CAPITOLO VII
CIRCUITI SEQUENZIALI SINCRONI

7.1) Introduzione.

Nel capitolo precedente sono stati presi in considerazione gli aspetti essenziali, sia dal punto di vista teorico che applicativo, dei circuiti sequenziali asincroni, con riferimento al modello fondamentale, esaminando in dettaglio i problemi relativi alla loro sintesi.

Nel presente capitolo verra' affrontato il problema dell'analisi e della sintesi dei circuiti sequenziali sincroni.

E' bene mettere in evidenza che per tale tipo di circuiti e' difficile individuare degli esempi che siano allo stesso tempo semplici e significativi. Per tale motivo gli esempi che verranno presi in considerazione avranno un carattere chiaramente artificiale e non dovranno essere considerati esempi significativi di sistemi reali.

7.2) Analisi dei circuiti sequenziali sincroni.

Come nel caso delle reti elettriche, anche nel caso dei circuiti sequenziali l'analisi e' piu' semplice che non la sintesi.

Per illustrare il procedimento sara' bene riferirsi ad un esempio; si consideri pertanto il circuito di fig. 7.2.1.

E' interessante notare che la struttura del circuito e' tale da coincidere perfettamente con il modello fondamentale. Si osservi inoltre che il segnale di clock non ha alcun contenuto informativo per quanto riguarda il funzionamento del circuito nella sua evoluzione da uno stato all'altro, ma serve unicamente a temporizzare le operazioni; esso potra' pertanto essere ignorato nel corso dell'analisi.

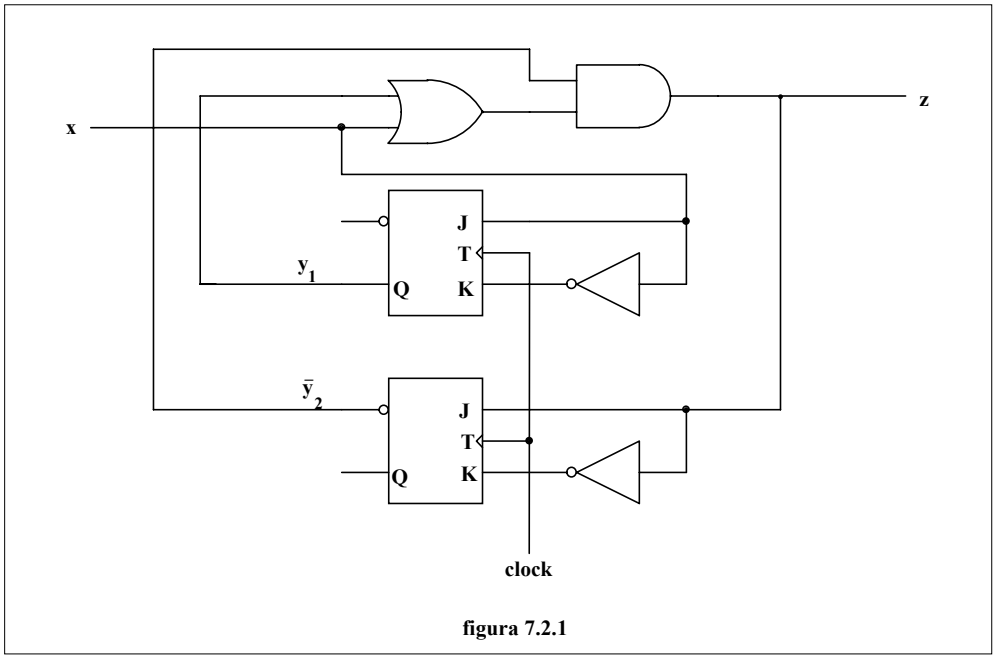


figura 7.2.1

Le equazioni di ingresso degli elementi di memoria (o **equazioni di eccitazione**) possono essere determinate dall'analisi della rete combinatoria. Si ottiene:

$$J_{y1} = x \qquad K_{y1} = \overline{x}$$

$$J_{y2} = z = (x + y_1).\overline{y_2} \qquad K_{y2} = \overline{(x + y_1).\overline{y_2}}$$

A partire da queste equazioni, utilizzando le equazioni di eccitazione del flip-flop JK, si determinano le equazioni di stato degli elementi di memoria y_1 e y_2. Poiche' per un flip - flop JK si ha:

$$y^{n+1} = J^n.\overline{y^n} + \overline{K^n}.y^n$$

si ottiene, tenendo presente che per ciascun elemento di memoria $K = \overline{J}$:

$$y_1^{n+1} = x^n.\overline{y_1^n} + x^n.y_1^n = x^n$$

$$y_2^{n+1} = (x^n + y_1^n)\overline{y_2^n} + \left[(x^n + y_1^n)\overline{y_2^n}\right].y_2^n = (x^n + y_1^n)\overline{y_2^n}$$

Da tali equazioni si puo' ottenere la tavola di transizione del circuito sequenziale che permette di valutare per ogni combinazione di x^n, y_1^n e y_2^n i valori di y_1^{n+1}, y_2^{n+1} e z^n.

E' semplice seguire sulla tavola di transizione l'evoluzione del circuito, tenendo presente che per ciascun impulso di clock si puo' avere un'unica variazione di stato.

Se ad esempio $x^n = y_2^n = 0$ e $y_1^n = 1$ si ha $z^n = 1$. All'istante n+1, cioe' dopo il successivo impulso di clock, si ha $y_1^{n+1} = 0, y_2^{n+1} = 1$ e nell'ipotesi che x^{n+1} sia sempre pari a 0, $z^{n+1} = 0$.

$y_1^n\, y_2^n$ \ x^n	**0**	**1**
00	**00/0**	**11/1**
01	**00/0**	**10/0**
11	**00/0**	**10/0**
10	**01/1**	**11/1**

$y_1^{n+1}\ y_2^{n+1}\ / z^n$

La tavola di transizione tende a diventare piuttosto voluminosa al crescere del numero di variabili. Si puo' semplificare la notazione codificando con q_1, q_2, q_3 e q_4 le quattro combinazioni di y_1 e y_2 ed operando analogamente nel caso di ingressi e uscite multiple, indicando rispettivamente con x_i e z_i le diverse combinazioni di ingressi e uscite. Combinando, come illustrato in fig. 7.2.2 (a) la parte relativa all'uscita e quella relativa al prossimo stato, si ottiene quella che viene chiamata tabella di stato.

La stessa informazione contenuta nella tabella di stato puo' essere visualizzata graficamente con il diagramma di stato, riportato in fig. 7.2.2 (b), il cui significato e' gia' stato

illustrato in precedenza. La rappresentazione e' evidentemente perfettamente equivalente a quella di Mealy adottata per le macchine sequenziali.

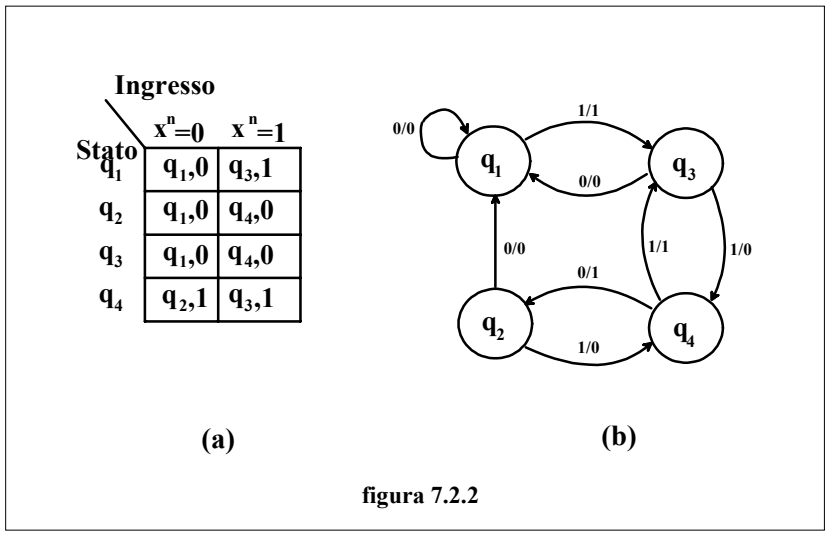


figura 7.2.2

7.3) Procedura di progetto.

La procedura di analisi illustrata al paragrafo precedente procede attraverso i seguenti passi:

La sintesi procede in direzione opposta a quella dell'analisi; la principale differenza consiste nel fatto che la sintesi parte ancor prima che esista un diagramma o una tabella di stato. Tale diagramma viene infatti di solito ottenuto da una qualsiasi descrizione del problema, che di solito consiste in una rappresentazione non ambigua a parole di cio' che il circuito deve fare. La procedura di sintesi e' illustrata in fig. 7.3.1.

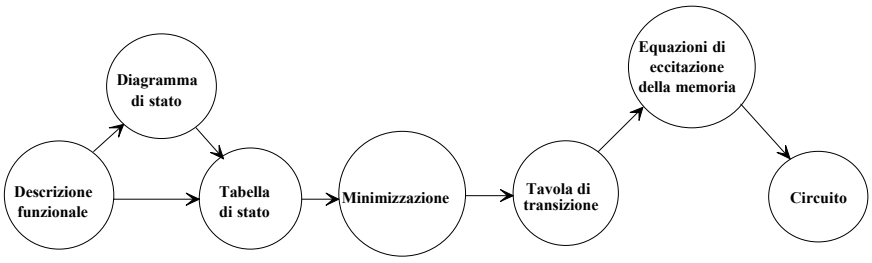


figura 7.3.1

Una notevole attenzione deve essere dedicata ai primi due passi, cioe' nel determinare la tabella di stato e nel ricavare poi la tabella di stato minima.

A partire da quest'ultima si puo' ottenere un circuito senza grosse difficolta'. Risulta invece notevolmente difficile ottenere il circuito piu' economico in quanto la complessita' delle equazioni di ingresso degli elementi di memoria varia con le diverse codificazioni dello stato.

7.4) Sintesi del diagramma di stato.

La determinazione della tavola di stato tramite un algoritmo e' possibile solamente se il comportamento del circuito e' descritto in forma di **espressione regolare**. Si lascera' tuttavia la discussione delle espressioni regolari a testi di teoria degli automi, assumendo per i nostri scopi che la prima descrizione formale di un circuito sequenziale sia il diagramma di stato. Attraverso l'uso di questo diagramma si puo' ottenere, come illustrato in fig. 7.2.2, la tabella di stato.

La determinazione del diagramma di stato si rivela particolarmente semplice per quei circuiti per i quali e' facilmente identificabile uno stato chiamato **stato di reset**. Quando tale stato esiste, vi deve essere la possibilita' di portare il circuito nella condizione di reset, a partire da un qualsiasi altro stato, con un'unica operazione. Essa puo' essere realizzata con una speciale colonna di ingresso sulla tabella di stato, oppure puo' essere considerata una funzione completamente indipendente da tale tabella. Ad esempio si puo' connettere una linea speciale a tutti i flip-flop per azzerarli al momento opportuno. Per compiere l'operazione di reset la linea puo' essere collegata sia agli ingressi considerati nell'implementazione della tabella di stato, sia, come spesso avviene, ad ingressi separati.

ESEMPIO 1 (Reset eseguito con meccanismo separato)

In un particolare sistema di comunicazione l'inizio di un messaggio sia identificato dalla presenza di tre 1 consecutivi sulla linea di ingresso x. I dati su questa linea siano sincronizzati da un impulso di clock. Si voglia progettare un circuito sequenziale sincrono la cui uscita diventi 1 solo in corrispondenza al riconoscimento di inizio messaggio. Vi sia inoltre un meccanismo separato di reset che possa riportare il circuito nello stato q_0 che precede l'inizio di un messaggio. Il comportamento tipico del circuito e' illustrato in fig. 7.4.1.

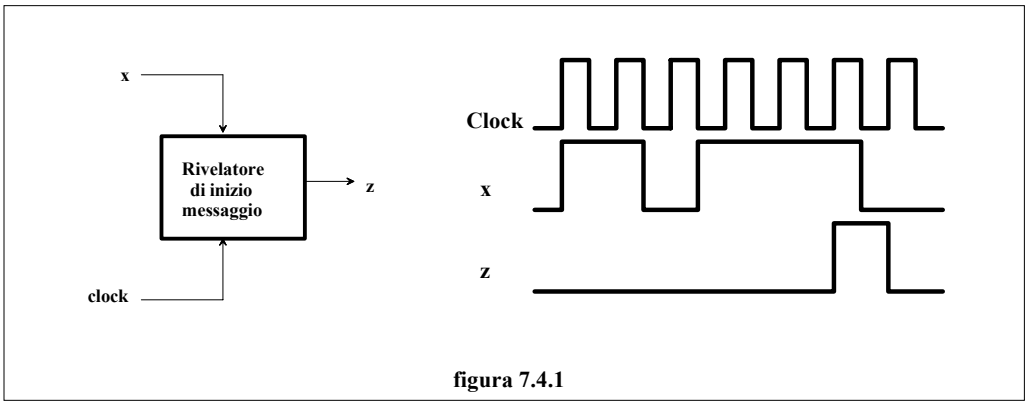


figura 7.4.1

La sintesi prende le mosse fissando lo stato di reset q_0 , come illustrato in fig. 7.4.2 (a). A partire da questo stato il circuito deve conteggiare il numero di 1 ricevuti consecutivamente; cio' viene realizzato passando ad un nuovo stato ogni volta che viene ricevuto un nuovo 1.

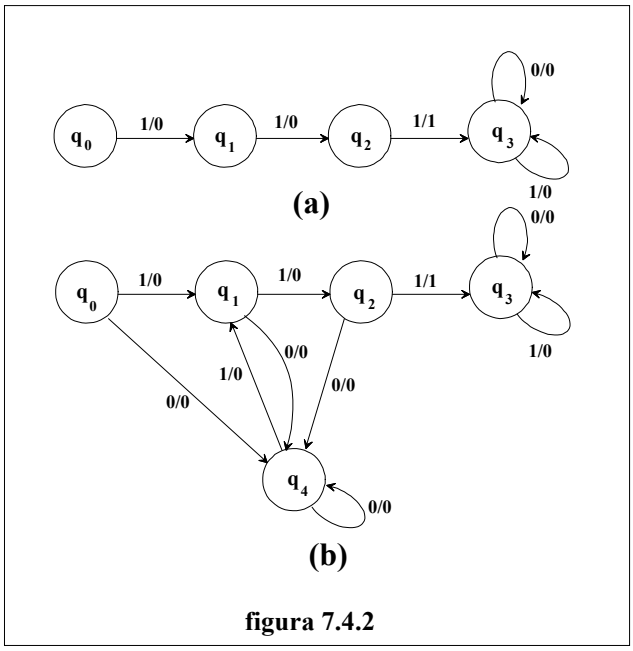


figura 7.4.2

Come e' illustrato in fig. 7.4.2 il circuito si porta nello stato q_1 quando viene ricevuto il primo 1 e nello stato q_2 quando viene ricevuto il secondo. L'uscita associata a ciascuna di queste transizioni e' 0. Il terzo 1 consecutivo che perviene in ingresso manda il circuito in q_3 e genera un'uscita 1. Giunto in q_3 il circuito vi rimane, qualsiasi sia l'ingresso, ed ha uscita nulla finche' non viene riportato in q_0 secondo un meccanismo esterno che non e' illustrato nella tabella di stato di fig. 7.4.3.

Stato \ Ingresso	0	1
q_0	q_4,0	q_1,0
q_1	q_4,0	q_2,0
q_2	q_4,0	q_3,1
q_3	q_3,0	q_3,0
q_4	q_4,0	q_1,0

figura 7.4.3

Poiche' uno 0 in ingresso si puo' presentare in qualsiasi periodo di clock, interrompendo una sequenza di 1, e' stato previsto uno stato q_4. Qualsiasi 0 in ingresso, a meno che il circuito non si trovi gia' nello stato q_3, genera un'uscita 0 e porta il circuito nello stato q_4; tutti gli 0 successivi e consecutivi non fanno altro che lasciare il circuito in q_4, mentre un 1, che potrebbe iniziare una sequenza di tre 1 consecutivi, riporta il circuito in q_1.

Tutte le possibili sequenze di ingresso sono state in tal modo prese in considerazione. E' immediato verificare che in corrispondenza a ciascun stato compaiono sul grafo le transizioni relative ad ambedue gli ingressi. Si ha cioe' una macchina sequenziale completa.

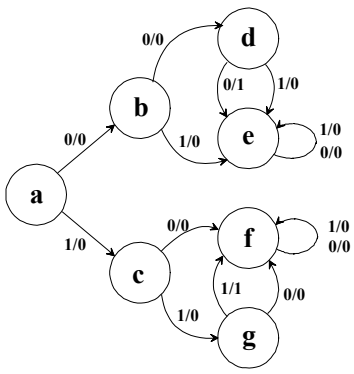
Si puo' notare che lo stato q_4 potrebbe essere eliminato riportando il circuito nello stato q_0 quando si ha in ingresso uno 0. In generale infatti, quando possibile, si utilizzano stati gia' esistenti come prossimo stato. Tuttavia essendo la costruzione del diagramma di stato una procedura intuitiva, qualora ci sia il dubbio se sia necessario definire un nuovo stato o utilizzarne uno gia' esistente, e' bene introdurre un nuovo stato per evitare seri errori.

ESEMPIO 2 (Reset eseguito con un opportuno ingresso)

Numeri compresi tra 0 e 7 vengono trasmessi in forma seriale binaria su una linea x, con i bit ordinati dal piu' al meno significativo. Si vuol progettare un circuito che fornisca un'uscita z = 1 in corrispondenza al terzo bit di uno dei due valori estremi 0 e 7. In caso contrario dovra' essere z = 0. Esista inoltre un ulteriore ingresso d tale che, se vale 1 ad un determinato istante di clock, allora al successivo impulso di clock si abbia l'arrivo del bit piu' significativo di un messaggio.

Si imponga che la condizione **d = 1** porti il circuito in uno stato a di reset. La tabella di stato puo' essere costruita considerando **d** come un meccanismo esterno di reset, e solo successivamente integrando **d** come ingresso nella tabella di stato.

Partendo dallo stato a si puo' costruire un grafo come illustrato in fig. 7.4.4. I numeri 0 e 7 sono rivelati all'istante del terzo ingresso e precisamente se uno 0 viene applicato allo stato d o un 1 allo stato g. Il terzo ingresso deve inoltre portare il circuito in uno stato tale che ulteriori uscite uguali a 1 siano impossibili. Gli stati e ed f soddisfano queste specifiche e il circuito dopo aver raggiunto uno di tali stati vi permane indefinitamente.



Stato \ Ingresso x	0	1
a	b/0	c/0
b	d/0	e/0
c	f/0	g/0
d	e/1	e/0
e	e/0	e/0
f	f/0	f/0
g	f/0	f/1

figura 7.4.4

Si puo' notare tuttavia che il circuito funziona allo stesso modo una volta che abbia raggiunto lo stato e oppure lo stato f. Quindi, a rigore, solo uno di tali stati e' necessario; ad esempio la riga f della tavola di flusso di fig. 7.4.4 puo' essere soppressa e nella tavola stessa f puo' essere sostituito ovunque da e senza alterare in alcunche' il funzionamento del circuito.

Tenendo conto anche dell'effetto dell'ingresso **d** e codificando l'ingresso secondo quanto illustrato in fig. 7.4.5 (a) si ottiene la tabella di stato di fig. 7.4.5 (b).

Le prime due colonne sono la copia della tavola di flusso di fig. 7.4.4 mentre i rimanenti due ingressi fanno evolvere il sistema verso lo stato a di reset.La tabella di stato di fig. 7.4.5 (b) realizza quindi quanto esposto nelle specifiche senza la necessita' di alcun meccanismo separato di reset.

d.x	X
00	1
01	2
10	3
11	4

(a)

Stato \ X	1	2	3	4
a	b/0	c/0	a/0	a/0
b	d/0	e/0	a/0	a/0
c	f/0	g/0	a/0	a/0
d	e/1	e/0	a/0	a/0
e	e/0	e/0	a/0	a/0
f	f/0	f/0	a/0	a/0
g	f/0	f/1	a/0	a/0

(b)

figura 7.4.5

Il metodo di determinazione del diagramma degli stati che e' stato esposto e' del tutto generale. In assenza di uno specifico stato di reset si puo' usare come punto di partenza un qualsiasi stato; tuttavia in assenza di uno stato di partenza ben determinato e distinguibile dagli altri, il grafo che si costruisce puo' raggiungere notevoli dimensioni. Per contenere tali dimensioni e' bene che gli archi del grafo puntino a stati esistenti, a partire dai quali si ha lo stesso comportamento futuro, piuttosto che a nuovi stati appena definiti.

7.5) Circuiti a memoria finita.

In fig. 7.5.1 e' illustrata una versione equivalente del circuito di fig. 7.2.1 in cui i flip-flop JK sono stati sostituiti dai flip-flop D.

Ricordando che l'uscita di un flip-flop di tipo D al tempo t^n e' uguale al valore dell'ingresso al tempo immediatamente precedente t^{n-1}, si vede che nel caso in esame l'uscita e' funzione solamente dell'ingresso attuale e dell'ingresso e dell'uscita all'istante immediatamente precedente.

Si ha cioe':

$$z^n = f\left(x^n, x^{n-1}, z^{n-1}\right)$$

E' questo un esempio di circuito a memoria finita, nel senso che esso puo' ricordare solamente un numero finito di ingressi e uscite passate per determinare l'uscita attuale e lo stato futuro.

Si noti che la dizione "**finito**" non ha nulla a che fare con la capacita' di memoria in bit, che e' ovviamente sempre e comunque limitata.

Una versione generalizzata di circuito a memoria finita e' illustrata in fig. 7.5.1 (b), in cui vengono memorizzati r ingressi precedenti e s uscite. L'uscita attuale e' funzione di queste informazioni memorizzate e dell'ingresso attuale x.

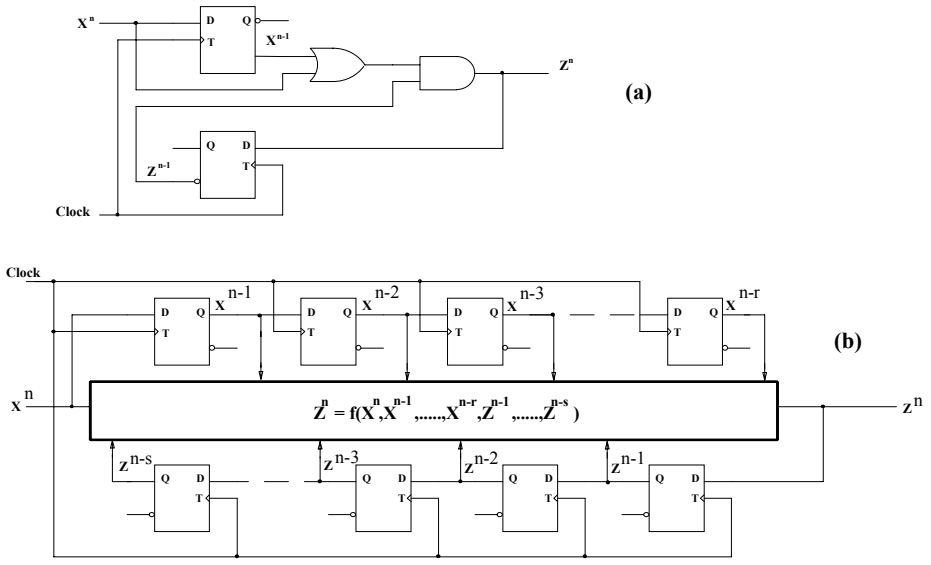


figura 7.5.1

Ovviamente x e z possono essere dei vettori rappresentanti rispettivamente parecchie linee di ingresso e di uscita. Allo stesso modo i flip-flop D possono rappresentare anziche' un singolo flip-flop, un insieme monodimensionale di flip-flop, uno per ciascuna linea di input o di output.

Non tutti i circuiti sequenziali sono a memoria finita. Negli esempi 1 e 2 del paragrafo precedente il circuito doveva fornire un'uscita in corrispondenza ad alcune particolari sequenze di ingresso, e non vi doveva essere nessuna altra uscita se prima non fosse stato applicato un opportuno segnale di reset. Tali circuiti quindi erano in grado di ricordare una particolare sequenza di ingresso o di uscita, presentatasi in precedenza, ad una distanza temporale (o se si vuole per un numero di impulsi di clock) arbitrariamente grande.

E' spesso possibile riconoscere un circuito a memoria finita dalle sue specifiche di progetto e tale riconoscimento puo' sovente semplificare la procedura di sintesi.

Quando infatti si sappia che il circuito e' a memoria finita, si puo' in genere specificare un limite superiore per la dimensione della memoria; si puo' cioe' fissare il massimo numero di stati interni e ricavare direttamente la tavola delle transizioni o quella di stato, come viene illustrato nel seguente esempio.

ESEMPIO 1

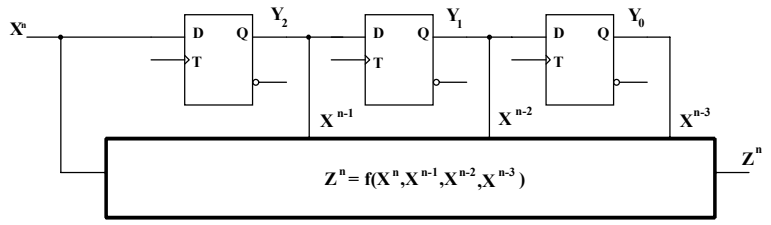
Alcuni bit di informazioni siano codificati su una linea x e sincronizzati da un clock. I bit siano codificati in modo che sulla linea x non possano mai apparire 2 o piu' simboli 1 o 4 o piu' simboli 0 consecutivi.

Si voglia progettare un circuito rivelatore di errore che dia un'uscita pari a 1 in corrispondenza o al quarto 0 o al secondo 1 consecutivo sulla linea x. Se, ad esempio, si dovessero avere tre 1 consecutivi sulla linea x, l'uscita dovrebbe rimanere alta per i due ultimi periodi di clock.

E' immediato riconoscere che il circuito da progettare e' un circuito a memoria finita. Anzi il caso che si sta esaminando potrebbe essere definito a memoria finita dell'ingresso, poiche' per un corretto funzionamento e' sufficiente memorizzare un certo numero di valori passati dell'ingresso. Nel caso peggiore il circuito dovra' memorizzare tre ingressi precedenti: se essi fossero stati 0 e l'ingresso attuale fosse ancora 0 l'uscita dovra' commutare a 1.Un circuito che soddisfi quanto enunciato puo' avere la struttura di fig. 7.5.2 (a). I bit in arrivo vengono memorizzati nella catena di flip-flop in modo che in qualsiasi istante

$$y_2^n = x^{n-1}$$
$$y_1^n = x^{n-2}$$
$$y_0^n = x^{n-3}$$

La rete logica combinatoria provvedera' poi a sintetizzare l'uscita sulla base dei dati memorizzati e dell'ingresso attuale. La tavola di transizione del circuito in questione puo' allora essere ricavata direttamente, come illustrato in fig. 7.5.2 (b).



(a)

$y_2y_1y_0$ \ X	0	1
000	000,1	100,0
001	000,0	100,0
010	001,0	101,0
011	001,0	101,0
100	010,0	110,1
101	010,0	110,1
111	011,0	111,1
110	011,0	111,1

(b)

Stato \ X		
a	a,1	e,0
b	a,0	e,0
c	b,0	f,0
d	b,0	f,0
e	c,0	g,1
f	c,0	g,1
g	d,0	h,1
h	d,0	h,1

(c)

figura 7.5.2

Come prima cosa e' necessario listare tutte le otto combinazioni delle variabili di stato. In corrispondenza a ciascuna di esse lo stato futuro resta determinato osservando che per ogni

impulso di clock le variabili di stato si spostano di una posizione a destra nella catena di flip-flop D e il posto che si rende disponibile viene occupato dell'ingresso attuale.

Per completare la tavola delle transizioni e' infine necessario assegnare il valore dell'uscita. Nel caso specifico si ottiene:

$$z = x \cdot y_2 + \overline{x} \cdot \overline{y_2} \cdot \overline{y_1} \cdot \overline{y_0}$$

Sebbene il circuito di fig 7.5.2 (a) sia perfettamente in grado di funzionare esso non e' certamente il minimo. Infatti in un solo caso e' necessario memorizzare i tre precedenti valori dell'ingresso. Se infatti $x^{n-1}= 1$, non e' necessario conoscere quanto valevano x^{n-2} e x^{n-3} ; e' quindi evidente che possono essere sufficienti meno di 8 stati interni.

In un circuito con memoria finita dell'ingresso del tipo di quello illustrato in fig. 7.5.2 la memoria agisce in sostanza come un convertitore serie parallelo. Le informazioni di ingresso, necessarie a determinare l'uscita, arrivano cioe' in una sequenza temporale e il circuito semplicemente le memorizza per utilizzarle poi in un unico istante tramite una rete combinatoria.

In conclusione, se si puo' determinare che ad ogni istante l'uscita di un circuito sequenziale dipende da non piu' di un certo numero di ingressi precedenti, allora per la determinazione della tavola delle transizioni si puo' utilizzare l'approccio descritto nell'esempio precedente.

Se la piu' lunga sequenza di ingresso da memorizzare e' lunga n, si dira' allora che la memoria finita dell'ingresso e' lunga n e saranno necessari n elementi di memoria per ciascuna linea di ingresso; nel caso in cui il segnale di ingresso sia unico la memoria potra' allora avere 2^n possibili stati, cioe' potranno essere memorizzate 2^n distinte sequenze di ingresso di lunghezza n. Se l'uscita deve essere diversa per ciascuna di queste 2^n sequenze allora il circuito di fig. 7.5.2 (a) e' il piu' economico e compito del progettista sara' solo quello di determinare la rete di uscita.

Se invece l'uscita in corrispondenza a diverse sequenze di ingresso e' sempre la stessa, allora puo' essere sufficiente un numero inferiore di elementi di memoria.

Nell'esempio 1 si e' presa in considerazione una memoria di lunghezza 3 e le sequenze memorizzabili sono in numero di 8.

Si e' tuttavia messo in evidenza che tutte le sequenze per le quali $x^{n-1}= 1$ producono la stessa uscita; di conseguenza non e' necessario che la memoria distingua le quattro configurazioni con tale caratteristica ed e' quindi possibile ridurre la memoria classificando i messaggi di ingresso all'atto del loro arrivo.

Riducendo in tal modo la memoria, la forma del circuito puo' differire da quella di fig. 7.5.2 nel senso che l'uscita dei flip-flop di memoria non sara' semplicemente uguale ai precedenti ingressi, ma sara' funzione di essi.

La sintesi tuttavia puo' essere eseguita anche a partire dalla tavola di stato ricavata dal circuito base, minimizzando poi il numero di stati necessari.

Quando la forma del circuito deve essere quella di fig. 7.5.2 la tavola delle transizioni, una volta che sia stata fissata la lunghezza della memoria, e' unica. In fig. 7.5.3 e 7.5.4 sono riportate le tavole di transizione per memorie di lunghezza 4, 3 e 2 rispettivamente.

		x^n	
$x^{n-1}x^{n-2}x^{n-3}x^{n-4}$	q^n	0	1
0000	a	a	i
0001	b	a	i
0010	c	b	l
0011	d	b	l
0100	e	c	m
0101	f	c	m
0110	g	d	n
0111	h	d	n
1000	i	e	p
1001	l	e	p
1010	m	f	q
1011	n	f	q
1100	p	g	r
1101	q	g	r
1110	r	h	s
1111	s	h	s

figura 7.5.3

Unico compito del progettista sara' in questo caso assegnare l'appropriato valore delle uscite.

		x^n	
$x^{n-1}x^{n-2}$	q^n	0	1
00	a	a	d
01	b	a	d
11	d	b	c
10	c	b	c

		x^n	
$x^{n-1}x^{n-2}x^{n-3}$	q^n	0	1
000	a	a	e
001	b	a	e
011	c	b	f
010	d	b	f
100	e	d	h
101	f	d	h
110	g	c	g
111	h	c	g

figura 7.5.4

Sebbene i circuiti con memoria finita dell'ingresso siano i piu' facili da riconoscere e da progettare, sono importanti anche i circuiti a memoria finita la cui struttura generale e'

riportata in fig. 7.5.1 (b). Infatti riconoscere il carattere di memoria finita puo' far notevolmente semplificare la procedura di progetto, come e' illustrato nell'esempio che segue.

ESEMPIO 2

Si voglia progettare un contatore bidirezionale a 3 bit (modulo 8). Il conteggio debba apparire in forma di numero binario su tre linee di uscita z_2, z_1 e z_0 con z_2 bit piu' significativo. Il conteggio procedera' con ogni impulso di clock e sara' un conteggio in avanti quando un ingresso x vale 1 mentre sara' all'indietro se x vale 0. Si riconosce immediatamente che il contatore bidirezionale e' un dispositivo a **memoria finita dell'uscita** con una lunghezza di memoria pari a 1. Il valore dell'uscita z e' infatti funzione solo del suo valore immediatamente precedente e del valore attuale dell'ingresso. Poiche' z e' un vettore di tre bit, di componenti z_2,z_1 e z_0, saranno necessari tre flip-flop per rappresentare un suo valore precedente. Una possibile configurazione circuitale e la corrispondente tavola delle transizioni sono riportate in fig. 7.5.5. La tavola delle transizioni e' costruita direttamente notando che il valore attuale desiderato per le uscite corrisponde con il valore futuro delle variabili di stato. E' evidente che per un contatore modulo 8 sono necessari 8 stati e quindi la tavola di fig. 7.5.5 (b) e' minima.

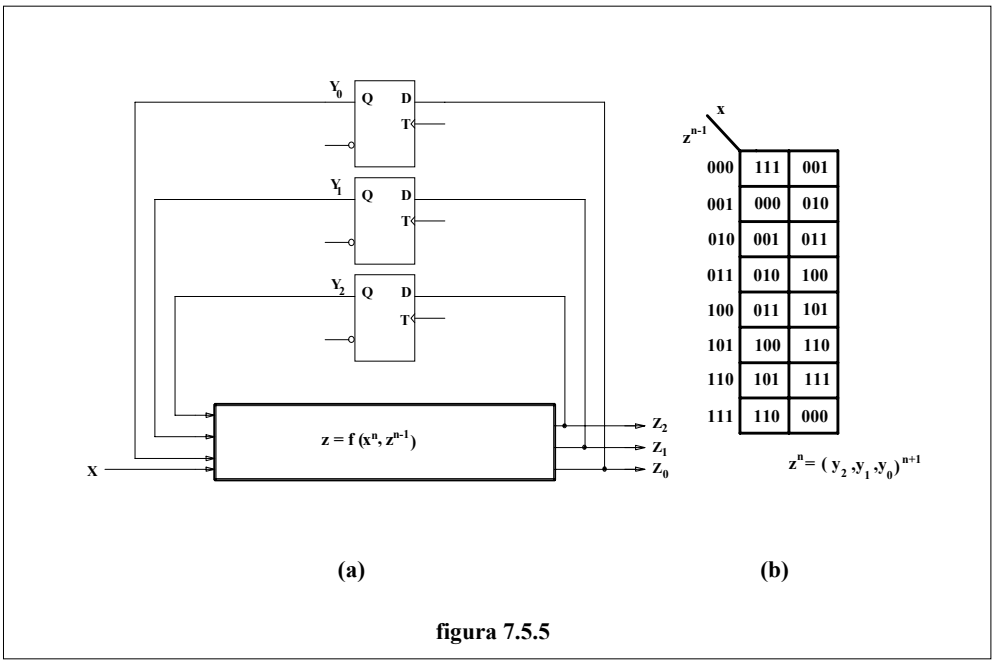
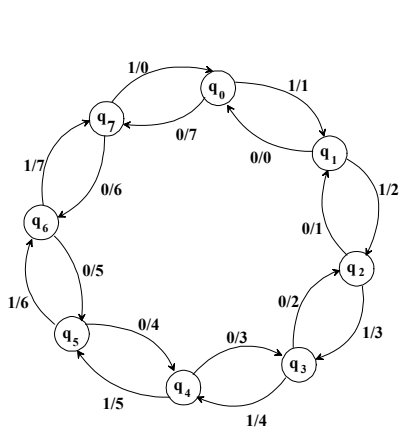


figura 7.5.5

Se la struttura del circuito di fig. 7.5.5 (a) e' soddisfacente si puo' procedere direttamente al progetto dei circuiti combinatori che realizzano l'uscita desiderata. Tuttavia, sebbene la tavola delle transizioni sia minima, la forma del circuito di fig. 7.5.5 (a) non e' la sola possibile.

E' possibile infatti usare differenti tipi di flip-flop e in tal caso non e' necessario che le variabili di stato siano uguali alla piu' recente uscita.

In fig. 7.5.6 sono riportati tavola di stato e diagramma degli stati del circuito di fig. 7.5.5 (a).



(a)

q^n \ x^n	0	1
q_0	q_7 /7	q_1 /1
q_1	q_0 /0	q_2 /2
q_2	q_1 /1	q_3 /3
q_3	q_2 /2	q_4 /4
q_4	q_3 /3	q_5 /5
q_5	q_4 /4	q_6 /6
q_6	q_5 /5	q_7 /7
q_7	q_6 /6	q_0 /0

(b)

figura 7.5.6

Nei due esempi di questo paragrafo e negli esempi dei paragrafi precedenti sono stati descritti due distinti approcci alla sintesi della tavola di stato. Talvolta la massima difficolta', specialmente per chi non abbia ancora accumulato una sufficiente esperienza, sta nel decidere quale metodo usare. Se il problema puo' essere riconosciuto senza equivoci come un caso di memoria finita, allora e' preferibile la sintesi diretta della tabella di stato o di quella delle transizioni.

Al contrario e' migliore l'approccio attraverso il diagramma di stato se si puo' individuare uno stato di reset o comunque uno stato di partenza. Di solito infatti una macchina sequenziale che possieda uno stato di reset e' una macchina a memoria non finita. Per tali macchine di solito si giunge in uno stato in cui si permane indefinitamente oppure si cicla su un sottoinsieme dei suoi stati dopo aver fornito la risposta desiderata. Per rendere la macchina riutilizzabile e' necessario riportarla con un'opportuna sequenza di ingresso nello stato iniziale. Una sequenza fissa che compia tale operazione e' detta ***sequenza di sincronizzazione***, mentre lo stato su cui si ritorna con tale operazione viene detto ***stato di reset***.

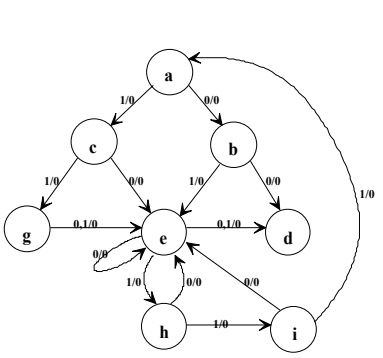
Spesso, come si e' visto negli esempi del paragrafo precedente, la sequenza di sincronizzazione viene realizzata con un singolo ingresso o con una speciale linea detta linea di reset.

ESEMPIO 3

Si riprogetti lo stesso circuito dell'esempio 2 del paragrafo 7.4 con la differenza che ora la linea d di reset non sia presente. Al suo posto l'inizio di un nuovo carattere sia segnalato da una sequenza di tre 1, che puo' iniziare in qualsiasi istante dopo che un carattere di 3 bit sia stato esaminato. Il diagramma di stato fino allo stato g e' lo stesso di quello di fig. 7.4.4 con gli stati e ed f combinati assieme. Si intuisce che lo stato e e' lo stato di tenuta, caratteristico dei circuiti a memoria non finita. L'operazione di reset e' ora realizzata con una sequenza di tre 1 che porta il circuito dallo stato e allo stato a attraverso gli stati h e i. La tavola di stato risultante e' illustrata in fig. 7.5.7 (b).

In alcuni problemi non e' possibile identificare ne' uno stato di reset, ne' una dimensione finita della memoria. In tali casi si puo' scegliere arbitrariamente un qualsiasi stato, la cui storia passata sia ricostruibile al meglio, ed utilizzarlo come stato iniziale. E' tuttavia

necessario porre in tal caso estrema attenzione nel generare la tavola di stato. Solo occasionalmente puo' avvenire di trovarsi in presenza di una macchina a memoria finita dotata di uno stato di reset: in tal caso l'uno o l'altro approccio sono ugualmente validi.



q	x=0	x=1
a	b,0	c,0
b	d,0	e,0
c	e,0	g,0
d	e,1	e,0
e	e,0	h,0
g	e,0	e,1
h	e,0	i,0
i	e,0	a,0

(a) **(b)**

figura 7.5.7

7.6) Minimizzazione degli stati.

Si e' gia' visto al capitolo V che, assegnata una macchina sequenziale *M*, e' sempre possibile determinare una macchina minima equivalente (o compatibile se la macchina di partenza non e' completa) *M'*. In campo circuitale ridurre il numero di stati significa ridurre il numero di elementi di memoria necessari, pervenendo quindi a circuiti piu' semplici e piu' economici.

Stato	x=0	x=1
q_0	$q_0/1$	$q_4/0$
q_1	$q_0/0$	$q_4/0$
q_2	$q_1/0$	$q_5/0$
q_3	$q_1/0$	$q_5/0$
q_4	$q_2/0$	$q_6/1$
q_5	$q_2/0$	$q_6/1$
q_6	$q_3/0$	$q_7/1$
q_7	$q_3/0$	$q_7/1$

figura 7.6.1

E' opportuno tuttavia notare che non sempre la riduzione del numero di stati porta ad una riduzione degli elementi di memoria; in effetti, cio' che conta a tal fine e' che venga ridotto il numero delle variabili necessarie a codificare gli stati.

Indicando con s il numero degli stati, il numero delle variabili usato per la codifica e' il piu' piccolo intero che approssima per eccesso il $\log_2$ s.

Si voglia ad esempio minimizzare la tavola degli stati ricavata per l'esempio 1 del paragrafo 7.5, e riportata in fig. 7.6.1.

La tabella di evoluzione delle coppie di stati compatibili rispetto all'uscita e' riportata in fig. 7.6.2.

Coppie α - compatibili \ Ingresso	0	1
q_1,q_2	q_0,q_1	q_4,q_5
q_1,q_3	q_0,q_1	q_4,q_5
q_2,q_3	q_1,q_1	q_5,q_5
q_4,q_5	q_2,q_2	q_6,q_6
q_4,q_6	q_2,q_3	q_6,q_7
q_4,q_7	q_2,q_3	q_6,q_7
q_5,q_6	q_2,q_3	q_6,q_7
q_5,q_7	q_2,q_3	q_6,q_7
q_6,q_7	q_3,q_3	q_7,q_7

figura 7.6.2

Le coppie q_1, q_2 e q_1, q_3 rimangono escluse in quanto non evolvono verso coppie compatibili rispetto all'uscita. Tra le coppie di stati compatibili puo' poi essere messa in luce una relazione di mutua compatibilita' tra gli stati q_4, q_5, q_6 e q_7.

Le classi di stati equivalenti sono quindi:

$$S_0 = \{q_0\} \quad S_1 = \{q_1\} \quad S_2 = \{q_2,q_3\}$$
$$S_3 = \{q_4,q_5,q_6,q_7\}$$

Pertanto a partire dalla macchina di fig. 7.6.3 (a), in cui sono evidenziate le classi di stati equivalenti, si perviene a quella minima di fig. 7.6.3 (b).

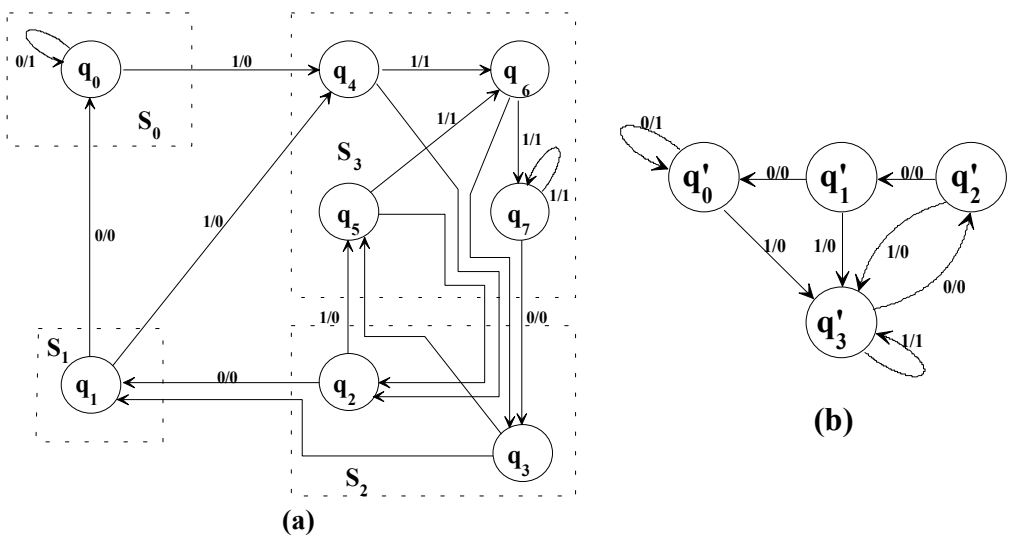


figura 7.6.3

7.7) Codificazione e determinazione delle equazioni di eccitazione.

Dopo che con il metodo di Ginsburg sia stata ottenuta la tavola di stato minima, rimane da realizzare il circuito il cui funzionamento e' descritto da tale tavola. E' bene dire subito che la ricerca della massima economia non si esaurisce con la minimizzazione del numero degli stati. E' infatti vero che il numero degli elementi di memoria diminuisce con il ridursi del numero di stati, ma e' anche vero che la massima economia si ottiene con uno sviluppo opportuno della rete combinatoria che sintetizza l'uscita e le variabili di eccitazione degli elementi di memoria. Puo' pertanto verificarsi il caso in cui una tavola di stato diversa da quella minima richieda una logica combinatoria meno complessa e quindi nel totale porti a una realizzazione piu' economica.

Se la sintesi della tavola di stato e' stata fatta con la tecnica dei sistemi a memoria finita e la riduzione del numero di stati non ha portato all'eliminazione di nessun elemento di memoria, allora la struttura del circuito sara' quella di fig. 7.5.1 (b). Poiche' secondo tale struttura gli ingressi e le uscite vengono semplicemente memorizzati, non vi e' alcuna necessita' di una rete combinatoria per realizzare lo stato futuro. Altre strutture circuitali potrebbero dare logiche di uscita piu' semplici, ma non e' affatto garantito che il circuito nel suo complesso risulti piu' economico.

Se la struttura a memoria finita non risulta appropriata, il problema di scegliere la migliore realizzazione e' sfortunatamente molto difficile da risolvere.

Si ricordi infatti che, se r sono le variabili utilizzate per codificare lo stato, allora ogni stato corrisponde a una delle 2^r combinazioni delle variabili. Il primo problema e' quindi quello di decidere quale delle 2^r combinazioni debba esser fatta corrispondere a ciascun stato.

Se il numero di stati m e' tale che:

$$2^{r-1} < m \leq 2^r$$

allora sono necessarie r variabili di stato e vi sono

$$\frac{2^r!}{(2^r - m)}$$

modi di codificare gli m stati.

Ad esempio, il problema di assegnare le combinazioni delle variabili di stato a ciascuno di quattro stati puo' essere visualizzato assegnando i quattro stati ai quattro vertici di un quadrato. In fig. 7.7.1 sono illustrate 3 di queste assegnazioni. Ogni altra possibile assegnazione puo' essere ottenuta come rotazione e/o specularita' di uno dei tre modi fondamentali di fig. 7.7.1 e corrisponde a variare l'ordine delle variabili e negarle. Tali operazioni non hanno effetto per quanto riguarda la complessita' del circuito.

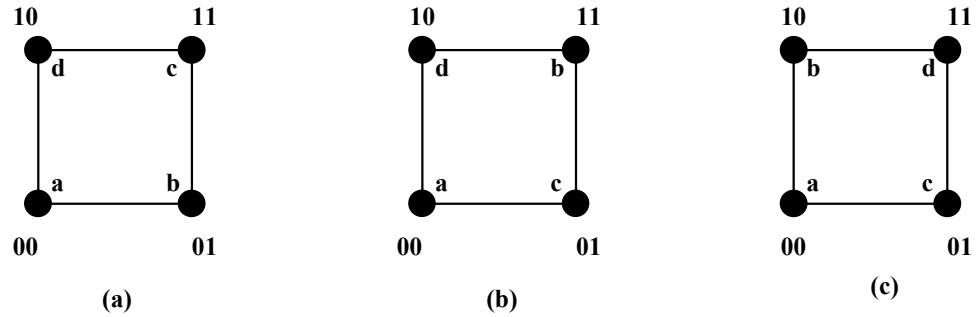


figura 7.7.1

In maniera analoga si puo' ragionare per qualsiasi numero di stati; il numero delle diverse assegnazioni distinte in funzione del numero di stati e' riportato in fig. 7.7.2.

Numero di stati	Numero di variabili	Assegnazione distinte
2	1	1
3	2	3
4	2	3
5	3	140
6	3	420
7	3	840
8	3	840
9	4	10.810.800

figura 7.7.2

Per i circuiti con soli due stati vi e' una sola possibilita' di scelta e non vi e' quindi alcun problema di assegnazione. Per circuiti con tre o quattro stati la via piu' efficace per risolvere il

problema e' quella di provare tutte le tre possibili assegnazioni e verificare quale di esse produce il circuito piu' economico.

Per piu' di quattro stati l'enumerazione completa e' ovviamente impossibile; si rende necessario un qualche metodo di sviluppo che porti ad una buona assegnazione a partire dalla tavola di stato.

Si consideri la tavola di stato di fig. 7.7.3, che e' la tavola minima per il circuito rivelatore di inizio messaggio dell'esempio 1 del paragrafo 7.4, dove con a e' stato indicato l'insieme degli stati $\{q_0,q_4\}$ e con b, c, d gli stati q_2, q_1 e q_3 rispettivamente.

In fig. 7.7.4 (a),(b),(c) sono riportate le tavole di flusso che si ottengono con le tre possibili assegnazioni di fig. 7.7.1. Poiche' vi e' una sola uscita, l'assegnazione della codifica non modifica in alcun modo il costo della rete di uscita e pertanto la parte della tavola di flusso relativa a quest'ultima grandezza viene omessa per motivi di chiarezza.

Stato \ Ingresso	0	1
a	a/0	c/0
b	a/0	d/1
c	a/0	b/0
d	d/0	d/0

figura 7.7.3

Il passo successivo della sintesi consiste nel tradurre la tavola di flusso nelle mappe di Karnaugh, o meglio nelle mappe di eccitazione, che rappresentano le equazioni di ingresso degli elementi di memoria.

stato attuale	Ingresso 0	Ingresso 1
a 00	00	11
b 01	00	10
c 11	00	01
d 10	10	10

(a)

stato attuale	Ingresso 0	Ingresso 1
a 00	00	01
b 11	00	10
c 01	00	11
d 10	10	10

(b)

stato attuale	Ingresso 0	Ingresso 1
a 00	00	01
b 10	00	11
c 01	00	10
d 11	11	11

(c)

figura 7.7.4

Al capitolo V sono state sviluppate le mappe che rappresentano lo stato futuro in funzione dello stato presente e degli ingressi per i piu' comuni tipi di flip-flop. Agli scopi che

ci si propone in questo capitolo e' opportuno modificare leggermente la forma in cui sono presentate queste informazioni.

Se, ad esempio, si considera un flip-flop JK la mappa ricavata in precedenza e' riportata in fig. 7.7.5 (a) mentre in fig. 7.7.5 (b) si ha quella che viene chiamata lista delle transizioni ed in cui viene assegnata la condizione degli ingressi necessaria ad ottenere una tra le quattro possibili transizioni di stato. Si noti che con i valori soprassegnati si intende la transizione dell'uscita verso il relativo valore, mentre i valori senza soprassegno indicano la permanenza dell'uscita al valore precedente.

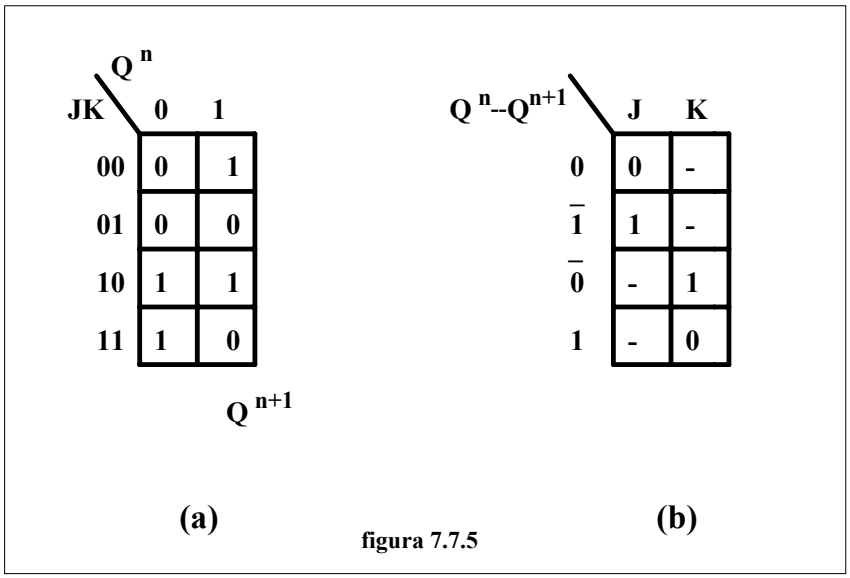


figura 7.7.5

Le liste di transizione per i flip-flop RS, D e T sono riportate in fig. 7.7.6.

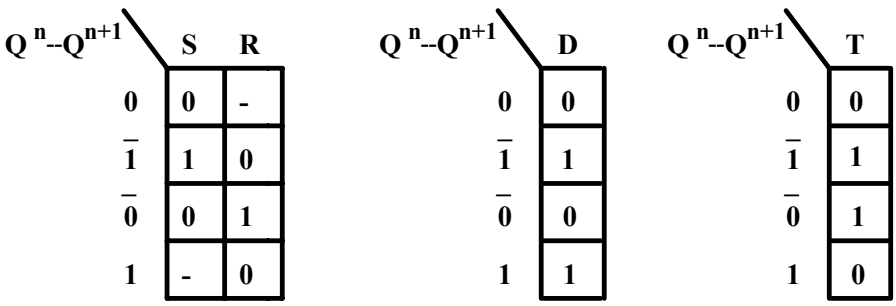


figura 7.7.6

Per sviluppare le mappe di eccitazione per l'assegnazione di fig. 7.7.4 (a) e' necessario considerare una variabile di stato alla volta e costruire le due mappe di eccitazione per J e K utilizzando la lista di transizione di fig. 7.7.5 (b). Si ottiene:

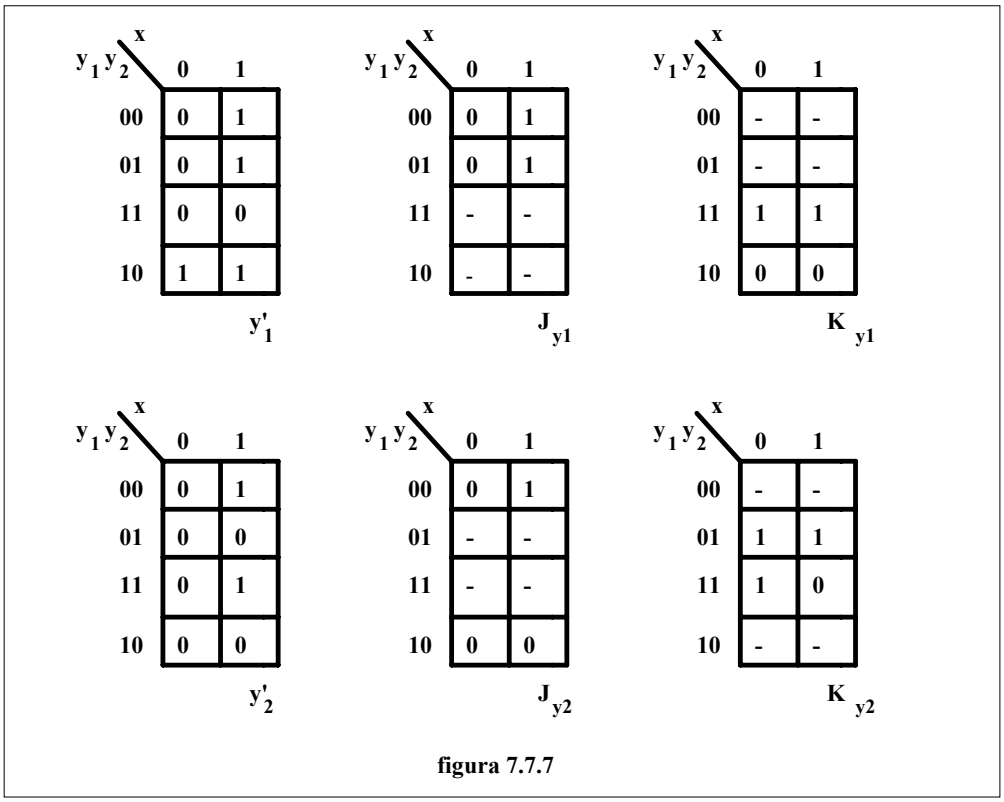


figura 7.7.7

Da tali mappe si puo' ottenere la realizzazione minima, facendo notare che il problema va affrontato come problema di minimizzazione di un circuito multiterminale. Le equazioni che si ricavano sono:

$$J_{y1} = x \qquad K_{y1} = y_2 \qquad J_{y2} = x.\overline{y_1} \qquad K_{y2} = \overline{x} + \overline{y_1}$$
$$z = x.\overline{y_1}.y_2$$

Procedendo analogamente per le assegnazioni di fig. 7.7.4 (b) e (c) si ottiene:

$$J_{y1} = x.y_2 \qquad K_{y1} = \overline{x}.y_2 \qquad J_{y2} = x.\overline{y_1} \qquad K_{y2} = \overline{x} + y_1$$
$$z = x.y_1.y_2$$

e rispettivamente

$$J_{y1} = x.y_2 \qquad K_{y1} = \overline{x}.\overline{y_2} \qquad J_{y2} = x \qquad K_{y2} = \overline{y_1}$$
$$z = x.y_1.\overline{y_2}$$

La prima e la terza assegnazione sono approssimativamente equivalenti; la terza assegnazione e' eventualmente preferibile in quanto usa solo dei gate AND mentre la prima usa anche un gate OR. Il relativo circuito e' riportato in fig. 7.7.8.

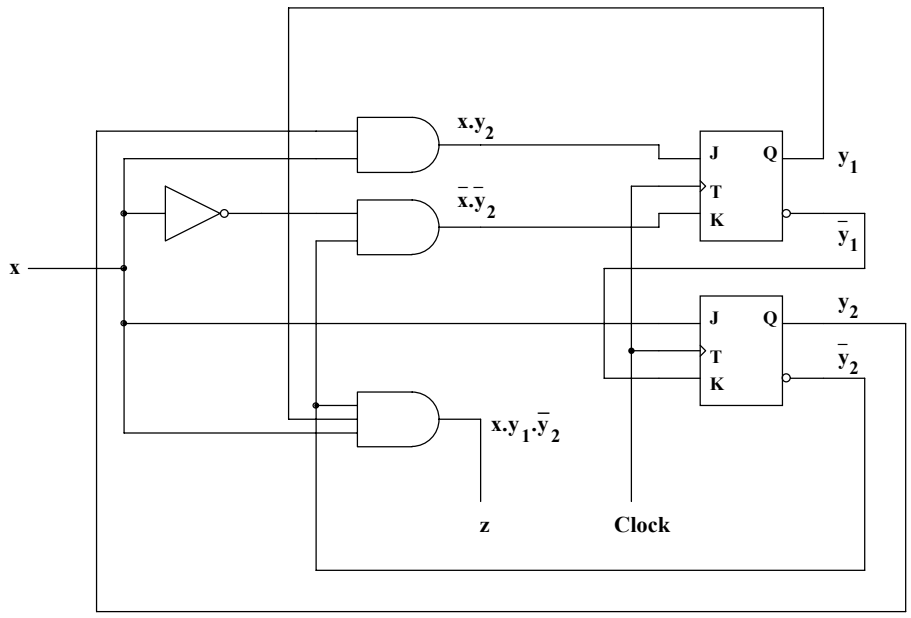


figura 7.7.8

A scopo di maggior chiarezza si consideri ancora il seguente esempio.

Si voglia realizzare il circuito relativo all'esempio 1 del paragrafo 7.5 utilizzando flip-flop RS.

La tavola di stato minima e' riportata in fig. 7.7.9 (a) mentre in fig. 7.7.9.(b) si ha la tavola di flusso relativa all'assegnazione 7.7.4 (b). Gli stati a, b, c, d corrispondono rispettivamente agli stati {a}, {c,d}, {e,f,g,h} e {b} della tavola di stato di fig. 7.5.2 (b).

Stato \ Ingresso	0	1
a	a/1	c/0
b	d/0	c/0
c	b/0	c/1
d	a/0	c/0

(a)

Stato	Ingresso		
a	00	00/1	01/0
c	01	11/0	01/1
b	11	10/0	01/0
d	10	00/0	01/0

(b)

figura 7.7.9

Le relative mappe di eccitazione dei due flip-flop RS sono:

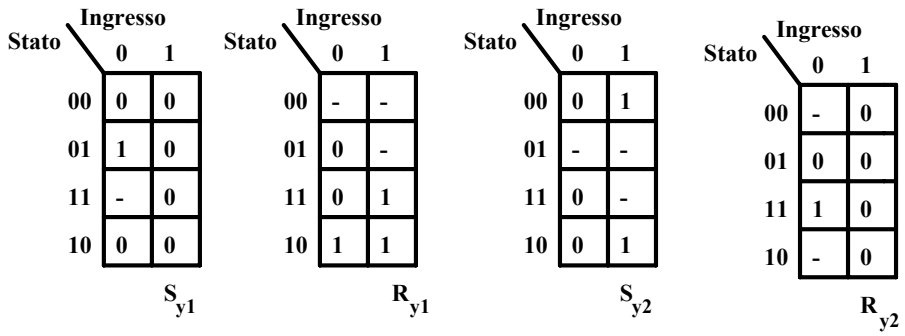


figura 7.7.10

Le equazioni di eccitazione sono:

$$s_{y1} = \bar{x}.y_2 \qquad R_{y1} = x + \overline{y_2} \qquad S_{y2} = x \qquad R_{y2} = \bar{x}.y_1$$
$$z = \bar{x}.\overline{y_1}.\overline{y_2} + x.\overline{y_1}.y_2$$

Le assegnazioni 7.7.4 (a) e (c) danno luogo rispettivamente alle equazioni:

$$S_{y1} = x + \overline{y_1}.y_2 \qquad R_{y1} = \bar{x}.y_1 \qquad S_{y2} = x \qquad R_{y2} = \bar{x}.\overline{y_1}$$
$$z = \bar{x}.\overline{y_1}.\overline{y_2} + x.y_1.y_2$$

e rispettivamente

$$S_{y1} = x.\overline{y_1}.y_2 \qquad R_{y1} = x + y_1.y_2 \qquad S_{y2} = x + y_1.\overline{y_2} \qquad R_{y2} = \bar{x}.y_2$$
$$z = \bar{x}.\overline{y_1}.\overline{y_2} + x.y_1.y_2$$

L'assegnazione 7.7.1 (b) da' evidentemente luogo alle equazioni piu' semplici a partire dalle quali e' immediato disegnare il circuito.

Avendo a che fare con circuiti a tre o quattro stati la procedura esposta, per quanto complessa e noiosa, e' quella che garantisce il miglior risultato. Tuttavia, quando gli stati siano cinque o piu', non e' evidentemente possibile provare tutte le assegnazioni che si possono realizzare. E' necessario pertanto individuare alcune regole e mettere a punto dei modi operativi che permettano di ottenere con uno sforzo ragionevolmente limitato un buon risultato. Per individuare tali regole e' opportuno esaminare piu' attentamente gli esempi precedenti allo scopo di vedere se vi sia qualche caratteristica che permetta di prevedere in anticipo quale assegnazione conduce alla realizzazione minima.

In fig. 7.7.11 sono riportate le tavole di stato per le tre assegnazioni possibili in riferimento agli ultimi due esempi visti, con gli stati ordinati secondo il codice ciclico della mappa di Karnaugh. L'assegnazione che da' luogo al risultato migliore e' evidenziata circondandola con una linea tratteggiata. Dall'osservazione delle due mappe racchiuse nel tratteggio si deduce che l'assegnazione scelta tende a riunire gli ingressi relativi a stati futuri uguali.

Ad esempio per il caso di fig. 7.7.11 (a) si puo' notare che lo stato futuro in corrispondenza all'ingresso x = 1 per gli stati attuali b e d e' lo stato d. Ne risulta che il valore delle variabili di stato per lo stato futuro sara' identico.

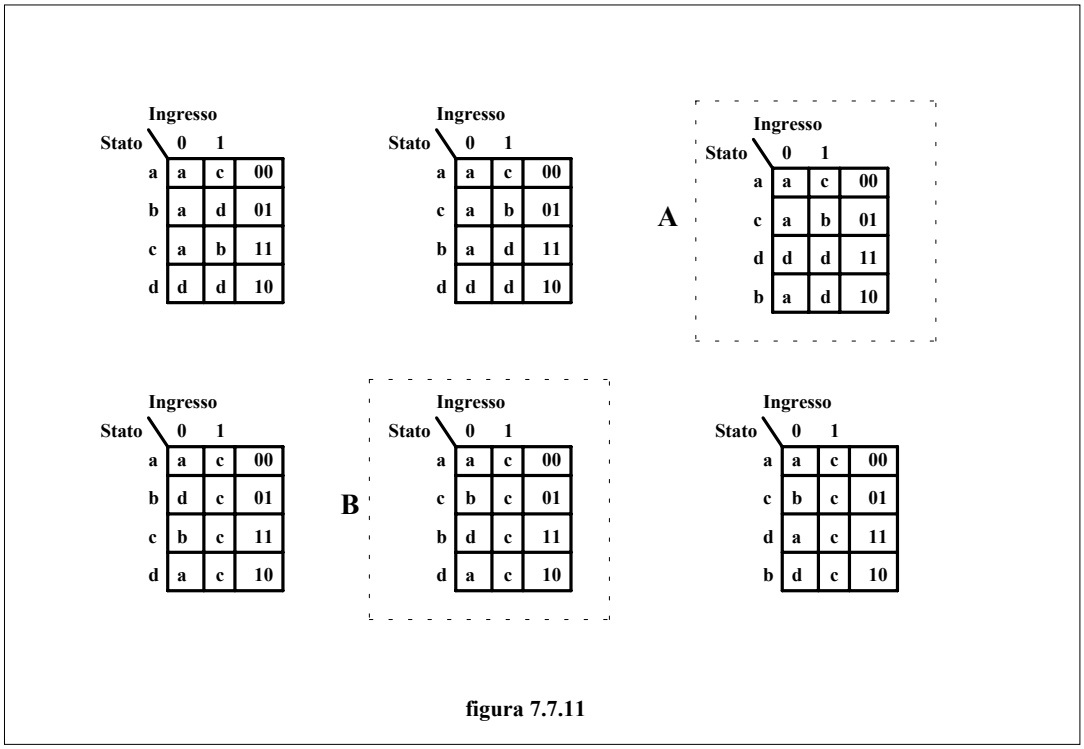


figura 7.7.11

Se quindi gli stati b e d vengono codificati con configurazioni che distano 1, allora il valore presente di una delle variabili di stato sara' identico per ambedue gli stati.

Di conseguenza la commutazione per tale variabile di stato sara' la stessa per ambedue i punti di ingresso e anche sulla tavola di eccitazione si avra' la medesima situazione.

Le condizioni messe in luce sono importanti in quanto la mappa di eccitazione dara' luogo a equazioni piu' semplici se gli 1, gli 0 e le condizioni non specificate saranno, per quanto possibile, raggruppati tra loro.

Nel caso che si sta esaminando, l'assegnazione prescelta attribuisce valore identico alla variabile y_1 per gli stati b e d, e pertanto le corrispondenti transizioni di y_1 per x=1 sono identiche. Il risultato si vede chiaramente nelle mappe di eccitazione di fig. 7.7.7 ; infatti in quella J_{y1} si hanno due condizioni non specificate, mentre in quella K_{y1} si hanno due zeri adiacenti.

Regola base per ottenere un buon risultato sara' pertanto quello di raggruppare, per quanto possibile, gli ingressi relativi a stati futuri uguali. Vi sono tuttavia molti modi per ottenere questi raggruppamenti e la regola base teste' enunciata puo' essere espansa e resa piu' specifica.

Si noti che nella tavola di stato di fig. 7.7.11 (b) gli stati a e d evolvono verso gli stessi stati per ambedue i valori dell'ingresso. Tale condizione verosimilmente produce una mappa piu' semplice di quella che si ottiene quando il prossimo stato e' lo stesso in una sola colonna, come nel caso che si e' appena preso in considerazione. Inoltre, se i due stati futuri a e c vengono resi adiacenti, allora una delle due variabili di stato sara' la stessa in ambedue le

colonne della stessa riga e dara' quindi luogo con elevata probabilita' ad un'ulteriore semplificazione.

La situazione descritta puo' essere individuata sulla mappa di eccitazione di fig. 7.7.10 in cui si hanno quattro zeri adiacenti nella mappa S_{y1} e un gruppo di 2 uni e 2 condizioni non specificate nella mappa R_{y1}.

Si consideri infine la tavola di stato di fig. 7.7.12.

In questa tabella gli stati a e c sono ancora gli stati futuri di a e d, ma non compaiono nella stessa colonna. Nondimeno, se si rendono a e c adiacenti, una variabile di stato sara' la stessa in ambedue le colonne di una data riga. Se inoltre si rendono anche a e d adiacenti, allora queste righe risulteranno accoppiate.

Stato \ Ingresso	0	1
a	c	a
b	d	c
c	b	a
d	a	c

figura 7.7.12

Da un esame delle tre possibili assegnazioni si vede che l'assegnazione di fig. 7.7.1 (b) in cui gli stati a e c e gli stati a e d sono adiacenti, da' luogo al miglior circuito che si possa ottenere per la tavola di stato di fig. 7.7.12.

Dalle osservazioni fin qui fatte si possono estrapolare le seguenti regole, utili ad ottenere una efficiente codifica di stato.

REGOLA 1

1) Si individuino sulla tavola di stato gli stati attuali che hanno in ciascuna colonna lo stesso stato futuro. Gli stati che soddisfano questa condizione devono essere resi adiacenti.
2) Si individuino sulla tavola di stato quegli stati che evolvono verso gli stessi stati futuri, ma con differente ordine delle colonne. Questi stati devono essere resi adiacenti se anche gli stati futuri verso cui evolvono possono essere resi adiacenti.
3) Gli stati relativi a righe che hanno lo stesso stato futuro per alcune, ma non per tutte le colonne, vanno resi adiacenti, dando la precedenza a quelle righe che godono di questa proprieta' per un maggior numero di colonne.

REGOLA 2

Gli stati futuri di una determinata riga devono essere resi adiacenti.

REGOLA 3

L'assegnazione prescelta deve semplificare per quanto possibile la mappa della funzione d'uscita.

E' importante notare che le tre regole enunciate sono in ordine di importanza decrescente. La loro applicazione puo' infatti portare a specifiche in conflitto tra di loro; tuttavia un' assegnazione adottata per soddisfare a una regola a priorita' piu' alta non dev'essere modificata per soddisfare una regola a priorita' minore.

Un'altra procedura operativa per la determinazione di un'assegnazione possibile e che tiene, almeno in parte, conto delle esigenze che sono state appena messe in luce, e' quella che fa ricorso alla cosidetta ***tavola inversa di flusso***. E' questa una tabella che considera per ogni stato e per ogni ingresso al tempo t^n , l'insieme degli stati in cui si trovava il circuito al tempo t^{n-1}.

Si consideri ad esempio la macchina sequenziale di fig. 7.7.13

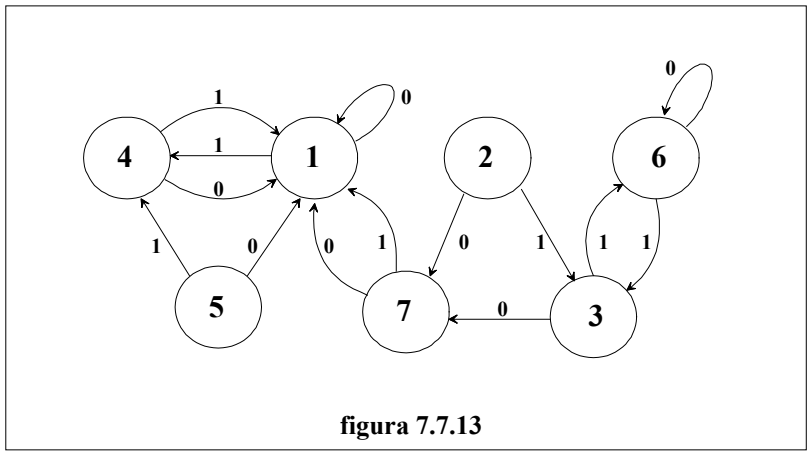


figura 7.7.13

La relativa tabella inversa di flusso e' riportata in fig. 7.7.14. Nell'ultima colonna e' conteggiato il numero di stati da cui si perviene allo stato attuale. Lo stato per cui tale conteggio e' massimo e' lo stato 1 con sei termini. Conviene pertanto codificare lo stato 1 con $y_1y_2y_3$= 000, in modo che nella tavola di flusso ci sia il massimo numero di zeri.

Le adiacenze vanno poi ricercate per righe e per colonne; le adiacenze per righe tengono conto delle transizioni verso lo stesso stato e si riconoscono in quanto formate da stati che stanno nella stessa casella (esclusi quelli della prima riga). Si individua in tal modo la necessita' di adiacenza tra gli stati 2,6 1,5 e 2,3.

La ricerca per colonne si fa a partire dalle coppie cosi' individuate e considerando la presenza o meno degli stati di tali coppie in ciascuna colonna. In tal modo la coppia 2,6 determina nella colonna x=0 l'esigenza di adiacenza per gli stati 6 e 7; la coppia 2,3 l'adiacenza degli stati 3,6 nella colonna x=1.

In definitiva e' quindi conveniente che siano adiacenti le coppie:

(2,3) (2,6) (1,5) (3,6) (6,7)

Stato attuale	Stato precedente		n. termini
	x=0	x=1	
1	1,4,5,7	4,7	6
2	--	--	0
3	--	2,6	2
4	--	1,5	2
5	--	--	0
6	6	3	2
7	2,3	--	2

figura 7.7.14

Queste condizioni non sono contemporaneamente soddisfacibili con tre variabili di stato. Una buona soluzione e':

y_3 \ y_1y_2	00	01	11	10
0	1	5	-	4
1	2	6	7	3

Nel problema che si e' preso in considerazione non era assegnata l'uscita; pertanto le coordinate della mappa possono essere assegnate in modo qualsiasi, purche' secondo un codice ciclico. Se viceversa l'uscita fosse stata specificata la scelta del codice di taratura doveva essere tale da minimizzare la rete di uscita.

Si applichi ora quanto esposto al seguente esempio.

ESEMPIO 1

Si voglia progettare un circuito che verifichi la parita' su caratteri binari da quattro bit, ricevuti in modo seriale su una linea x, a partire da uno stato di reset.

All'istante in cui viene ricevuto il quarto bit l'uscita deve essere 1 se e solo se il numero totale di 1 nel carattere e' pari. In tutti gli altri istanti l'uscita deve essere 0 e dopo la ricezione del quarto bit il circuito deve ritornare nello stato di reset pronto a ricevere il successivo carattere.

E' evidente che a ciascun istante le uniche informazioni importanti consistono in quanti bit siano stati ricevuti e se il numero totale di 1 sia pari o dispari. Queste considerazioni conducono direttamente al diagramma di stati di fig. 7.7.15, che e' anche il diagramma della macchina minima.

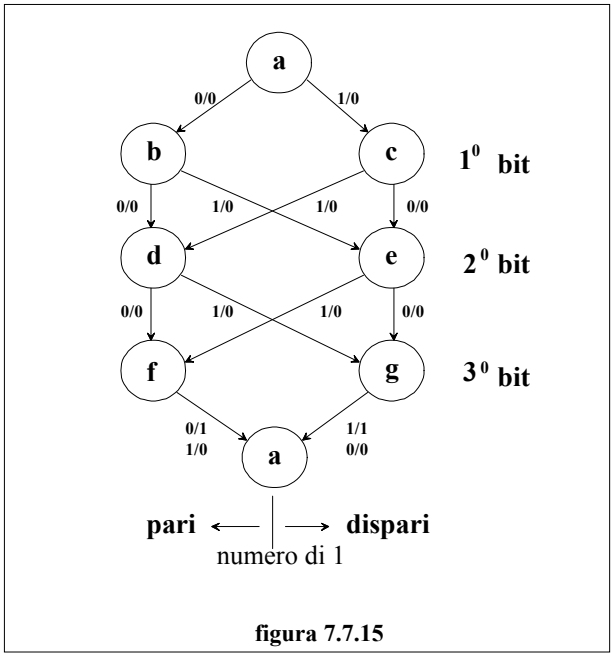


figura 7.7.15

La corrispondente tavola degli stati e':

stato \ ingresso	0	1
a	b/0	c/0
b	d/0	e/0
c	e/0	d/0
d	f/0	g/0
e	g/0	f/0
f	a/1	a/0
g	a/0	a/1

Applicando la regola 1(1) enunciata in precedenza, gli stati f e g devono essere resi adiacenti. La regola 1 (2) a sua volta richiede che gli stati d ed e siano resi adiacenti se e' possibile rendere adiacenti f e g, e che b e c siano adiacenti se adiacenti sono d ed e. La codifica dello stato a e' apparentemente arbitraria e , a meno che non vi siano fondati motivi per evitarlo, lo si codifichera' con 000. Tale scelta e' tanto piu' giustificata in quanto lo stato a e' lo stato di reset e con una codifica nulla esso puo' venir raggiunto molto facilmente sem-

plicemente azzerando tutti i flip-flop della memoria. Una buona codifica e la relativa tavola di flusso sono allora le seguenti:

y_3 \ y_1y_2	00	01	11	10
0	a	f	g	-
1	b	d	e	c

$y_1y_2y_3$		x = 0	x = 1
a	000	001/0	101/0
b	001	011/0	111/0
d	011	010/0	110/0
f	010	000/1	000/0
	100	--/-	--/-
c	101	111/0	011/0
e	111	110/0	010/0
g	110	000/0	000/1

Le equazioni finali che si ricavano sono, nell'ipotesi di utilizzare flip-flop di tipo JK:

$$J_{y1} = x.\overline{y_2} + x.y_3 \qquad K_{y1} = x + \overline{y_3} \qquad J_{y2} = y_3 \qquad K_{y2} = \overline{y_3}$$

$$J_{y3} = \overline{y_2} \qquad K_{y3} = y_2 \qquad z = x.y_1.\overline{y_3} + \overline{x}.\overline{y_1}.y_2.y_3$$

Il metodo della tabella inversa di stato in questo caso non porta invece a risultati significativi.

E' importante osservare a questo punto che le regole e i procedimenti esposti non sono procedimenti algoritmici completi atti ad individuare la migliore tra le possibili codificazioni. Essi tutto al piu' permettono di ridurre il numero di possibili alternative, consentendo di ottenere una buona soluzione in un numero ridotto di tentativi.

Si consideri infatti di nuovo la codifica della tavola di stato di fig. 7.7.3. La regola 1(1) impone che gli stati b e d siano resi adiacenti. Ora sia la codifica racchiusa nel tratteggio di fig. 7.7.11 (A) che quella immediatamente precedente soddisfano tale esigenza. La regola 1(1) quindi non individua la codifica migliore, ma permette tuttalpiu' di scartare la prima delle tre codifiche possibili.

A maggior conferma di quanto detto si consideri la tavola di stato di fig. 7.7.9.

L'applicazione della regola 1(1) porta a scegliere la codifica di fig. 7.7.11 (B), che produce il circuito migliore quando i flip-flop usati sono di tipo RS. Se viceversa si usassero dei flip-flop JK la codifica migliore sarebbe la prima tra quelle di fig. 7.7.11, in quanto con i flip-flop JK meta' delle condizioni di ingresso sono condizioni non specificate,indipendentemente da quale sia la codifica dello stato. C'e' tuttavia da osservare che in questo caso la codifica di fig. 7.7.11 (B) e' peggiore per un solo gate; pertanto la codifica individuata attraverso l'applicazione delle regole enunciate e' comunque accettabile.

7.8) Partizione degli stati e codificazione.

Nelle procedure di determinazione di una macchina sequenziale minima e' stato introdotto il concetto di partizione dell'insieme degli stati in classi di equivalenza. Due stati

appartenevano alla stessa classe (p = q) se per qualsiasi ingresso x erano soddisfatte le condizioni:

$$w(p,x) = w(q,x)$$
$$S(p,x) = S(q,x)$$

Si considerino ora le partizioni in classi di equivalenza che soddisfano solamente la seconda di queste due condizioni. Due stati cioe' appartengono alla stessa classe di equivalenza se, per ciascun ingresso, gli stati verso cui evolvono appartengono ancora ad una stessa classe di equivalenza.

Si consideri ad esempio la tavola degli stati di fig. 7.8.1.:

stato \ ingresso	0	1
0	7	1
1	0	2
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	0

figura 7.8.1

Le partizioni {0,2,4,6}, {1,3,5,7}e {0,4}, {1,5}, {2,6}, {3,7} soddisfano la condizione enunciata.

Infatti per gli stati {0,4} gli stati futuri sono {3,7} per l'ingresso x=0 e {1,5} per x=1; per la coppia {1,5} sono rispettivamente {0,4} e {2,6} per x=0 e x=1 e cosi' via. Per la partizione {0,2,4,6} gli stati futuri sono {1,3,5,7} per ambedue gli ingressi e viceversa. Quindi per tutti gli stati di una determinata classe di equivalenza gli stati futuri, per qualsiasi ingresso, si trovano tutti nella medesima classe di equivalenza.

Le partizioni che godono di questa proprieta' vengono dette **partizioni chiuse**.

Per illustrare in che modo partizioni di questo tipo vengano usate nella codifica dello stato si assegni una delle variabili di stato ai due blocchi {0,2,4,6} e {1,3,5,7}, ad esempio $y_1 = 0$ al primo e $y_1 = 1$ al secondo. Poiche' gli stati attuali evolveranno sempre verso stati dell'altro blocco, l'equazione di stato per y_1 sara':

$$y_1^{n+1} = \overline{y_1^n}$$

In effetti assegnare le variabili di stato induce sempre delle partizioni degli stati in gruppi che corrispondono alle varie combinazioni di valori delle variabili. Se si possono assegnare le variabili di stato in modo da indurre partizioni chiuse, si ottengono generalmente

delle equazioni piu' semplici di quelle che si possono ottenere con una codifica completamente casuale.

Se una partizione e' chiusa allora il blocco che contiene lo stato futuro e' individuato univocamente dal blocco che contiene lo stato presente ed eventualmente dal valore dell'ingresso. Poiche' vi sono meno blocchi che non stati, essi sono individuati da un sottoinsieme delle variabili di stato. Il risultato e' quello di ridurre il numero di variabili nelle equazioni di stato, dando luogo quindi a equazioni, che in generale sono piu' semplici.

Realizzare partizioni chiuse non permette solamente di ottenere delle codifiche vantaggiose, ma influisce anche in modo diretto sulla struttura del circuito sequenziale. Nell'esempio che si e' appena esaminato, la variabile y_1 e' funzione solamente di se stessa e quindi la parte di circuito relativa non richiede altri ingressi. In generale le partizioni chiuse danno luogo ad una separazione del circuito completo in un certo numero di sottocircuiti, ciascuno adibito all'elaborazione di un certo numero di variabili del sistema e dipendente a sua volta da un numero limitato di variabili.

la fig. 7.8.2 mostra in che modo un circuito con quattro variabili di stato possa esser suddiviso in un certo numero di sottocircuiti.

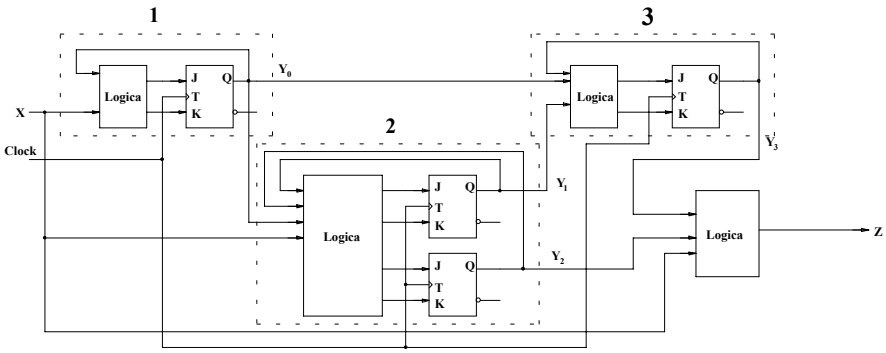


figura 7.8.2

Si noti che ciascun sottocircuito ha come ingressi alcune delle variabili di stato, ma non tutte. Ad esempio il flip-flop del sottocircuito 1 ha come ingressi la sua stessa uscita e l'ingresso effettivo x. In modo analogo il sottocircuito 2 ha come ingressi x, y_0 e la sua stessa uscita.

In generale una buona partizione degli stati di un circuito sequenziale corrisponde ad una suddivisione del circuito in un certo numero di sottocircuiti collegati da un numero di interconnessioni molto limitato.

Molte ricerche [*] sono state condotte relativamente al problema dell'individuazione delle partizioni piu' vantaggiose, molte delle quali indirizzate a formalizzare procedure di ricerca di partizioni chiuse o di altri tipi di partizione con proprieta' simili.

---

[*]

HARTMANIS J. - STEARNS R.E. *" Algebraic structure theory of sequential Machines "*
PRENTICE-HALL Englewood Cliffs- N.J 1966
HARTMANIS J. *" On the state assignement Problem for sequential machines "*
IRE trans. on Electronic Computers, EC-10:2,157- 165 (giugno 1961)
HARTMANIS J. - STEARNS R.E. *" On the state assignement problem for sequential machines "*
IRE Trans. on Electronic Computers EC-10:4,593-603 (dicembre 1961)

Sfortunatamente tali procedure sono molto complesse e noiose da utilizzare. Per tale ragione anziche' cercare metodi formali di partizione degli stati e' piu' opportuno individuare quelle partizioni degli stati che possono essere suggerite dalla formulazione originale del problema. Spesso i sottocircuiti realizzano delle sottofunzioni chiaramente individuabili, come conteggi, funzioni di scorrimento, riconoscimento di parita', ecc. Se queste funzioni vengono riconosciute gia' durante la prima fase di approccio al problema, la codificazione degli stati puo' essere notevolmente semplificata.

ESEMPIO 1

Si riprenda l'esempio 1 del paragrafo 7.7, relativo al circuito in grado di controllare la parita' di parola di 4 bit trasmesse serialmente su una linea x. Dimenticando che il diagramma degli stati e' gia' stato determinato in precedenza, si tenti ora di affrontare il problema cercando di determinare eventuali sottofunzioni.

Ovviamente una funzione da soddisfare e' la memorizzazione della parita' all'atto dell'arrivo di ciascun bit. Per soddisfare tale esigenza e' sufficiente un unico flip-flop.

Poiche' poi all'arrivo del quarto bit il circuito deve comportarsi in maniera diversa da quanto avviene per i tre bit precedenti, e' inoltre necessario memorizzare qual'e' numero di bit giunti e tale funzione puo' essere assolta da un semplice contatore a quattro stati.

Le due sottofunzioni messe in luce sono illustrate mediante i relativi diagrammi di stato in fig. 7.8.3

Il diagramma di fig. 7.8.3 (b), in cui l'uscita e' sempre 0, e' valido per i primi tre bit; il quarto bit riporta il circuito allo stato di partenza (corrispondente ad un numero di bit uguali a 1 pari) senza che in tale operazione interessi il valore di questo quarto bit. [fig. 7.8.3 (c)]. L'uscita passa a 1 se e solamente se la parita' e' verificata su tutti i quattro bit.

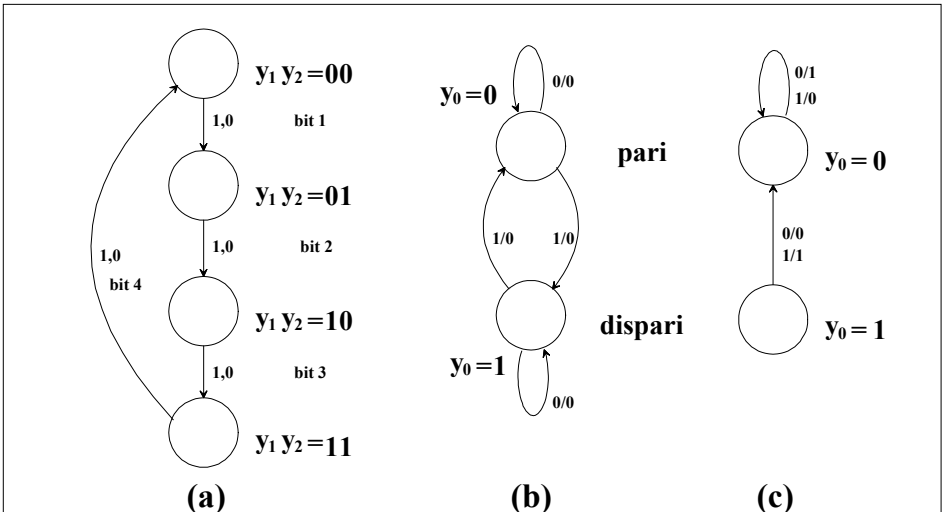


figura 7.8.3

Si noti che i valori di y_1 e y_2 non dipendono da x e da y_0. Tuttavia i valori di y_0^{n+1} e z sono funzione dei valori attuali x, y_1, y_2 e y_0.

BOOTH T. *" Sequential machines and automata theory "*
Wiley & Sons - New York - 1967

Per riunire le due funzioni in un unico circuito sequenziale e' opportuno tabulare la y_0^{n+1} come illustrato in fig. 7.8.4 (a).

Le due righe contrassegnate con l'asterisco corrispondono alla situazione illustrata in fig. 7.8.3 (c), valida per l'istante in cui arriva il quarto bit e cioe' quando $y_1 = y_2 = 1$. Le rimanenti righe corrispondono alla situazione di fig. 7.8.3 (b). Nella tabella compaiono due situazioni non specificate in quanto con $y_1 = y_2 = 0$ deve essere $y_0 = 0$. La figura 7.8.4 (b) e' la tavola di stato di un contatore binario da 2 bit. Da queste due tabelle si derivano le equazioni di ingresso degli elementi di memoria; se questi sono flip-flop JK si ha:

$$J_{y2} = K_{y2} = 1 \qquad J_{y1} = K_{y1} = y_2 \qquad J_{y0} = x.\left(\overline{y_1} + \overline{y_2}\right) \qquad K_{y0} = x + y_1 . y_2$$

$$z = y_1 . y_2 .\left(x . y_0 + \overline{x}.\overline{y_0}\right)$$

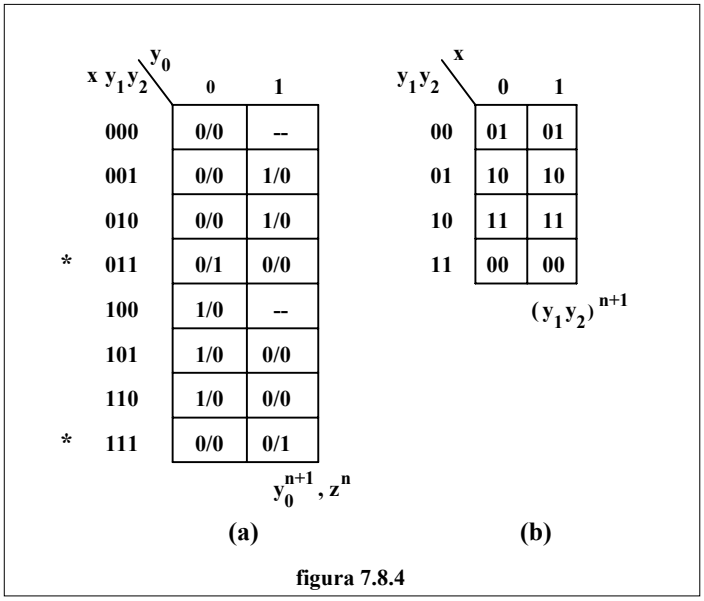


figura 7.8.4

Queste equazioni sono piu' semplici di quelle ottenute in precedenza. Si noti che il flip-flop y_2 funziona come flip-flop T, commutando ad ogni impulso di clock. Con la scelta fatta per le variabili di stato il circuito risulta separabile in due distinti sottocircuiti, come illustrato in fig. 7.8.5.

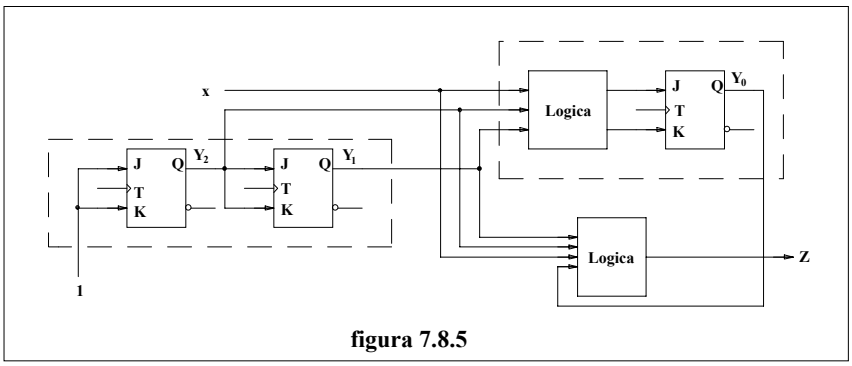


figura 7.8.5

Le tabelle di fig. 7.8.4 (a) e (b) possono essere combinate per ottenere la tavola di flusso di fig. 7.8.6 (a) e la tabella di stato di fig. 7.8.6 (b). Poiche' y_0 non e' un ingresso del contatore

binario a 2 bit, ci si dovrebbe aspettare di trovare una partizione chiusa degli stati in quattro classi, una per ciascuna combinazione di valori di y_1 e y_2. Questa partizione esiste ed e' la {0,4}, {1,5}, {2,6}, {3,7}. Che questa partizione sia chiusa puo' essere verificato dalla tabella di fig. 7.8.6 (c).

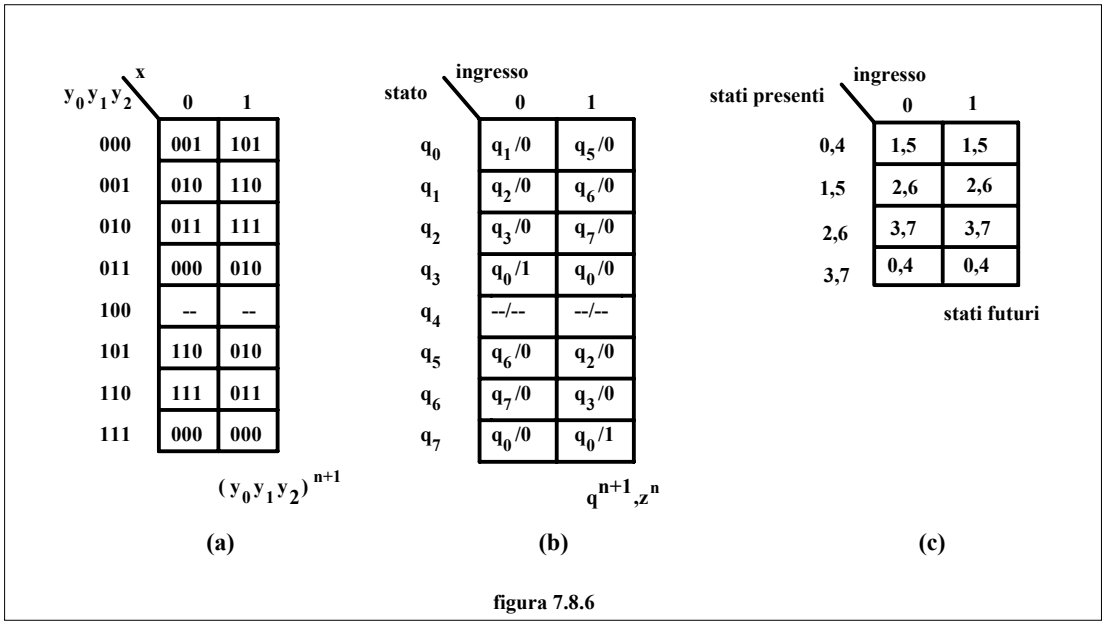


figura 7.8.6

Ritornando alle espressioni delle equazioni di eccitazione e di uscita si puo' notare che esse sono nettamente piu' semplici di quelle che si possono ottenere con l'applicazione delle regole 1,2 e 3 enunciate al paragrafo precedente. Individuare una partizione chiusa e' quindi molto spesso una via estremamente efficace per ottenere una realizzazione efficiente. Tuttavia il battere con successo tale strada dipende in maniera determinante dalla struttura del problema da risolvere e dall'abilita' del progettista nel riconoscere questa struttura. Non si suggerisce pertanto di tornare a tecniche di progetto intuitive, quanto ad essere sempre pronti a cogliere quando una struttura e' partizionabile.

In alcuni casi la struttura del problema suggerisce solo in modo parziale la codifica da adottare; in questi casi la codifica va completata mediante l'applicazione delle regole esposte al paragrafo precedente.

ESEMPIO 2

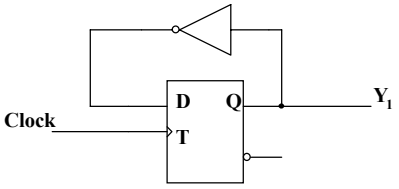
Si consideri nuovamente il contatore avanti-indietro modulo 8 introdotto all'inizio del presente paragrafo e la cui tavola degli stati e' illustrata in fig. 7.8.1. Per tale contatore si e' gia' visto che esistono due partizioni chiuse facilmente individuabili. Ci si dimentichera' tuttavia di questo fatto, tentando invece di individuare quelle partizioni che discendono in maniera diretta dalla struttura del circuito.

Una partizione naturale di ogni contatore e' quella che suddivide gli stati in due blocchi, uno corrispondente ai conteggi pari e uno ai conteggi dispari. Tale partizione sara' chiusa se il numero degli stati e' pari, poiche' ogni conteggio pari e' senz'altro seguito da uno dispari e viceversa. Per il caso in esame la partizione sara' (0,2,4,6) e (1,3,5,7), cui si assegnera' la variabile y_1. In fig. 7.8.7 e' riportata la tabella degli stati del contatore, la tavola di flusso del circuito che sviluppa y_1 e la relativa realizzazione con flip-flop di tipo D.

Scopo di tale circuito e' evidentemente quello di identificare se ci si trovi in una situazione di conteggio pari o dispari. La relativa equazione e':

$$y_1^{n+1} = \overline{y_1^n}$$

Stato \ ingresso	0	1
0	7	1
1	0	2
2	1	3
3	2	4
4	3	5
5	4	6
6	5	7
7	6	0



blocco attuale	0	1	y_1^n	y_1^{n+1} 0	y_1^{n+1} 1
(0,2,4,6)	(1,3,5,7)	(1,3,5,7)	0	1	1
(1,3,5,7)	(0,2,4,6)	(0,2,4,6)	1	0	0

figura 7.8.7

Ciascuno degli stati deve poi essere identificato da un'unica combinazione delle relative variabili. Se si usano tre variabili, come nel caso in esame, la seconda variabile dovra' dividere i due blocchi di quattro stati a meta', realizzando quattro blocchi da due stati; la terza variabile eseguira' un'ulteriore suddivisione a meta', fornendo otto blocchi da uno stato solo.

Si stabilisca pertanto che la variabile $y_2 = 0$ identifichi primo e terzo stato di ciascuno dei due blocchi precedenti, mentre il valore $y_2 = 1$ identifichi il secondo e quarto stato. La variabile y_2 induce cioe' la partizione (0,1,4,5) e (2,3,6,7). Allora le variabili y_1 e y_2 considerate assieme inducono la partizione:

y_2y_1	Gruppo
00	(0,1,4,5)∩(0,2,4,6)=(0,4)
01	(0,1,4,5)∩(1,3,5,7)=(1,5)
11	(2,3,6,7)∩(1,3,5,7)=(3,7)
10	(2,3,6,7)∩(0,2,4,6)=(2,6)

E' immediato verificare che anche questa e' una partizione chiusa e che quindi i valori futuri delle variabili di stato non sono funzioni di tutte le variabili. Si puo' allora sviluppare la tavola di flusso e il sottocircuito di fig. 7.8.8.

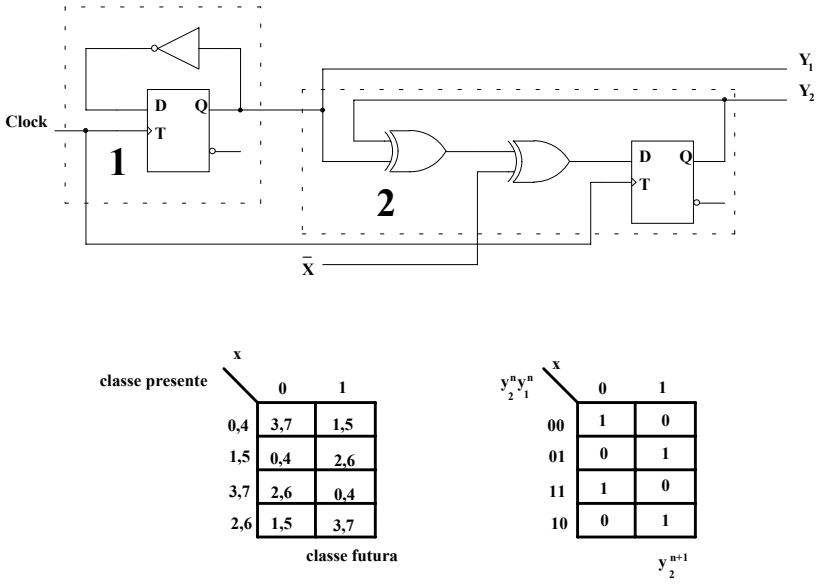


figura 7.8.8

La relativa equazione di stato e':

$$y_2^{n+1} = x^n \cdot \left(y_1^n \oplus y_2^n\right) + \overline{x^n} \cdot \left(\overline{y_1^n \oplus y_2^n}\right) = \overline{x^n} \oplus \left(y_1^n \oplus y_2^n\right)$$

Infine la variabile y_3 dovra' dividere questi quattro blocchi in due, inducendo la partizione (0,1,2,3) e (4,5,6,7). Poiche' i circuiti per sviluppare y_1 e y_2 sono gia' stati realizzati e' ovvio che il sottocircuito che si sta esaminando dovra' sviluppare solo y_3.

La tavola di flusso per questo sottocircuito e' riportata in fig. 7.8.9 assieme alla relativa realizzazione circuitale.

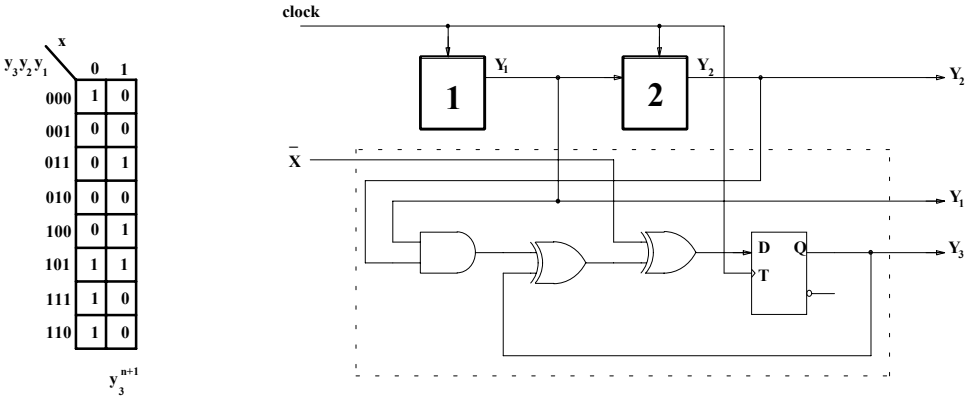


figura 7.8.9

Si noti che nello schema si sono indicati con 1 e 2 i blocchi realizzati in precedenza per y_1 e y_2.

L'equazione che si ricava per y_3 e':

$$y_3^{n+1} = \left(\left(y_1^n \cdot y_2^n\right) \oplus y_3^n\right) \oplus \overline{x^n}$$

Puo' non risultare immediatamente evidente, ma il circuito e' il piu' economico che si puo' realizzare con flip-flop di tipo D. Qualsiasi altra codifica dello stato fa si' che y_1 e y_2 siano funzione di un maggior numero di variabili, portando quindi a circuiti piu' complessi.

Le relazioni che esistono tra codifica dello stato, partizioni e struttura del circuito, messe in luce nei precedenti esempi, portano al quesito se la minimizzazione degli stati sia sempre conveniente e desiderabile.

Nell'esempio 3 del paragrafo 7.5 si era pervenuti ad una tavola con 8 stati, che richiedeva quindi tre variabili di stato. Se tuttavia si esamina la struttura base si notano due distinte sequenze di conteggio da tre stati. Pertanto una struttura piu' semplice e piu' naturale si puo' ottenere con quattro variabili di stato, due per ciascuna sequenza. La variabile di stato in piu' fa evidentemente si' che sia necessario un ulteriore flip- flop; tuttavia le reti combinatorie di eccitazione e di uscita ne risultano semplificate in modo notevole.

Un altro caso tipico e' quello dell'esempio 1 dello stesso paragrafo; considerando il circuito come sistema a memoria finita e realizzandolo nella forma di fig. 7.5.2 (a) si rendono necessari tre flip-flop D e l'unico costo ulteriore e quello relativo alla realizzazione della logica di uscita.

Lo stesso circuito e' stato preso come base per l'esempio di minimizzazione del paragrafo 7.6 che ha portato al diagramma a quattro stati di fig. 7.6.3 (b).

La realizzazione con flip-flop SR porta ad una rete logica di uscita di complessita' paragonabile a quella della realizzazione precedente. Per quanto riguarda invece lo stato, nella realizzazione a otto stati non era necessaria alcuna logica combinatoria, ma era sufficiente un'opportuna interconnessione dei flip-flop. Nella realizzazione a quattro stati invece la rete combinatoria si rende senz'altro indispensabile e le stesse interconnessioni risultano piu' complicate. Pertanto, malgrado che la realizzazione a quattro stati faccia risparmiare un flip-flop, essa e' probabilmente piu' costosa che non quella a otto stati.

7.9) Conclusioni.

Nel presente capitolo e' stata esposta la procedura di progetto dei circuiti sequenziali sincroni. Si intuisce immediatamente che il risultato che si ottiene e' notevolmente piu' influenzato dall'esperienza del progettista che non nel caso dei circuiti combinatori e al limite dei circuiti sequenziali asincroni.

La mancanza di formalismo e rigorosita' puo' far pensare che la procedura proposta sia una specie di progetto basato sulla prova e l'errore; cio' non e' assolutamente vero in quanto la procedura e' organizzata con estrema attenzione per portare il progettista, con un numero di tentativi il piu' possibile limitato, dalle iniziali e imprecise specificazioni al circuito finale.

Il primo passo, che consiste nella determinazione del diagramma degli stati, e' il piu' importante. Le specifiche che vengono assegnate sono di solito abbastanza vaghe e incomplete. Le procedure dei paragrafi 7.4 e 7.5 aiutano a chiarire il funzionamento del circuito e a determinarne il comportamento in ogni possibile circostanza.

Molto spesso, probabilmente a causa della fatica e delle difficolta' che questo passo comporta, il progettista e' portato a interconnettere in maniera quasi empirica dei sottocircuiti, quali contatori o circuiti di test di parita',ecc., intuiti dalle specifiche complessive, ottenendo

pero' il piu' delle volte dei comportamenti assolutamente insospettati. Ora non e' che procedure intuitive non possano essere utilizzate, ma il momento in cui si possono applicare e' solo dopo aver analizzato e capito a fondo il comportamento che si desidera per il circuito.

Se il circuito e' a memoria finita e la tavola di flusso viene realizzata su questa base, l'unica operazione che il progettista deve compiere e' quella di stabilire l'uscita; se inoltre la struttura a memoria finita e' anche applicabile, essa produce sempre il circuito piu' economico.

Se il circuito non e' a memoria finita e' necessario minimizzare il numero degli stati. Questa minimizzazione non da' luogo necessariamente al circuito piu' economico, tuttavia e' sempre bene partire da una tavola con il numero di stati minimo ed aggiungere altri stati ausiliari solo se la struttura del problema rende ovvio che e' conveniente operare con un maggior numero di variabili di stato.

Il passo finale e' quello della codifica, difficoltoso e molto spesso frustrante. Secondo quanto e' stato esposto e' conveniente usare alcune regole semiempiriche, unite ad una buona dose di intuizione, per ottenere un risultato soddisfacente. Ancora oggi disgraziatamente non esiste alcuna buona procedura formale di codifica, e quindi una specie di intuizione guidata rimane ancora il miglior metodo di progetto.