6 *CONTATORI E REGISTRI*

6.1 Contatore modulo 10

Supponiamo di voler progettare un contatore modulo 10: per definire 10 stati avremo bisogno di almeno quattro elementi di memoria. Poiché alle combinazioni delle quattro linee delle uscite delle memorie dovremo associare i contenuti decimali *m* del contatore, appare ovvio assegnare gli stati decimali ai termini minimi di peso corrispondente. Un tale contatore, il cui funzionamento è definito nella tabella 6.1.1, è detto *decimale codificato binario (BCD).*

Tabella 6.1.1

n	A^n	B^n	C^n	D^n	A^{n+1}	B^{n+1}	C^{n+1}	D^{n+1}
0	0	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0
2	0	1	0	0	1	1	0	0
3	1	1	0	0	0	0	1	0
4	0	0	1	0	1	0	1	0
5	1	0	1	0	0	1	1	0
6	0	1	1	0	1	1	1	0
7	1	1	1	0	0	0	0	1
8	0	0	0	1	1	0	0	1
9	1	0	0	1	0	0	0	0

Notiamo che i termini minimi m_{10}, m_{11}, m_{12}, m_{13}, m_{14} e m_{15} non fanno parte della sequenza. Questi termini sono detti *ridondanti* e possono essere utilizzati, analogamente a quanto visto nel §3.5, per la semplificazione delle equazioni applicative.

Ricaviamo dalla tabella 6.1.1 le equazioni applicative del nostro circuito sequenziale. Esse saranno:

$$A^{n+1} = (\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,B\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,C\overline{D} + \overline{A}BC\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D)^n$$

$$= (\overline{A}\overline{D} + \overline{A}\overline{B}\overline{C})^n = [\overline{A}(\overline{B}\overline{C} + \overline{D})]^n \quad (6.1.1)$$

$$B^{n+1} = (A\overline{B}\overline{C}\overline{D} + \overline{A}B\overline{C}\overline{D} + A\overline{B}C\overline{D} + \overline{A}BC\overline{D})^n$$

$$= (\overline{A}B\overline{D} + A\overline{B}\overline{D})^n = [B(\overline{A}\overline{D}) + \overline{B}(A\overline{D})]^n \quad (6.1.2)$$

$$C^{n+1} = (AB\overline{C}\overline{D} + \overline{A}\overline{B}C\overline{D} + A\overline{B}C\overline{D} + \overline{A}BC\overline{D})^n$$

$$= [C(\overline{A}\overline{D} + \overline{B}\overline{D}) + \overline{C}(AB\overline{D})]^n \quad (6.1.3)$$

$$D^{n+1} = (ABC\overline{D} + \overline{A}\overline{B}\overline{C}D)^n$$

$$= [D(\overline{A}\overline{B}\overline{C}) + \overline{D}(ABC)]^n \quad (6.1.4)$$

Le semplificazioni sono ottenute dai grafici dalla fig. 6.1.1 alla fig. 6.1.4.

Dal confronto di (6.1.1), (6.1.2), (6.1.3) e (6.1.4) con (5.4.6) si verifica che:

$$g_{1A} = 0 \qquad g_{2A} = \overline{B}\overline{C} + \overline{D} \quad (6.1.5)$$

$$g_{1B} = \overline{A}\overline{D} \qquad g_{2B} = A\overline{D} \quad (6.1.6)$$

$$g_{1C} = \overline{B}\overline{D} + \overline{A}\overline{D} \qquad g_{2C} = AB\overline{D} \quad (6.1.7)$$

$$g_{1D} = \overline{A}\overline{B}\overline{C} \qquad g_{2D} = ABC \quad (6.1.8)$$

Dalle (6.1.5), (6.1.6), (6.1.7) e (6.1.8), supponendo di voler realizzare il contatore con J-K, si ottengono dalle (5.4.7), e (5.4.8) le equazioni degli ingressi.

Prima di fare ciò tuttavia ci chiediamo se si possano usare le ridondanze per semplificare le equazioni applicative.

Nelle fig. 6.1.1, 6.1.2, 6.1.3 e 6.1.4 sono dati i grafici delle quattro equazioni applicative assieme ai termini ridondanti.

Si vede che associando i termini ridondati di indice pari alla (6.1.1) otteniamo:

$$A^{n+1} = \overline{A}^n \quad (6.1.9)$$

Associando m_{10}, e m_{14} alla (6.1.2) otteniamo:

$$B^{n+1} = [B(\overline{A}) + \overline{B}(A\overline{D})]^n \quad (6.1.10)$$

Associando tutte le ridondanze, esclusa m_{15}, alla (6.1.3) otteniamo:

$$C^{n+1} = [C(\overline{A} + \overline{B}) + \overline{C}(AB)]^n \quad (6.1.11)$$

Associando le ridondanze di indice pari alla (6.1.4) otteniamo:

$$D^{n+1} = [D(\overline{A}) + \overline{D}(ABC)]^n \quad (6.1.12)$$

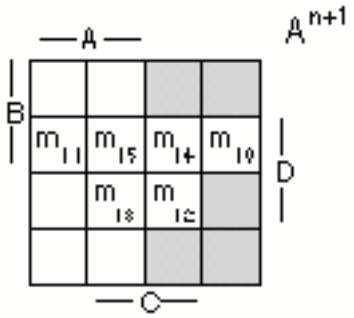

Figura 6.1.1

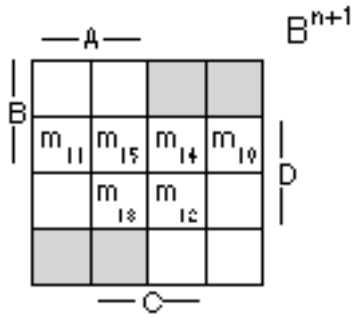

Figura 6.1.2

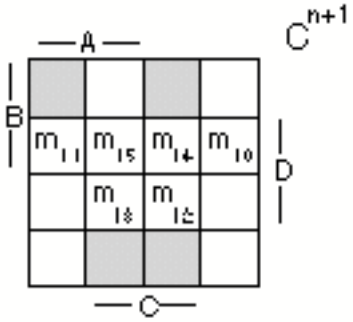

Figura 6.1.3

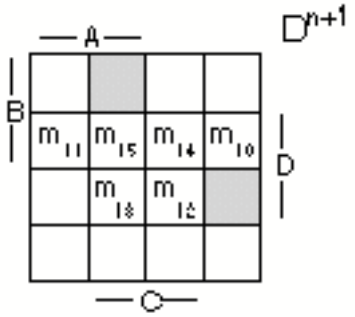

Figura 6.1.4

Nel caso della (6.1.4) non conviene associare m_{15}, che pur semplificherebbe l'espressione perché scomparirebbe un termine con $\overline{D}$ e l'equazione appplicativa non sarebbe più nella forma (5.3.1). Avendo utilizzato le ridondanze otteniamo delle espressioni per g_1 e g_2 notevolmente più semplici delle (6.1.5), (6.1.6), (6.1.7) e (6.1.8):

$$g_{1A} = 0 \qquad g_{2A} = 1 \tag{6.1.13}$$
$$g_{1B} = \overline{A} \qquad g_{2B} = A\overline{D} \tag{6.1.14}$$
$$g_{1C} = \overline{A} + \overline{B} \qquad g_{2C} = AB \tag{6.1.15}$$
$$g_{1D} = \overline{A} \qquad g_{2D} = ABC \tag{6.1.16}$$

Ci chiediamo tuttavia se l'uso delle ridondanze comporti qualche rischio. Aver utilizzato nelle equazioni applicative le ridondanze equivale a completare la tabella 6.1.1 con gli stati della tabella 6.1.2 dove, nelle colonne corrispondenti al tempo (n+1), si sono posti degli **1** in corrispondenza ai termini ridondati utilizzati.

Si vede che se, fortuitamente (per un disturbo o per mancanza di azzeramento iniziale), il contatore entrasse in uno stato ridondante con uno o al massimo due impulsi di conteggio ritornerebbe alla sequenza corretta.

Tabella 6.1.2

m	A^n	B^n	C^n	D^n	A^{n+1}	B^{n+1}	C^{n+1}	D^{n+1}
m_{10}	0	1	0	1	1	1	0	1
m_{11}	1	1	0	1	0	0	1	0
m_{12}	0	0	1	1	1	0	1	1
m_{13}	1	0	1	1	0	0	1	0
m_{14}	0	1	1	1	1	1	1	1
m_{15}	1	1	1	1	0	0	0	0

Ricordando la (5.4.7) e la (5.4.8), le equazioni degli ingressi per elementi di memoria J-K saranno:

$$J_A = g_{2A} = 1 \qquad K_A = \overline{g}_{1A} = 1 \tag{6.1.17}$$
$$J_B = g_{2B} = A\overline{D} \qquad K_B = \overline{g}_{1B} = A \tag{6.1.18}$$
$$J_C = g_{2C} = AB \qquad K_C = \overline{g}_{1C} = AB \tag{6.1.19}$$
$$J_D = g_{2D} = ABC \qquad K_D = \overline{g}_{1D} = A \tag{6.1.20}$$

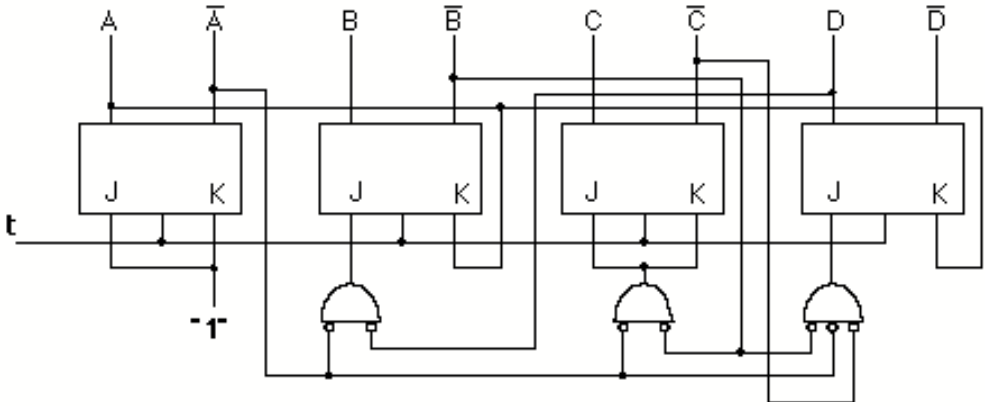

Figura 6.1.5

Il circuito che realizza il contatore è dato in fig. 6.1.5. Un contatore di questo tipo è detto *sincrono* perché il segnale di orologio è applicato a tutti gli elementi di memoria allo stesso istante. La massima frequenza di conteggio sarà inversamente proporzionale al tempo che impiegano gli ingressi a raggiungere uno stato stabile dopo un impulso di conteggio.

6.2 Contatore sincrono modulo 5

Supponiamo di voler progettare un contatore a 5 stati: per definirli (tabella 6.2.1) saranno necessarie almeno 3 linee.

Tabella 6.2.1

m	A^n	B^n	C^n	A^{n+1}	B^{n+1}	C^{n+1}
0	0	0	0	1	0	0

1	1	0	0	0	1	0
2	0	1	0	1	1	0
3	1	1	0	0	0	1
4	0	0	1	0	0	0

Si tenga presente che ci sono tre termini minimi, m_5, m_6, m_7, che non vengono considerati perché non fanno parte della sequenza. Anche in questo caso i termini ridondanti potranno essere utilizzati per la semplificazione delle equazioni applicative.

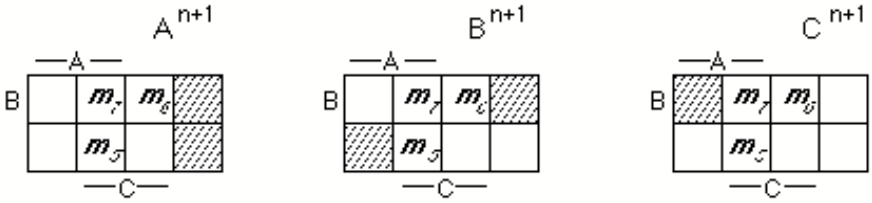

Figura 6.2.1

Nella figura 6.2.1 sono date le mappe delle tre equazioni applicative, con gli stati ridondanti, utilizzate per le semplificazioni.

L'associazione dei termini ridondanti equivale a completare la tabella 6.2.1 come indicato nella 6.2.1 bis.

Anche se fortuitamente il circuito cadesse in uno stato ridondante (m_5, m_6 o m_7), ad esempio all'accensione, dopo un impulso di orologio rientrerebbe nella sequenza corretta, passando a m_0 o a m_2.

Le equazioni applicative semplificate risultano essere pertanto:

$$A^{n+1} = (\overline{A}\overline{C})^n \qquad (g_1 = 0; g_2 = \overline{C})_A \tag{6.2.1}$$
$$B^{n+1} = (B\overline{A} + A\overline{B})^n \quad (g_1 = \overline{A}; g_2 = A)_B \tag{6.2.2}$$
$$C^{n+1} = (AB\overline{C})^n \qquad (g_1 = 0; g_2 = AB)_C \tag{6.2.3}$$

Supponendo di voler utilizzare elementi del tipo J-K, applicando le (5.4.7) e (5.4.8), otteniamo le equazioni degli ingressi:

$$J_A = g_{2A} = \overline{C} \qquad K_A = \overline{g}_{1A} = 1 \tag{6.2.4}$$
$$J_B = g_{2B} = A \qquad K_B = \overline{g}_{1B} = A \tag{6.2.5}$$
$$J_C = g_{2C} = AB \qquad K_C = \overline{g}_{1C} = 1 \tag{6.2.6}$$

Tabella 6.2.1 bis

m	A^n	B^n	C^n	A^{n+1}	B^{n+1}	C^{n+1}
0	0	0	0	1	0	0
1	1	0	0	0	1	0

2	0	1	0	1	1	0
3	1	1	0	0	0	1
4	0	0	1	0	0	0
5	*1*	*0*	*1*	*0*	*1*	*0*
6	*0*	*1*	*1*	*0*	*1*	*0*
7	*1*	*1*	*1*	*0*	*0*	*0*

Lo schema generale del contatore è dato in fig. 6.2.2.

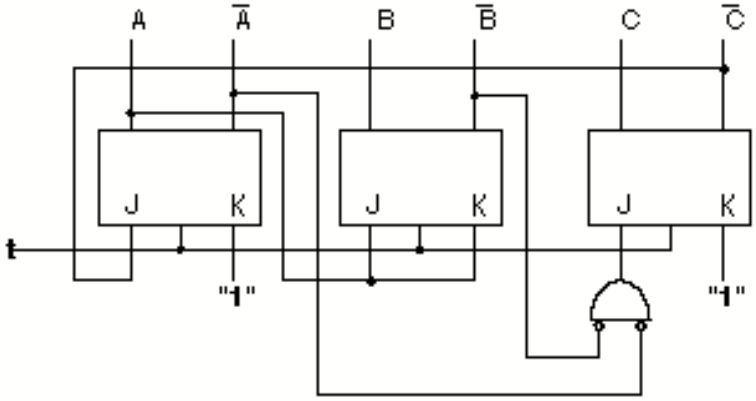

Figura 6.2.2

E' interessante notare che ciò che si è ottenuto è praticamente lo schema del popolare circuito integrato 7490.

Da un contatore modulo 5 è possibile ottenere un contatore modulo 10, detto *parzialmente sincrono*, col circuito di fig. 6.2.3 nel caso che i flip-flop commutino sul fronte positivo del segnale di orologio.

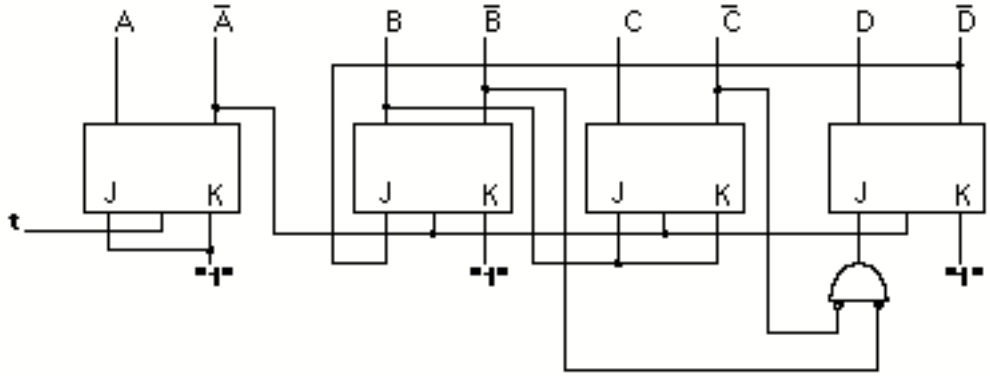

Figura 6.2.3

6.3 Contatore Johnson

Supponiamo di voler realizzare un contatore come quello descritto dalla tabella 6.3.1.

Il contatore è a modulo 6 ed è evidente che applicando la regola riassunta dalla tabella ad un sistema ad m elementi di memoria il modulo del contatore risulta essere $2m$.

Tabella 6.3.1

	A^n	B^n	C^n	A^{n+1}	B^{n+1}	C^{n+1}
m_0	0	0	0	1	0	0
m_1	1	0	0	1	1	0
m_3	1	1	0	1	1	1
m_7	1	1	1	0	1	1
m_6	0	1	1	0	0	1
m_4	0	0	1	0	0	0

Due termini minimi sono ridondanti (m_2, m_5) e possono essere utilizzati per la semplificazione delle equazioni applicative. Queste vengono direttamente mappate in fig. 6.3.1 e scritte in forma semplificata:

$$A^{n+1} = \overline{C}^n \tag{6.3.1}$$
$$B^{n+1} = A^n \tag{6.3.2}$$
$$C^{n+1} = B^n \tag{6.3.3}$$

La realizzazione del circuito, quindi, risulta estremamente semplice. Ad esempio, utilizzando elementi di tipo D, si ottiene lo schema di fig. 6.3.2.

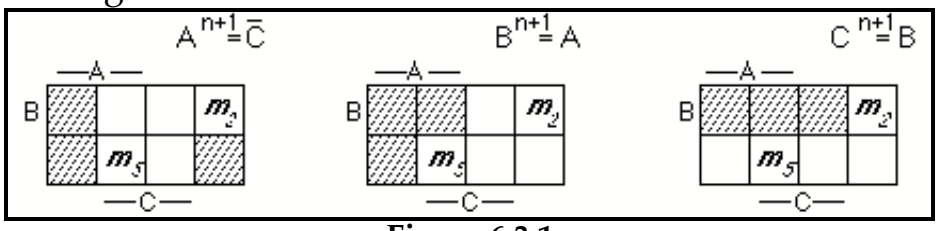

Figura 6.3.1

Se completiamo la tabella 6.3.1 tenendo conto dell'uso fatto dei termini ridondanti, otteniamo la tabella 6.3.1 bis.

Osserviamo che se casualmente, all'accensione o per un disturbo, il sistema assumesse uno degli stati ridondanti non ne uscirebbe più. Questa condizione è estremamente pericolosa e va evitata.

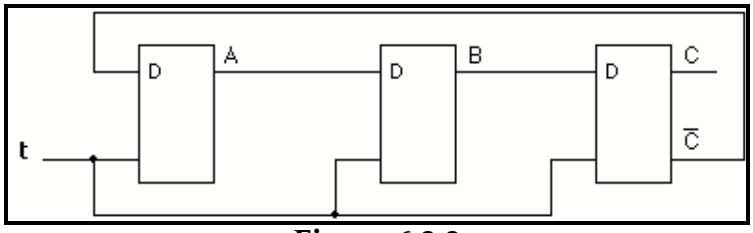

Figura 6.3.2

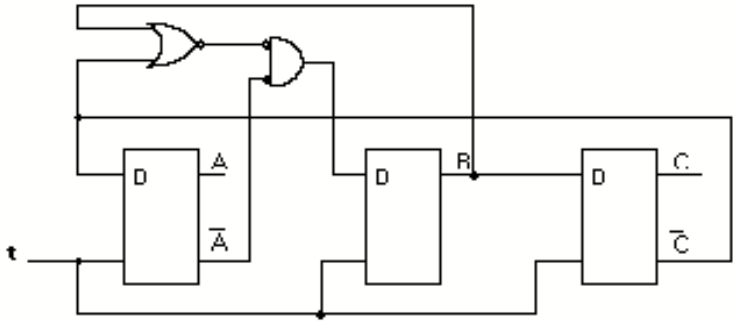

Figura 6.3.3

Tabella 6.3.1 bis

	A^n	B^n	C^n	A^{n+1}	B^{n+1}	C^{n+1}
	0	0	0	1	0	0
	1	0	0	1	1	0
	1	1	0	1	1	1
	1	1	1	0	1	1
	0	1	1	0	0	1
	0	0	1	0	0	0
m_2	*0*	*1*	*0*	*1*	*0*	*1*
m_5	*1*	*0*	*1*	*0*	*1*	*0*

Osservando la tabella si intuisce che l'unico modo per correggere questo comportamento consiste nell'eliminare uno qualsiasi degli **1** nel lato destro della tabella, in corrispondenza ai termini ridondanti. Le possibili equazioni applicative sono pertanto le seguenti:

$$A^{n+1} = (A\overline{C} + \overline{B}\,\overline{C})^n = [\overline{C}(A + \overline{B})]^n \qquad B^{n+1} = A^n \qquad C^{n+1} = B^n \qquad (6.3.4)$$

$$B^{n+1} = (AB + A\overline{C})^n = [A(B + \overline{C})]^n \qquad A^{n+1} = \overline{C} \qquad C^{n+1} = B^n \qquad (6.3.5)$$

$$C^{n+1} = (AB + BC)^n = [B(A + C)]^n \qquad A^{n+1} = \overline{C}^n \qquad B^{n+1} = A^n \qquad (6.3.6)$$

Sono tutte equivalenti come complessità circuitale. In fig. 6.3.3 è dato lo schema secondo (6.3.5), utilizzando sempre flip-flop D.

Analogamente un contatore a 5 stadi (modulo 10) si può realizzare come in fig. 6.3.4.

La decodifica decimale di un contatore Johnson a modulo 10 si ottiene con solo 10 porte AND (o NOR) a due ingressi ed è esente da incertezza di commutazione poiché fra due stati successivi varia un bit soltanto.

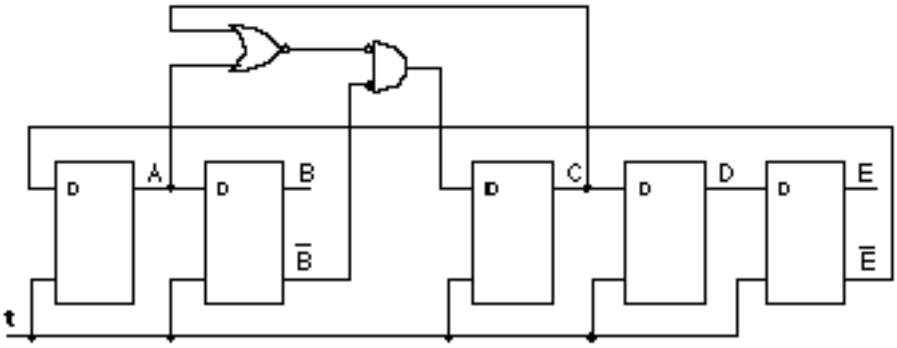

Figura 6.3.4

6.4 Contatore Gray

Alla fine del paragrafo precedente abbiamo notato che il codice dei contatori Johnson varia di un solo bit fra due stati adiacenti, con conseguente mancanza di incertezza di commutazione.

Tuttavia lo stesso risultato si può ottenere con molta maggior efficienza con contatori che seguano il codice Gray della tabella 6.4.1. Il caso del contatore Gray decimale è considerato nella tabella 6.4.2.

Tabella 6.4.1

A^n	B^n	C^n	D^n	A^{n+1}	B^{n+1}	C^{n+1}	D^{n+1}
0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0
1	1	0	0	0	1	0	0
0	1	0	0	0	1	1	0
0	1	1	0	0	0	1	0
0	0	1	0	1	0	1	0
1	0	1	0	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	0	1	1
1	0	1	1	0	0	1	1
0	0	1	1	0	1	1	1
0	1	1	1	0	1	0	1
0	1	0	1	1	1	0	1
1	1	0	1	1	0	0	1
1	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0

Tabella 6.4.2

A^n	B^n	C^n	D^n	A^{n+1}	B^{n+1}	C^{n+1}	D^{n+1}
0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0
1	1	0	0	0	1	0	0
0	1	0	0	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
0	1	0	1	1	1	0	1
1	1	0	1	1	0	0	1
1	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0

Diamo prima la soluzione per il contatore a 16 stati. La soluzione per il contatore modulo 10 si otterrà in modo analogo sfruttando, eventualmente, per la semplificazione i sei stati ridondanti.

Le equazioni applicative si ricavano dalla tabella 6.4.1 e sono:

$$A^{n+1} = (\overline{B}\,\overline{D} + ABC + B\overline{C}D)^n \tag{6.4.1}$$
$$B^{n+1} = (A\overline{D} + \overline{A}CD + \overline{A}B\overline{C})^n \tag{6.4.2}$$
$$C^{n+1} = (\overline{B}C + AC + \overline{A}B\overline{D})^n \tag{6.4.3}$$
$$D^{n+1} = (AD + CD + BD + ABC)^n \tag{6.4.4}$$

Per la soluzione circuitale adottiamo, in questo caso, la tecnica suggerita nel § 3.7, insieme ad elementi di memoria di tipo D la cui soluzione per gli ingressi è data dalla (5.4.9). Il circuito è dato in fig. 6.4.1.

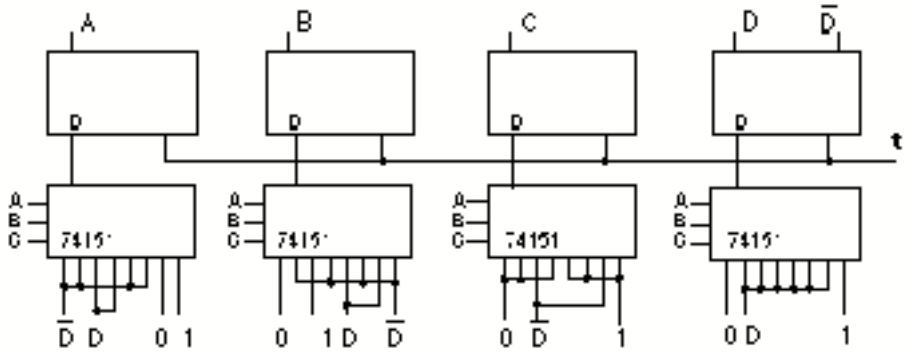

Figura 6.4.1

Per il contatore modulo 10, sfruttando le ridondanze m_4, m_5, m_7, m_{12}, m_{13}, m_{15} ricaviamo le seguenti equazioni applicative, adattate ad una soluzione del tipo (5.4.9):

$$A^{n+1} = (\overline{B}\,\overline{D} + B\overline{C}D)^n \tag{6.4.5}$$

$$B^{n+1} = (A\overline{D} + \overline{A}B)^n \quad (6.4.6)$$
$$C^{n+1} = (\overline{A}B\overline{D})^n \quad (6.4.7)$$
$$D^{n+1} = (AD + BD + C)^n \quad (6.4.8)$$

Il circuito si realizza come in fig. 6.4.2 utilizzando multiplexers a 4 ingressi.

Notiamo tuttavia che il codice Gray può essere ottenuto anche, partendo da un numero binario, attraverso funzioni combinatorie: nel §3.6 è stata discussa la conversione da codice binario a Gray e viceversa. Conviene notare che il circuito di fig. 3.6.2 può avere ritardi di propagazione pari a (*n*-1) ritardi di XOR se *n* è il numero di bit da trasformare.

Alcune considerazioni si impongono nel caso della codifica Gray di un contatore a modulo 10.

Un normale contatore BCD definito come nel §6.1 viene codificato in codice Gray da un circuito quale quello di fig. 3.6.1?

Evidentemente no. Dal confronto della tabella 6.4.2 con la 6.4.1 si vede che il Gray modulo 10, non utilizza i codici centrali da 0010 a 0011, pertanto un contatore modulo 10, del tipo definito nel §6.1, per essere codificato in Gray dal circuito di fig. 3.6.1 dovrebbe contare secondo la sequenza, non pesata, m_0, ... m_4, m_{11}, ... m_{15}.

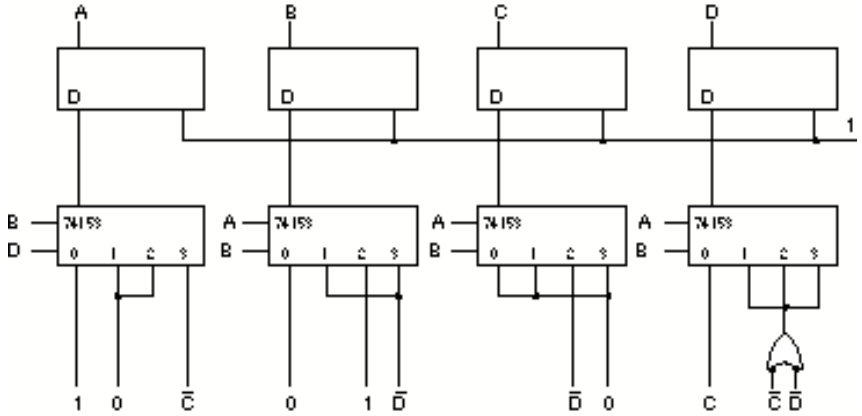

Figura 6.4.2

Inoltre se si vogliono porre in cascata più contatori Gray a modulo 10, per conservare la caratteristica della variazione di un solo bit nell'intero codice, bisogna adottare una sequenza di conteggio "bustrofedica" del tipo:
0 ..... 9, 19 ..... 10, 20 ..... 29, 39 ..... 30, 40 ..... 49, 59 ..... 50, 60 ..... 69, 79 ..... 70, 80 ..... 89, 99 ..... 90, 190 ..... 199, 189 ..... 180, 170 ..... 179, 169 ..... 160, etc.

I contatori delle singole cifre devono essere del tipo avanti/indietro e il controllo del senso di conteggio risulta assai macchinoso, nonostante il cambiamento del senso di conteggio

avanti/indietro si ottenga con la semplice inversione del bit D (vedi tabella 6.4.2).

6.5 Registri a scorrimento

Nella fig. 3.8.4 sono indicati dei segnali di selezione, *T*, che possono essere ottenuti, ad esempio, decodificando attraverso un circuito opportuno (§ 3.10) le uscite di un contatore di adeguata capacità.

Un altro modo semplice per ottenere una simile serie di segnali, è quello di usare un registro a scorrimento nel quale circoli un solo bit secondo quanto definito dalla tabella 6.5.1.

Tabella 6.5.1

A^n	B^n		N^n	A^{n+1}	B^{n+1}		N^{n+1}
0	0	.	0	1	0	.	0
1	0	.	0	0	1	.	0
0	1	.	0	0	0	.	0
.	.	.	.	.	.	.	.
0	0	.	1	0	0	.	0

La tabella prevede un istante in cui non sia presente alcun bit nel registro. Si può, a seconda delle esigenze, definire la sequenza come in tabella 6.5.2, dove si prevede che il bit uscente dall'ultimo flip-flop rientri direttamente nel primo.

Tabella 6.5.2

A^n	B^n		N^n	A^{n+1}	B^{n+1}		N^{n+1}
1	0	.	0	0	1	.	0
0	1	.	0	0	0	.	0
.	.	.	.	.	.	.	.
0	0	.	1	1	0	.	0

Facendo uso di tutte le ridondanze dalla tabella 6.5.1, usando flip-flop di tipo D, si ottengono le equazioni applicative :

$$A^{n+1} = (\overline{A} \bullet \overline{B} \bullet .. \overline{N})^n \qquad (6.5.1)$$

$$B^{n+1} = A^n \qquad (6.5.2)$$

etc.

Da queste equazioni si ricava il circuito di fig. 6.5.1. Si vede immediatamente che qualora entrasse più di un bit nel registro, o il registro contenesse **1** in tutte le celle, si realizzerebbe comunque la sequenza desiderata dopo un certo numero di impulsi di orologio.

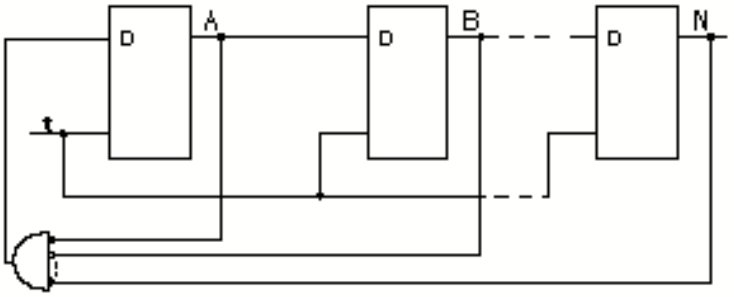

Figura 6.5.1

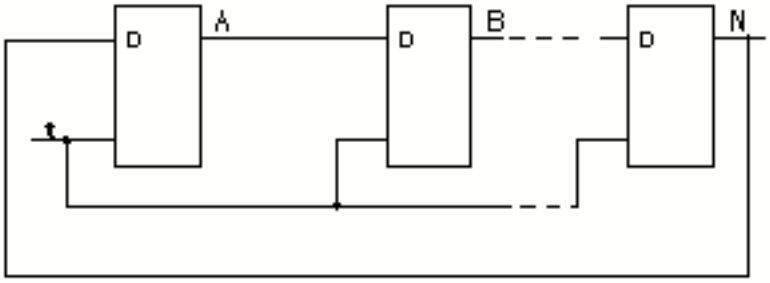

Figura 6.5.2

Nel caso della tabella 6.5.2., facendo uso di tutte le ridondanze, si otterrebbero le equazioni:

$$A^{n+1} = N^n \qquad (6.5.3)$$

$$B^{n+1} = A^n \qquad (6.5.4)$$

etc.

che porterebbero al circuito di fig. 6.5.2 che perderebbe la sequenza desiderata ogniqualvolta entrasse più di un bit nel registro. In particolare se le celle diventassero tutte **0** o **1**, il contenuto del registro resterebbe bloccato in quella configurazione.

Più in generale un registro a scorrimento come quello di fig. 6.5.1 o 6.5.2, privo della porta connessa al primo flip-flop, può essere considerato un registro adatto a rendere accessibili in parallelo, sulle sue uscite, dati che entrino serialmente, sincronizzati dall'orologio, nell'ingresso del primo flip-flop.

Analogamente un registro a scorrimento potrebbe rendere seriale un dato caricato in parallelo sul registro stesso.

In questo caso basta realizzare una cella del tipo di fig. 6.5.3 che, su controllo della linea *P*, renda disponibili all'ingresso della cella, o la linea *M* o la linea *N*. Connettendo le celle come nelle fig. 6.5.4, si ottiene un registro che, sincrono con l'orologio, fa scorrere i dati quando *P* è **0** e carica i dati dalle linee parallele quando *P* è **1**, fornito inoltre di una linea di azzeramento $\overline{C}$.

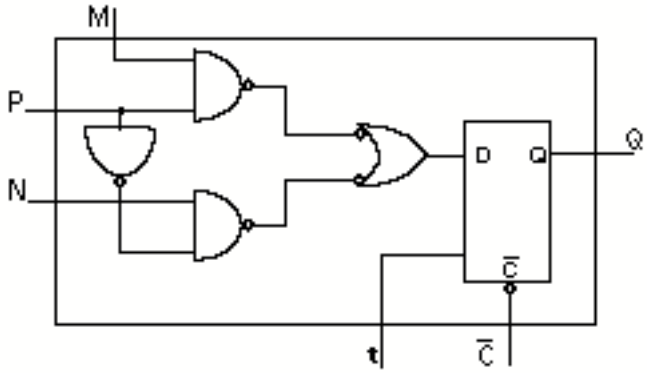

Figura 6.5.3

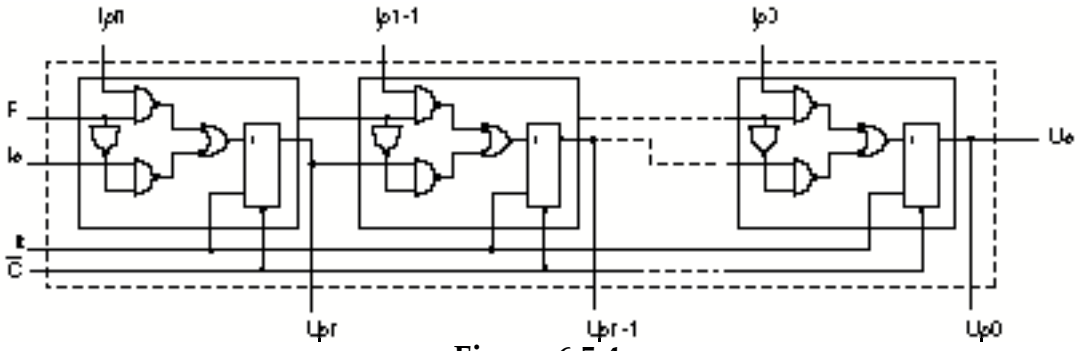

Figura 6.5.4

In generale i registri a scorrimento sono usati per il trattamento seriale dell'informazione. Nel sommatore della fig. 3.1.1, ad esempio, si suppone che gli addendi siano depositati in due registri con uscita parallela e il risultato della somma finisca in un analogo registro con ingresso parallelo.

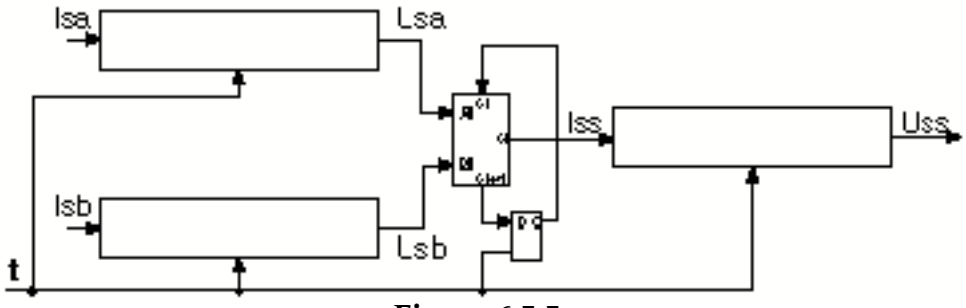

Figura 6.5.5

Qualora si voglia una struttura seriale si può, a spese della velocità di esecuzione, ottenere lo stesso risultato con una sola cella di sommatore, un flip-flop che memorizzi il riporto generato e tre registri a scorrimento, connessi come nella fig. 6.5.5, controllati da un orologio comune. Il bit meno significativo sarà quello spostato per primo.

6.6 Modulo di memoria ad accesso casuale

Supponiamo di voler realizzare una tabella costituita da 16 righe di registri, di quattro flip-flop ciascuno, nella quale scrivere, per righe, un'informazione binaria di 4 bit. Lo schema a blocchi di questo circuito è dato nella fig. 6.6.1, che avrà quattro linee, *A*, *indirizzi*, per selezionare la riga, e quattro linee, *D*, per i *dati*.

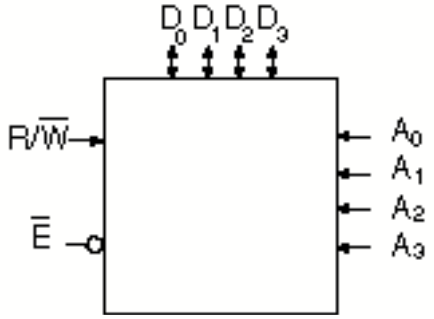

Figura 6.6.1

Poiché vogliamo essere in grado di leggere e scrivere nella riga selezionata, dovremo avere una linea di controllo, $R/\overline{W}$, che metterà i dati sulle linee d'uscita quando sarà allo stato **1**, mentre alla transizione da **0** a **1** scriverà i dati, presenti sulle linee *D*, nella riga selezionata. Le linee *D* sono pertanto bidirezionali e convogliano i dati verso l'esterno quando $R/\overline{W}$ è uguale a **1**, mentre acquisiscono i dati dall'esterno quando $R/\overline{W}$ è uguale a **0**.

Inoltre, per rendere estendibile a un numero maggiore di registri o di bit il modulo, introduciamo anche una linea di abilitazione $\overline{E}$, attiva a livello **0**.

Vediamo anzitutto come può essere realizzato il modulo di fig. 6.6.1, utilizzando componenti già studiati nei paragrafi precedenti. Possiamo decidere di utilizzare flip-flop di tipo D, per le celle di memoria, e demultiplexer per distribuire il segnale di comando scrittura e di lettura alla riga selezionata dall'indirizzo.

Essendo inoltre le linee *D* bidirezionali, conviene utilizzare, per controllare la direzione di trasferimento, elementi tri-state.

Lo schema completo è dato nella fig. 6.6.2. Il modulo può essere utilizzato come elemento costituente una memoria più grande che può diventare, a sua volta, il modulo per un'altra memoria di maggiori dimensioni e così via, con un processo ricorrente. Nella fig. 6.6.3 è dato un esempio di memoria a 8 bit (byte) con 64 locazioni definite da 6 linee di indirizzo, realizzata con 8 moduli del tipo di fig. 6.6.2. Essendo anch'essa fornita di una linea di abilitazione $\overline{E}$, può essere usata, a sua volta, per realizzare memorie più grandi.

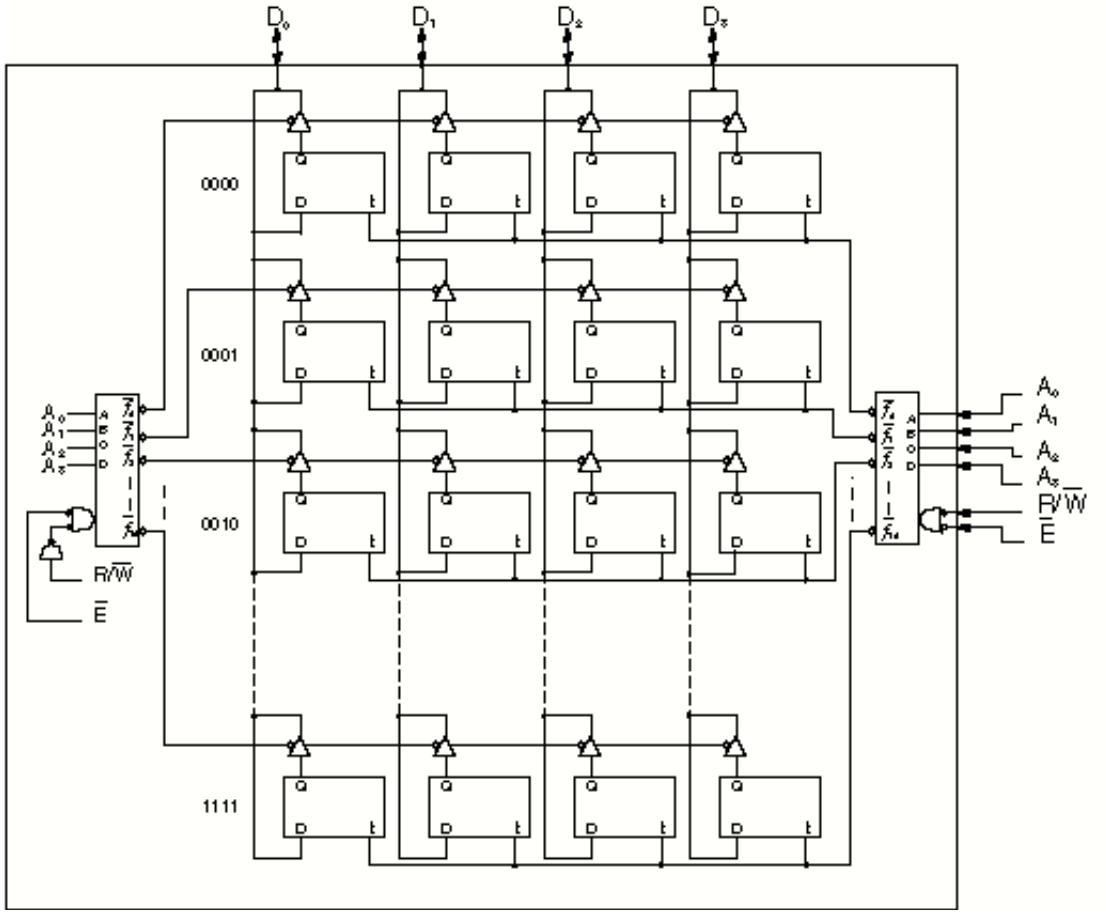

Figura 6.6.2

Naturalmente lo schema proposto è valido solamente come esempio di architettura. In realtà le moderne memorie a semiconduttore, al fine di raggiungere elevate densità e bassa dissipazione , sono realizzate con strutture CMOS.

Le memoria del tipo descritto è detta *statica* perché ritiene l'informazione per un tempo indefinito, purché alimentata, anche se non viene letta o scritta.

Vi sono memorie *dinamiche*, a più alta densità, realizzate con elementi di memoria che richiedono periodici cicli di lettura al fine di non perdere l'informazione.

Le memorie, sia statiche che dinamiche, ad *accesso casuale*, sono dette RAM (Random Access Memory). Le memorie che hanno un contenuto predefinito in fase di costruzione o mediante una speciale procedura di scrittura, sono dette ROM (Read Only Memory) quando non sono riprogrammabili e EPROM (Erasable Read Only Memory) quando invece lo sono.

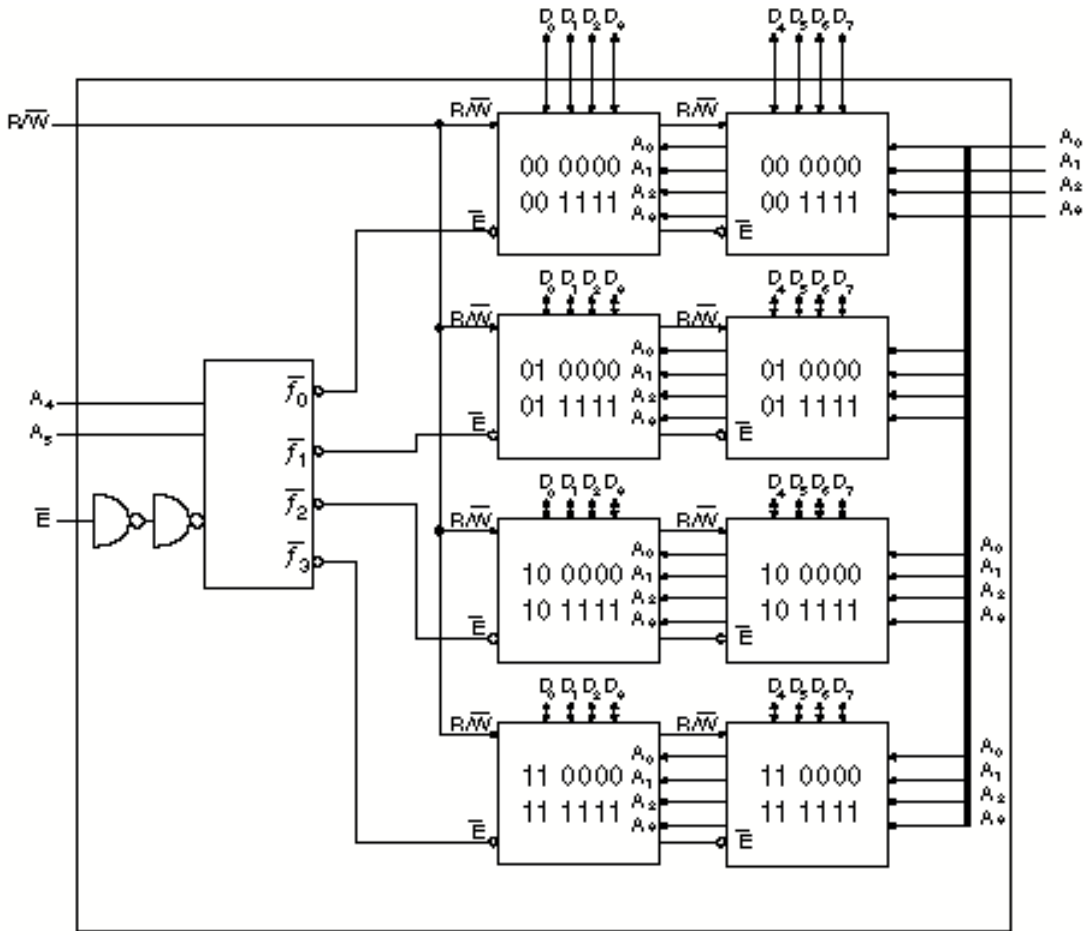

Figura 6.6.3

Usando memorie ROM, o EPROM, come tabelle è possibile realizzare funzioni combinatorie e quindi, come vedremo, anche funzioni sequenziali in modo semplice e modulare.

6.7 Funzioni booleane realizzate con ROM

Abbiamo visto nel capitolo 2 che le funzioni combinatorie booleane si descrivono e sintetizzano mediante le tavole della verità, dalle quali poi si ricava la *f* espressa come somma di prodotti o prodotto di somme.

Disponendo di memorie nella quali poter definire un contenuto opportuno, possiamo pensare di scrivere in esse le tavole della verità, ricavando la *f* direttamente dalle sue uscite.

Supponiamo di disporre di una memoria ROM a sedici locazioni da 4 bit, *E*,*F*,*G*,*H* che richiede quattro linee di indirizzo, *A*,*B*,*C*,*D*.

Se consideriamo le linee di indirizzo *A*,*B*,*C*,*D* come le variabili della *f*, possiamo scrivere, ad esempio nel bit *E* della

locazione individuata dal termine minimo m_i, il valore f_i che assume in corrispondenza la f.

L'uscita E della memoria realizzerà la f al variare delle variabili, indirizzi, A, B, C, D.

Essendo il dato della memoria di quattro bit, è chiaro che con la ROM di questo esempio, possono realizzarsi quattro diverse funzioni delle stesse quattro variabili.

Nel realizzare funzioni combinatorie con ROM non si pone il problema delle semplificazioni dato che la memoria contiene l'intera tavola della verità.

Con memorie ROM e un registro di flip-flop, tenendo presente lo schema generale di fig. 5.3.1, si possono quindi realizzare anche funzioni sequenziali.

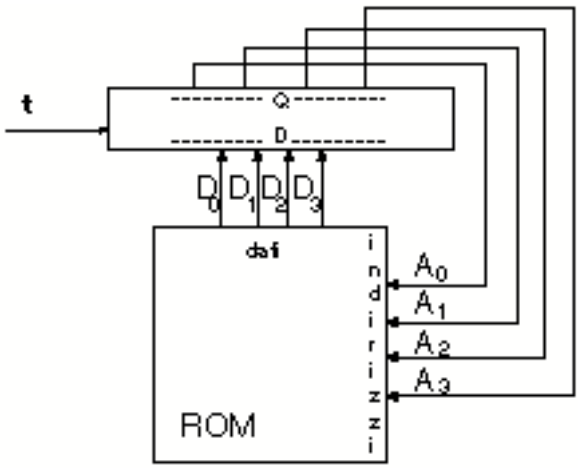

Figura 6.7.1

In particolare un contatore, con codice qualsiasi, si può realizzare come in fig. 6.7.1, scrivendo nella locazione individuata dal codice dello stato n, il codice dello stato (n+1).

6.8 Esercizi

a)-Disegnare un contatore che esegua la sequenza:

$$m_0, m_1, m_7, m_{10}, m_{11}, m_{12}, m_{14}, m_{15}$$

Confrontare la soluzione con 4 flip-flop con altre possibili soluzioni.

b)-Disegnare un contatore BCD avanti indietro, sincrono, con senso di conteggio controllato da una linea U. Valutare la possibilità di realizzare un contatore parzialmente sincrono.

c)-Realizzare un circuito che generi ripetitivamente le forme d'onda di fig. 6.81.

Figura 6.8.1

d)-Realizzare dispositivo, detto registro ad approssimazioni successive (SAR), a *n* bit che, pilotato da un orologio **t**, attraversi *n* fasi, separate da un periodo di azzeramento, e si comporti nel seguente modo:
il bit della posizione *n*-esima passa comunque nella (*n*+1)-esima ad ogni impulso di **t** e rimane o meno nella *n*-esima in funzione dello stato della linea *I* all'istante (*n*+1). *I*=**0** il bit scompare dalla posizione *n*, *I*=**1** il bit resta nella posizione *n*.
Tenere presente che può essere necessario utilizzare più di un registro di flip-flop.

e)-Realizzare mediate una ROM di dimensioni adeguate una unità che moltiplichi due numeri binari interi, positivi, di quattro bit ciascuno.

f)-Realizzare mediante una ROM di dimensioni adeguate un contatore Gray del tipo di tabella 6.4.1.