengono memorizzati nel registro tutti nello stesso istante, ossia in parallelo, in una lettura sono resi disponibili sui terminali di uscita, pure in parallelo. Come elementi di memoria sono in genere utilizzati latch o flip-flop D o JK di tipo sincronizzato.

Lo schema di principio è illustrato nella Fig. 6.2. In scrittura, supponiamo che il segnale a livelli *load* sia 0: in tal caso l'uscita di entrambe le porte AND sugli ingressi di ciascun flip-flop sono 0 ed i flip -flop stessi mantengono il loro stato. Se invece il segnale *load* è 1, all'ingresso *J* di ogni flip-flop è applicato il bit corrispondente del dato *I*, mentre all'ingresso *K* è applicato il complemento di tale bit: il flip-flop si trova in condizione di effettuare un'operazione di set o di reset a seconda che il bit ad esso applicato valga 1 oppure 0: in ogni caso memorizza tale bit in corrispondenza dell'impulso di clock. Per effettuare un'operazione di lettura, il segnale **oe** (output enable), applicato in parallelo sulle uscite di tutti i flip-flop, viene posto ad 1 e ciò determina l'attivazione delle "porte" rappresentate nella figura con dei quadratini le quali rendono il dato memorizzato nei flip-flop disponibile sulle uscite corrispondenti, immediatamente nel caso di dispositivi trasparenti, al successivo impulso di clock nel caso di flip-flop master-slave.

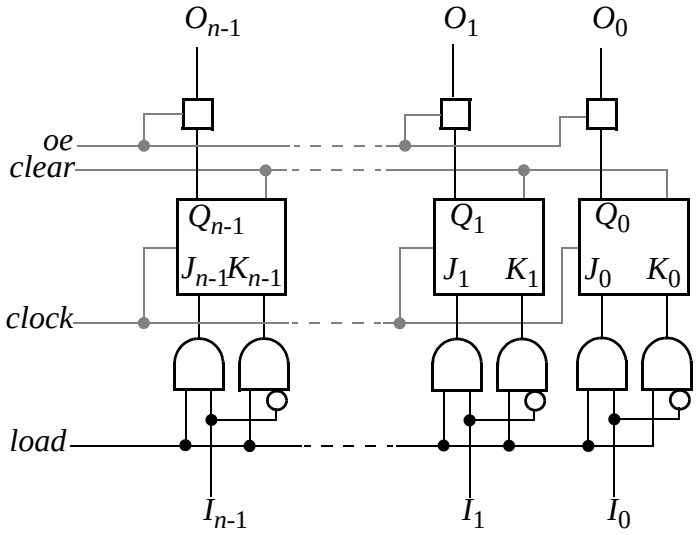


Fig. 6.2

Le porte controllate dal segnale *oe* sono dispositivi speciali detti **porte 3-state**, che non implementano alcuna funzione logica, ma consentono o impediscono il propagarsi di un

segnale dall'ingresso all'uscita, comportandosi come una specie di interruttori comandati da un segnale di controllo (Fig. 6.3); il loro uso rende un dispositivo digitale dotato di *alta impedenza* sull'uscita.

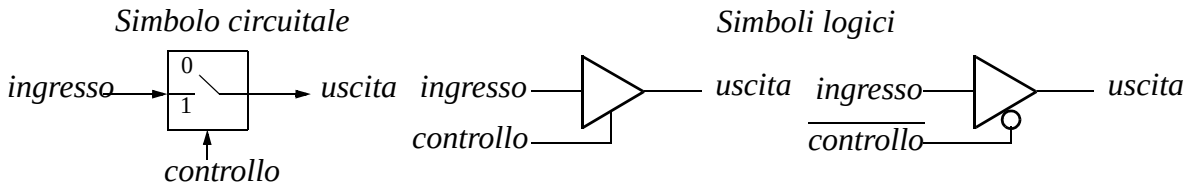


Fig. 6.3

Le porte 3-state sono molto utilizzate nei circuiti digitali, principalmente per due scopi: a) realizzare funzioni di **OR cablato**, cioè soluzioni circuitali in cui una linea di uscita è fisicamente collegata a più linee di ingresso, di una sola delle quali può di volta in volta assumere il livello di segnale: per ottenere ciò occorre che tutte, tranne quella alimentante, possano essere poste in alta impedenza. b) Realizzare linee bidirezionali, ossia linee sulle quali possano essere istradati tanto segnali di entrata quanto di uscita (Fig. 6.4):

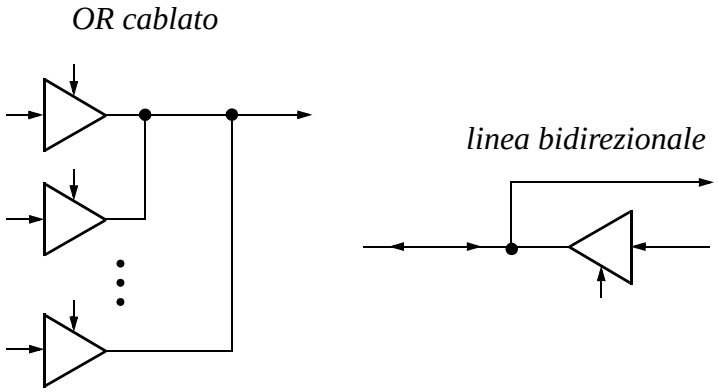


Fig. 6.4

In pratica le porte 3-state possono anche non essere elementi circuitali fisicamente distinti, ma è molto più comune che la funzione di alta impedenza sull'uscita sia realizzata direttamente sul componente che ne deve essere dotato (talvolta invece di alta impedenza si parla di *open-collector*, in riferimento ad una particolare tecnica costruttiva dei dispositivi digitali: elettronicamente c'è differenza tra alta impedenza e oper collector, ma dal punto di vista logico si possono considerare equivalenti).

Infine il segnale **clear**, asincrono rispetto al clock come il segnale *load*, determina l'azzeramento del contenuto del registro, indipendentemente dal dato presente in ingresso.

Le realizzazioni pratiche possono differenziarsi da questo schema per il numero di segnali di controllo e per il modo come essi sono utilizzati; per esempio il segnale *clear* può mancare; i segnali *load*, *clear*, *oe* possono essere attivi quando valgono 0 e così via; inoltre può essere presente un ulteriore segnale a livelli detto **select** il quale, in AND con il *load*, abilita o meno il registro: la sua funzione in una struttura multiregistro è quella di effettuare una selezione del registro o dei registri di volta in volta abilitati ad operare.

6.1.2 - **Registri di traslazione**

Questi registri, detti anche **shift registers** o **registri seriali**, sono caratterizzati dal fatto che l'ingresso del dato è **seriale**, dal bit meno significativo (registri di traslazione sinistra) o da quello più significativo (registri di traslazione destra); anche l'uscita è in genere seriale, dal bit più significativo o da quello meno significativo nei due casi, ma non mancano realizzazioni di registri di traslazione con uscite parallele.

Ad ogni impulso di clock la configurazione binaria memorizzata nel registro viene fatta traslare rispettivamente verso sinistra o verso destra di un bit, mentre il bit più (o meno) significativo viene perduto e quello meno (o più) significativo assume il valore del bit presente in ingresso. Se il registro è lungo *n* bit occorrono *n* impulsi di clock per sostituire completamente il dato memorizzato con un altro. Per esempio consideriamo un registro di traslazione sinistra lungo quattro bit, contenente inizialmente 1011 e sia 0110 il dato che deve esservi caricato; l'operazione è descritta nella seguente figura:

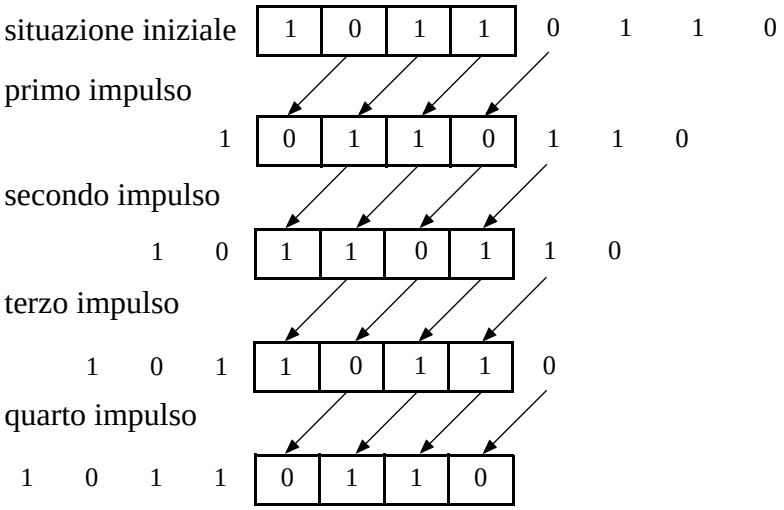


Fig. 6.5

Normalmente uno stesso registro può effettuare sia traslazioni destre sia traslazioni sinistre, a seconda del valore di un segnale di controllo **right/left** (per esempio R/L = 0 ≡ traslazione destra, R/L = 1 ≡ traslazione sinistra), mentre un segnale **serial-in**, logicamente equivalente al segnale *load* dei registri paralleli, abilita l'operazione.

Questo modo di funzionamento è reso possibile connettendo l'uscita di un flip-flop all'entrata del flip-flop immediatamente a destra o a sinistra rispettivamente per lo shift destro o sinistro, mentre l'ingresso del flip-flop più significativo è l'ingresso esterno e l'uscita di quello meno significativo l'uscita seriale esterna o viceversa a seconda che si tratti di uno shift destro o sinistro. La Fig. 6.6 mostra uno schema di principio:

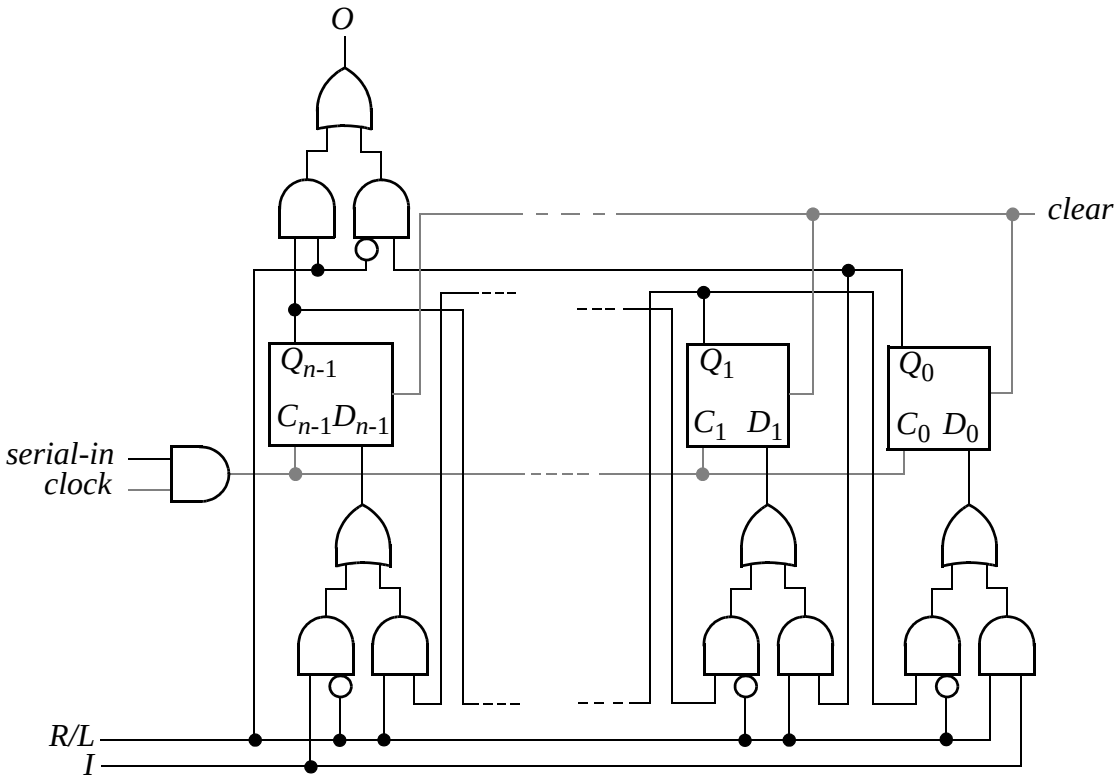


Fig. 6.6

Come si può vedere, il segnale *serial-in* inibisce l'arrivo del clock quando è 0, impedendo qualunque operazione del registro. Quando invece *serial-in* è 1, il segnale *R/L*, che agisce come segnale di controllo del multiplexer formato dalle *n* terne di porte AND e OR, sceglie se il dato *I* deve essere caricato dal bit meno significativo e corrispondentemente il dato di uscita *O* prelevato dal bit più significativo o viceversa.

Nelle realizzazioni pratiche le funzioni di registro seriale e di registro parallelo si trovano spesso combinate insieme nello stesso componente, il quale può costituire un registro con ingresso seriale e/o parallelo ed uscita pure seriale e/o parallela, il quale viene indicato con il nome di **registro universale**. La Fig. 6.7 illustra un registro universale a quattro bit.

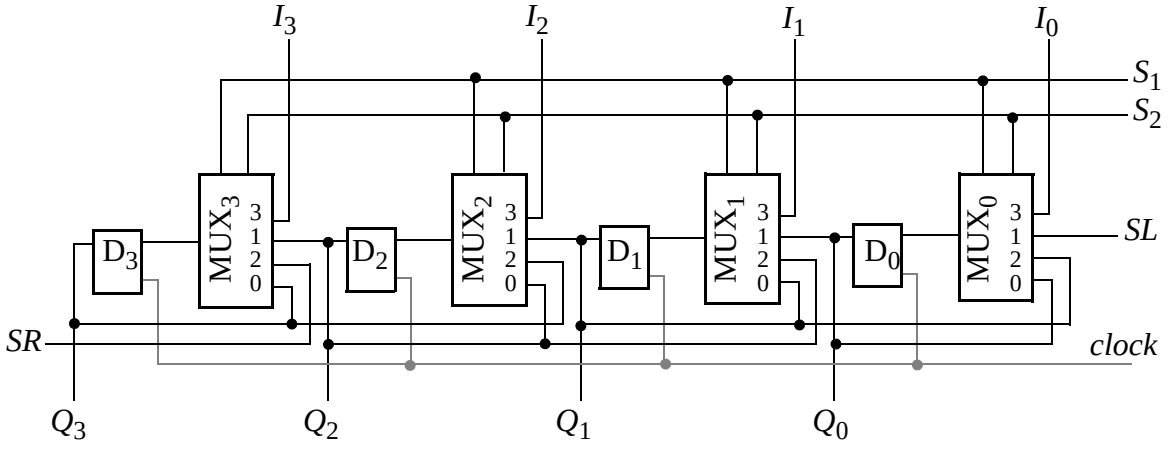


Fig. 6.7

Gli elementi di registro sono latch o flip-flop D, contrassegnati con D_0, D_1, D_2, D_3, ed i loro ingressi sono ottenuti dall'uscita di altrettanti multiplexer, controllati dai segnali S_1 ed S_2, i quali operano secondo la seguente tabella:

S_1, S_2	
0 0	nessuna operazione
0 1	shift left: $D_0 \leftarrow$ SL
1 0	shift right: $D_3 \leftarrow$ SR
1 1	caricamento parallelo

6.2 - Memorie di lettura-scrittura (RAM)

Una RAM (Random Access Memory) è un dispositivo costituito da *N* elementi binari di memoria, organizzati in *M* gruppi o **locazioni** di *n* elementi ciascuno, in modo che sia $N = M \times n$. Ogni locazione è in grado di memorizzare l'informazione che può essere codificata con *n* bit, ossia con una **parola** di *n* bit ed è individuata da un indirizzo numerico nell'intervallo [0, *M*-1]. L'accesso ad una locazione avviene in modo indipendente dalla posizione della locazione stessa e in tempo costante: da qui deriva l'appellativo di memoria ad accesso casuale, che a rigore competerebbe anche alle ROM, pur non essendo per esse utilizzato.

Le quantità *M* ed *n* sono dette rispettivamente **capacità** della memoria e **lunghezza della parola**; la capacità è misurata correntemente in migliaia o milioni di parole, per lo più aventi una lunghezza di 8 bit (byte), utilizzando suffissi moltiplicativi quali K (chilo), $1K = 2^{10} \approx 10^3$, M (mega), $1\ M = 2^{20} \approx 10^6$, G (giga), $1G = 2^{30} \approx 10^9$. La lunghezza di parola è espressa per lo più in multpli di byte. Per esempio con il termine RAM 128K×8 si fa riferimento ad una memoria RAM di $128 \times 2^{10} = 131072$ parole da 8 bit.

Le operazioni che possono essere compiute su una RAM sono la **lettura**, ossia la generazione di una copia del contenuto di una locazione e la **scrittura**, ovvero la modifica distruttiva del contenuto di una locazione. Per questo la RAM è provvista di un opportuno numero di linee di ingresso, di uscita e di linee bidirezionali, alcune delle quali servono per segnali di controllo, altre per i dati. La struttura interna è formata da una matrice $M \times n$ di elementi di memoria, di solito latch asincroni, e da altri circuiti combinatori di controllo e di decodifica (Fig. 6.8).

Le linee denominate *address*, in numero pari a $\lceil \log_2 M \rceil$, definiscono gli indirizzi delle locazioni di memoria; esse sono decodificate internamente in modo da generare, per ogni indirizzo, un segnale di selezione di tutti i latch della locazione avente quell'indirizzo. Il segnale *S* (select) ha il compito di selezionare l'intero dispositivo, in modo da renderlo operativo; la scelta del tipo di operazione è invece effettuata con il segnale $\overline{WE}$ (write enable);

per esempio $\overline{WE} = 1$ può significare un'operazione di scrittura, $\overline{WE} = 0$ una lettura o viceversa.

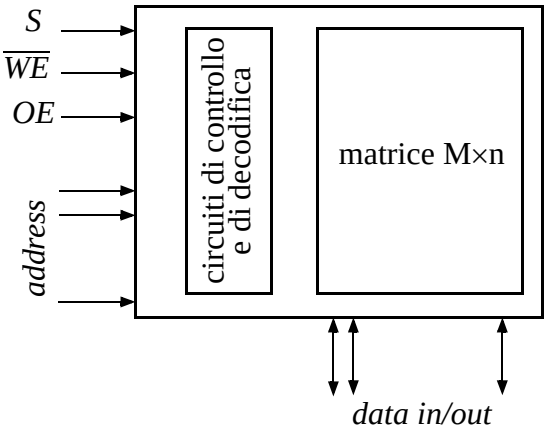


Fig. 6.8

Uno schema di principio per l'organizzazione interna di una RAM è mostrato in Fig. 6.9, limitatamente ai bit della locazione *i*-esima:

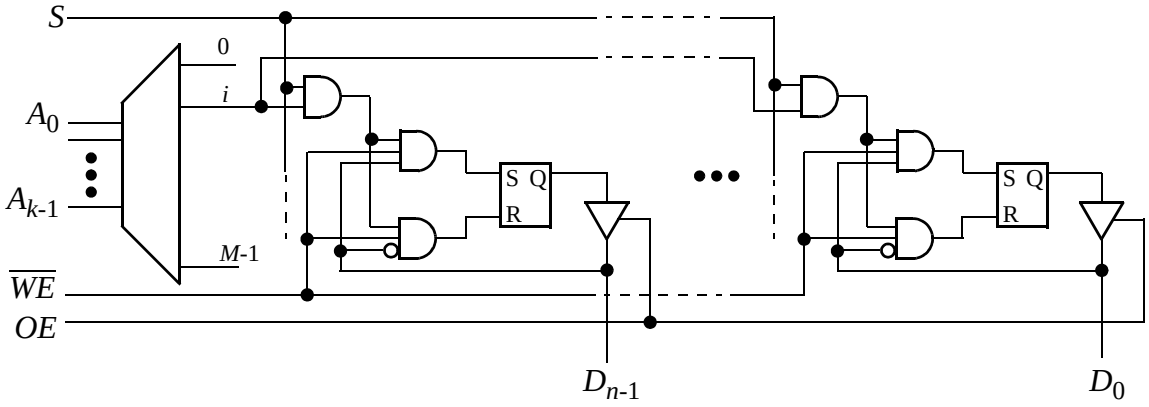


Fig. 6.9

Il segnale *S* abilita il dispositivo, consentendo o inibendo l'arrivo del segnale di selezione della riga *i*-esima, generato dal decodificatore degli indirizzi: il segnale $\overline{WE}$ definisce il tipo di operazione mentre il segnale *OE* controlla lo stato di alta impedenza delle uscite dei latch. Durante il ciclo di scrittura $\overline{WE}$ viene posto ad 1 ed *OE* a 0, per cui le uscite dei latch sono in alta impedenza ed i segnali presenti sulle linee dei dati $D_0, \ldots, D_{n-1}$ possono essere memorizzati nei rispettivi latch. Invece durante il ciclo di lettura $\overline{WE}$ vale 0 in modo che gli ingressi dei latch sono inibiti; *OE* è posto ad 1 e consente che le uscite dei latch siano collegate alle linee dei dati. In assenza di selezione della locazione *i*-esima, gli ingressi dei latch sono tutti a 0 e pertanto la configurazione memorizzata nella locazione si mantiene.

Si definisce **tempo di accesso** l'intervallo che intercorre tra l'istante in cui l'indirizzo viene presentato sulle line *A* e l'istante in cui il dato è disponibile sulle linee di uscita; esso è in genere lo stesso anche per il ciclo di scrittura, per cui non si fa distinzione tra lettura e lettura. Valori tipici del tempo di accesso sono intorno a 100÷200 ns per memorie di capacità variabile da 64

Kbyte a 1 Mbyte. Il **tempo di ciclo** invece è il tempo che occorre attendere dall'inizio di un'operazione prima di potere dare inizio ad un'altra operazione; in genere coincide con il tempo di accesso, ma può essere anche superiore ad esso, in quanto a definirlo concorrono ritardi aggiuntivi, non strettamente legati all'operazione effettuata.

Le memorie RAM basate su latch sono dette **statiche**, dal momento che l'informazione una volta memorizzata non ha più bisogno di essere manipolata, ma rimane inalterata finchè il dispositivo è alimentato.

Sono molto comuni anche RAM **dinamiche**, nelle quali per conservare l'informazione si sfrutta una carica elettrica immagazzinata nella capacità di canale di un transistore MOS. Le memorie dinamiche sono preferibili in quanto consentono una densità, intesa come rapporto tra capacità e superficie di silicio necessaria, molto elevata ed un costo più basso delle memorie statiche; per contro hanno l'inconveniente di richiedere periodicamente un ciclo di refresh che, dovendo essere intercalato alle normali operazioni con periodicità dell'ordine del millisecondo, provoca un aumento del tempo di ciclo rispetto alle memorie statiche.

Qualunque sia il tipo di celle usate come elementi di memoria, in genere esse sono organizzate in forma di matrice per due ragioni principali:

a) una disposizione fisica (**layout**) secondo un array rettangolare facilita la stesura dei collegamenti tra le celle e la circuiteria di controllo, in quanto consente di allocarli negli spazi tra una cella e l'altra.

b) L'indirizzo di una cella, A_i, può essere partizionato in d componenti, che costituiscono un vettore a d dimensioni (A_{i1}, A_{i2}, ..., A_{id}). Ogni componente viene decodificata da un decodificatore separato, che pertanto ha dimensioni e complessità ridotte e al tempo stesso maggiore velocità. La cella di indirizzo A_i viene selezionata attivando simultaneamente tutte le d uscite dei decodificatori, ossia per **coincidenza** di d segnali di indirizzamento.

La più semplice organizzazione ad array è quella unidimensionale, illustrata in Fig. 6.10 per una memoria di M celle:

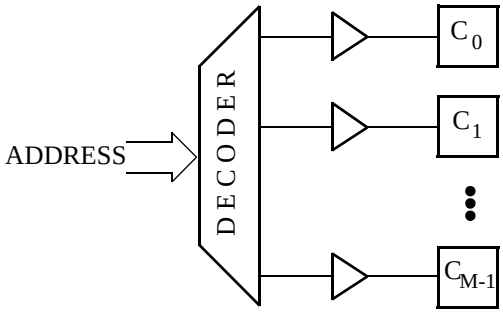


Fig. 6.10

Assai comunemente, specie quando la capacità della memoria è elevata, si usa un layout a due dimensioni, organizzando le celle in una matrice di M_x righe ed M_y colonne, in modo che $M_x \times M_y = M$. A sua volta l'indirizzo è diviso in due componenti dette **componente di riga** $a_x = \lceil \log_2 N_x \rceil$ e **componente di colonna** $a_y = \lceil \log_2 N_y \rceil$, ed una cella è selezionata per coincidenza di due segnali X ed Y ottenuti dalla decodifica delle componenti di riga e di colonna. Questa

organizzazione consente una riduzione non indifferente delle dimensioni del bus degli indirizzi e della circuiteria di accesso rispetto al caso unidimensionale. Infatti se $M_x = M_y \cong M^{1/2}$, invece di un decodificatore $\lceil \log_2 M \rceil \times M$, ed M amplificatori di pilotaggio sono necessari soltanto due decodificatori $\lceil \log_2 M \rceil \times M^{1/2}$ e $2M^{1/2}$ amplificatori. La Fig. 6.11 schematizza questo tipo di organizzazione.

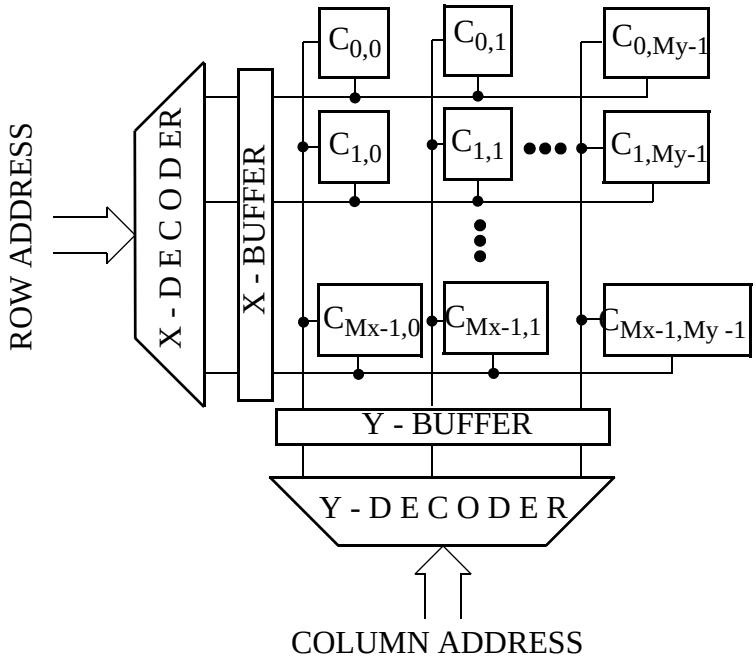


Fig. 6.11

Per esaminare la temporizzazione tipica delle memorie RAM, prendiamo in considerazione un componente commerciale assai diffuso, recante la sigla 4116. Si tratta di un chip di RAM dinamica da 16K parole da 1 bit, il cui schema a blocchi è riportato in Fig. 6.12 . Come si vede l'indirizzamento avviene per righe e per colonne tramite un unico bus indirizzi di 7 bit; pertanto sia l'indirizzo di riga sia quello di colonna devono essere memorizzati in appositi buffer, essendo presentati in tempi successivi; la caratterizzazione di un indirizzo come indirizzo di riga o di colonna avviene tramite i segnali di selezione $\overline{RAS}$ (Row Address Select) e $\overline{CAS}$ (Column Address Select) e la logica di controllo, alla quale arriva anche il segnale $R/\overline{W}$ per la specifica dell'operazione; *D* è il dato di ingresso: esso viene memorizzato in un registro (**data in**), separato dal registro MDR che è utilizzato solo come registro di uscita.

Questo dispositivo di memoria consente vari modi di operazione, dei quali esamineremo solo il modo *read*, il modo *write* ed il modo *read modify-write*. I tempi di ciclo dell'operazione *read modify-write*, che è la più lunga, vanno da 375 ns a 515 ns, a seconda delle versioni.

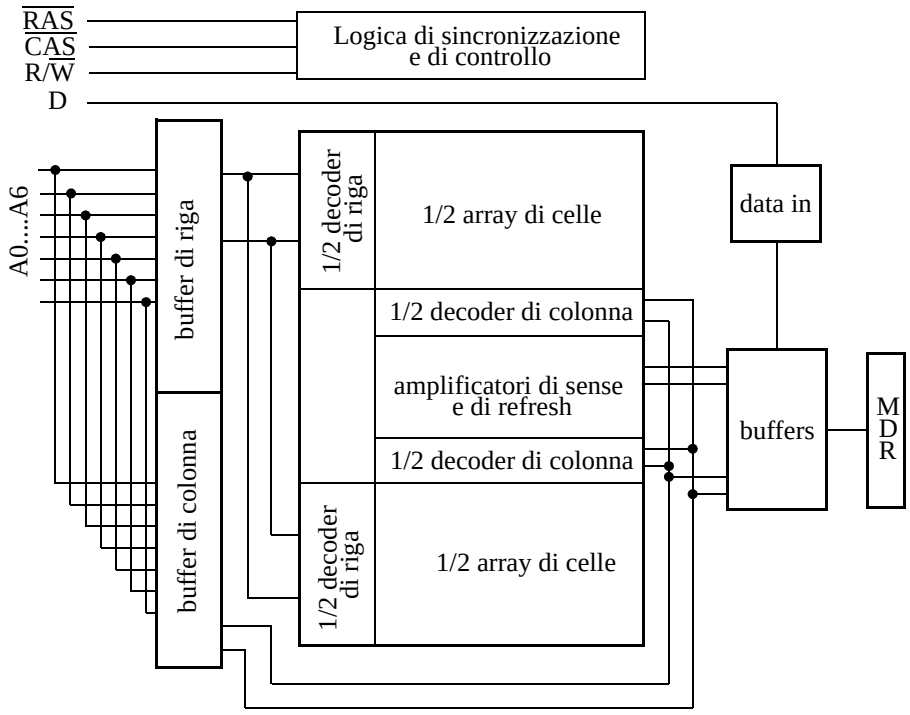


Fig. 6.12

Il diagramma temporale del ciclo di lettura è mostrato in Fig. 6.13. Quando il segnale $\overline{RAS}$ scende a zero, l'indirizzo di riga viene memorizzato nei buffer di ingresso e viene abilitato il decodificatore di riga; con un ritardo non inferiore a t_{RLCL}, entro il quale l'indirizzo di colonna deve essersi stabilizzato, anche il segnale $\overline{CAS}$ scende a zero, provocando la memorizzazione dell'indirizzo di colonna e l'attivazione del relativo decodificatore. Prima dell'azzeramento di $\overline{CAS}$, il segnale di lettura/scrittura sale ad uno e dopo un tempo di accesso $t_{a(C)}$, misurato dalla caduta di $\overline{CAS}$, il dato letto diventa valido in uscita e rimane tale per un tempo t_{PXZ} dal momento della risalita di $\overline{CAS}$; il $\overline{RAS}$ risale per ultimo e deve rimanere alto per un tempo minimo $t_{w(RH)}$. La somma dei ritardi $t_{w(RL)}$ e $t_{w(RH)}$ determina la durata del ciclo di lettura $t_{c(r)}$. Nei diagrammi le zone tratteggiate hanno il significato di valori di segnale non significativi.

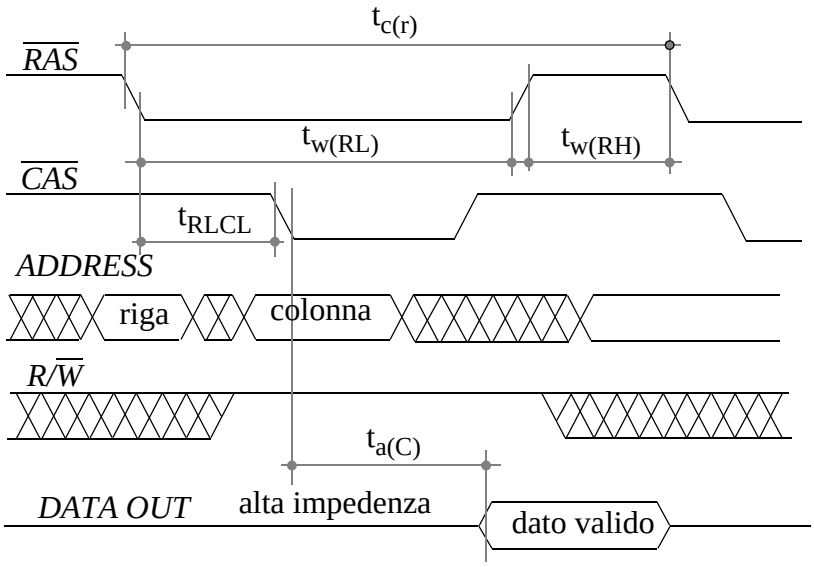


Fig. 6.13

Nell'operazione di scrittura, la temporizzazione dell'indirizzamento è analoga a quella del ciclo di lettura (prima viene memorizzato l'indirizzo di riga, poi quello di colonna); quello tra i due segnali $\overline{CAS}$ e $R/\overline{W}$ che viene fatto cadere per primo determina l'acquisizione del dato nel registro-buffer *data in*, in modalità **early write operation** o **write operation** rispettivamente; il dato non può cambiare prima di un ritardo $t_{H(DRL)}$ dalla caduta di $\overline{RAS}$. Il tempo di ciclo è ancora determinato dal periodo del $\overline{CAS}$. Il diagramma temporale è illustrato in Fig. 6.14a.

Infine l'operazione read modify-write consiste in una lettura seguita da una scrittura allo stesso indirizzo, che pertanto deve essere selezionato solo una volta, con risparmio di tempo. Le due operazioni sono eseguite su controllo del segnale $R/\overline{W}$: quando esso cade viene letto il dato con ritardo $t_{a(C)}$ dalla caduta di $\overline{CAS}$; quindi, mentre il dato rimane valido in uscita, $R/\overline{W}$ diventa alto, consentendo l'acquisizione nel registro *data in* del dato in ingresso (Fig. 6.14b).

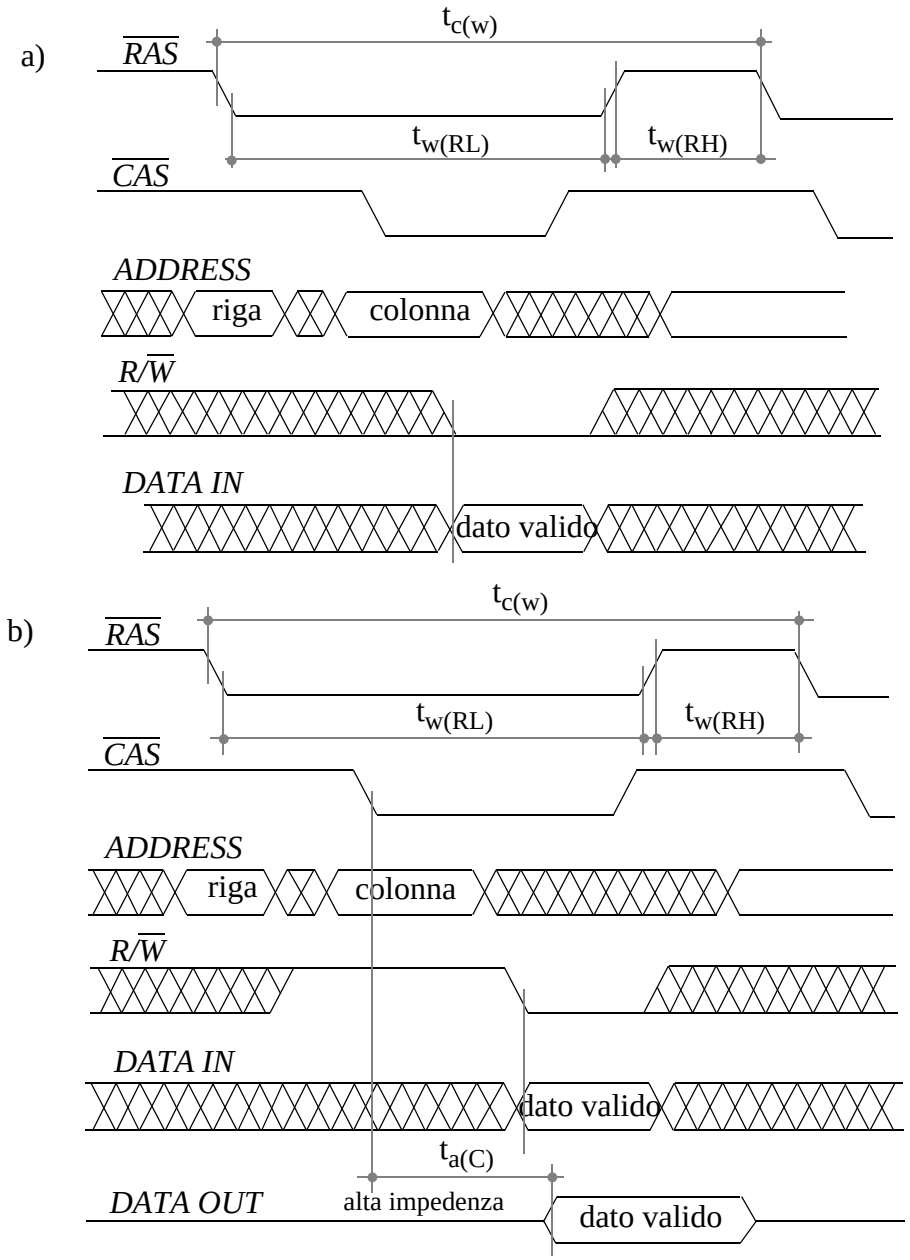


Fig. 6.14

6.3 - Contatori

Effettuare un'operazione di conteggio in un campo numerico di *n* cifre in base β significa incrementare di una unità per volta il valore del numero in esso rappresentato. Partendo da 0 e operando secondo le regole dell'aritmetica in base, dopo β^n incrementi, il numero ha un valore pari a β^n-1 ed è rappresentato da una sequenza di *n* cifre tutte uguali a quella che rappresenta

il valore numerico β-1; un ulteriore incremento di una unità provoca l'azzeramento del campo e la generazione di un riporto in posizione n+1: poichè il campo di rappresentazione ha n posizioni, il conteggio è modulo β^n, ovvero è possibile contare da 0 fino a β^n-1, per ripartire poi da 0 in modo ciclico.

Se il campo è binario, il conteggio è modulo 2^n ed un dispositivo in grado di effettuarlo è un **contatore binario** ad n stadi. Come è stato visto a suo tempo, il flip-flop T è un circuito sequenziale caratterizzato dal fatto che il suo stato cambia ogni volta che viene applicato un impulso all'ingresso di sincronizzazione, mentre l'ingresso T è 1: esso è intrinsecamente un elemento di contatore binario ad uno stadio; pertanto c'è da aspettarsi che in generale un contatore ad n stadi sia realizzabile con n flip-flop T, pur non escludendo la possibilità di utilizzare anche latch o altri flip-flop, in particolare i JK.

Oltre a contare il numero di impulsi di ingresso e a memorizzare un numero che rappresenta il valore del conteggio effettuato, un contatore può realizzare anche altre importanti funzioni: 1) può generare un treno di impulsi derivati da impulsi di ingresso, ma a frequenza più bassa, ossia può effettuare **divisioni di frequenza**; 2) può fornire una sequenza di stringhe binarie da usare in varie applicazioni quale l'indirizzamento di una memoria.

6.3.1 - Contatori sincroni

Affrontiamo il problema della sintesi di un contatore osservando che se n è il numero di stadi è naturale identificare gli stati di uscita, che possono essere codificati con le 2^n combinazioni di n variabili logiche, con gli stati interni, aderendo quindi al modello di Moore di macchina sequenziale; con riferimento al caso di n = 3, senza che peraltro questo pregiudichi la generalità della trattazione, si ricava immediatamente il grafo degli stati riportato in Fig. 6.15:

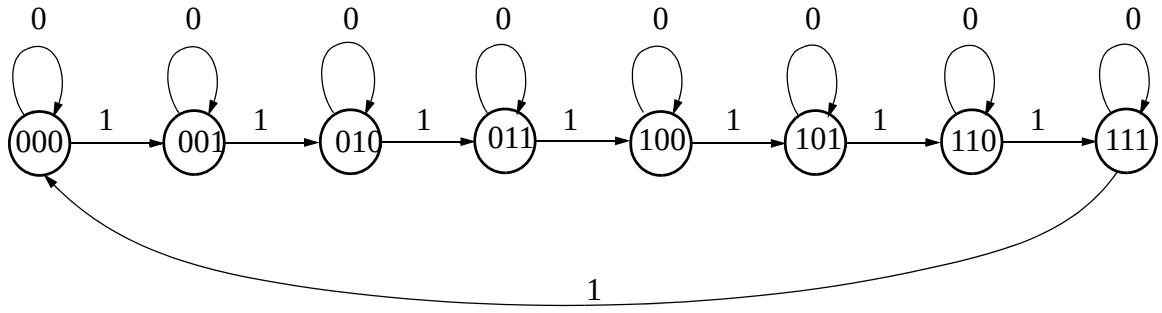


Fig. 6.15

Sviluppando da questo il procedimento di sintesi di una rete sequenziale sincronizzata, e tenendo conto dell'utilizzo di flip-flop T sincronizzati quali elementi di memoria, si ottengono le seguenti equazioni caratteristiche del contatore, il quale è detto **sincrono** per indicare il fatto che tutti i flip-flop sono impulsati insieme:

$$T_0 = x,\ T_1 = y_0x,\ T_2 = y_1y_0x = y_1T_1$$

In generale per lo stadio *i*-esimo avremo:

$$T_i = y_{i-1}y_{i-2}\ldots y_0x = y_{i-1}T_{i-1}$$

avendo indicato con *x* il segnale applicato all'ingresso *T* del flip-flop di ordine più basso.

Riferendoci al caso esemplificato, la Fig. 6.16 mostra le due possibili realizzazioni, ottenute sulla base delle equazioni caratteristiche sopra ricavate, che mettono in evidenza come la commutazione del flip-flop *i*-esimo avvenga quando tutti i precedenti si trovano nello stato 1 ed arriva un impulso in ingresso; in a) è schematizzata la versione con **riporto parallelo**, in b) quella con **riporto seriale**. Nella prima versione, poichè ogni flip-flop può commutare quando tutti gli ingressi della rispettiva porta AND sono 1, la frequenza di conteggio è indipendente dal numero degli stadi ed è data dall'espressione:

$$f \leq \frac{1}{t_{ff} + t_g}$$

dove t_{ff} e t_g sono rispettivamente il tempo di risposta di un flip-flop e di una porta logica. D'altra parte però lo stadio *i*-esimo richiede una porta AND con *i* ingressi e ciò può costituire un limite realizzativo se il numero degli stadi è abbastanza elevato.

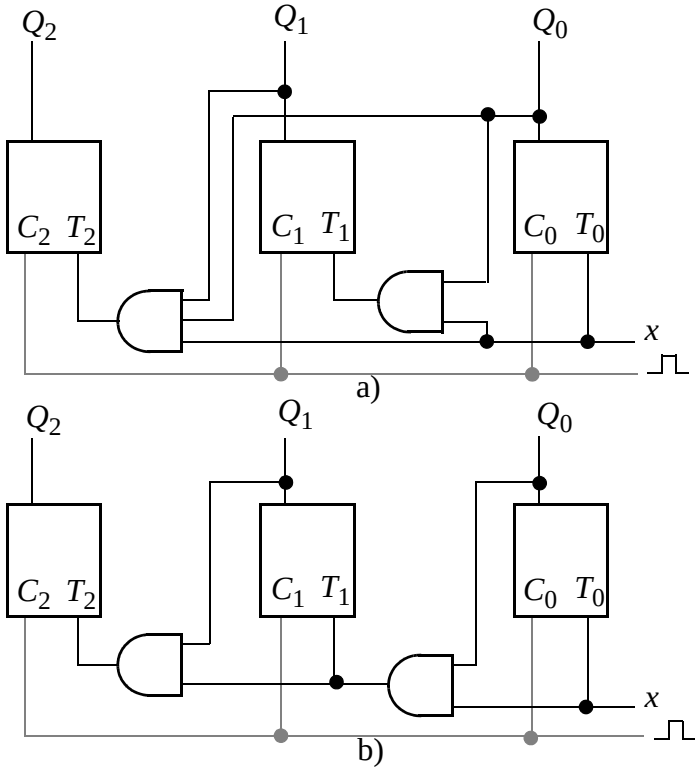


Fig. 6.16

Viceversa nella versione con riporto seriale, le uscite dei vari flip-flop, che concorrono a

determinare la commutazione dello stadio *i*-esimo, devono attraversare un numero di porte AND crescente con la distanza da *i*, fino ad un massimo di *n*-1 porte per l'uscita dallo stadio più significativo; pertanto la frequenza di conteggio diminuisce al crescere del numero degli stadi secondo la legge:

$$f \leq \frac{1}{t_{ff} + (n-1)t_g}$$

ed è generalmente più bassa di quella di un contatore con riporto parallelo; in compenso ogni porta AND ha soltanto due ingressi.

Secondo gli schemi della Fig. 6.16, di fatto il segnale che provoca l'avanzamento del conteggio è il clock, mentre il segnale *x* ha la funzione di segnale di abilitazione (*x* = 1) o di disabilitazione (*x* = 0); se questa funzione non è richiesta, *x* è tenuto fisso ad 1, per cui è sufficiente applicarlo all'ingresso T_0, eliminando di conseguenza la porta AND tra il primo ed il secondo stadio in entrambi gli schemi e riducendo di uno il numero di ingressi alle altre porte AND nello schema a). La Fig. 6.17 mostra le forme d'onda delle uscite Q_0, Q_1 e Q_2, supponendo che i flip-flop commutino sul fronte in salita del clock:

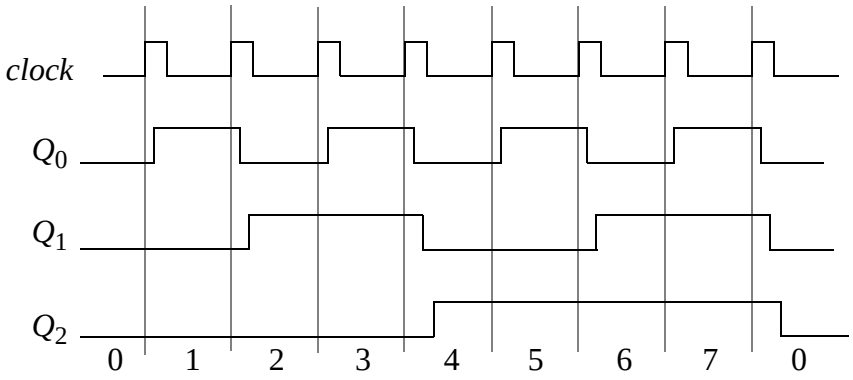


Fig. 6.17

6.3.2 - Contatori ripple-carry o asincroni

La forma più semplice di contatore binario è quella indicata nello schema di Fig. 6.18 per *n* = 3, basato sull'uso di flip-flop T asincroni eccitati sul fronte in discesa dell'ingresso; poichè la commutazione di ogni flip-flop, a parte il primo, è dovuta al fronte che si genera sull'uscita del flip-flop precedente, questo contatore viene detto **ripple carry** (a propagazione del fronte del riporto) o anche **asincrono**, in relazione al fatto che i vari stadi cambiano stato in tempi diversi, in conseguenza della propagazione sequenziale del fronte del riporto.

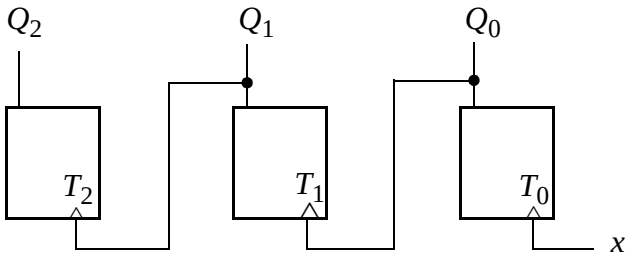


Fig. 6.18

Le forme d'onda delle uscite di questo contatore sono dettagliate nella Fig. 6.19:

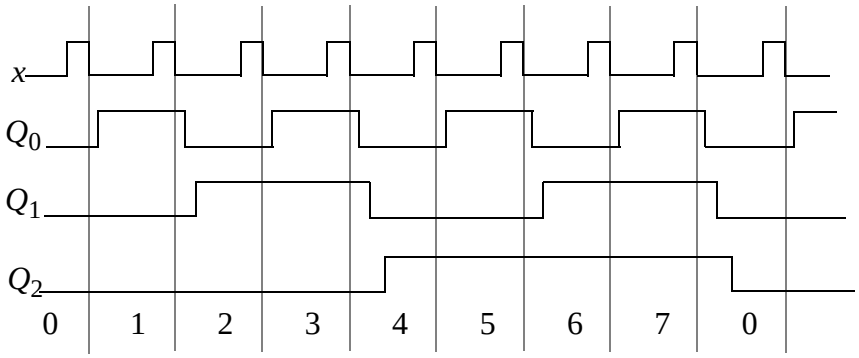


Fig. 6.19

Se l'eccitazione di ogni flip-flop oltre al primo viene prelevata dall'uscita complementata $\overline{Q}_i$ del flip-flop precedente anzichè da quella affermata come in Fig. 6.19, si ottiene un contatore ripple carry asincrono decrementante. Lo stesso comportamento si ottiene con flip-flop eccitati sul fronte in salita, prelevando il segnale di eccitazione di ciascuno, tranne il primo, dall'uscita complementata o da quella affermata del precedente, rispettivamente per un contatore incrementante o decrementante.

Nella Fig. 6.20 le funzioni di conteggio per incremento e per decremento sono riunite nello stesso dispositivo, utilizzando in ingresso ai flip-flop successivi al primo dei multiplexer comandati da un segnale a livelli **up/down**;

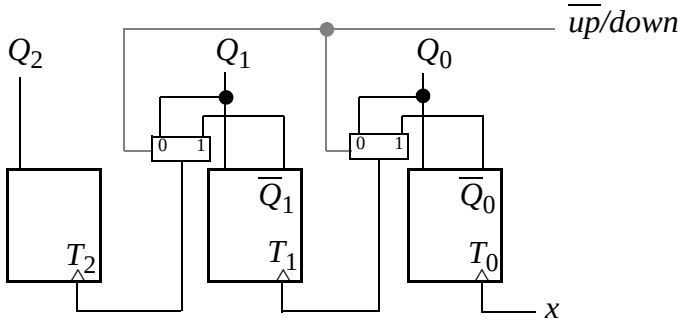


Fig. 6.20

In molte situazioni ha interesse che l'uscita di un contatore non sia il numero binario che

rappresenta il suo stato interno, bensì un segnale su una di 2^n linee, se n è il numero di stadi del contatore. Questo può essere ottenuto semplicemente decodificando le uscite di un contatore binario; tuttavia se si ricorre ad un contatore ripple carry, sulle uscite decodificate si possono presentare degli impulsi spurii (**spikes**) dovuti al modo caratteristico con cui in questo tipo di contatori avviene l'eccitazione dei vari stadi e ai ritardi interni di commutazione dei flip-flop. Ciò è evidenziato in Fig. 6.21, dove si vede per esempio che l'uscita decodificata $Z_1 = \overline{Q}_0 Q_1 \overline{Q}_2$ presenta, oltre al normale andamento, uno spike che la riporta ad 1 erroneamente.

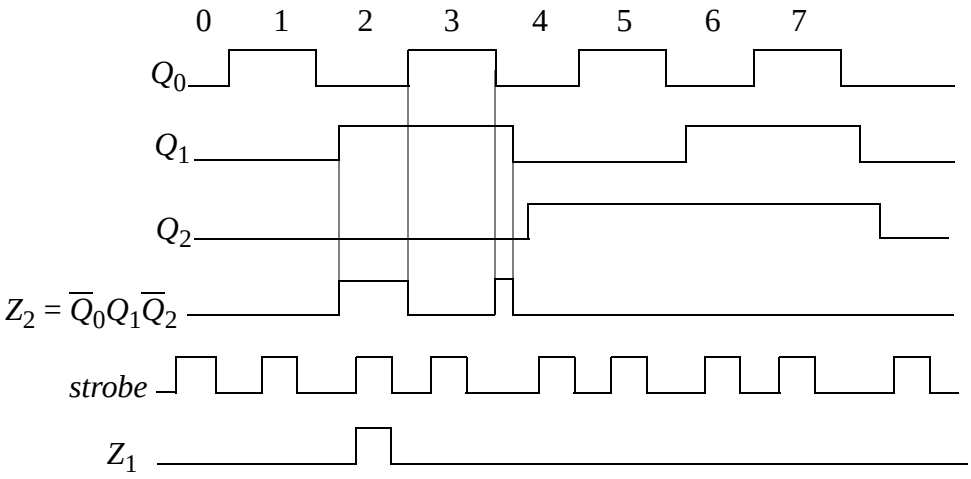


Fig. 6.21

Poichè il problema degli spike è dovuto alla commutazione di due o più flip-flop, la soluzione consiste nel fare in modo che tutti i flip-flop cambino simultaneamente, ricorrendo a contatori sincroni a riporto parallelo, oppure nel far commutare solo un flip-flop per volta. Un'altra tecnica consiste nell'utilizzare un impulso, detto **strobe**, che attiva il decodificatore solo quando lo stato del contatore si è stabilizzato, come mostra la stessa Fig. 6.21.

6.3.3 - Contatori non binari

Un limite dei contatori binari è il fatto che essi contano modulo una potenza di 2, mentre può essere necessario contare modulo un numero qualsiasi, soprattutto quando questi dispositivi sono usati come divisori di frequenza. Il problema può essere risolto sintetizzando un contatore modulo N, con N qualsiasi, secondo il consueto processo di sintesi delle reti sequenziali oppure, quando sia già disponibile un contatore binario con modulo $2^n > N$, creando connessioni opportune tra le sue uscite ed i suoi ingressi; in particolare nel caso di contatori con clear, si può generare un impulso di azzeramento mediante una porta AND ai cui ingressi sono applicate le uscite dei flip-flop che sono ad 1 per lo stato N.

a) Per esempio un contatore modulo 5 sincrono con ingresso di abilitazione può essere realizzato in base al primo approccio secondo lo schema di Fig. 6.22d;

b) In alternativa, supponendo di avere a disposizione un contatore binario sincrono dotato di reset asincrono, si può ricorrere alla soluzione di Fig. 6.23.

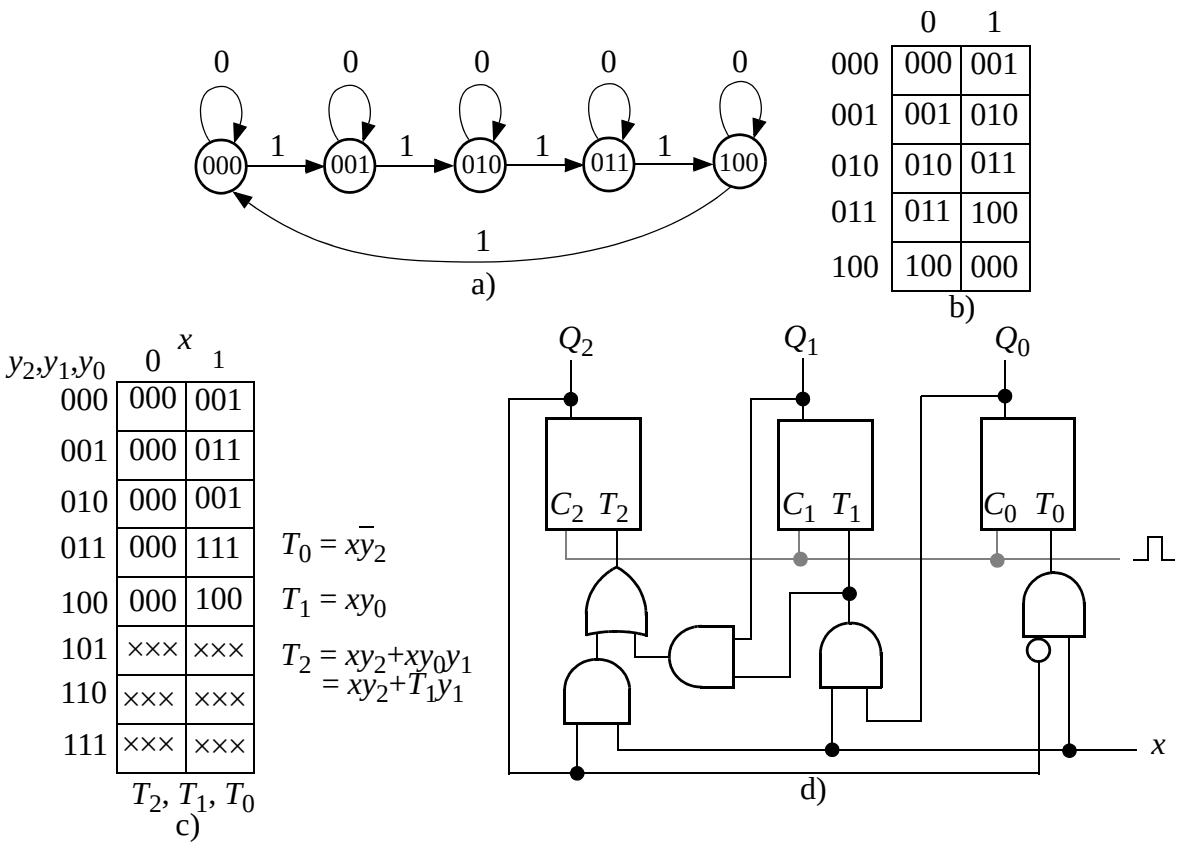


Fig. 6.22

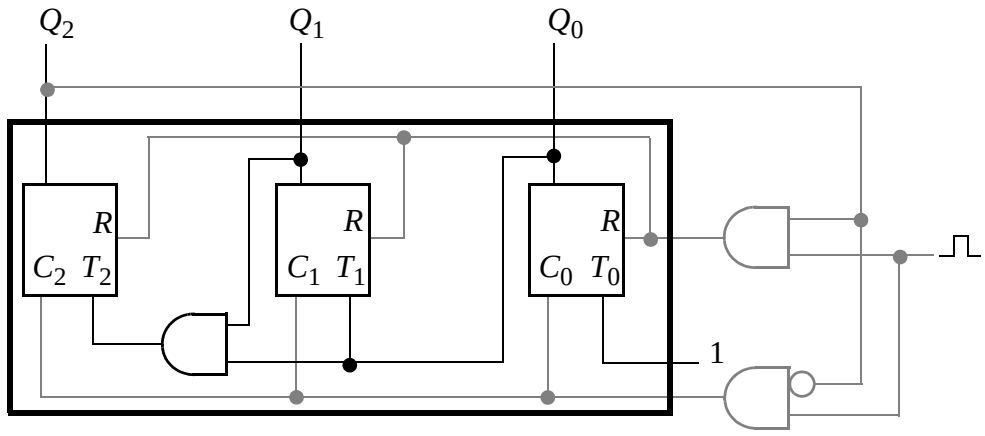


Fig. 6.23

Un'altra situazione frequente è quella nella quale è richiesto un contatore che percorra una sequenza di stati diversa da quella naturale; in questi casi si può usare un contatore binario con una rete combinatoria sulle uscite che converta dalla sequenza naturale a quella richiesta, oppure si può progettare un contatore in grado di fornire direttamente tale sequenza. Un tipico caso è quello di un contatore BCD, il quale percorre una sequenza di dieci stati corrispondenti alle cifre decimali codificate in binario; la Fig. 6.24 riporta lo schema di un contatore di tal

genere, per il quale si utilizzano flip-flop JK:

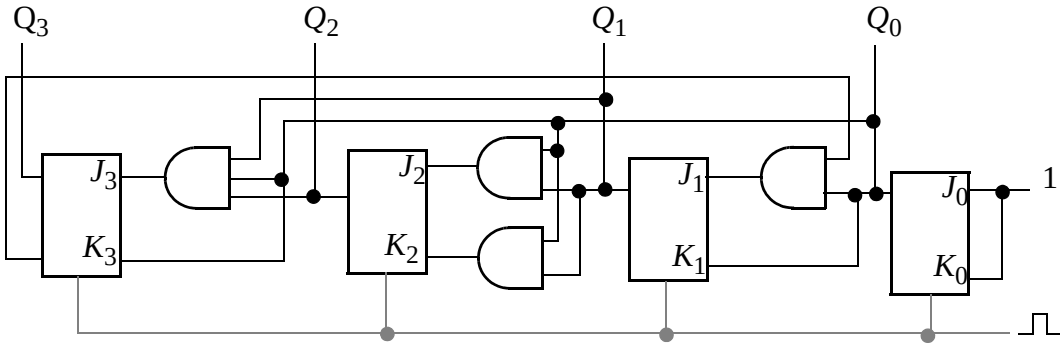


Fig. 6.24

6.3.4 - Contatori ad anello

Utilizzando registri di traslazione è possibile realizzare altre famiglie di contatori che, pur non avendo un numero di stadi minimo in rapporto al modulo (il numero di stati è uguale al numero di stadi), tuttavia hanno caratteristiche interessanti per quanto riguarda la facilità di disporre delle uscite decodificate; sono detti genericamente **contatori a traslazione** e comprendono in particolare i **contatori ad anello** (ring counters).

Un esempio di contatore ad anello a cinque stadi è mostrato in Fig. 6.25; consiste in un registro di shift a cinque bit nel quale l'uscita più significativa è connessa all'ingresso meno significativo:

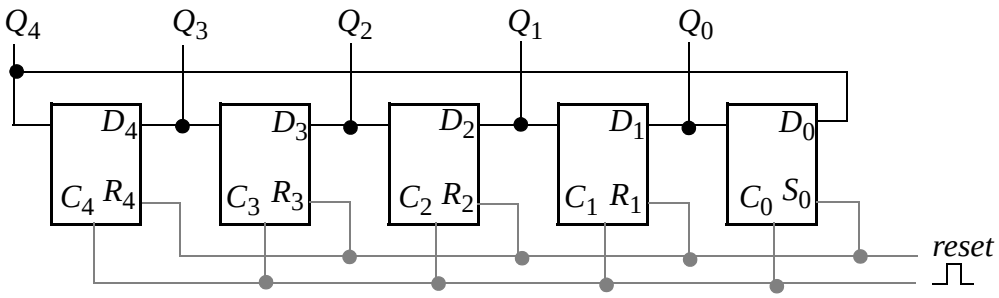


Fig. 6.25

I singoli stadi del registro sono flip-flop D sincronizzati, dei quali il meno significativo ha un ingresso di set, attraverso il quale è possibile scrivere 1 nel suo stato per inizializzare il contatore; lo stesso impulso serve ad azzerare tutti gli altri stadi, attraverso l'ingresso di reset dei rispettivi flip-flop. Si tratta di un contatore il quale percorre una sequenza ciclica di cinque stati nella quale un'uscita per volta viene posta ad 1; in altri termini le uscite sono direttamente decodificate.

Tuttavia questo circuito presenta l'inconveniente di poter entrare in una sequenza ciclica spuria, se a causa di disturbi viene assunto uno stato non previsto, contenente più di un 1. Per questo, come è stato detto, necessita di essere posto in uno stato iniziale valido. Per rimediare all'inconveniente si usa uno schema modificato come quello di Fig. 6.26, nel quale una porta

AND genera il termine $\overline{Q}_3\overline{Q}_2\overline{Q}_1\overline{Q}_0$ (in generale per *n* stadi il termine $\overline{Q}_{n-2}\ldots\overline{Q}_0$) che costituisce l'ingresso dello stadio meno significativo e vale 1 quando tutti gli stadi tranne il più significativo che non interessa sono a 0. In questo modo se il contatore entra in uno stato spurio all'uscita della porta AND si presenta 0 per *n*-1passi: a quel punto tutti gli ingressi della porta AND sono 1 e pertanto il contatore ritorna automaticamente nella configurazione di partenza 0...01 valida: si tratta di un contatore **autocorrettore** e **autoinizializzante**; per esempio se lo stato spurio è 01011, viene realizzata la sequenza: 01011, 10110, 01100, 11000, 10000, **00001**.

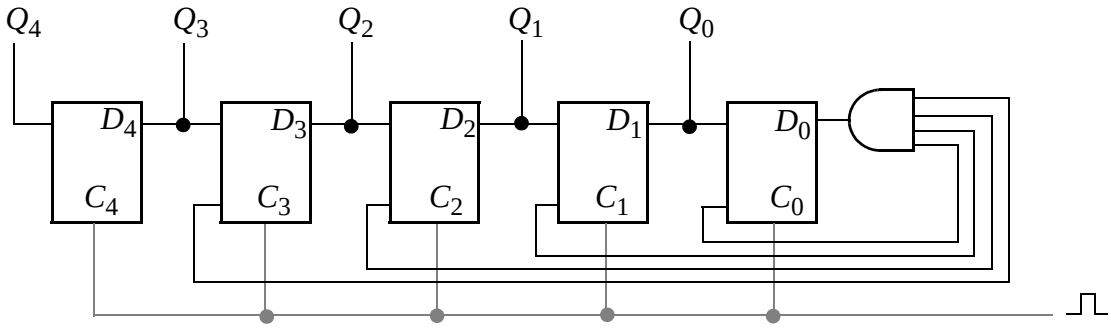


Fig. 6.26

ESERCIZI

1) Progettare un contatore binario che conta per tre modulo 8.

2) Un contatore binario a tre stadi con clear possiede un ingresso di controllo *x* che agisce nel modo seguente. Quando *x* = 0, il contatore conta secondo la sequenza 0, 1, 2, ..., 7, 0, ...; quando *x* = 1 conta secondo la sequenza 0, 2, 4, 6, 0, ... Realizzare il contatore.

3) Progettare un contatore decadico BCD, ovvero cun contatore che fornisce in uscita il codice 8-4-2-1 degli interi decimali 0 ... 9.

4) Sintetizzare un circuito contatore a tre stadi che percorre ciclicamente la sequenza 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 1 2 ... , utilizzando nel progetto flip flop JK.

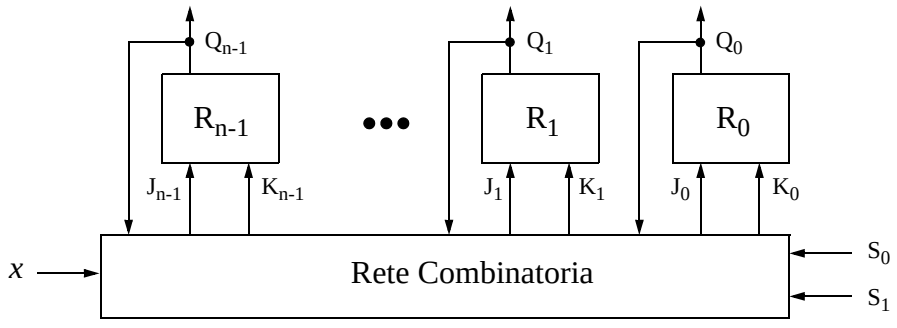
5) Si vuole realizzare un registro universale a quattro bit utilizzando come elementi di memoria dei flip-flop JK. Il registro deve possedere le seguenti funzionalità:
   - azzeramento;
   - ingresso parallelo;
   - shift logico sinistro/destro (coinvolge tutti i bit);
   - shift aritmetico sinistro/destro (lascia inalterato il bit più significativo);
   - rotazione logica sinistra/destra.

   Progettare una rete combinatoria per il controllo delle funzionalità del registro.

6) Progettare una rete combinatoria da accoppiare ad un registro R ad *n* bit come illustrato in figura, in modo da farlo funzionare in uno dei seguenti modi:

   1) contatore binario modulo n, sincrono con riporto seriale;
   2) $R_{i \bmod n} \leftarrow R_{(i-1) \bmod n}$ i = 0, ..., n-1;
   3) contatore ad anello autoinizializzante.

   Istanziare l'esercizio nel caso *n* = 4 e nell'ipotesi che i flip flop del registro siano di tipo JK.



7) Progettare un registro R a quattro bit, realizzati con flip-flop SR, con le seguenti modalità di funzionamento:

1) ingresso parallelo;
2) scorrimento circolare, secondo le specifiche: $R_i \leftarrow R_{i-2}$ e $R_{i-2} \leftarrow R_i$, i = 3, 4 con R_i bit i-esimo del registro;
3) contatore ad anello autoinizializzante e autocorrettore.

8) Progettare un contatore modulo 7 con due ingressi di controllo s_1 ed s_2 tali che con $s_1 = 1$ ed $s_2 = 0$ viene precaricato con un valore arbitrario compreso tra 0 e 6; con $s_1 = 0$ ed $s_2 = 0$ viene percorsa una sequenza di conteggio decrescente a partire dal valore preimpostato; con $s_1 = 0$ ed $s_2 = 1$ viene percorsa una sequenza di conteggio crescente, sempre a partire dal valore preimpostato.

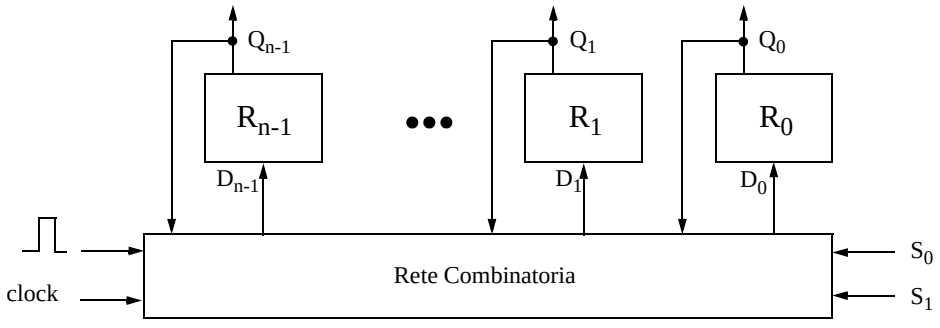
9) Progettare un contatore decimale con uscita BCD, il quale segue la sequenza di conteggio in verso crescente fino a 9, quindi al proseguire del conteggio, presenta sulle uscite una configurazione di tutti 1; il contatore deve essere riportato a 0 con un impulso di reset.

10) Si vuole progettare una struttura di conteggio che percorra ciclicamente la sequenza 0123 21 2345 43 4567 65 6701 07 ..., utilizzando un contatore binario up/down a tre stadi, un contatore modulo 6 ed un latch SR, oltre a consuete porte logiche. Disegnare lo schema logico della struttura richiesta.

11) Progettare una rete combinatoria da accoppiare ad un registro *R* ad *n* bit come illustrato in figura, in modo da farlo funzionare in uno dei seguenti modi:

   1) contatore incrementante modulo 10;
   2) registro di rotazione aritmetica verso sinistra: $R_i \leftarrow R_{i-1}$, $i = 1, ..., n-2$, $R_0 \leftarrow R_{n-2}$, $R_{n-1} \leftarrow R_{n-1}$;
   3) registro di traslazione aritmetica verso destra di 2 bit (la traslazione aritmetica propaga verso destra il bit più significativo);
   4) registro parallelo sull'ingresso

   Fare l'ipotesi che i flip-flop siano di tipo D; **attenzione**, questo implica che per la funzione di contatore occorre prendere opportuni provvedimenti.



12) Progettare un contatore modulo 13 in salita e modulo 15 in discesa. Il modo di conteggio è definito da una variabile a livelli *m* (*m* = 0, modo in salita e *m* = 1, modo in discesa). Il cambiamento del valore di *m* durante il funzionamento del contatore porta

quest'ultimo a seguire la sequenza di conteggio opposta a quella corrente, partendo dal valore raggiunto al momento della transizione di *m*.

13) Progettare un registro universale con le seguenti modalità di funzionamento:
    a) shift logico sinistro;
    b) shift aritmetico destro, con estensione del segno;
    c) contatore asincrono modulo 11 incrementante;
    d) contatore asincrono modulo 12 decrementante.

14) Progettare un contatore BCD a due cifre, ottenuto collegando opportunamente in cascata due contatori modulo 10, secondo lo schema di figura: