VINCENZO FALZONE

CIRCUITI DIGITALI

- sistemi di numerazione e codici
- algebra booleana
- circuiti elementari di logica e di memoria
- circuiti combinatori
- circuiti sequenziali
- tecniche digitali
- calcolatori elettronici

EDIZIONI SCIENTIFICHE SIDEREA

VINCENZO FALZONE

CIRCUITI DIGITALI

SIDEREA

Le copie non firmate dall'Autore si ritengono contraffatte

Stampato presso il laboratorio fotolitografico della
Edizioni Scientifiche SIDEREA
Via Terme di Traiano 5A - 00184 ROMA

PREFAZIONE

Questo libro deriva dalle dispense scritte per i corsi di «Teoria e Tecnica dei Circuiti Digitali» da me tenuti presso la «Scuola Superiore di Specializzazione in Telecomunicazioni» dell'I.S.P.T.. Contiene gli argomenti trattati nei corsi stessi e molti altri che mi sono sembrati utili per risolvere i problemi che più comunemente si presentano nel progetto logico delle apparecchiature digitali.

Il volume ha carattere introduttivo e, benché le lezioni da cui ha origine siano dedicate a ingegneri elettronici, è scritto in forma molto semplice e non presuppone precedenti conoscenze di algebra astratta o di teoria della commutazione.

Il contenuto del libro è il seguente: i primi due capitoli comprendono un cenno sui sistemi di numerazione e sui codici binari, nonché una esposizione sintetica dell'algebra booleana, limitatamente alle parti che trovano applicazione nel testo. Il cap.III descrive i circuiti elementari di logica e di memoria, realizzati con componenti discreti o a circuiti integrati. Il cap.IV tratta l'analisi e la sintesi dei circuiti combinatori a uno e più terminali, compresi – tra questi ultimi – i selettori e le matrici. I tre capitoli seguenti sono dedicati alla teoria delle macchine e dei circuiti sequenziali sincroni e asincroni; gli ultimi capitoli, infine, espongono i criteri praticamente seguiti per la realizzazione dei circuiti sequenziali con componenti integrati e descrivono l'organizzazione generale e alcuni circuiti tipici dei calcolatori elettronici.

Vincenzo Falzone

INDICE

INDICE

CAPITOLO VI: Circuiti sequenziali asincroni

CAPITOLO VII: Circuiti sequenziali sincroni

CAPITOLO VIII: Tecniche digitali

CAPITOLO IX: Calcolatori elettronici

*

CAPITOLO I

CENNI SUI SISTEMI DI NUMERAZIONE E SUI CODICI

1.1 - Sistemi di numerazione.

Si chiama *sistema di numerazione* l'insieme di un numero finito di simboli e delle regole che assegnano uno ed un solo significato ad ogni scrittura formata coi simboli stessi.

I moderni sistemi di numerazione sono posizionali: tutti i simboli (o *cifre*) vengono ordinati in modo che ognuno abbia un valore di una unità più alto di quello della cifra precedente; quando più cifre vengono combinate insieme, il valore del numero rappresentato dipende anche dalle loro posizioni relative.

La trattazione completa dei sistemi numerici esula dai nostri scopi; nei prossimi paragrafi ci limiteremo a descrivere i sistemi più usati nei calcolatori, derivandone le proprietà da quelle del sistema decimale, a tutti familiare.

1.2 - Sistema decimale.

Le cifre del sistema numerico a base 10 (sistema decimale) sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

In ogni numero decimale, la cifra all'estrema destra ha il valore minore (cifra meno significativa); quella all'estrema sinistra il valore maggiore (cifra più significativa).

Il significato della scrittura 3847 è:

3 migliaia + 8 centinaia + 4 decine + 7 unità

ovvero:

$$\begin{aligned} & 3000 + 800 + 40 + 7 \\ = \; & 3 \times 1000 + 8 \times 100 + 4 \times 10 + 7 \times 1 \\ = \; & 3 \times 10^3 + 8 \times 10^2 + 4 \times 10^1 + 7 \times 10^0 \; . \end{aligned}$$

L'ultima espressione mette chiaramente in evidenza il valore posizionale di ogni cifra.

Nei numeri frazionari, o misti, ogni cifra alla destra della virgola deve intendersi moltiplicata per una potenza negativa di 10, a partire da -1 per la cifra più vicina alla virgola. Ad esempio 307,251 significa:

$$\begin{aligned} & 3 \times 10^2 + 0 \times 10^1 + 7 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2} + 1 \times 10^{-3} \\ & 300 + 0 + 7 + 0{,}2 + 0{,}05 + 0{,}001 \; . \end{aligned}$$

Le regole esposte sono estensibili immediatamente ad ogni sistema posizionale di base B. Precisamente:

– *Base* di un sistema di numerazione è il numero di cifre diverse che esso comprende.

– La cifra di minor valore è sempre lo 0; le altre sono, nell'ordine: 1,2,... ..., B - 1, se B > 10, occorre introdurre B - 10 simboli in aggiunta alle cifre decimali.

– Un numero intero N si rappresenta con la scrittura $C_n C_{n-1} \ldots C_2 C_1 C_0$. Vale la relazione:

$$N = C_n B^n + C_{n-1} B^{n-1} + \ldots + C_2 B^2 + C_1 B^1 + C_0 B^0$$

essendo C_k la cifra nella generica posizione k.

– Un numero frazionario N' si rappresenta con la scrittura $0, C_1 C_2 \ldots C_n$. Vale la relazione:

$$N' = C_1 B^{-1} + C_2 B^{-2} + \ldots + C_n B^{-n}$$

essendo C_k la cifra nella generica posizione K.

– Un numero misto N^* si rappresenta con la scrittura N, N', essendo:

$$N^* = N + N' \; .$$

Nel seguito, prenderemo in considerazione i sistemi binario ed esadecimale per i quali si ha, rispettivamente, B = 2 e B = 16.

1.3 - Sistema binario.

La base 2 è la più piccola teoricamente possibile per un sistema di numerazione. Le cifre 0 e 1 si indicano col nome di *bit* (abbreviazione di *binary digit*). Il valore posizionale di ogni bit è legato alle potenze di 2: così le scritture 1011001 e 110,101 significano, rispettivamente:

$$1011001 = 1\times2^6 + 0\times2^5 + 1\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0 =$$
$$= 1\times64 + 0\times32 + 1\times16 + 1\times8 + 0\times4 + 0\times2 + 1\times1 =$$
$$= 64 + 16 + 8 + 1 = 89$$

$$110{,}101 = 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} =$$
$$= 4 + 2 + \frac{1}{2} + \frac{1}{8} =$$
$$= 4 + 2 + 0{,}5 + 0{,}125 = 6{,}625 \ .$$

Per evitare confusione, ogni volta che si usano sistemi di numerazione diversi, conviene scrivere – come indici dei relativi numeri – le basi, ad esempio:

$$(1011001)_2 = (89)_{10}$$
$$(110{,}101)_2 = (6{,}625)_{10} \ .$$

Il sistema binario è adoperato nei circuiti digitali in generale, e nei calcolatori numerici in particolare, in quanto esistono numerosi componenti elettronici che possono trovarsi nell'uno o nell'altro di due condizioni di funzionamento opposte, facilmente interpretabili coi due valori delle cifre binarie.

1.4 - Sistema esadecimale.

I 16 simboli del sistema sono: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F; il loro valore corrisponde, ordinatamente, ai numeri decimali 0... ...15.

Il significato posizionale di una cifra esadecimale è legato alle potenze di 16; ad esempio, la scrittura 2BF, C6 significa:

$$2BF,C6 = 2\times16^2 + 11\times16^1 + 15\times16^0 + 12\times16^{-1} + 6\times16^{-2} =$$
$$= 2\times256 + 11\times16 + 15 + 12/16 + 6/256 =$$
$$= 512 + 176 + 15 + 0{,}75 + 0{,}023 =$$
$$= (703{,}773)_{10} \ .$$

Il sistema esadecimale viene usato soprattutto perché, essendo ogni sua cifra esprimibile con quattro bit, fornisce un metodo semplice per scrivere – con pochi simboli – dei grossi numeri binari, altrimenti rappresentati da lunghe stringhe di 0 e 1. Dividendo i bit di tali numeri, da destra verso sinistra, in gruppi di quattro e sostituendo ad ogni gruppo la corrispondente cifra esadecimale, si ottengono infatti rappresentazioni molto sintetiche.

Ad esempio, il numero:

$$1100101010111011111$$

diviene:

$$\begin{array}{ccccc} 110 & 0101 & 0101 & 1101 & 1111 \\ 6 & 5 & 5 & D & F \end{array} = (655DF)_{16} \ .$$

Se il numero non è intero, la divisione in gruppi di quattro bit va fatta, a partire dalla virgola, verso destra e verso sinistra:

$$\begin{array}{l} 1100110111110111,\ 001101111 \quad = \\ 1 \cdot 1001 \cdot 1011 \cdot 1111 \cdot 0111,\ 0011 \cdot 0111 \cdot 1 = \\ 1 \quad 9 \quad B \quad F \quad 7 \quad 3 \quad 7 \quad 8 = (19BF7,378)_{16} \ . \end{array}$$

Il passaggio inverso, dal numero esadecimale a quello binario, è immediato: basta sostituire, a ogni cifra, i quattro bit corrispondenti:

$$B\,3CA,FAD\,8 = 1011001111001010,1111101011011$$

infatti:

B	3	C	A	F	A	D	8
1011	0011	1100	1010	1111	1010	1101	1000

A titolo d'esempio, nella tabella di fig. I.1 sono mostrate le rappresentazioni, nei tre codici decimale, binario ed esadecimale, dei numeri interi da 0 a $(20)_{10}$.

TABELLA I.1 - Numeri interi da 0 a $(20)_{10}$ in binario e in esadecimale.

Decimale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Binario	0	1	01	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000	10001	10010	10011	10100
Esadecim.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14

Fig. I.1

I.5 - Conversione tra i sistemi di numerazione.

Le regole illustrate nel paragrafo precedente sono esempi particolarmente semplici di conversioni, cioè dei metodi per esprimere lo stes-

so numero in sistemi di numerazione diversi. Vedremo ora alcuni procedimenti per le conversioni tra i sistemi binario ed esadecimale ed il sistema decimale; per chiarezza, tratteremo separatamente le conversioni dei numeri interi e frazionari, che avvengono secondo regole diverse.

1.5.1 - Conversione di numeri interi dal sistema decimale al binario.

Il metodo più semplice è quello della divisione, la cui giustificazione è immediata. Si divide ripetutamente il numero decimale per 2, fino ad ottenere un quoziente nullo. Il numero binario equivalente è composto dai resti delle successive divisioni; la cifra più significativa è lo ultimo resto; la meno significativa, il primo.

Esempio: *Conversione in binario del numero* $(26)_{10}$.

$\frac{26}{2} = 13 + 0$ ← cifra meno significativa

$\frac{13}{2} = 6 + 1$

$\frac{6}{2} = 3 + 0$ ↑ numero binario

$\frac{3}{2} = 1 + 1$

$\frac{1}{2} = 0 + 1$ ← cifra più significativa.

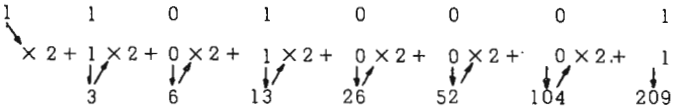
Pertanto: $(26)_{10} = (11010)_2$.

1.5.2 - Conversione di numeri interi dal sistema binario al decimale.

Il metodo migliore, valido anche per i numeri frazionari o misti, è quello, implicitamente già descritto negli esempi precedenti, dell'espansione del numero in potenze di 2.

Una regola pratica per abbreviare le operazioni necessarie è la seguente: si raddoppia il bit più significativo e si aggiunge il secondo bit; si raddoppia la somma e si aggiunge il terzo bit ... si continua così fino al bit meno significativo: l'ultima somma rappresenta il numero decimale.

Esempio: *Conversione in decimale del numero* $(11010001)_2$.



Pertanto: $(11010001)_2 = (209)_{10}$.

1.5.3 - Conversione di numeri interi dal sistema decimale all'esadecimale.

Si divide ripetutamente il numero decimale per 16, fino ad ottenere un quoziente nullo. I resti, convertiti in esadecimale, rappresentano il risultato della conversione: il primo resto è la cifra meno significativa.

Esempio: *Conversione in esadecimale del numero* $(714)_{10}$.

$$\frac{714}{16} = 44 + 10\ (=A) \quad \longleftarrow \text{cifra meno significativa}$$

$$\frac{44}{16} = 2 + 12\ (=C) \quad \uparrow \text{numero esadecimale}$$

$$\frac{12}{16} = 0 + 2\ (=2) \quad \longleftarrow \text{cifra più significativa}.$$

Pertanto: $(714)_{10} = (2CA)_{16}$.

1.5.4 - Conversione di numeri interi dal sistema esadecimale al decimale.

Oltre al metodo generale, dell'espansione secondo potenze di 16, si può usare la seguente regola pratica: si moltiplica per 16 l'equivalente decimale della cifra più significativa; si aggiunge l'equivalente decimale della seconda cifra; si moltiplica la somma per 16, e si aggiunge la terza cifra .... L'ultima somma ottenuta rappresenta il numero cercato.

Esempio: *La conversione in decimale di* $(2CA)_{16}$ *si può fare in due modi:*

1)
$$(2CA)_{16} = 2 \times 16^2 + 12 \times 16^1 + 10 \times 16^0 =$$
$$= 2 \times 256 + 12 \times 16 + 10 \times 1 =$$
$$= 512 + 192 + 10 = (714)_{10}$$

2)
$$(2CA)_{16} = 2 \times 16 = 32$$
$$(32)_{10} + (C)_{16} = 32 + 12 = 44$$
$$44 \times 16 = 704$$
$$(704)_{10} + (A)_{16} = 704 + 10 = (714)_{10}.$$

1.5.5 - Conversione di frazioni dal sistema decimale al binario.

La conversione avviene per successive moltiplicazioni della frazione per due. I valori degli interi ottenuti come risultati costituiscono, nell'ordine, le cifre del numero binario; i soli valori frazionari vengono usati nelle successive moltiplicazioni. Il procedimento ha termine quando la parte frazionaria è diventata nulla o quando si siano trovate un numero sufficiente di cifre binarie.

Esempi.

1) *Convertire in binario il numero* $(0,65625)_{10}$.

Frazione	Prodotto	Parte intera	
0,65625 × 2 =	1,31250	1	← cifra più significativa
0,31250 × 2 =	0,62500	0	
0,62500 × 2 =	1,25000	1	numero binario
0,25000 × 2 =	0,50000	0	
0,50000 × 2 =	1,0000	1	← cifra meno significativa

$$(0,65625)_{10} = (0,10101)_2 \quad .$$

2) *Convertire in binario il numero* $(0,6352)_{10}$.

Frazione	Prodotto	Parte intera	
0,6352 × 2 =	1,2704	1	← cifra più significativa
0,2704 × 2 =	0,5408	0	
0,5408 × 2 =	1,0816	1	
0,0816 × 2 =	0,1632	0	numero binario
0,1632 × 2 =	0,3264	0	
0,3264 × 2 =	0,6528	0	
0,6528 × 2 =	1,3056	1	← cifra meno significativa

$$(0,6352)_{10} = (0,1010001)_2 \quad .$$

1.5.6 - Conversione di frazioni dal sistema binario al decimale.

Conviene usare il metodo generale di conversione per espansione del numero in potenze negative di 2. Per facilitare le operazioni, nella tabella della fig.I.2 sono riportati i valori decimali di 2^{-n} per n=1, 2...10.

TABELLA I.2 - Valori delle potenze negative di 2.

n	1	2	3	4	5	6	7	8	9	10
2^{-n}	0,5	0,25	0,125	0,0625	0,03125	0,015625	0,0078125	0,0039062	0,0019531	0,0009765

Fig.I.2

Esempio: *Convertire in decimale il numero 0,011001.*

$$(0,011001)_2 = 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 0 \times 2^{-5} + 1 \times 2^{-6} =$$
$$= 0,25 + 0,125 + 0,015625 = (0,390625)_{10} .$$

I.5.7 - Conversione di frazioni dal sistema decimale all'esadecimale.

Si usa il procedimento illustrato nel par. I.5.5, moltiplicando per 16 invece che per 2.

Esempio: *Convertire in esadecimale il numero* $(0,828125)_{10}$.

Frazione	Prodotto	Parte intera	Cifra esadecimale	
0,828125 × 16 =	13,25	13	D	← cifra più significativa
0,25 × 16 =	4,00	4	4	← cifra meno significativa

$$(0,828125)_{10} = (0,D4)_{16} .$$

I.5.8 - Conversione di frazioni dal sistema esadecimale al decimale.

Il metodo da usare è quello delle divisioni ripetute per 16, invertendo il procedimento esposto al punto I.5.4. Occorre dividere l'equivalente decimale della cifra esadecimale più a destra per 16, aggiungere al quoziente la seconda cifra esadecimale da destra e dividere di nuovo per 16; si continua fino ad esaminare tutte le cifre della frazione: l'ultimo quoziente ottenuto rappresenta il numero cercato.

Esempio: *Convertire in decimale il numero* $(0,D4)_{16}$.

$$(4)_{16} = (4)_{10}$$

$$\frac{4}{16} = 0,25$$

$$(D)_{16} + (0,25)_{10} = 13 + 0,25 = (13,25)_{10}.$$

$$\frac{13,25}{16} = 0,828125$$

$$(0,D4)_{16} = (0,828125)_{10} \ .$$

Per facilitare i calcoli delle espressioni frazionarie fino a 3 cifre decimali, si può usare la tabella della fig.I.3, che riporta valori decimali di 0,X – 0,0X – 0,00X, dove X è una qualsiasi cifra esadecimale.

TABELLA I.3 - Equivalente decimale di frazioni esadecimali.

X	0,X	0,0X	0,00X
1	0,0625	0,00390625	0,000244140625
2	0,1250	0,00781250	0,000488281250
3	0,1875	0,01171875	0,000732421875
4	0,2500	0,01562500	0,000976562500
5	0,3125	0,01953125	0,001220703125
6	0,3750	0,02343750	0,001464843750
7	0,4375	0,02734375	0,001708984375
8	0,5000	0,03125000	0,001953125000
9	0,5625	0,03515625	0,002197265625
A	0,6250	0,03906250	0,002441406250
B	0,6875	0,04296875	0,002685546875
C	0,7500	0,04687500	0,002929687500
D	0,8125	0,05078125	0,003173828125
E	0,8750	0,05468750	0,003417968750
F	0,9375	0,05859375	0,003662109375

L'uso della tabella è molto semplice. Per convertire, ad esempio, il numero $(0,A61)_{16}$ basta leggere i tre valori:

$$\begin{aligned} 0,A &= 0,625 \\ 0,06 &= 0,02343750 \\ 0,001 &= 0,000244140625 \end{aligned}$$

e sommarli: $(0,A61)_{16} = (0,648681640625)_{10}$.

1.6 - Rappresentazione dei numeri con segno.

I numeri binari con segno si rappresentano:

1) con modulo e segno;

2) in complemento a 2;

3) in complemento a 1.

1.6.1 - Rappresentazione con modulo e segno.

Il valore assoluto del numero viene rappresentato nel sistema di numerazione binario; a sinistra del bit più significativo, si premette poi il bit di segno: uno 0 per i numeri positivi, un 1 per quelli negativi.

Ad esempio, il numero +11 si rappresenta con 01011; il numero -11 con 11011. Tra il bit di segno e il bit più significativo possono essere introdotti zeri non significativi: sono, cioè equivalenti le scritture 00001011 e 01011; 10001011 e 11011.

In questa rappresentazione, esiste uno zero positivo (0000...0) e uno zero negativo (10...0): pertanto, il valore assoluto del più grande numero positivo eguaglia quello del più piccolo numero negativo rappresentabile con n bit.

1.6.2 - Rappresentazione in complemento a 2.

In un sistema di numerazione a base B, si chiama complemento a B di un numero $(N)_B$ ad n cifre, il numero:

$$B^n - N \ .$$

Tale numero si ottiene aggiungendo 1 al complemento a (B - 1) di ogni cifra x (B - 1 - x).

Ad esempio, il complemento a 10 del numero 6485 è 3515 (9 - 6 = = 3; 9 - 4 = 5; 9 - 8 = 1; 9 - 5 = 4; 3514 + 1 = 3515). Sommando 6485 + 3515 si ottiene infatti 10000, cioè 10^4.

Il complemento a 2 di un numero binario N è ottenuto aggiungendo 1 al complemento a 1 di ogni bit, cioè al suo inverso.

Ad esempio, il complemento a 2 di 11001010110 è 00110101001 + + 1 = 00110101010.

Per rappresentare i numeri binari con segno, si adottano le seguenti convenzioni:

– i numeri positivi sono rappresentati in modulo e segno;

– i numeri negativi hanno un 1 nella posizione più significativa, e sono rappresentati in complemento a 2.

Ad esempio, il numero +11 si rappresenta con 01011; il numero -11 con 10101.

Un numero di m bit si può rappresentare con n bit (m > n) espandendo verso sinistra, di (m - n) posizioni, il bit di segno.

Così, le scritture 01011 e 00001011 rappresentano entrambi il numero +11, e le scritture 10101 e 11111010l rappresentano il numero -11.

In questa rappresentazione, lo zero è rappresentato come un numero positivo, cioè con una stringa di n zeri, il primo dei quali è il bit di segno. Così, per n = 7 si ha:

0000000 .

Non esiste lo zero negativo, pertanto il valore assoluto del più grande numero positivo rappresentabile con n bit è inferiore di 1 al più piccolo numero negativo di n bit.

Ad esempio, il più grande numero positivo di 16 bit è:

0111111111111111 cioè $2^{15} - 1 = (32.767)_{10}$;

mentre il più piccolo numero negativo è:

1000000000000000 cioè $-2^{15} = (-32.768)_{10}$.

1.6.3 - Rappresentazione in complemento a 1.

In un sistema di numerazione a base B, si chiama complemento a B - 1 di un numero $(N)_B$ a n cifre, il numero:

$$B^n - 1 - N \ .$$

Tale numero si ottiene facendo il complemento a B - 1 di ogni cifra.

Ad esempio, il complemento a 9 di 6485 è 3514.

Il complemento a 1 di un numero binario N è ottenuto invertendo ogni suo bit.

Ad esempio, il complemento a 1 di 11001010110 è 00110101001.

Per la rappresentazione di numeri binari con segno, si adottano le seguenti convenzioni:

– i numeri positivi sono rappresentati in modulo e segno;

– i numeri negativi hanno un 1 nella posizione più significativa, e sono rappresentati in complemento a 1.

Ad esempio, il numero +11 si rappresenta con 01011; il numero -11 con 10100.

In questa rappresentazione, vale la regola dell'espansione del segno; esiste uno zero positivo e uno zero negativo, per cui il valore assoluto del più grande numero positivo eguaglia quello del più piccolo numero negativo rappresentabile con n bit.

1.7 - Aritmetica binaria.

Per comprendere il funzionamento di alcuni importanti circuiti numerici, è necessario saper effettuare le operazioni aritmetiche direttamente in binario; le relative regole, esposte di seguito, sono simili a quelle valide per il sistema decimale. Esamineremo dapprima le operazioni sui numeri senza segno, poi quelle sui numeri con segno.

1.7.1 - Addizione binaria.

Le regole dell'addizione di due bit sono le seguenti:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ con riporto di 1

l'ultima regola si può scrivere:

$$(1)_2 + (1)_2 = (10)_2$$

e significa, semplicemente: $(1 + 1 = 2)_{10}$.

L'addizione di due numeri si fa tenendo conto degli eventuali riporti.

Esempio: *Addizionare i numeri 1011011 e 1011010.*

```
 11 1    <---- riporti
1011011
1011010 +
--------
10110101
```

1.7.2 - Sottrazione binaria.

Le regole della sottrazione di due bit sono le seguenti:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$0 - 1 = 1 \quad \text{con un prestito di 1 .}$$

L'ultima regola si può scrivere:

$$(10)_2 - (1)_2 = (1)_2$$

e significa: $(2 - 1 = 1)_{10}$.

La sottrazione di due numeri si fa tenendo conto degli eventuali prestiti.

Esempio: *Sottrarre, 10101010 da 11011001.*

```
 1 111   <-- prestiti
11011001
10101010 -
--------
00101111
```

1.7.3 - Moltiplicazione binaria.

Le regole della moltiplicazione di 2 bit sono:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1 \ .$$

Nella moltiplicazione di 2 numeri, poiché ogni cifra del moltiplicatore non può essere che 0 o 1, ogni prodotto parziale è uguale al moltiplicando, oppure è nullo: nel primo caso, il moltiplicando viene copiato; nel secondo, viene fatto scorrere di un posto a sinistra; successivamente, si addizionano i prodotti parziali.

Esempi:

1) 1100111×100

```
1100111 x
    100
---------
110011100
```

2) 1100111×11

$$\begin{array}{r} 1100111 \times \\ 11 \\ \hline 1100111 \\ 1100111 \\ \hline 100110101 \end{array}$$

3) $1100{,}111 \times 10{,}1$

$$\begin{array}{r} 1100{,}111 \times \\ 10{,}1 \\ \hline 1100111 \\ 1100111 \\ \hline 100000{,}0011 \end{array}$$

I.7.4 - Divisione binaria.

La divisione binaria avviene confrontando gli n bit più significativi del dividendo con gli n bit del divisore. La prima cifra del quoziente sarà 1 o 0 a seconda che il divisore contenga o no il dividendo. Per il resto, valgono le regole della divisione decimale.

Esempio: *Divisione, in binario, di* $(72)_{10}$ *per* $(6)_{10}$.

$$72 = 1001000$$
$$6 = 110$$

Divisione:

$$\begin{array}{r|l} 1001000 & 110 \\ -110 & \hline 1100 \\ \hline 00110 & \\ -110 & \\ \hline 000 & \end{array}$$

$$1001000/110 = 1100\ (=12)_{10}.$$

I.8 - Operazioni sui numeri binari con segno.

I.8.1 - Addizione e sottrazione.

Considerando i numeri con segno, l'addizione e la sottrazione vengono trattate come un'unica operazione. La rappresentazione più conveniente per effettuare quest'operazione è quella in complemento a 2 o in complemento a 1.

1.8.1.1 - Rappresentazione in complemento a 2.

Nel sistema decimale, l'operazione 4378 - 3515 (= 863) può essere scritta 4378 + (10000 - 3515) - 10000 = 4378 + (6485) - 10000 dove il termine tra parentesi indica il complemento a 10 del sottraendo. Il risultato si ottiene sommando al minuendo il complemento a 10 del sottraendo: 4378 + 6485 = 10863 e trascurando la cifra più significativa.

Nel sistema binario, l'operazione $N_1 - N_2$ $(N_1 \geq N_2 > 0)$ si effettua complementando N_2, sommando N_1 al complemento di N_2, e trascurando il bit più significativo del risultato. Se occorre, si applica la regola dell'espansione del segno per eguagliare la lunghezza di N_2 a quella di N_1.

Esempio: *10001 - 11* $(= 17 - 3)_{10}$.

Nella rappresentazione con complemento a 2, il complemento di 11 si scrive 101; si ha quindi:

```
                    010001 +
bit di segno  -->   111101
                  ---------
                  1|001110
```

Pertanto $10001 - 11 = 1110 \, (= 14)_{10}$.

L'addizione e la sottrazione di 2 numeri con segno a n bit si fa sommando i numeri stessi, e trascurando eventuali bit nella posizione n + 1; la sottrazione va preceduta dal complemento del sottraendo (che, se negativo, diventa così positivo).

Esempi:

1) $(8 + 21)_{10}$ $(= 29)_{10}$

```
(8)     001000
(21)    010101 +
        ------
(29)    011101
```

2) $(64 - 25)_{10}$ $(= 39)_{10}$

```
(64)      01000000
(-25)     11100111
         ---------
(39)     1|00100111
```

3) $(25 - 64)_{10}$ $(= -39)_{10}$

```
(25)      00011001
(-64)     11000000
          --------
(-39)     11011001
```

In qualche caso, tuttavia, le regole esposte cadono in difetto.

Esempio:

4) $(15+15)_{10}\ (=30)_{10}$

(15)	01111
(15)	01111
(−2)	11110

Il risultato corretto si ha leggendo il numero 11110 come se avesse un bit di segno 0 a sinistra (011110 = 30).

Esempio:

5) $(-13-15)_{10} = (-28)_{10}$

(−13)	10011
(−15)	10001
(+4)	1|00100

Il risultato esatto si ha leggendo anche il bit di segno (100100 = −28).

Ciò avviene perché la rappresentazione adottata fissa rigidamente il numero e la posizione dei bit significativi, di modo che, se la somma di 2 numeri di n bit significativi contiene (n + 1) bit, l'ultimo di questi varia il bit di segno, e falsa il risultato. Si dice, allora, che si è verificato un trabocco (*overflow*).

Per evitare il trabocco, basta aumentare i bit degli addendi, come mostrano i due esempi seguenti.

Esempi:

6) $(15+15)_{10}\ (=30)_{10}$

(15)	001111
(15)	001111 +
(30)	011110

7) $(-13-15)_{10}\ (=-28)_{10}$

(−13)	110011
(−15)	110001
(−28)	1|100100

Il risultato di un'addizione è esatto se non c'è stato nessun riporto sia nel bit di segno che alla sua sinistra, oppure se c'è stato un riporto in entrambi; è invece avvenuto un trabocco se, nelle due posizioni, si è verificato un solo riporto. Gli esempi 4), 5), 6), 7) illustrano, rispettivamente, i quattro casi possibili.

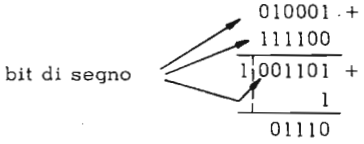
1.8.1.2 - Rappresentazione in complemento a 1.

Nel sistema decimale, l'operazione 4378 − 3515 (= 863) può essere scritta 4378 + (9999 − 3515) − 9999 = 4378 + (6484) − 9999 dove il termine tra parentesi indica il complemento a 9 del sottraendo. Il risultato si ottiene sommando al minuendo il complemento a 9 del sottraendo (4378 + + 6484 = 10862) trascurando la cifra più significativa, e aggiungendola alle cifre rimaste (862 + 1 = 863).

Nel sistema binario, l'operazione $N_1 - N_2$ $(N_1 > N_2 \geq 0)$ si effettua complementando N_2, sommando N_1 al complemento di N_2, trascurando il bit più significativo del risultato, e sommando 1 al numero ottenuto. Se occorre, si applica la regola dell'espansione del segno ad N_2.

Esempio: $10001 - 11$ $(17 - 3)_{10}$.

Nella rappresentazione con complemento a 1, il complemento di 11 si scrive 100, pertanto:



L'addizione – e la sottrazione – di due numeri con segno a n bit si fa sommando i numeri stessi; la sottrazione va preceduta dal complemento del sottraendo (che, se negativo, diventa così positivo). Eventuali bit 1 nella posizione $(n + 1)$ vanno aggiunti al risultato.

Esempi:

1) $(8 + 21)_{10}$ $(= 29)_{10}$

(8)	001000
(21)	010101 +
(29)	011101

2) $(64 - 25)_{10}$ $(= 39)_{10}$

(64)	01000000
(−25)	11100110 +
	1 00100110
	1 +
(39)	00100111

L'aggiunta di 1 al risultato non va fatta se non esiste il bit 1 in posizione $n + 1$: in questo caso, infatti, il risultato è negativo e va letto direttamente.

Esempi:

3) $(+ 12 - 15)_{10}$

4) $(25 - 64)_{10}$ $(= -39)_{10}$

(25)	00011001
(−64)	10111111
(−39)	11011000

Valgono per il resto le osservazioni fatte nel paragrafo precedente circa le possibilità di evitare trabocchi.

1.8.2 - Moltiplicazione.

La rappresentazione più conveniente per la moltiplicazione è quella con modulo e segno.

In questo caso, il modulo del prodotto è il prodotto dei moduli, mentre il segno è 0 se i bit di segno sono uguali, 1 se sono diversi.

Esempio: $(+15)_{10} \times (-2)_{10}$

(+15)	01111
(−2)	10010

prodotto dei moduli: $1111 \times 0010 = 11110$

bit di segno: 1

risultato: $111110\ (= 30)_{10}$.

1.8.3 - Divisione binaria.

La rappresentazione più conveniente è quella con modulo e segno: il modulo del quoziente è il quoziente dei moduli; il segno è 0(1) se i bit di segno sono uguali (diversi).

Esempio: $(+72)_{10}/(-6)_{10}$

(+72)	01001000
(−6)	1110

quoziente dei moduli: 1100

bit di segno: 1

risultato: $11100\ (= -12)_{10}$.

Quelli descritti sono i metodi più semplici per effettuare le operazioni aritmetiche in binario. Essi presuppongono che i numeri vengono espressi nella rappresentazione con «modulo e segno» o in quella «complementata» a seconda dell'operazione da effettuare. Esistono, tuttavia, delle regole per eseguire le operazioni nello stesso sistema di rappresentazione, evitando ogni conversione. Per esse, rimandiamo ai testi in bibliografia (particolarmente a [8]).

1.9 - I codici.

Codice è un insieme [C] di simboli adottato per rappresentare gli elementi di un insieme [C*]. Sono *codici*, ad esempio, l'insieme delle

parole di una lingua o dei numeri decimali; *simboli* di questi codici sono, rispettivamente, le lettere dell'alfabeto e le cifre da 0 a 9. *Parole* di un codice sono le combinazioni di simboli che hanno significato; *codificazione* è l'operazione per cui una parola del codice [C] viene fatta corrispondere a un elemento dell'insieme [C*].

Un codice si dice *non ambiguo* quando la corrispondenza tra le sue parole e gli elementi di [C*] è univoca; *ambiguo* quando almeno una parola di [C] rappresenta almeno due elementi di [C*].

Perché un codice a K simboli rappresenti in modo non ambiguo N elementi di [C*], le sue parole debbono avere una lunghezza non inferiore a una determinata lunghezza minima. Precisamente, considerando le parole come numeri di base K, la lunghezza minima eguaglia il numero di cifre necessarie per scrivere il numero N. Poiché il più alto numero rappresentabile con n cifre è:

$$K^n - 1 ,$$

tale numero è il più piccolo intero n tale che:

$$K^n \geq N$$

cioè:

$$n \geq \log_k N \quad .$$

Ad esempio, per codificare 500 elementi in un codice binario, occorrono 9 bit, perché:

$$2^9 = 512 > 500 > 2^8 = 256 \quad .$$

Coi simboli finora adottati, un codice si dice *efficiente* se le sue parole hanno lunghezza $l = n$; *ridondante* se $l > n$; *ambiguo* se $l < n$.

I codici sono usati in svariati campi, con finalità estremamente diverse; rimandando ai trattati specializzati per la relativa teoria, intendiamo qui illustrare alcuni dei codici più usati per la trasmissione e l'elaborazione, col solo uso dei simboli 0 e 1, di messaggi alfabetici e/o numerici.

1.10 - Codici per la rappresentazione di cifre decimali.

I codici più semplici sono quelli che si propongono di rappresentare i numeri decimali codificandone ogni cifra indipendentemente dalle altre.

Poiché una cifra decimale si esprime con 4 bit, e $2^4 = 16$, rimangono 6 configurazioni non sfruttate. Per rappresentare un numero, occor-

rono quindi più bit che nel sistema di numerazione binario; in compenso, le espressioni numeriche ottenute sono più semplicemente interpretabili.

I.10.1 - Codice BCD.

La codificazione delle cifre 0 ÷ 9 nel codice BCD (Binary-Coded Decimal) avviene secondo il sistema di numerazione binario (fig.I.4). Non vengono adoperate le configurazioni corrispondenti ai numeri decimali 10 ÷ 15.

TABELLA I.4 - Codice BCD

Decimale	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Fig.I.4

Esempio: *I numeri 37, 15, 91 hanno le rappresentazioni:*

37 : 00110111
15 : 00010101
91 : 10010001

Il codice BCD è *ponderato*: si chiamano così quei codici binari in cui ogni bit ha un valore, non necessariamente positivo, a seconda della posizione che occupa nella parola. In questo caso, i valori dei bit sono, da sinistra verso destra: 8, 4, 2, 1.

I.10.2 - Codice «Eccesso a tre».

La codificazione della cifra K (K = 0 ... 9) avviene esprimendo nel sistema di numerazione binario il numero K + 3 (fig.I.5).

TABELLA I.5 - Codice «Eccesso a tre»

Decimale	0	1	2	3	4	5	6	7	8	9
Eccesso a tre	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100

Fig.I.5

Esempio: *I numeri 37, 15, 91 hanno le rappresentazioni:*

37 : 01101010
15 : 01001000
91 : 11000100

Il codice *eccesso a tre* non è ponderato, ma è *autocomplementare*: il complemento a 9 di ogni cifra – o numero – decimale si ottiene, cioè, semplicemente per inversione dei bit.

Esempio: Il complemento a 9 di 3 (0110) è 6 (1001): il complemento a 9 di 37 (01101010) è 62 (10010101).

I codici autocomplementari sono stati usati in alcuni elaboratori, perché realizzano con facilità le operazioni aritmetiche.

I.10.3 - Codice «Aiken 2421».

E' un codice – insieme – ponderato ed autocomplementare. I valori dei bit per ogni cifra, eccetto lo zero, sono: 2, 4, 2, 1. La codificazione avviene secondo la tabella della fig.I.6.

TABELLA I.6 - Codice di Aiken

Decimale	0	1	2	3	4	5	6	7	8	9
Aiken	0000	0001	0010	0011	0100	1011	1100	1101	1110	1111

Fig.I.6

Altri codici per la rappresentazione delle cifre decimali verranno illustrati nel prossimo paragrafo:

I.11 - Codici ridondanti.

I codici ridondanti adoperano, nella codificazione degli N elementi di un insieme [C*], un numero (m) di bit superiore a quello (n) strettamente necessario. Come l'alta ridondanza di una lingua, parlata o scritta, permette spesso la comprensione di una comunicazione, anche se il messaggio che la codifica è ricevuto in forma non completamente corretta, così la ridondanza di un codice permette di rilevare o correggere gli errori intervenuti durante qualsiasi fase di trasmissione di un messaggio.

La ridondanza viene creata aggiungendo, secondo una determinata legge, agli n bit che codificano il messaggio, K bit di controllo. Viene quindi trasmesso un messaggio di $m = n + K$ bit. Si chiama *ridondanza il rapporto:*

$$R = \frac{m}{n} = 1 + \frac{K}{n} \quad .$$

La mancata osservanza, in ricezione, della legge che lega i bit di controllo a quelli di informazione, denuncia un errore. La verifica va fatta, ovviamente, su tutti i bit del messaggio.

Delle 2^m configurazioni diverse che è possibile formare con m bit, solo 2^n hanno significato, cioè sono *parole* del codice; le restanti $2^m - 2^n = 2^n (2^K - 1)$ si chiamano *configurazioni non significative*. Un errore viene rivelato se trasforma una parola in una configurazione non significativa; la probabilità di rivelare gli errori è, dunque, tanto maggiore quanto più alta è la ridondanza.

I codici ridondanti trovano impiego negli elaboratori e nei sistemi di telesegnalazione e trasmissione dati.

I.11.1 - Definizioni.

Si chiama *peso* il numero di bit 1 presenti in una certa configurazione, significativa o no. Ad esempio, le configurazioni 0110100 e 0010000 hanno, rispettivamente, pesi 3 e 1.

Distanza tra 2 configurazioni, C_1 e C_2, dello stesso codice, è il numero delle posizioni in cui i bit di C_1 hanno valore diverso da quelli di C_2. Ad esempio, le configurazioni 0000 e 1010 hanno distanza 2; le configurazioni 0101 e 1010 hanno distanza 4.

Invertendo qualche bit, gli errori di trasmissione possono trasformare una parola P in una configurazione non significativa C_p. *Molteplicità* dell'errore è la distanza tra la parola P trasmessa e la configurazione C_p ricevuta. Errori *singoli*, *doppi*, *tripli*, ... sono quelli che originano configurazioni C_p a distanza 1, 2, 3, ... da una certa P.

Si chiama *distanza minima di Hamming (h)* la minima distanza fra tutte le possibili coppie di parole di un codice. Poiché gli errori sono riconoscibili se trasformano una parola P in una configurazione non significativa C_p, ma non lo sono se la trasformano in una parola P', saranno sicuramente individuabili i soli errori la cui molteplicità è inferiore ad h. Si chiamano *rivelatori d'errori* quei codici con $h > 1$.

L'individuazione di un errore non è sufficiente a correggerlo. Se, però, h è abbastanza grande e si suppone che C_p provenga dalla P che si trova alla distanza minore, si può effettuare la correzione, interpretando C_p come P. I codici che funzionano in questo modo si chiamano *autocorrettori*, e debbono avere una ridondanza sensibilmente elevata.

I.11.2 - Probabilità totale di errore.

La probabilità totale di errore di un codice ridondante si calcola conoscendo la probabilità p, dipendente dalle caratteristiche fisiche del sistema di trasmissione, che un'impulso venga interpretato in maniera errata.

Supposti gli errori tutti indipendenti tra loro, la probabilità che r bit di una parola siano ricevuti in maniera errata è p^r. Se le parole sono di m bit, $(1-p)^{m-r}$ è la probabilità che i restanti $(m-r)$ bit siano esatti. La probabilità P_r che una parola P si trasformi in una determinata configurazione C_r^*, a distanza r da P, poiché C_r^* ha r bit eguali ed $(m-r)$ bit diversi da P stessa, è:

$$P_r = p^r (1-p)^{m-r} \quad .$$

L'errore intervenuto verrà riconosciuto se C_r^* non è una parola del codice, non lo sarà nel caso opposto.

Se, a distanza r, si trovano C_r parole del codice, la probabilità di avere un errore di molteplicità r non rivelato è:

$$P_r = C_r \cdot p^r (1-p)^{m-r} \quad .$$

Per una determinata parola, la probabilità totale di errore non rivelato è:

$$P_t = \sum_{r=h}^{m} C_r p_r (1-p)^{m-r} \quad .$$

Se $p \ll 1$, come è sempre in pratica, si può assumere, per qualsiasi r:

$$(1-p)^{m-r} \cong 1$$

e, trascurando l'influenza degli errori di molteplicità $>h$:

$$P_t \cong C_h p^h \quad .$$

Per un codice caratterizzato dalla lunghezza m e dalla distanza h, P_t sarà noto se si conoscerà il numero C_h delle parole a distanza h da una sua generica parola.

I.11.3 - Codice a controllo di parità.

I codici a controllo di parità si costruiscono aggiungendo agli n bit del segnale un bit di controllo, il cui valore è tale da rendere pari il peso degli $n+1$ bit di ogni parola.

Ad esempio, con il controllo di parità, le configurazioni 0111 e 0110 rappresentanti le cifre decimali 7 e 6 nel codice BCD diventano 10111 e 00110.

Un codice a controllo di parità ha $h = 2$, e rivela tutti e soli gli errori di molteplicità dispari.

Per dare un'idea dei valori di R, C_h e P_t, facciamo il caso pratico di segnali di 6 bit. Si ha: $n = 6$, $k = 1$, $m = 7$, $R = m/n = 7/6 = 1,16$; esistono $2^6 = 64$ parole e altrettante configurazioni non significative. Il valore di C_h è dato dalla relazione:

$$C_h = \binom{m}{h} = \binom{7}{2} = \frac{7!}{2!\,5!} = 21 \ .$$

Supponendo $p = 0,01$, si ha una probabilità totale di errore non rivelato:

$$P_t = C_h\, p^h = C_2\, p^2 = 21 \cdot (10^{-2})^2 = 0,21\,\% \ .$$

I.11.4 - Codici a peso costante.

Nei codici a peso costante, sono *parole* tutte le configurazioni aventi un determinato peso p; in essi, pertanto: $h = 2$.

Se le parole hanno lunghezza m, esistono $\binom{m}{p}$ parole, e $[2^m - \binom{m}{p}]$ configurazioni non significative. Il numero delle parole, a parità di m, è minore di quello dei codici a controllo di parità, che pure hanno lo stesso valore di h. Si ottiene, in compenso, una maggior protezione dagli errori, essendo rivelati, oltre agli errori dispari, anche quelli pari che originano combinazioni di peso diverso da p.

Il numero delle parole a distanza 2 da ogni parola del codice è:

$$C_2 = p\,(m - p)$$

in ogni parola, infatti, esistono p bit di valore 1 ed $(m - p)$ bit di valore 0, quindi $(m - p)$ modi diversi di scambiare un 1 con uno 0 senza alterare il peso.

Due codici di questo tipo, particolarmente importanti, sono quelli usati per codificare le cifre $0 \div 9$ del sistema decimale: il codice *2 su 5* e il codice *biquinario*.

I.11.4.1 - Codice 2 su 5.

Ogni cifra decimale viene rappresentata con 5 bit, in modo che il suo peso sia costantemente uguale a 2. I bit che rappresentano ogni ci-

fra, ad eccezione dello zero, hanno, da sinistra verso destra, i valori 0, 1, 2, 3, 6 (fig.I.7).

TABELLA I.7 - Codice 2 su 5										
Decimale	0	1	2	3	4	5	6	7	8	9
2 su 5	01100	11000	10100	10010	01010	00110	10001	01001	00101	00011

Fig.I.7

La ridondanza per cifra decimale è:

$$R = \frac{5}{4} = 1,25 \ .$$

La probabilità totale d'errore, per essere $C_2 = 2(5-2) = 6$, nel caso di $p = 0,01$ è:

$$P_t = 6 \cdot (10^{-2})^2 = 0,06\ \% \ .$$

I.11.4.2 - Codice Biquinario.

E' un codice ad elevata ridondanza, con $m = 7$ e $p = 2$, nel quale ogni cifra decimale è codificata con 7 bit i cui valori, da sinistra verso destra, sono: 0, 5; 0, 1, 2, 3, 4 (fig.I.8).

TABELLA I.8 - Codice Biquinario										
Decimale	0	1	2	3	4	5	6	7	8	9
Biquinario	1010000	1001000	1000100	1000010	1000001	0110000	0101000	0100100	0100010	0100001

Fig.I.8

Sui 7 bit di ogni parola vengono effettuati due controlli, per verificare che le prime 2 e le ultime 5 posizioni abbiano peso 2. E' così possibile rilevare una parte degli errori che trasformano la parola in una configurazione di peso 2.

La ridondanza per cifra decimale è:

$$R = \frac{7}{4} = 1,75 \ .$$

La probabilità totale di errore, per essere $C_r = 5$, supponendo $p = 0,01$, è:

$$P_t = 5 \cdot (10^{-2}) = 0,05\% \ .$$

I.11.5 - Codici di Hamming.

R.W. Hamming ha proposto dei codici, con $h = 3$ ed $h = 4$, da usare come rivelatori o autocorrettori d'errore. I codici con $h = 3$ individuano gli errori singoli e doppi, oppure correggono i soli errori singoli, senza individuare i doppi perché, interpretando ogni configurazione non significativa come originata dalla parola più vicina, trasformano quelle con 2 bit errati in una parola diversa dall'originale.

I codici con $h = 4$ individuano gli errori singoli, doppi, tripli; se usati come autocorrettori, correggono gli errori singoli e individuano i doppi che però, originando configurazioni equidistanti da almeno 2 parole, non possono essere corretti.

In generale, indicando con r il numero di errori che si possono rilevare, e con c quelli che possono essere corretti, si ha la relazione:

$$r = h - 1$$

per i codici rivelatori d'errore, e le relazioni:

$$\begin{cases} c < \dfrac{h}{2} \\ c + r < h \end{cases}$$

per gli autocorrettori.

I.11.5.1 - Codice di Hamming con $h = 3$.

Questo codice viene costruito aggiungendo agli n bit di un codice efficace, K bit di controllo, ognuno dei quali verifica la parità di un gruppo degli $m = n + k$ bit risultanti. Più precisamente, lo i^{mo} bit di controllo va messo nella posizione 2^{i-1} della parola, e controlla la parità di gruppi alternati di 2^{i-1} bit, a cominciare dalla posizione 2^{i-1}.

Il primo dei bit di controllo, dunque, va nella prima posizione, e controlla la parità dei bit 1, 3, 5, 7, 9, ...; il secondo bit, nella seconda posizione, controlla i bit (2, 3), (6, 7), (10, 11), ...; il terzo bit, nella quarta posizione, controlla i bit (2, 5, 6, 7), (12, 13, 14, 15), ...

In ricezione, verificata la parità di tutti i gruppi, a cominciare dal primo, si scrive 0 per ogni verifica esatta, 1 per ogni verifica errata;

i k bit ottenuti, letti da destra verso sinistra, formano un numero (N_c) detto *numero di controllo*. Se $N_c = 0$, il messaggio è esatto; ognuno degli altri $2^k - 1$ possibili valori di N_c indica, invece, in quale delle m posizioni si è verificato un errore di molteplicità 1. Deve pertanto essere:

$$m \leq 2^k - 1 \quad . \tag{I.1}$$

Per costruire uno di questi codici, si comincia col calcolare k, il cui valore è determinato da n. Il valore minimo di k è 2, per n = 1; dalla (I.1) si ha poi:

$$k = 3 \quad \text{per} \quad n = 2, 3, 4$$
$$k = 4 \quad \text{per} \quad n = 5, 6, \ldots, 10, 11 \; .$$

Si chiamano *ottimi* quei codici per cui $m = 2^k - 1$: essi, a parità di k, hanno n massimo, contengono cioè il massimo numero di parole.

Per chiarire le idee, supponiamo di dover codificare 16 simboli, allora:

$$n = 4$$
$$k = 3$$
$$m = n + k = 7 \; .$$

I bit $k_1 k_2 k_3$ di controllo vanno inseriti nelle posizioni 1, 2, 4 della parola, la cui composizione pertanto, se b_i è il bit i^{mo} del messaggio, sarà:

$$k_1 k_2 b_1 k_3 b_2 b_3 b_4$$

k_1 controlla le posizioni 1, 3, 5, 7 ($k_1 b_1 b_2 b_4$);

k_2 controlla le posizioni 2, 3, 6, 7 ($k_2 b_1 b_3 b_4$);

k_3 controlla le posizioni 4, 5, 6, 7 ($k_3 b_2 b_3 b_4$).

Aggiungendo i bit di controllo, per esempio al messaggio 1001 si otterrà la parola:

$$k_1 k_2 1 k_3 001$$

e, per essere $k_1 = k_2 = 0$, $k_3 = 1$, la parola codificata sarà:

$$0011001 \; .$$

Supponiamo che un errore singolo sul terzo bit trasformi la parola stessa nella configurazione non significativa:

$$0001001 \; .$$

Alla verifica, i primi due controlli di parità (posizioni 1, 3, 5, 7 e 2, 3, 6, 7) danno risultati errati, quindi gli ultimi 2 bit di N_c sono 1. Il terzo controllo (posizioni 4, 5, 6, 7) è esatto: si ha $N_c = 011$: pertanto il bit errato è quello nella terza posizione.

Si noti che N_c individua anche errori intervenuti sui bit di controllo; per esempio, per la configurazione 1011001, proveniente da 0011001, si ha $N_c = 001$.

La ridondanza del codice in esame è:

$$k = \frac{7}{4} = 1{,}75 \ .$$

Il numero delle parole a distanza 3 da ogni altra è certamente < 15; se $p = 0{,}01$, si ha una probabilità totale di errore:

$$P_t = C \cdot p^3 < 15\,(10^{-2})^3 = 0{,}0015\ \% \ .$$

I.11.5.2 - Codice di Hamming con h = 4.

I codici con $h = 4$ si costruiscono aggiungendo a quelli con $h = 3$ un quarto di bit di controllo, in ultima posizione, per controllare la parità di tutti i precedenti.

Con $h = 4$, il minimo valore di k è 3, per $n = 1$; si ha poi:

$$k = 4 \qquad \text{per} \quad n = 2, 3, 4$$
$$k = 5 \qquad \text{per} \quad n = 5, 6, \ldots, 9, 10 \ .$$

La verifica di un segnale si fa controllandone i k bit di parità. Se i primi $(k - 1)$ controlli sono non tutti esatti, mentre l'ultimo lo è, si è verificato un doppio errore, non correggibile. Se i primi $(k - 1)$ sono esatti e l'ultimo no, c'è un errore nell'ultimo bit di controllo. Se sono errati l'ultimo bit di controllo e qualcuno dei $(k - 1)$ precedenti, c'è stato un errore singolo, correggibile attraverso N_c.

Supponendo di derivare un codice con $h = 4$ da quello con $h = 3$ ed $n = 4$ del paragrafo precedente, si ha:

$$R = \frac{8}{4} = 2 \ .$$

Le parole del codice sono 16; quelle a distanza 4 da una certa parola sono certamente < 15; supponendo ancora $p = 0{,}01$; la probabilità totale di errore non rivelato è:

$$P_t < 15 \cdot p^4 = 1{,}5 \cdot 10^7 \quad .$$

Aggiungendo un altro bit di controllo si è, perciò, diminuito fortemente P_1.

I.12 - Codici riflessi.

I codici riflessi sono largamente usati nei convertitori analogico-digitali perché è possibile disporre le loro parole in ordine tale che ognuna si trovi sempre a distanza 1 dalla precedente e dalla seguente. Così, se si ordinano le parole stesse secondo i numeri naturali da 0 a N, il numero k viene rappresentato da una configurazione a distanza 1 dai numeri (k - 1) e (k + 1).

I codici riflessi non sono adoperati in trasmissione, per il basso valore di C_r, e risultano di non agevole uso negli elaboratori, perché non sono ponderati (le posizioni dei bit, cioè, non hanno un particolare significato). L'elaborazione dei numeri ottenuti dalla quantizzazione di grandezze continue e rappresentati in questi codici, richiede perciò la conversione in un opportuno codice binario. Tra i numerosi codici riflessi esistenti acquista, a questo proposito, particolare importanza il codice di Gray, che può essere semplicemente convertito in forma binaria.

I.12.1 - Codice di Gray.

Nella fig.I.9 è mostrato il codice di Gray a 4 bit, per rappresentare i numeri decimali da 0 a 15.

Dal codice $[G_n]$ di Gray a n bit si può ricavare quello $[G_{n+1}]$ a (n + 1) bit con il seguente procedimento:

- si assegna al numero decimale $2^n + k$ la stessa configurazione di bit che rappresenta il numero $2^n - (k + 1)$, per $k = 0, 1, \ldots, (2^n - 1)$;
- si premette a tutti i numeri $< 2^n$ un bit 0, e a tutti quelli $\geq 2^n$ un bit 1.

In pratica, basta ribaltare attorno a un asse di simmetria speculare il codice a n bit e premettere 0 e 1 alle 2 immagini ottenute. Ad esempio, ribaltando intorno ad un asse di simmetria speculare il codice di Gray a 2 bit (00 - 01 - 11 - 10):

00
01
11
10

10
11
01
00

si ottiene il codice a 3 bit: 000 - 001 - 011 - 010 - 110 - 111 - 101 - 100.

Come si vede dalla fig.I.9, anche il passaggio dal numero 7 al numero 8, che nel sistema binario cambia tutti i bit (da 0111 a 1000) avviene con una sola inversione (da 0100 a 1100): l'errore derivante da una incertezza nella valutazione di una grandezza analogica, nel primo caso assolutamente intollerabile, diviene quindi accettabile.

TABELLA I.9 - Codice di Gray per i numeri decimali 0 ÷ 15

Decimale	Gray	Decimale	Gray
0	0000	8	1100
1	0001	9	1101
2	0011	10	1111
3	0010	11	1110
4	0110	12	1010
5	0111	13	1011
6	0101	14	1001
7	0100	15	1000

Fig.I.9

La conversione di una parola dal codice di Gray (P_G) al codice binario (P_B) avviene con le seguenti regole:

– si procede da sinistra verso destra: fino al primo bit 1 di P_G, $P_B \equiv P_G$. In seguito:

– quando il bit k^{mo} di $P_G = 0$, il bit k^{mo} di P_B è uguale al bit $(k-1)^{mo}$ di P_B;

– quando il bit k^{mo} di $P_G = 1$, il bit k^{mo} di P_B è opposto al bit $(k-1)^{mo}$ di P_B.

Esempio: *Convertire in codice binario la parola del codice di Gray:*

$$P_G \equiv 010101100111000111100001 \quad .$$

I primi 2 bit di P_B sono eguali a quelli di P_G, trovandosi il primo bit 1 in seconda posizione. Il terzo bit di P_B, per esser il terzo bit di $P_G = 0$, è uguale al secondo bit di P_B (1). Il quarto bit di P_B, essendo il corrispondente $P_G = 1$, è opposto al terzo bit di P_B. La conversione completa è riportata nella fig.I.10.

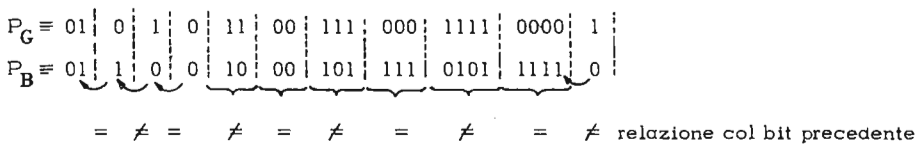


Fig.I.10 - Conversione dal codice di Gray al codice binario.

La conversione di una parola dal codice binario (P_B) al codice di Gray (P_G) avviene con le seguenti regole:

– si procede da sinistra verso destra, paragonando il k^{mo} col $(k-1)^{mo}$ bit di P_B. Se i due bit sono uguali, il k^{mo} bit di P_G è 0; se sono diversi è 1;

– il primo bit di P_G va confrontato con 0.

Esempio: *Convertire in codice di Gray la parola binaria:*

$$P_B \equiv 11101010101111110 \ .$$

Il primo bit di P_B è diverso da 0, quindi P_G inizia con 1.
Il secondo bit di P_B è uguale al precedente: il corrispondente $P_G = 0$.
La conversione completa è riportata nella fig.I.11.

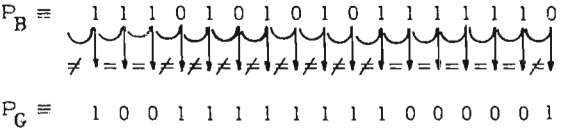


Fig.I.11 - Conversione dal binario al codice di Gray.

*

BIBLIOGRAFIA

1. SMITH D.E. & GINSBURG J.: *From Numbers to Numerals and form Numerals to Computation* - The World of Mathematics, vol.I, pagg.442-464 - New York 1956.

2. BROOKS F.P.- IVERSON K.E.: *Automatic Data Processing* - John Wiley & Sons, 1963.

3. RICHARDS R.K.: *Arithmetic Operations in Digital Computers* - D.Van Nostrand Co. Princeton, 1955.

4. PETERSON W.W.: *Error Correcting Codes* - John Wiley & Sons, 1960.

5. KANTZ W.M.: *Codes and Coding Circuity for automatic Error Correction within Digital Systems* - Reaunaancy Techniques for Computing Systems. Wilcox & Mann, 1962.

6. MILLER E.R.: *Switching Theory* - vol.I - John Wiley & Sons, 1965.

7. *A Programmer's Introduction to the IBM System/360 Architecture, Instructions and Assembler Language* - IBM - Student Text, 1967.

8. CHU Y.: *Digital Computer Design Fundamentals* - Mc Graw-Hill, 1962.

9. BARON R.C. - PICCIRILLI A.T.: *Digital Logic and Computer Operations* - Mc Graw-Hill, 1968.

*

CAPITOLO II

ALGEBRA BOOLEANA

II.1 - Generalità.

Il matematico inglese George Boole (1815-1864) nel 1847 espose nel libro «*Mathematical Analysis of Logic*» le regole fondamentali di un algoritmo per studiare i problemi della logica deduttiva, algoritmo che sviluppò e completò in una seconda opera pubblicata nel 1858 «*An Investigation of the laws of tought*». L'algebra binaria cui egli pervenne contemplava due soli valori, l'1 e lo 0, che rappresentavano il vero e il falso in una proposizione, e rimase una teoria matematica applicata al solo studio della logica fino al 1938, quando Claude Shannon ne adottò il simbolismo per l'analisi dei circuiti di commutazione a contatti. Da allora, l'algebra booleana è diventata uno strumento di fondamentale importanza per il progetto dei circuiti logici, in particolare per quelli elettronici dei calcolatori numerici.

Shannon ebbe l'idea di usare i valori 0 e 1 dell'algebra booleana per definire lo stato di apertura o di chiusura di un generico contatto o di un circuito a più contatti; in seguito l'1 e lo 0 indicarono la presenza o l'assenza di un determinato segnale in un punto di un circuito elettronico o, più generalmente, gli stati di qualsiasi elemento capace di assumere l'uno o l'altro di due stati opposti.

In questo capitolo esporremo i concetti e le proprietà fondamentali dell'algebra logica, limitatamente all'uso che ne faremo nello studio dei circuiti di commutazione, e del tutto indipendentemente dall'algebra delle classi e delle proposizioni da cui, più elegantemente, si usa oggi farla derivare.

II.2 - Definizioni e postulati.

L'algebra booleana considera l'esistenza di due elementi distinti, indicati con i simboli «0» e «1». Tali simboli non rappresentano numeri né quantità definite, ma debbono intendersi piuttosto come stati o condizioni caratteristiche, mutuamente escludentisi. Una costante binaria può dunque valere soltanto 0 o 1.

Tra due costanti si definisce un'operazione di somma, indicata col segno «+», per la quale valgono le seguenti regole:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 1 \ . \end{aligned}$$

Si definisce anche un'operazione di prodotto, indicato col segno «·», per la quale si ha:

$$\begin{aligned} 0 \cdot 0 &= 0 \\ 0 \cdot 1 &= 0 \\ 1 \cdot 0 &= 0 \\ 1 \cdot 1 &= 1 \ . \end{aligned}$$

Si noti che:

a) le operazioni di somma e di prodotto sono commutative;

b) le regole del prodotto si ottengono da quelle della somma scambiando lo 0 con lo 1 e il + con il · (principio di dualità).

Si definisce infine una terza operazione, chiamata negazione e indicata sol simbolo «¯», tale che inverte il valore della costante su cui opera.

$$\begin{aligned} \overline{0} &= 1 \\ \overline{1} &= 0 \ . \end{aligned}$$

Per definizione:

$$\begin{aligned} \overline{\overline{0}} &= 0 \\ \overline{\overline{1}} &= 1 \ . \end{aligned}$$

II.3 - Variabili.

Variabile binaria indipendente è una grandezza matematica capace di assumere l'uno o l'altro dei due valori 0 e 1.

Anche per le variabili si definiscono le operazioni di somma, prodotto, negazione.

Se $x_1 x_2 \ldots x_n$ sono n variabili binarie indipendenti, la loro somma Σ ha l'espressione:

$$\Sigma = x_1 + x_2 + \ldots + x_n \ ;$$

si ha $\Sigma = 1$ se almeno una delle x vale 1; $\Sigma = 0$ se $x_1 = x_2 = \ldots = x_n = 0$.

Il prodotto Π ha l'espressione:

$$\Pi = x_1 \cdot x_2 \cdot \ldots \cdot x_n \ ;$$

si ha $\Pi = 1$ se $x_1 = x_2 = \ldots = x_n = 1$, $\Pi = 0$ se almeno una delle x è 0.

La negazione della variabile x si indica con:

$$\overline{x}$$

(si legge «non x») ed è quell'operazione per cui:

$$\text{se} \quad x = 1, \qquad \overline{x} = 0$$
$$\text{se} \quad x = 0, \qquad \overline{x} = 1 .$$

Per le operazioni di somma e prodotto valgono le relazioni:

$x + 1 = 1$	e le duali	$x \cdot 0 = 0$
$x + 0 = x$		$x \cdot 1 = x$
$x + x = x$		$x \cdot x = x$.

Ognuna di queste relazioni può essere provata dando alla x i 2 valori possibili. Ad esempio, per verificare che $x \cdot 0 = 0$, basta mostrare che se $x = 0$, si ha:

$$x \cdot 0 = 0 \ \rightarrow \ 0 \cdot 0 = 0 ,$$

e se $x = 1$:

$$x \cdot 0 = 0 \ \rightarrow \ 1 \cdot 0 = 0 .$$

Valgono anche le proprietà:

- commutativa: $x_1 + x_2 = x_2 + x_1 \quad (x_1 x_2 = x_2 x_1)$;
- associativa: $x_1 + x_2 + x_3 = x_1 + (x_2 + x_3) \quad (x_1 \cdot x_2 \cdot x_3 = x_1 \cdot (x_2 \cdot x_3))$;
- distributiva: $x_1 x_2 + x_1 x_3 = x_1 (x_2 + x_3) \quad ((x_1 + x_2)(x_1 + x_3) = x_1 + x_2 x_3)$.

Per l'operazione di inversione si hanno invece le relazioni:

$$x + \overline{x} = 1 \qquad (x \cdot \overline{x} = 0)$$

$$\overline{\overline{x}} = x .$$

La cui prova è immediata, se si pone prima $x = 0$, poi $x = 1$.

Le variabili binarie $x_1 x_2 \ldots x_n$, nel loro insieme, possono dar luogo a 2^n diverse configurazioni di valori 0 e 1. La scrittura $x_1 x_2 x_3 = 011$ considera, tra le $2^3 = 8$ configurazioni possibili di $x_1 x_2 x_3$, quella in cui x_1 vale 0 ed x_2, x_3 valgono 1. Questa configurazione si indica semplicemente, per convenzione, col prodotto $\overline{x}_1 x_2 x_3$. Le configurazioni di n variabili coincidono, dal punto di vista formale, con l'espressione dei numeri $0, 1, \ldots, 2^n - 1$ nel sistema di numerazione binario.

II.4 - Funzioni.

Si dice che una variabile y è funzione delle n variabili indipendenti $x_1 x_2 \ldots x_n$, e si scrive:

$$y = F(x_1, x_2, \ldots, x_n)$$

quando esiste un criterio che fa corrispondere in modo univoco, ad ognuna delle 2^n configurazioni delle x, un valore della y.

Il numero delle funzioni di n variabili è, pertanto, 2^{2^n}.

Dalla definizione, deriva una prima rappresentazione delle funzioni, quella ottenuta scrivendo tutte le possibili ennuple di valori (o configurazioni) delle n variabili con – accanto – i valori della y (tabella di verità della funzione).

A titolo d'esempio, nella fig.II.1 è riportata la tabella di verità per una qualsiasi delle $2^{2^3} = 256$ funzioni di 3 variabili.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Fig.II.1 - Tabella di verità per una funzione di 3 variabili.

Dalla tabella di verità, dalla rappresentazione di una generica configurazione di variabili, e dalla definizione di somma, si può ricavare una prima espressione algebrica di una funzione logica. Poiché, infatti, ogni funzione y di $x_1 x_2 \ldots x_n$ vale 1 per un certo numero p di configurazioni delle variabili, ed ogni configurazione si esprime col prodotto delle variabili dirette o negate, la y può scriversi come la somma di tali prodotti.

Ad esempio, la funzione della fig.II.1 vale 1 per le configurazioni 000, 010, 011, 101 delle variabili $x_1x_2x_3$, pertanto si scriverà:

$$y = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2\overline{x}_3 + \overline{x}_1x_2x_3 + x_1\overline{x}_2x_3 \ .$$

Le espressioni così ottenute, che nei prossimi paragrafi trasformeremo in vari modi, si chiamano *forme canoniche in somme di prodotti* della funzione, o semplicemente *forme canoniche*. I prodotti che compaiono come addendi della somma si chiamano *prodotti elementari* o *minterm* (termini minimi) della y: in seguito, adotteremo esclusivamente quest'ultima denominazione.

Un ulteriore modo, estremamente sintetico, per rappresentare una funzione è quello di scriverla come somma dei numeri decimali che corrispondono ai minterm letti come numeri binari.

Ad esempio, la funzione della fig.II.1, interpretando le configurazioni 000, 010, 011, 101 come i numeri decimali 0, 2, 3, 5, si scrive:

$$y = \Sigma\ (0, 2, 3, 5) \ .$$

Riportiamo, ora, i principali teoremi dell'algebra logica, illustrandone il significato, ma tralasciandone la dimostrazione, per la quale rimandiamo ai testi citati nella bibliografia.

II.5 - Teoremi.

Chiamate x, y, z tre generiche grandezze booleane (costanti, variabili o funzioni), si hanno i seguenti teoremi:

Teorema 1 – $x + xy = x$ (primo teorema dell'assorbimento): il valore dell'espressione $x + xy$ dipende esclusivamente dal valore di x. L'espressione duale di questo teorema, che si può verificare dando a x e y tutti i possibili valori, è: $x \cdot (x + y) = x$.

Teorema 2 – $x + \overline{x}y = x + y$ (secondo teorema dell'assorbimento): il valore dell'espressione $x + \overline{x}y$ è uguale a quello della somma $x + y$. La verifica è analoga a quella precedente; l'espressione duale è: $x(\overline{x} + y) = xy$.

Teorema 3 – $xy + yz + \overline{x}z = xy + \overline{x}z$ (terzo teorema dell'assorbimento): il prodotto yz ha valore sempre uguale a quello della somma $xy + \overline{x}z$. Si può verificare dando a x, y, z tutte le otto possibili terne di valori. La espressione duale è: $(x + y)(y + z)(\overline{x} + z) = (x + y)(\overline{x} + z)$.

Teorema 4 – $\overline{(x + y)} = \overline{x} \cdot \overline{y}$: la negazione di una somma ha lo stesso valore del prodotto degli addendi negati. L'espressione duale è $\overline{xy} = \overline{x} + \overline{y}$.

I due teoremi possono essere riuniti in uno solo, il fondamentale *teorema di De Morgan*, che si esprime simbolicamente con:

$$\overline{f}(x_1, x_2, \ldots, x_n, +, \cdot) = f(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n, \cdot, +) ,$$

cioè: la negazione di una funzione si ottiene negando ogni variabile e scambiando tra loro le operazioni di somma e di prodotto.

Esempio: la negazione della funzione:

$$F_1 = \overline{x}y + x\overline{y}$$

è la:

$$F_2 = \overline{F}_1 = (x + \overline{y})(\overline{x} + y) = xy + \overline{x}\,\overline{y} .$$

Infatti, si vede dalla tabella di verità delle due funzioni (fig.II.2) che F_1 vale 1 per le configurazioni per cui F_2 vale 0, e viceversa.

Fig.II.2 - Tabelle di verità di una funzione e della sua negazione.

x	y	F_1
0	0	0
0	1	1
1	0	1
1	1	0

x	y	F_2
0	0	1
0	1	0
1	0	0
1	1	1

Il teorema di De Morgan ci permette di ricavare una seconda espressione algebrica delle funzioni logiche, deducibile anch'essa dalla tabella di verità. La funzione y si può concepire come la negazione della funzione $\overline{y}$, scritta – in forma canonica – come somma dei k_0 minterm che corrispondono agli 0 della funzione y stessa. Applicando all'espressione della $\overline{y}$ il teorema di De Morgan, si ottiene la y come prodotto di k_0 somme delle variabili $x_1 x_2 \ldots x_n$. Questa nuova rappresentazione della y si chiama *forma canonica come prodotto di somme*: ognuna delle somme delle n variabili, dirette o negate, prende il nome di *maxterm* (termine massimo).

Ad esempio, considerando la funzione y di 3 variabili la cui tabella di verità è riportata nella fig.II.1, si hanno i valori 0, quindi i valori 1 della $\overline{y}$, in corrispondenza alle configurazioni: 001, 100, 110, 111. Quindi si scriverà:

$$\overline{y} = \overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1x_2\overline{x}_3 + x_1x_2x_3$$

e, applicando il teorema di De Morgan:

$$y = \overline{\overline{y}} = \overline{\overline{x}_1\overline{x}_2x_3 + x_1\overline{x}_2\overline{x}_3 + x_1x_2\overline{x}_3 + x_1x_2x_3}$$

$$= (x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_2 + x_3)(\overline{x}_1 + \overline{x}_2 + x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) .$$

L'espressione della y come prodotto di maxterm si può ottenere direttamente dalla tabella di verità: ci sono tanti maxterm quanti sono gli 0 della y; ogni maxterm è la somma di tutte le variabili, dirette o negate a seconda che la corrispondente configurazione della tabella di verità contiene 0 o 1.

Si noti che, se la forma canonica come prodotto di maxterm contiene q termini, quella come somma di minterm ne contiene $2^n - q$; la prima comprende, dunque, un minor numero di termini della seconda se $q < 2^{n-1}$.

II.6 - Un'interpretazione fisica dell'algebra booleana.

La prima applicazione pratica dell'algebra booleana venne fatta ai circuiti di commutazione a contatti. Vogliamo qui accennare brevemente, anche se intendiamo occuparci soltanto di circuiti logici elettronici, sia per ragioni storiche, sia perché la tecnica usa ancora largamente i circuiti a contatti, per esempio in campo telefonico.

Nei circuiti a contatti, i simboli 0 e 1 indicano, rispettivamente, che un circuito elettrico è aperto o chiuso. Una variabile x può indicare un interruttore o un contatto mossò da un relè. Se la stessa lettera indica più contatti, questi si intendono tutti azionati simultaneamente dallo stesso organo. I simboli x e $\overline{x}$ indicano due contatti comandati simultaneamente, ma sempre con valori opposti.

Stabilita la corrispondenza tra variabili e contatti, vediamo il significato delle operazioni di somma e prodotto, e quello delle funzioni.

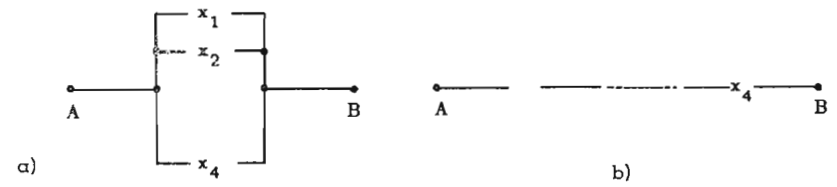


Fig.II.3 - Raggruppamento di contatti in parallelo (a) e in serie (b).

Si consideri un insieme di contatti $x_1 x_2 \ldots x_n$ in parallelo tra i punti A e B (fig.II.3a): la continuità tra A e B si ha quando almeno uno dei contatti è chiuso: la somma $x_1 + x_2 + \ldots + x_n$ descrive dunque il comportamento elettrico di n contatti in parallelo. La continuità tra 2 punti A e B che comprendono n contatti in serie (fig.II.3b) si ha invece solo se tutti i contatti sono chiusi: comportamento, questo, che può essere

descritto dal prodotto logico $x_1 x_2 \ldots x_n$. Le relazioni introdotte permettono di scrivere l'espressione analitica di ogni raggruppamento in serie-parallelo di contatti, e di interpretare in questo senso ogni espressione logica.

A titolo d'esempio, nella fig.II.4 è mostrato il circuito a contatti descritto dall'espressione:

(II.1) $w[xy + \bar{y}z + \bar{x}y(t+z)]$.

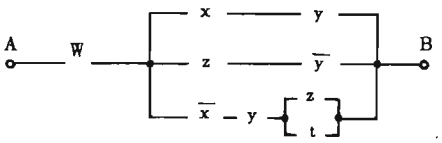


Fig.II.4 - Circuito a contatti serie parallelo.

L'espressione logica che indica sotto quali condizioni un circuito è connesso elettricamente tra due punti A e B si chiama *funzione di trasmissione* o, più brevemente, *funzione* del circuito. La (II.1) è, ad esempio, la funzione di trasmissione del circuito della fig.II.4.

I concetti esposti di algebra logica permettono di trattare analiticamente i problemi relativi ai circuiti a contatti. Tali problemi possono essere di due generi diversi: di analisi o di sintesi, a seconda che – dato lo schema del circuito – si voglia la descrizione del suo funzionamento o che, da quest'ultimo, si voglia il progetto del circuito. Esamineremo un esempio assai semplice per ognuno di questi problemi, traendone conclusioni di carattere generale.

Esempio 1: *Analizzare il circuito della fig.II.5, composto di una lampada L e due interruttori A e B, ognuno con 2 contatti opposti* ($A, \bar{A}$ *e* $B, \bar{B}$).

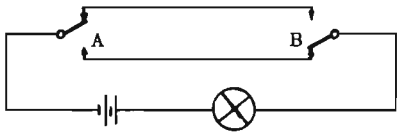


Fig.II.5 - Circuito con una lampada e due interruttori.

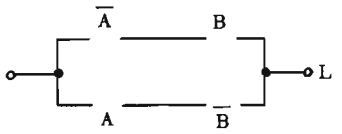


Fig.II.6 - Schema logico del circuito di fig.II.5.

Il circuito può essere schematizzato come indicato nella fig.II.6: con L si è chiamata la funzione del circuito, intendendo che quando $L = 1$ la lampada sia accesa e quando $L = 0$ sia spenta.

L'espressione analitica della funzione L è:

$$L = \bar{A}B + A\bar{B} \ .$$

Da questa si ottiene la tabella di verità (figura II.7), mettendo $L = 1$ in corrispondenza alle configurazioni 01 e 10 di AB. La lampada è pertanto accesa ogni volta che uno e uno solo dei due interruttori è chiuso.

A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

Fig.II.7 - Tabella di verità per il circuito di figura II.6.

Esempio 2: *Progettare un circuito per accendere o spegnere una lampada da uno qualsiasi di tre interruttori indipendenti.*

Siano A, B, C i tre interruttori: ricerchiamo le configurazioni di ABC per cui la funzione L è 1, corrispondenti alle posizioni degli interruttori per cui la lampada è accesa.

Quando i tre interruttori sono aperti (ABC = 000), la lampada è evidentemente spenta (L = 0). Chiudendo uno qualsiasi di essi (ABC = 001, 010, 100) la lampada si accende (L = 1), e si spegne (L = 0) se si chiude un altro interruttore (ABC=011,101,110). La lampada, infine, si riaccende se si chiude l'ultimo interruttore (ABC = 111) o se ne apre uno qualsiasi: nella fig.II.8 è mostrata la tabella di verità della funzione L.

Si ha quindi:

$$L = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + ABC \quad . \tag{II.2}$$

A	B	C	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Fig.II.8 - Tabella di verità di un circuito a 3 interruttori.

Il circuito che realizza la funzione è rappresentato nella fig.II.9a.

Scrivendo la L come prodotto di maxterm:

$$L = (A+B+C)(A+\overline{B}+\overline{C})(\overline{A}+B+\overline{C})(\overline{A}+\overline{B}+C) \tag{II.2'}$$

si ricava il circuito equivalente della fig.II.9b. Si noti che nei due circuiti ci si è serviti della proprietà distributiva della somma per eliminare un contatto A e un $\overline{A}$, scrivendo le equazioni (II.2) e (II.2') nella forma:

$$L = \overline{A}\,(\overline{B}C + B\overline{C}) + A\,(BC + \overline{B}\,\overline{C})$$

$$L = \left[A + (B+C)\,(\overline{B}+\overline{C})\right]\left[\overline{A} + (B+\overline{C})\,(\overline{B}+C)\right] \quad .$$

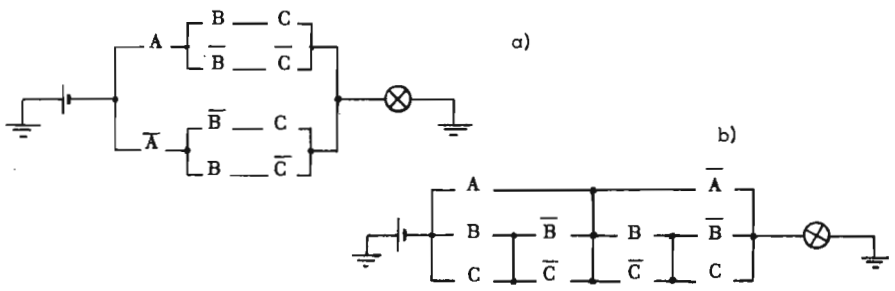


Fig.II.9 - Realizzazioni circuitali di un circuito a 3 interruttori.

È questo un punto da sottolineare: espressioni che contengono un minor numero di lettere rispetto ad altre equivalenti conducono a circuiti con un minor numero di contatti, quindi più economici.

Appare chiara l'utilità di pervenire a quella forma della funzione, se esiste, che porta al circuito col minimo numero di componenti (problema della semplificazione). I componenti dei circuiti a relè sono i contatti; nel prossimo capitolo vedremo quali sono i componenti elettronici e quali funzioni logiche adempiano: per ora tratteremo la semplificazione delle espressioni logiche unicamente dal punto di vista algebrico.

II.7 - Semplificazione delle espressioni logiche.

Nei paragrafi precedenti abbiamo introdotto il concetto di funzione logica di n variabili e mostrato vari modi di rappresentazione delle funzioni stesse: la tabella di verità, la somma di minterm, il prodotto di maxterm. Le due ultime rappresentazioni non esauriscono le espressioni analitiche di una funzione: come qualsiasi relazione algebrica, anche quelle logiche possono essere trasformate in un certo numero di espressioni formalmente diverse, ma sostanzialmente equivalenti. Le proprietà studiate ci permettono già, del resto, di operare alcune di queste trasformazioni, come mostra l'esempio seguente.

Esempio: *La funzione rappresentata dalla tabella di verità della fig.II.10:*

x_1	x_2	x_3	y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Fig.II.10 - Tabella di verità di una funzione a 3 variabili.

si scrive in forma canonica:

(II.3) $$y = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 + \overline{x}_1x_2x_3 + x_1x_2x_3 \quad .$$

Per la proprietà distributiva, si ha:

$$y = \overline{x}_1\overline{x}_2\,(\overline{x}_3 + x_3) + x_2x_3\,(\overline{x}_1 + x_1)$$

e, per essere $x + \overline{x} = 1$, $x \cdot 1 = x$, si ha infine:

(II.3') $$y = \overline{x}_1\overline{x}_2 + x_2x_3 \quad .$$

Le espressioni (II.3) e (II.3') sono perfettamente equivalenti, ma la seconda è molto più semplice.

Chiamaremo *equivalenti* due funzioni che hanno la stessa tabella di verità; fra tutte le forme equivalenti di una funzione y, è particolarmente importante quella *minima*, che contiene il minor numero di lettere. Nel numero di lettere si conta ogni apparizione di una variabile, diretta o negata: ad esempio, la (II.3) contiene 12 lettere, la (II.3') ne contiene 4. Si chiama *semplificazione* il processo che porta alla forma minima o, più generalmente, a una forma contenente meno lettere di quella canonica (espressione semplificata). Si noti che, in qualche caso, l'espressione minima coincide con quella canonica.

Espressioni semplificate si ottengono, come già fatto nell'esempio precedente, anche applicando alle espressioni canoniche le relazioni fondamentali dell'algebra: questa strada richiede però una pratica eccezionale, non assicura di aver raggiunto l'espressione minima e può seguirsi solo per un limitato numero di variabili.

Esempio: *Semplificare la funzione:*

$$y = x_1x_2x_3x_4 + x_1x_2x_3\overline{x}_4 + x_1x_2\overline{x}_3x_4 + \overline{x}_1x_2x_3x_4 + x_1x_2\overline{x}_3\overline{x}_4 \quad .$$

Per la proprietà distributiva, il primo termine si può semplificare col secondo:

$$x_1x_2x_3x_4 + x_1x_2x_3\overline{x}_4 = x_1x_2x_3\,(x_4 + \overline{x}_4) = x_1x_2x_3 \quad .$$

Per la proprietà $x + x = x$, si può ancora semplificare il primo termine col terzo e col quarto, ed il terzo termine con il quinto:

$$x_1x_2x_3x_4 + x_1x_2\overline{x}_3x_4 = x_1x_2x_4$$

$$x_1x_2x_3x_4 + \overline{x}_1x_2x_3x_4 = x_2x_3x_4$$

$$x_1x_2\overline{x}_3x_4 + x_1x_2\overline{x}_3\overline{x}_4 = x_1x_2\overline{x}_3 \quad .$$

Si può quindi scrivere:

$$y = x_1x_2x_3 + x_1x_2x_4 + x_2x_3x_4 + x_1x_2\overline{x}_3 \quad .$$

Il primo e l'ultimo termine di questa relazione possono ancora essere semplificati:

$$x_1x_2x_3 + x_1x_2\overline{x}_3 = x_1x_2 \quad .$$

Per il primo teorema dell'assorbimento $(x + xy = x)$, il termine x_1x_2 assorbe $x_1x_2x_4$. La funzione diventa allora:

$$y = x_1x_2 + x_2x_3x_4$$

e, mettendo in evidenza x_1:

$$y = x_1\,(x_2 + x_3x_4) \quad .$$

Per la descrizione dei metodi di semplificazione, sono utili le seguenti definizioni:

Implicante di una funzione booleana $y = f(x_1 x_2 \ldots x_n)$ è un prodotto p di m variabili ($n \leq m \leq 1$) contenuto nella y, tale cioè che, quando p = 1, anche y = 1.

Esempio: *I prodotti a, ab, $a\bar{b}$, c sono implicanti della funzione:*

$$y = ab + a\bar{b} + c$$

come appare dalla seguente tabella di verità:

abc	y	ab	$a\bar{b}$	a	c
000	0	0	0	0	0
001	1	0	0	0	1
010	0	0	0	0	0
011	1	0	0	0	1
100	1	0	1	1	0
101	1	0	1	1	1
110	1	1	0	1	0
111	1	1	0	1	1

Implicante primo di una y è un implicante che non è contenuto in nessun altro implicante della y.

Esempio: *a, c sono implicanti primi della $y = ab + a\bar{b} + c$; ab, $a\bar{b}$ non lo sono perché risultano entrambi contenuti in a.*

Forma irridondante di una y è ogni somma di implicanti primi che contiene tutti i minterm della y. Tra le forme irridondanti esistono una o più *forme minime*.

Implicante essenziale è un implicante primo che compare in ogni forma irridondante di una y.

In definitiva, la somma (Σ) degli implicanti essenziali copre un certo numero di un minterm della y. La forma minima sarà formata da Σ e da un sottoinsieme degli implicanti primi non essenziale che contiene tutti i minterm non coperti da Σ.

La ricerca della forma minima avviene per tentativi; quella degli implicanti con il metodo di Karnaugh, con il metodo di Quine-Mc Cluskey, o con quello accennato nel par.II.9.1.

II.7.1 - Il metodo delle mappe di Karnaugh.

Il metodo grafico proposto da Karnaugh serve a ricavare gli implicanti primi di una funzione, teoricamente a qualsiasi numero di variabili; in pratica, è usato fino a 5-6 variabili.

Le mappe di Karnaugh costituiscono, anzitutto, un ulteriore metodo per rappresentare le funzioni logiche: sono dei rettangoli, o dei quadrati, divisi in tante caselle quante sono le configurazioni diverse di n variabili (2^n): quindi 4 caselle per 2 variabili, 8 per 3, 16 per 4.

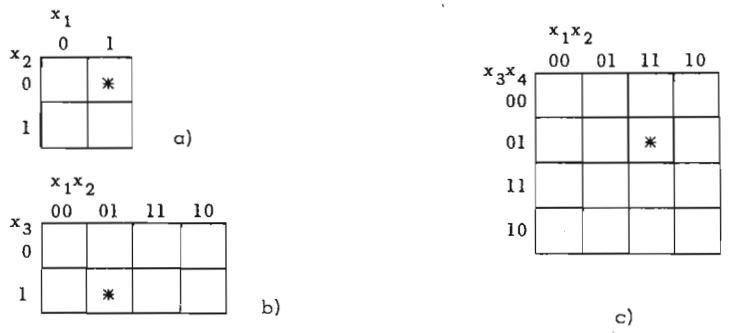


Fig.II.11 - Mappe di Karnaugh per 2 (a), 3 (b), 4 (c) variabili.

Ogni casella ha n coordinate e viene messa in corrispondenza con la omonima configurazione delle variabili. Nella fig.II.11 sono rappresentate le mappe di Karnaugh per 2, 3, 4 variabili. Le caselle segnate con l'asterisco, di coordinate $x_1 = 1$, $x_2 = 0$ (fig..II.11a); $x_1 = 0$, $x_2 = 1$, $x_3 = 1$ (fig.II.11b); $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$ (fig.II.11c), rappresentano, rispettivamente, le configurazioni $x_1\overline{x}_2$, $\overline{x}_1x_2x_3$, $x_1x_2\overline{x}_3x_4$ delle variabili x_1x_2, $x_1x_2x_3$, $x_1x_2x_3x_4$.

Per rappresentare una funzione y sulle mappe, basta scrivere 1 in corrispondenza delle caselle le cui coordinate eguagliano le configurazioni della tabella di verità per cui $y = 1$. I seguenti esempi chiariranno quanto detto.

Esempio.

1) La fig.II.12b rappresenta sulle mappe di Karnaugh la funzione a 2 variabili della figura II.12a.

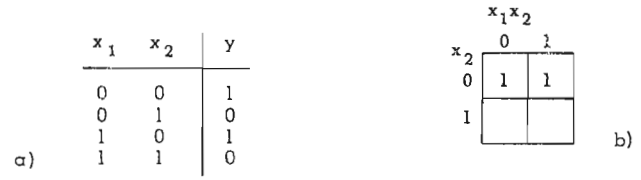


Fig.II.12 - Tabella di verità (a) e mappa di Karnaugh (b) per una funzione di 2 variabili.

2) La funzione di tre variabili:

$$y = x_1x_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1x_2x_3$$

è rappresentata sulla mappa di Karnaugh della fig.II.13.

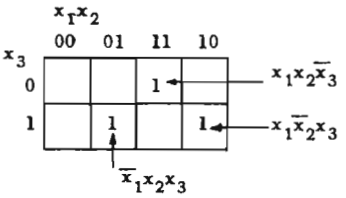


Fig.II.13 - Mappa di Karnaugh per una funzione di 3 variabili.

3) La funzione di quattro variabili:

$$y = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3x_4$$

si rappresenta su una mappa a quattro variabili nel modo illustrato nella fig.II.14.

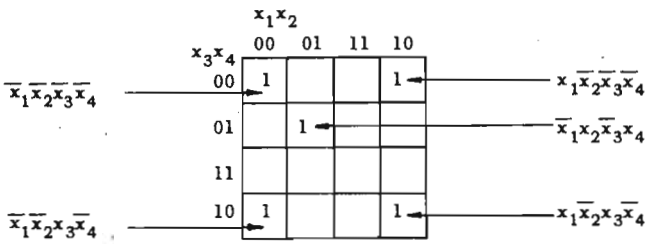


Fig.II.14 - Mappa di Karnaugh per una funzione di 4 variabili.

Sulla mappa di Karnaugh, i minterm corrispondenti a due caselle adiacenti, cioè con un lato in comune, differiscono per una sola variabile, che è diretta nell'una, negata nell'altra, e sono quindi contenuti nel prodotto delle (n - 1) variabili in comune.

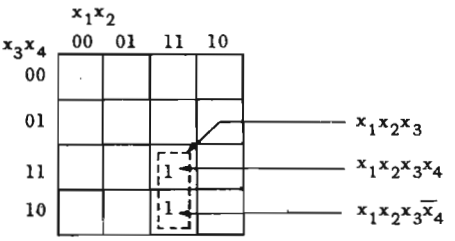


Fig.II.15 - Minterm adiacenti in una mappa di Karnaugh a 4 variabili.

Ad esempio, le due caselle corrispondenti ai minterm $x_1x_2x_3\bar{x}_4$ e $x_1x_2x_3x_4$ differiscono per la sola variabile x_4 (fig.II.15) e sono contenute nel blocco tratteggiato, di coordinate $x_1x_2x_3 = 111$, ed espressione $x_1x_2x_3$.

Due caselle vanno considerate adiacenti, come è facile verificare dal confronto delle coordinate, anche quando si trovano nelle posizioni indicate nella fig.II.16: le mappe vanno, a questo riguardo, immaginate chiuse su se stesse.

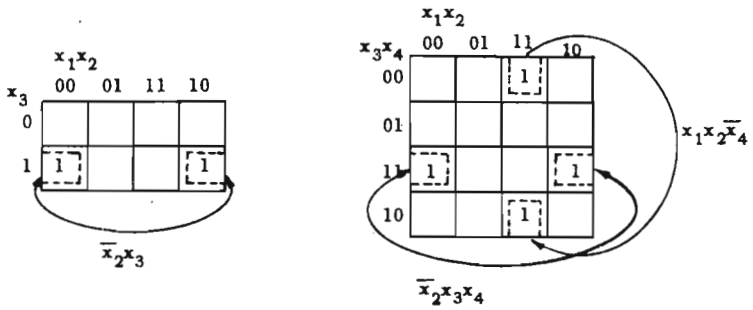


Fig.II.16 - Configurazioni di 2 minterm adiacenti sulle mappe di Karnaugh.

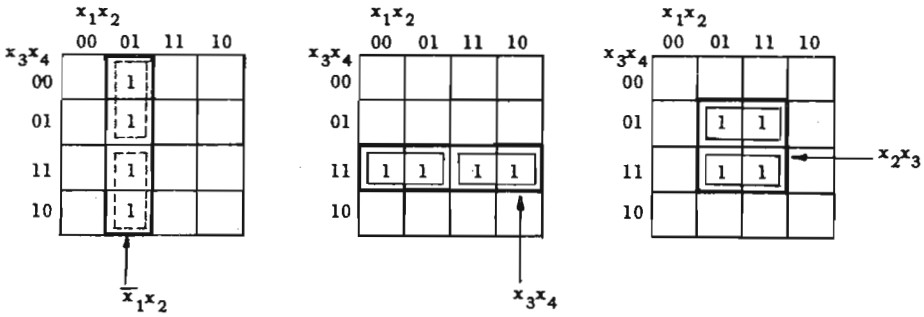


Fig.II.17 - Configurazioni di 4 minterm adiacenti sulle mappe di Karnaugh.

Le considerazioni precedenti si possono estendere ai gruppi di quattro caselle sulla stessa colonna, o sulla stessa riga, o intorno allo stesso vertice (fig.II.17): queste configurazioni vanno intese come due gruppi di 2 minterm adiacenti, rappresentati col prodotto di (n - 1) variabili che differiscono, a loro volta, soltanto per l'affermazione e la negazione di una variabile. I quattro minterm relativi sono pertanto contenuti nel prodotto delle (n - 2) variabili in comune.

Vanno considerate adiacenti anche le configurazioni mostrate nella fig.II.18.

Con lo stesso criterio, è possibile passare a blocchi di 8 caselle adiacenti.

E' ovvio, a questo punto che ogni blocco rappresenta un implicante della funzione; un blocco non compreso in altri più grandi rappresenta un implicante primo; un blocco che è l'unico a coprire uno o più caselle rappresenta un implicante essenziale. In definitiva, per semplificare una funzione y, rappresentata su una mappa di Karnaugh, basta raccoglierne le caselle nel minor numero di blocchi contenenti 8, 4, 2 caselle adiacenti, in modo da prenderle tutte almeno una volta.

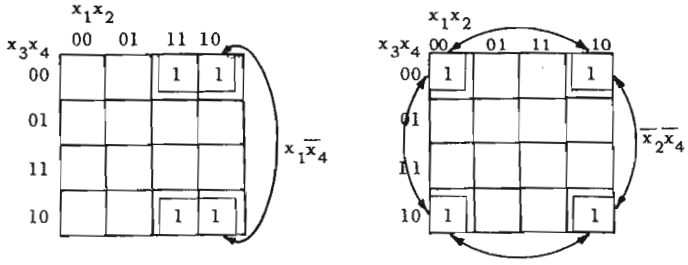
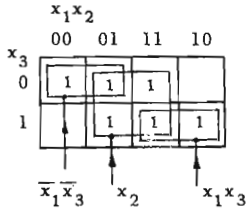


Fig.II.18 - Configurazioni di 4 minterm adiacenti sulle mappe di Karnaugh.

Esempio 1: *Semplificazione della funzione a 3 variabili rappresentata nella mappa della fig.II.19.*

Fig.II.19 - Semplificazione di una funzione di 3 variabili.



La forma minima è:

$$y = x_2 + x_1x_3 + \overline{x}_1\overline{x}_3 \quad .$$

Esempio 2: *Semplificazione della funzione a 4 variabili rappresentata nella mappa della fig.II.20.*

La forma minima è:

$$y = \overline{x}_2 + \overline{x}_1x_4 \quad .$$

La y non è funzione di x_3, ma soltanto di x_1, x_2, x_4: si dice in questo caso che y è una funzione *degenere* di x_3: essa può essere considerata, a tutti gli effetti, una funzione di tre variabili, la cui mappa di Karnaugh è una delle due metà orizzontali di quella della fig.II.20.

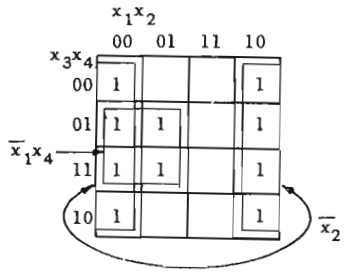


Fig.II.20 - Semplificazione di una funzione di 4 variabili.

Esempio 3: *Semplificazione della funzione a quattro variabili della fig.II.21.*

La forma minima è:

$$y = \overline{x}_1 x_2 + x_2 x_3 x_4 + \overline{x}_1 \overline{x}_3 \overline{x}_4 + x_1 \overline{x}_2 \overline{x}_3 x_4 \quad .$$

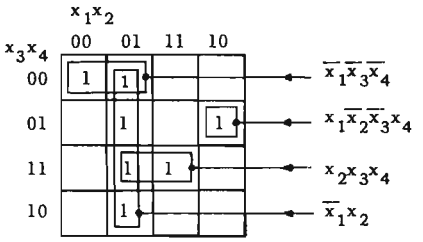


Fig.II.21 - Semplificazione di una funzione di 4 variabili.

Esempio 4: *Semplificazione della funzione a quattro variabili della fig.II.22.*

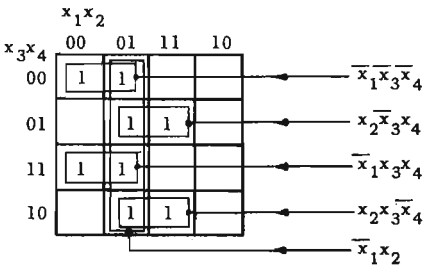


Fig.II.22 - Semplificazione di una funzione di 4 variabili.

In questo caso, l'implicante primo $\overline{x}_1 x_2$ non è essenziale perché tutti i suoi minterm sono compresi nei restanti implicanti primi, che sono tutti essenziali. La forma minima è perciò:

$$y = \overline{x}_1 \overline{x}_3 \overline{x}_4 + x_2 \overline{x}_3 x_4 + \overline{x}_1 x_3 x_4 + x_2 x_3 \overline{x}_4 \quad .$$

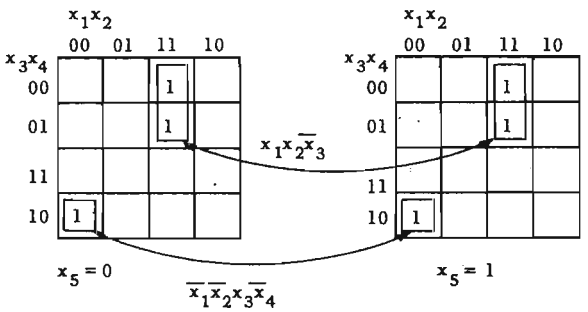


Fig.II.23 - Minterm adiacenti di una funzione di 5 variabili.

Le mappe di Karnaugh per cinque variabili sono formate da due mappe per quattro variabili (fig.II.23). I blocchi di caselle adiacenti, quin-

di semplificabili, sono – oltre a quelli già visti – tutti quelli che è possibile formare con caselle occupanti le stesse posizioni sulle due tavole.

Nella fig.II.23 sono indicati alcuni gruppi di caselle adiacenti.

Le mappe di Karnaugh per sei variabili sono formate da quattro mappe a quattro variabili disposte in quadrato (fig.II.24). Sono da considerarsi adiacenti, quindi semplificabili, oltre a quelli già visti, tutti i blocchi formati da caselle occupanti le stesse posizioni su due mappe adiacenti orizzontalmente o verticalmente. Nella fig.II.24 sono indicati alcuni di questi blocchi.

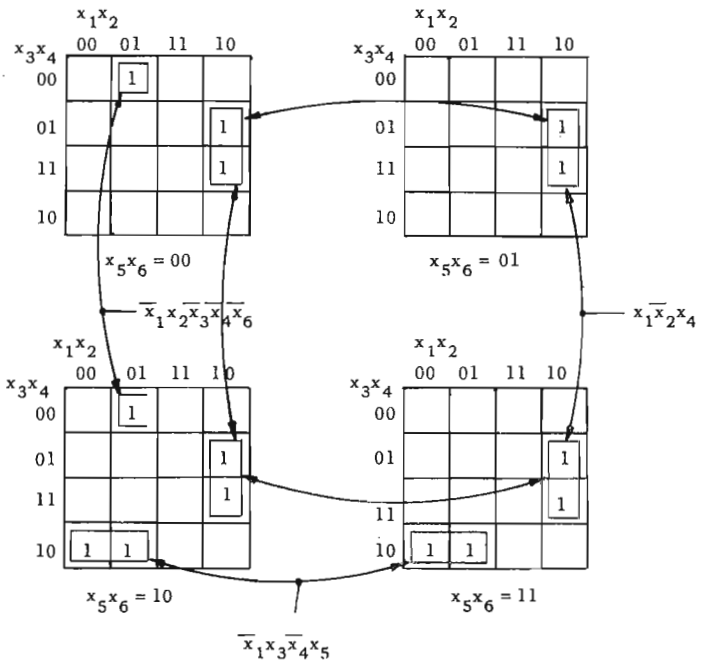


Fig.II.24 - Minterm adiacenti di una funzione di 6 variabili.

A titolo d'esempio, nella fig.II.25 è riportata la semplificazione della funzione di cinque variabili:

$$y = x_1\overline{x}_2x_3x_4x_5 + x_1\overline{x}_2x_3x_4\overline{x}_5 + x_1\overline{x}_2x_3\overline{x}_4x_5 + x_1\overline{x}_2x_3\overline{x}_4\overline{x}_5 + x_1\overline{x}_2\overline{x}_3x_4x_5 + x_1\overline{x}_2\overline{x}_3x_4\overline{x}_5 + x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5 + x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5 + \overline{x}_1x_2x_3x_4x_5 + x_1x_2x_3x_4x_5 + \overline{x}_1\overline{x}_2x_3x_4x_5 + \overline{x}_1x_2\overline{x}_3\overline{x}_4x_5$$

la cui forma minima è:

$$y = x_1\overline{x}_2 + x_3x_4x_5 + \overline{x}_1x_2\overline{x}_3\overline{x}_4x_5 \ .$$

Abbiamo visto che si può rappresentare una funzione come somma di numeri decimali (par.II.4). Attribuendo i pesi 1, 2, 4, 8,... alle va-

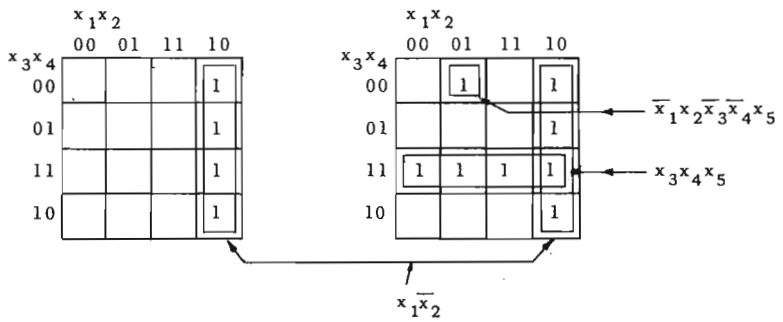


Fig.II.25 - Semplificazione di una funzione di 5 variabili.

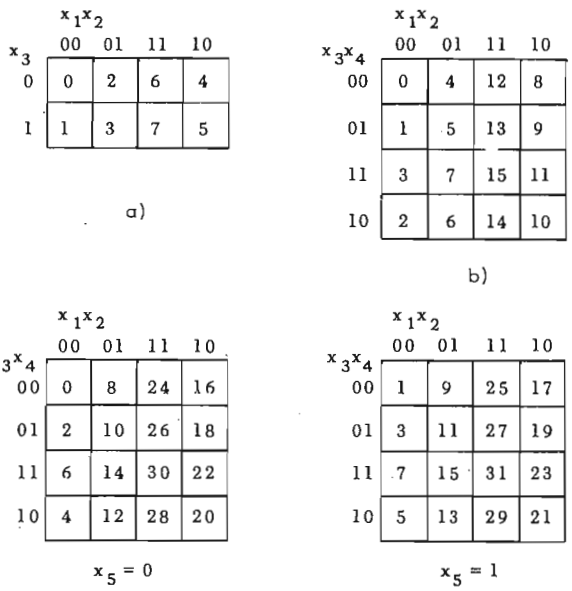


Fig.II.26 - Valori decimali delle caselle delle mappe di Karnaugh a 3 (a), 4 (b) e 5 (c) variabili.

riabili x_n, x_{n-1}, x_{n-2}, x_{n-3}, ... si possono numerare in decimale le caselle delle mappe, per riportarvi immediatamente una funzione scritta in questa forma. Nella fig.II.26 sono riportati i valori decimali delle varie caselle, secondo la suddetta convenzione.

II.7.2 - Il metodo di Quine - Mc Cluskey.

Il metodo di Quine - Mc Cluskey è un procedimento tabellare per minimizzare le funzioni logiche in forma di somme di prodotti. Consiste in una serie di confronti fra tutti i minterm della funzione da semplificare, basati sulla relazione $fx + f\bar{x} = f$, per cui due prodotti che differiscono per l'affermazione e la negazione di una sola variabile sono compresi nell'implicante corrispondente al prodotto delle variabili comuni.

In pratica si procede nel modo seguente:

1) Partendo dalla tabella di verità della funzione y, si dividono tutti i minterm in gruppi aventi lo stesso numero di 1 (livelli), si ordinano i livelli in ordine crescente e si numerano i minterm coi corrispondenti numeri decimali.

Ad esempio, i minterm $\bar{x}_1\bar{x}_2\bar{x}_3$, $\bar{x}_1\bar{x}_2x_3$, $\bar{x}_1x_2\bar{x}_3$, $\bar{x}_1x_2x_3$, $x_1x_2x_3$ vanno raggruppati in quattro livelli, e ordinati come mostrato nella figura II.27.

Livello	Numero	Minterm
0	0	0 0 0
1	1	0 0 1
	2	0 1 1
2	3	0 1 1
3	7	1 1 1

Fig.II.27 - Costruzione di una tabella di Quine - Mc Cluskey.

2) Si confrontano i minterm del livello k con tutti quelli del livello (k + + 1); le semplificazioni avvengono tra minterm confrontabili che differiscono in un solo bit. Sono, così, semplificabili 0001101 e 0101101; non lo sono 0001101 e 0110101.

a)

0•1	0	0	-
0•2	0	-	0
1•3	0	-	1
2•3	0	1	-
3•7	-	1	1

b)

0	0	0	0	✓
1	0	0	1	✓
2	0	1	0	✓
3	0	1	1	✓
7	1	1	1	✓

Fig.II.28 - Prima serie di confronti nel metodo di semplificazione.

Si costruisce una seconda tabella dove si indicano le semplificazioni avvenute con una lineetta e i relativi implicanti coi numeri dei due

minterm che li hanno originati. Nella prima tabella si segnano, inoltre, tutti i minterm semplificati almeno una volta. Dopo questa prima serie di confronti, la tabella della fig.II.27 origina la tabella della fig. II.28a; i minterm originari vanno tutti segnati (figura II.28b).

La semplificazione tra 0 (000) e 1 (001) che dà luogo a 0·1 (00-) significa: $\overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2x_3 = \overline{x}_1\overline{x}_2$; la lineetta indica appunto che la terza variabile (x_3) non compare nel prodotto che rappresenta i minterm 0 e 1.

3) Nella seconda tabella, si confrontano tutti gli implicanti di (n-1) variabili sul livello k con quelli sul livello (k - 1). Sono semplificabili gli implicanti differenti per un solo bit e già semplificati rispetto alla stessa variabile, cioè con una lineetta nella stessa posizione. Così, 00-110, che rappresenta $\overline{x}_1\overline{x}_2x_4x_5\overline{x}_6$ è semplificabile con 01-110, che rappresenta $\overline{x}_1x_2x_4x_5\overline{x}_6$, e dà per risultato 0--110, cioè $\overline{x}_1x_4x_5\overline{x}_6$. Le semplificazioni avvenute si riportano su una terza tabella; nella seconda si segnano tutti gli implicanti semplificati almeno una volta.

Nell'esempio della fig.II.28a, l'espressione (0·1) è semplificabile con la (2·3), e dà per risultato l'espressione 0·1·2·3 (0--). Lo stesso risultato si ottiene semplificando (0·2) con (1·3). Non è possibile, invece, semplificare la (3·7) che – pertanto – non viene segnata nella seconda tabella.

4) Si continua nelle semplificazioni, e nella costruzione delle relative tabelle, finché è possibile.

Nel nostro esempio, non esistono altre semplificazioni.

5) Si scrive una lettera maiuscola accanto a tutti i prodotti che, nelle varie tabelle, non sono stati segnati (*primi implicanti* della y).

Nell'esempio considerato, i primi implicanti sono soltanto 2 (figura II.29).

L'espressione minima della y è, come noto, la somma del minor numero di primi implicanti con cui si coprono tutti i minterm della y stessa.

La scelta più conveniente dei primi implicanti, immediata per le funzioni di poche variabili, diventa molto complessa già per n = 6, soprattutto a causa del gran numero di primi implicanti che si può avere. È opportuno, allora, servirsi di un reticolo avente i minterm sulle verticali e i primi implicanti sulle orizzontali. Su questo reticolo si indicano con una x i minterm di ogni primo implicante, si segnano poi – con un cerchio – le x che si trovano da sole su ogni verticale: il minterm corrispondente fa parte di un solo primo implicante il quale, pertanto, è essenziale e deve comparire nella copertura minima. Ogni primo implicante essenziale comprende, tuttavia, altri minterm: questi risultano auto-

maticamente coperti, e non vanno più considerati. Eliminati i primi implicanti essenziali, è molto più semplice trovare – per tentativi – la copertura minima dei minterm rimasti.

0	0	0	0	✓
1	0	0	1	✓
2	0	1	0	✓
3	0	1	1	✓
7	1	1	1	✓

0•1	0	0	–	✓
0•2	0	–	0	✓
1•3	0	–	1	✓
2•3	0	1	–	✓
3•7	–	1	1	A

0•1•2•3	0	–	–	B

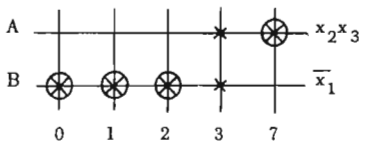
Fig.II.29 - Passi successivi per la semplificazione secondo Quine-Mc Cluskey della funzione della fig.II.27.

Nell'esempio illustrato, il reticolo ha l'aspetto della fig.II.30. Entrambi i primi implicanti sono essenziali. Si ha quindi:

$$y = A + B = \overline{x}_1 + x_2 x_3 \quad .$$

In questo caso, il reticolo è del tutto superfluo; la sua utilità è invece evidente nel prossimo esempio, che chiarisce l'applicazione del metodo.

Fig.II.30 - Reticolo per la ricerca della copertura minima.



Esempio: *Semplificare la funzione:*

$$y = \Sigma(1, 3, 4, 6, 7, 9, 10, 11, 12, 13, 14, 15) \quad .$$

I minterm della y, scritti come numeri binari, sono:

0001, 0011, 0100, 0110, 0111, 1001, 1010, 1011, 1100, 1101, 1110, 1111

Ordinandoli secondo il numero di 1, si ottiene la tabella della fig.II.31.

1	0001
4	0100
3	0011
6	0110
9	1001
10	1010
12	1100
7	0111
11	1011
13	1101
14	1110
15	1111

Fig.II.31 - Tabella dei minterm di una funzione di 4 variabili.

Le tabelle delle semplificazioni, e i risultati dei confronti, sono riportati nella fig.II.32.

1•3	0 0 – 1	✓
1•9	– 0 0 1	✓
4•6	0 1 – 0	✓
4•12	– 1 0 0	✓
3•7	0 – 1 1	✓
3•11	– 0 1 1	✓
6•7	0 1 1 –	✓
6•14	– 1 1 0	✓
9•11	1 0 – 1	✓
9•13	1 – 0 1	✓
10•11	1 0 1 –	✓
10•14	1 – 1 0	✓
12•13	1 1 0 –	✓
12•14	1 1 – 0	✓
7•15	– 1 1 1	✓
11•15	1 – 1 1	✓
13•15	1 1 – 1	✓
14•15	1 1 1 –	✓

1•3•9•11	– 0 – 1	A
4•6•12•14	– 1 – 0	B
3•7•11•15	– – 1 1	C
6•7•14•15	– 1 1 –	D
9•11•13•15	1 – – 1	E
10•11•14•15	1 – 1 –	F
12•13•14•15	1 1 – –	G

Fig.II.32 - Semplificazione della funzione della fig.II.31.

Esistono sette primi implicanti; nella fig.II.33 è mostrato il reticolo per la scelta della copertura minima, formato da 12 verticali e 7 orizzontali.

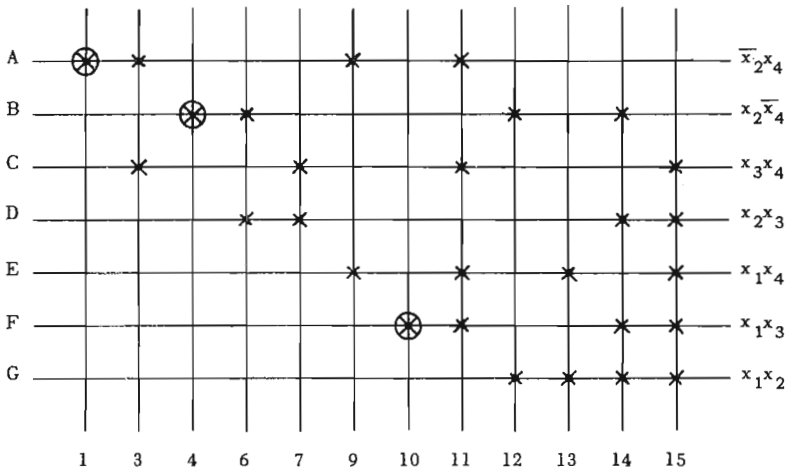


Fig.II.33 - Reticolo per la ricerca della copertura minima della funzione della fig.II.31.

I minterm 1, 4, 10 sono compresi in un solo primo implicante: pertanto A, B ed F fanno parte della copertura minima. La loro scelta elimina i minterm 3, 9, 11, 6, 12, 14, 15; per coprire i rimanenti (7 e 13) è indifferente scegliere il primo implicante C o D in-

sieme con E o G, non essendocene nessuno che li comprenda entrambi. Esistono aunque quattro soluzioni minime equivalenti:

$$y = A + B + F + D + E = \overline{x}_2x_4 + x_2\overline{x}_4 + x_1x_3 + x_2x_3 + x_1x_4$$
$$y = A + B + F + D + G = \overline{x}_2x_4 + x_2\overline{x}_4 + x_1x_3 + x_2x_3 + x_1x_2$$
$$y = A + B + F + C + E = \overline{x}_2x_4 + x_2\overline{x}_4 + x_1x_3 + x_3x_4 + x_1x_4$$
$$y = A + B + F + C + G = \overline{x}_2x_4 + x_2\overline{x}_4 + x_1x_3 + x_3x_4 + x_1x_2 \quad .$$

Come vedremo nel par.II.8, un'espressione equivalente con lo stesso numero di lettere si poteva ricavare più rapidamente partendo dalle configurazioni delle variabili per cui $y = 0$.

Il reticolo per la ricerca della copertura minima si può adoperare, evidentemente, anche per il metodo di Karnaugh.

II.8 - Fattorizzazione.

Applicando la proprietà associativa della somma, è spesso possibile diminuire il numero delle lettere che compaiono nell'espressione minima come somma di prodotti, ottenibile coi due metodi di semplificazione illustrati. Questo procedimento, che introduce nell'espressione della funzione uno o più ordini di parentesi, si chiama *fattorizzazione* ed è in pratica molto importante se le funzioni semplificate debbono essere realizzate mediante circuiti a contatti: in questo caso, infatti, le operazioni di somma e prodotto non hanno alcun costo, e le m variabili prese come fattori comuni tra k termini richiedono l'impiego di m contatti, invece di k·m.

Ad esempio, la funzione minima come somma di prodotti:

$$y = x_1x_2x_3x_4 + x_1x_2\overline{x}_3\overline{x}_4 + x_1x_2x_3\overline{x}_5$$

contiene 12 lettere, e si realizza con altrettanti contatti.

La stessa funzione, scritta nella forma fattorizzata:

$$y = x_1x_2\,[\overline{x}_3\overline{x}_4 + x_3\,(x_4 + \overline{x}_5)]$$

contiene 7 lettere, e permette un risparmio di 5 contatti.

L'importanza della fattorizzazione è minore nei circuiti elettronici, per le ragioni che vedremo nel prossimo capitolo; può tuttavia essere applicata con buoni risultati nei circuiti che non hanno particolari requisiti di velocità e non contengono soltanto elementi passivi.

II.9 - Espressione minima di una funzione sotto forma di prodotto di somme.

Come avviene per le forme canoniche, anche per le forme minime si può arrivare facilmente ad espressioni del tipo *prodotto di somme*. Con qualsiasi metodo di semplificazione, basta partire dagli 0 della funzione: ottenuta la forma minima, come somma di prodotti, della $\overline{y}$, si ottiene la forma cercata applicando a quest'ultima il teorema di De Morgan.

Se non esistono vincoli sulla forma della funzione, conviene anzi procedere in questo modo quando gli 1 della y sono in numero maggiore di 2^{n-1}.

Esempio 1: *Semplificare sulle mappe di Karnaugh la funzione della fig.II.34, ricavando la forma minima come prodotto di somme.*

L'espressione minima della funzione $\overline{y}$, ottenuta dalla considerazione delle caselle vuote, è:

$$\overline{y} = x_1x_2 + x_2\overline{x}_4 \ .$$

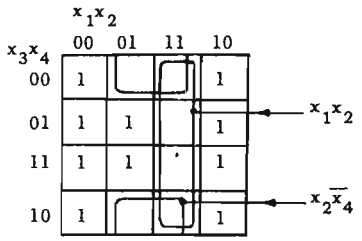


Fig.II.34 - Semplificazione di una funzione negata.

Per il teorema di De Morgan, si ha quindi:

$$y = \overline{\overline{y}} = \overline{x_1x_2 + x_2\overline{x}_4} = (\overline{x}_1 + \overline{x}_2)(\overline{x}_2 + x_4) \ .$$

Si confronti questo risultato con l'esempio 2 del par.II.6.1, la cui mappa è rappresentata nella fig.II.20.

Esempio 2: *Semplificare col metodo di Quine-Mc Cluskey la funzione:*

$$y = \Sigma(1,3,4,6,7,9,10,11,12,13,14,15)$$

dando la forma minima come prodotto di somme.

La funzione $\overline{y}$ è data dai minterm i cui valori decimali da 0 a 15 non sono compresi nella y:

$$\overline{y} = \Sigma(0,2,5,8) \ .$$

La semplificazione della y secondo Quine-Mc Cluskey (fig.II.35) dà:

$$y = A + B + C = \overline{x}_1 x_2 \overline{x}_3 x_4 + \overline{x}_1 \overline{x}_2 \overline{x}_4 + \overline{x}_2 \overline{x}_3 \overline{x}_4$$

quindi:

$$y = \overline{\overline{y}} = \overline{\overline{x}_1 x_2 \overline{x}_3 x_4 + \overline{x}_1 \overline{x}_2 \overline{x}_4 + \overline{x}_2 \overline{x}_3 \overline{x}_4} =$$

$$= (x_1 + \overline{x}_2 + x_3 + \overline{x}_4)(x_1 + x_2 + x_4)(x_2 + x_3 + x_4) \ .$$

Confrontando i risultati ottenuti con quelli preceaenti, si vede che la forma minima come prodotto di somme ha lo stesso numero di lettere delle forme come somme di prodotti, ma si raggiunge molto più rapidamente (v. figg.II.32, II.33). Inoltre, la forma finale della y si può ottenere direttamente dalle espressioni dei primi implicanti, secondo il metodo esposto nel par.II.5.

0	0 0 0 0	✓
2	0 0 1 0	✓
8	1 0 0 0	✓
5	0 1 0 i	A

0·2	0 0 - 0	B
0·8	- 0 0 0	C

Fig.II.35 - Semplificazione di una funzione negata.

II.9.1 - Ricerca dei primi implicanti di una funzione.

Una interessante e poco nota proprietà delle forme *prodotti di somme*, dimostrata in [16], è la seguente:

Data una funzione y espressa come prodotto di somme:

$$(1) \qquad y = \prod_{1}^{m}{}_i \, S_i$$

la forma $\sum_{1}^{n}{}_j \, P_j$ *ottenuta effettuando i prodotti indicati nella (1) contiene tutti e soli i primi implicanti della* y.

I vantaggi di questo metodo per la ricerca dei primi implicanti sono evidenti: si noti, tra l'altro, che la (1) può essere sia in forma canonica che semplificata.

Esempio 1: *Effettuando i prodotti indicati nella funzione y dell'esempio 1 del paragrafo precedente, si ottiene, come nell'esempio 2 del par.II.6.1:*

$$y = (\overline{x}_1 + \overline{x}_2)(\overline{x}_2 + x_4) = \overline{x}_2 + \overline{x}_1 x_4$$

(il termine $\overline{x}_2 x_4$ viene eliminato, per assorbimento da $\overline{x}_2$).

Esempio 2: *Effettuando i prodotti indicati nella funzione y dell'esempio 2 del paragrafo precedente, si ottiene:*

$$(2) \qquad y = (x_1 + \overline{x}_2 + x_3 + \overline{x}_4)(x_1 + x_2 + x_4)(x_2 + x_3 + x_4) =$$

$$= x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_3x_4 + \overline{x}_2x_4 + x_2\overline{x}_4 \quad .$$

I prodotti che compaiono nella (2) (dalla quale sono stati eliminati i prodotti del tipo $\overline{x}_2x_3x_4$, assorbiti da quelli del tipo $\overline{x}_2x_4$) coincidono con i primi implicanti ottenuti col metodo di Quine-Mc Cluskey, indicati nella fig.II.32.

II.10 - Le condizioni non specificate e le funzioni di funzioni.

Nella tabella di verità di una funzione y di n variabili, esistono k configurazioni ($k < 2^n$, se $y \neq 1$) per cui $y = 1$ e $2^n - k$ per cui $y = 0$. Possono però presentarsi delle situazioni in cui si ha $y = 1$ per k configurazioni, $y = 0$ per m configurazioni, e $m + k < 2^n$.

Le restanti $2^n - (m + k)$ configurazioni, per le quali non è definito alcun valore della y, si chiamano *condizioni non specificate* (don't care conditions: d.c.c.).

In pratica, le condizioni non specificate si presentano frequentemente: ad esempio ogni volta che alcune configurazioni delle variabili di ingresso in un circuito non possono mai verificarsi o, esistendo, rendono priva di significato la funzione d'uscita.

Dal punto di vista analitico, le d.c.c., compaiono nello studio delle funzioni di funzioni, cioè nelle funzioni y di un certo numero di variabili $w_1w_2 \ldots w_m$ ognuna delle quali è, a sua volta, funzione delle n variabili indipendenti $x_1x_2 \ldots x_n$:

$$y = F(w_1, w_2, \ldots, w_m)$$

$$w_i = F_i(x_1, x_2, \ldots, x_n) \qquad (i = 1, 2, \ldots, m) \quad .$$

Le d.c.c. per la y si ricavano dalla tabella di verità delle w_i in funzione delle $x_1x_2 \ldots x_n$, e sono tutte e sole le configurazioni di w_i che non esistono nella tabella stessa.

Esempio: *Trovare le d.c.c. per una funzione:*

$$y = F(w_1, w_2, w_3)$$

sapendo che le w_i *sono funzioni delle variabili indipendenti* $x_1x_2x_3x_4$, *secondo le relazioni:*

$$w_1 = x_1 (x_2x_3 + \overline{x}_2\overline{x}_3)$$

$$w_2 = x_4 (x_2\overline{x}_3 + \overline{x}_2x_3)$$

$$w_3 = x_1x_2x_3 + \overline{x}_1\overline{x}_2\overline{x}_3 \quad .$$

La tabella di verità per le w_i è mostrata nella fig.II.36.

Le terne diverse dei possibili valori di $w_1w_2w_3$ sono 5 (001, 000, 010, 100, 101). Le tre configurazioni mancanti sono quindi da considerarsi, per la y, condizioni non specificate.

x_1	x_2	x_3	x_4	w_1	w_2	w_3
0	0	0	0	0	0	1
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	1	0	1

Fig.II.36 - Tabella di verità per 3 funzioni di 4 variabili.

Le d.c.c. vanno usate, tutte o in parte, come ulteriori configurazioni per cui y = 1, in modo da semplificare ulteriormente la y stessa. Prima di vedere come l'esistenza delle d.c.c. modifica i metodi di semplificazione, è necessario saper rappresentare le funzioni che contengono condizioni non specificate.

a) Sulla tabella di verità, le condizioni non specificate si indicano con una lineetta in corrispondenza delle configurazioni non ammesse. Nella fig.II.37 è mostrata la tabella di verità per una generica y, funzione delle variabili $w_1w_2w_3$, supponendo le w_i funzioni delle variabili $x_1x_2x_3x_4$, secondo la tabella della fig.II.36.

w_1	w_2	w_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	–
1	0	0	1
1	0	1	0
1	1	0	–
1	1	1	–

Fig.II.37 - Tabella di verità di una funzione incompletamente specificata.

b) Nella rappresentazione della y in forma canonica, le d.c.c. si indicano tra parentesi. Così, la y della fig.II.37 si scrive:

$$y = \overline{w}_1\overline{w}_2\overline{w}_3 + w_1\overline{w}_2\overline{w}_3\ (+\overline{w}_1 w_2 w_3 + w_1 w_2\overline{w}_3 + w_1 w_2 w_3)\ .$$

c) Nella rappresentazione della y con numeri decimali, le d.c.c. si indicano con una seconda sommatoria con indice ϕ. La y della fig.II.37 si scrive:

$$y = \Sigma(0,4) + \Sigma_{\phi}(3,6,7)\ .$$

d) Sulla mappa di Karnaugh, le d.c.c. si rappresentano con il simbolo ϕ nelle caselle le cui coordinate corrispondono alle condizioni non specificate. Nella fig.II.38 è rappresentata la y della fig.II.37.

Fig.II.38 - Mappa di Karnaugh per una funzione incompletamente specificata.

w_3 \ w_1w_2	00	01	11	10
0	1		ϕ	1
1		ϕ	ϕ	

II.10.1 - Semplificazione di una y con d.c.c. sulle mappe di Karnaugh.

Le semplificazioni vanno fatte in modo da coprire tutte e sole le caselle 1, ma i blocchi formati possono comprendere quante si vogliono caselle ϕ. In altre parole, alle caselle ϕ si assegna il valore 1 o 0 a seconda che convenga o no considerarle per formare, con le caselle 1, dei blocchi che altrimenti non esisterebbero.

Esempio: *Semplificare la y la cui mappa di Karnaugh è rappresentata nella figura II.39.*

Conviene assegnare il valore 1 a quattro delle caselle ϕ, e il valore 0 alle altre, per realizzare i due blocchi mostrati in figura. Si ha:

$$y = x_1x_3 + \overline{x}_1\overline{x}_3\ .$$

Se non si fossero considerate le d.c.c., si sarebbe avuto:

$$y = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 + \overline{x}_1x_2\overline{x}_3x_4 + x_1x_2x_3x_4 + x_1\overline{x}_2x_3\overline{x}_4\ .$$

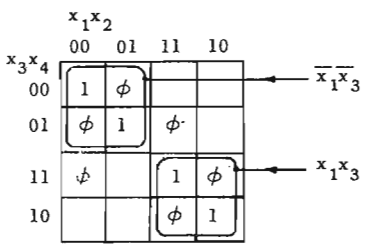


Fig.II.39 - Semplificazione di una funzione incompletamente specificata.

II.10.2 - Semplificazione di una y con d.c.c. col metodo di Quine-Mc Cluskey.

Le tabelle e confronti vanno fatti per tutte le configurazioni per le quali $y = 1$ e $y = \phi$; la copertura, però, va realizzata per le sole configurazioni del primo tipo, costruendo il reticolo senza d.c.c.

0	0 0 0 0	✓
1	0 0 0 1	✓
2	0 0 1 0	✓
4	0 1 0 0	✓
3	0 0 1 1	✓
9	1 0 0 1	✓
10	1 0 1 0	✓
7	0 1 1 1	✓
11	1 0 1 1	✓
13	1 1 0 1	✓
14	1 1 1 0	✓
15	1 1 1 1	✓

0•1	0 0 0 -	✓
0•2	0 0 - 0	✓
0•4	0 - 0 0	A
1•3	0 0 - 1	✓
1•9	- 0 0 1	✓
2•3	0 0 1 -	✓
2•10	- 0 1 0	✓
3•7	0 - 1 1	✓
3•11	- 0 1 1	✓
9•11	1 0 - 1	✓
9•13	1 - 0 1	✓
10•11	1 0 1 -	✓
10•14	1 - 1 0	✓
7•15	- 1 1 1	✓
11•15	1 - 1 1	✓
13•15	1 1 - 1	✓
14•15	1 1 1 -	✓

0•1•2•3	0 0 - -	B
1•3•9•11	- 0 - 1	C
2•3•10•11	- 0 1 -	D
3•7•11•15	- - 1 1	E
9•11•13•15	1 - - 1	F
10•11•14•15	1 - 1 -	G

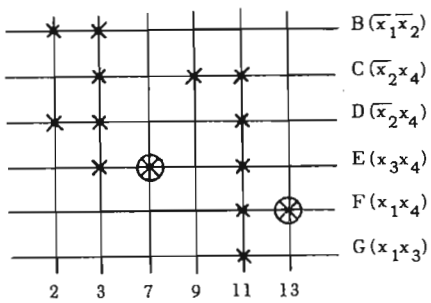


Fig.II.40 - Semplificazione di una funzione incompletamente specificata.

Esempio: *Semplificare col metodo di Quine-Mc Cluskey la funzione:*

$$y = \Sigma\,(2, 3, 7, 9, 11, 13) + \Sigma_{\phi}\,(0, 1, 4, 10, 14, 15) \quad .$$

Nella fig.II.40 sono riportate le tabelle e il reticolo per la copertura minima. Risulta:

$$y = E + F + D = x_3x_4 + x_1x_4 + \overline{x}_2x_3$$

oppure:

$$y = E + F + B = x_3x_4 + x_1x_4 + \overline{x}_1\overline{x}_2 \quad .$$

Si noti che il reticolo non comprende il primo implicante A, perché questo non contiene che minterm d.c.c..

II.11 - Funzioni universali.

Tutte le forme analitiche, canoniche, minime o fattorizzate con cui finora si è rappresentata una funzione logica, contengono, salvo casi particolari, le tre operazioni di somma, prodotto, inversione.

Ci si può chiedere, a questo punto, se non sia possibile esprimere una funzione rappresentata su una tabella di verità con un insieme di operazioni diverse da quelle finora usate o, al limite, se non esista qualche operazione che, da sola, permetta di esprimere qualsiasi funzione.

Dal punto di vista matematico, questi problemi sono assai interessanti. Da quello tecnico, lo sono nella misura in cui esistono elementi che realizzano operazioni diverse dalla somma, prodotto, inversione. Alcuni componenti elettronici, come vedremo nel prossimo capitolo, si trovano appunto in questa condizione.

La teoria della commutazione insegna che esistono delle funzioni, dette *universali*, che permettono di esprimere con un solo operatore qualsiasi funzione logica. Non tutte le funzioni universali sono realizzabili in maniera semplice ed economica; d'altra parte i componenti più comuni realizzano soltanto alcune funzioni, universali o no. In pratica, hanno quindi importanza due problemi:

– esprimere algebricamente, nel modo più semplice, una funzione y con gli operatori corrispondenti alle funzioni universali realizzabili fisicamente;

– esprimere algebricamente, nel modo più semplice, una funzione y con un insieme di operazioni non universali, ma realizzabili con componenti fisicamente compatibili.

Risolveremo i due problemi nei prossimi capitoli, dopo aver descritto le funzioni realizzate dai componenti elettronici. Per concludere questi cenni di algebra logica, intendiamo ora fare una osservazione sull'insieme delle operazioni di somma, prodotto e inversione, ed esporre le più importanti proprietà delle funzioni universali NAND e NOR.

II.11.1 - Osservazione sulle operazioni di somma, prodotto, inversione.

Abbiamo imparato a esprimere analiticamente ogni funzione con le operazioni di somma, prodotto, inversione. Non è però strettamente necessario usare le tre operazioni: applicando il teorema di De Morgan, si dimostra che ogni funzione può essere espressa usando l'operazione di inversione e una sola delle operazioni di somma e prodotto.

Infatti, per trasformare un'espressione contenente delle variabili dirette e negate insieme con gli operatori di somma e prodotto in un'altra equivalente contenente soltanto negazioni e prodotti, basta eliminare le somme con una doppia inversione.

Esempio:

$$x y + \overline{y}\,\overline{z} = \overline{\overline{x\,\overline{y}}\;\overline{\overline{y}\,\overline{z}}}$$

Per trasformare un'espressione contenente delle variabili dirette e negate, insieme con gli operatori di somma e prodotto, in una equivalente contenente soltanto negazioni e somme, basta scrivere l'espressione in forma di prodotto di somme, ed eliminare i prodotti con una doppia negazione.

Esempio:

$$x y + \overline{y}\,\overline{z} = \overline{\overline{x} + \overline{y}} + \overline{y + z} \quad .$$

II.11.2 - Definizione e proprietà dell'operazione NAND.

L'operazione NAND tra due variabili x_1 e x_2 si indica col segno / e si definisce come la negazione del prodotto $x_1 x_2$: NAND deriva da NOT-AND, termini con cui si indicano, in inglese, le operazioni di prodotto e inversione.

$$x_1/x_2 = \overline{x_1 x_2} = \overline{x}_1 + \overline{x}_2 \quad .$$

Nella fig.II.41 è mostrata la tabella di verità della funzione NAND.

L'operazione NAND su più variabili $x_1 x_2 \ldots x_n$ è, per definizione, la negazione del loro prodotto:

x_1	x_2	x_1/x_2
0	0	1
0	1	1
1	0	1
1	1	0

Fig.II.41 - Tabella di verità della funzione NAND.

$$x_1/x_2/\ldots/x_n = x_1 x_2 \ldots x_n = \overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_n \quad .$$

Le principali proprietà dell'operatore NAND sono:

a) $x_1/x_2 = x_2/x_1$: l'operazione è commutativa;

b) $x/0 = 1$;

c) $x/1 = \overline{x}$;

d) $x/x = x/ = \overline{x}$;

e) $\overline{x}_1/\overline{x}_2 = x_1 + x_2$;

f) $x/\overline{x} = 1$;

g) $(x_1 + x_2)/x_2 = \overline{x}_2$;

h) $x_1/x_2/x_3 = \overline{x_1/x_2}/x_3 \neq (x_1/x_2)/x_3$: l'operazione non è associativa; la associazione di più variabili deve essere accompagnata dalla loro negazione. Ad esempio:

$$x_1/x_2/x_3/x_4/x_5/x_6 = (\overline{x_1/x_2})/x_3/(\overline{x_4/x_5/x_6}) \ ;$$

questa proprietà è importante per la sintesi dei circuiti NAND;

i) $(x_1x_2)/(x_3x_4x_5) = (x_1x_2x_3)/(x_4x_5) = x_1/x_2/x_3/x_4/x_5$;

l) $(\overline{x}_1 + x_2 + x_3)/(x_1 + x_2 + x_3) = \overline{x_2 + x_3}$.

Dalle relazioni d) ed e), e da quanto esposto nel par.II.10.1, si deduce che la funzione NAND è una funzione universale. Le regole, derivanti dalle proprietà a) ... l) per scrivere una funzione y col solo segno / sono:

1) si scrive la y nella forma *somma di prodotti*;

2) si mette un segno / al posto di ogni somma o prodotto. Si chiudono tutti i prodotti fra parentesi, e si lasciano invariati i segni di negazione;

3) si invertono le variabili che compaiono da sole;

4) si sostituiscono le variabili negate con le corrispondenti dirette seguite dal segno /, e chiuse tra parentesi.

Esempi.

1) $ABC + DE + F = (A/B/C)/(D/E)/(\overline{F}) =$

$= (A/B/C)/(D/E)/(F/)$

2) $AB + \overline{C} = (A/B)/C$

3) $\overline{A}\,\overline{B} + \overline{A}D + \overline{B}CD + B\overline{C}D = (\overline{A}/\overline{B})/(\overline{A}/D)/(\overline{B}/C/D)/(B/\overline{C}/D) =$

$= [(A/)/(B/)]/[(A/)D]/[(B/)/C/D]/[B/(C/)/D]$.

II.11.3 - Definizione e proprietà dell'operazione NOR.

L'operazione NOR tra due variabili x_1 e x_2 si indica col segno $\downarrow$ e si definisce come la negazione della somma $x_1 + x_2$: NOR deriva da NOT-OR, termine con cui, in inglese, si indicano le operazioni di inversione e di somma.

$$x_1 \downarrow x_2 = \overline{x_1 + x_2} = \overline{x}_1\overline{x}_2 \ .$$

Nella fig.II.42 è mostrata la tabella di verità della funzione NOR.

L'operazione NOR su più variabili $x_1 x_2 \ldots x_n$ è, per definizione, la negazione della loro somma:

$$x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 + x_2 + \ldots + x_n} = \overline{x}_1 \overline{x}_2 \ldots \overline{x}_n \ .$$

Le principali proprietà dell'operazione NOR sono:

x_1	x_2	$x_1 \downarrow x_2$
0	0	1
0	1	0
1	0	0
1	0	0

Fig.II.42 - Tabella di verità della funzione NOR.

a) $x_1 \downarrow x_2 = x_2 \downarrow x_1$: l'operazione è commutativa;

b) $x \downarrow 0 = \overline{x}$;

c) $x \downarrow 1 = 0$;

d) $x \downarrow x = x \downarrow = \overline{x}$;

e) $\overline{x}_1 \downarrow \overline{x}_2 = x_1 x_2$;

f) $x \downarrow \overline{x} = 0$;

g) $x_1 x_2 \downarrow x_2 = \overline{x}_2$;

h) $x_1 \downarrow x_2 \downarrow x_3 = \overline{x_1 \downarrow x_2} \downarrow x_3 \neq (x_1 \downarrow x_2) \downarrow x_3$: l'operazione non è associativa. Ogni associazione di variabili va accompagnata dalla loro negazione. Ad esempio:

$$x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4 \downarrow x_5 = (\overline{x_1 \downarrow x_2}) \downarrow (\overline{x_3 \downarrow x_4}) \downarrow x_5 \ ;$$

questa proprietà è importante per la sintesi dei circuiti NOR.

i) $(x_1 + x_2) \downarrow (x_3 + x_4 + x_5) = (x_1 + x_2 + x_3) \downarrow (x_4 + x_5) = x_1 \downarrow x_2 \downarrow x_3 \downarrow x_4 \downarrow x_5$;

l) $(x_1 x_2 x_3) \downarrow (\overline{x}_1 x_2 x_3) = \overline{x_2 x_3}$.

Dalle relazioni d) ed e), e da quanto esposto nel par.II.10.2, si deduce che il NOR è una funzione universale. Le regole per scrivere una funzione y col solo segno $\downarrow$ derivano dalle proprietà a) ... l):

1) si scrive la y come prodotto di somma;

2) si mette un segno $\downarrow$ al posto di ogni segno di somma o prodotto, lasciando tra parentesi le somme, e invariati i segni di negazione;

3) si invertono le variabili che compaiono da sole;

4) si sostituiscono le variabili negate con le corrispondenti dirette seguite dal segno $\downarrow$ e chiuse fra parentesi.

Esempi.

1) $(x_1 + x_2 + x_3)(x_4 + x_5) x_6 = (x_1 \downarrow x_2 \downarrow x_3) \downarrow (x_4 \downarrow x_5) \downarrow x_6 = (x_1 \downarrow x_2 \downarrow x_3) \downarrow (x_4 \downarrow x_5) \downarrow (x_6 \downarrow)$

2) $(\overline{x}_1+\overline{x}_2+\overline{x}_3)(\overline{x}_1+x_4)(\overline{x}_2+x_4) = (\overline{x}_1 \downarrow \overline{x}_2 \downarrow \overline{x}_3) \downarrow (\overline{x}_1 \downarrow x_4) \downarrow (\overline{x}_2 \downarrow x_4) =$

$$= [(x_1\downarrow)\downarrow(x_2\downarrow)\downarrow(x_3\downarrow)]\downarrow[(x_1\downarrow)\downarrow x_4]\downarrow[(x_2\downarrow)\downarrow x_4] \ .$$

Dalle definizioni stesse di NAND e NOR, si ricavano infine due notevoli proprietà:

$$\overline{x_1/x_2/\ldots/x_n} = \overline{x}_1 \downarrow \overline{x}_2 \downarrow \ldots \downarrow \overline{x}_n$$

$$\overline{x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n} = \overline{x}_1/\overline{x}_2/\ldots/\overline{x}_n \ .$$

Da queste. deriva una forma generalizzata del teorema di De Morgan:

$$\overline{F}(x_1, x_2, \ldots, x_n, +, \cdot, \downarrow, /) = F(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n, \cdot, +, /, \downarrow)$$

cioè: la negazione di un'espressione dove compaiono le variabili $x_1 x_2 \ldots$ $\ldots x_n$ insieme con gli operatori somma, prodotto, NAND, NOR, si ottiene negando tutte le variabili e scambiando il prodotto con la somma e il NAND con il NOR.

A proposito di questa relazione, si noti che, pur essendo NAND e NOR due funzioni universali, nulla vieta di scrivere delle espressioni dove essi compaiono con altri operatori. Praticamente, anzi, il segno / (o il segno $\downarrow$) si usa insieme col segno di inversione: vedremo in seguito l'opportunità di questa notazione.

*

BIBLIOGRAFIA

1. BOOLE G.: *The Mathematical Analysis of Logic* - Cambridge, 1847.
2. BOOLE G.: *An Investigation of the Laws of Thought* - London, 1854. Ristampa Dover (New York) 1954.
3. RIGHI R.: *Algebra Booleana e applicazioni* - Siderea, 1966.
4. BIRKHOFF G.: *A Survey of Modern Algebra* - Macmillan, 1953.
5. CALDWELL S.: *Switching Circuits and Logical Design* - John Wiley & Sons, 1958.
6. HOHN F.F.: *Applied Boolean Algebra, An Elementary Introduction* - Macmillan, 1960.
7. KARNAUGH M.: *The Map Method of Synthesis of Combinational Logic Circuits* - Communications and Electronics - n.9, Novembre 1953.
8. PHISTER M.J.: *Logical Design of Digital Computers* - John Wiley & Sons, 1958.
9. SHANNON C.E.: *A Symbolic Analysis of Relay and Switching Circuits* - Transactions of the AIEE - Vol.57, 1938.
10. CURTIS H.A.: *A new Approach to the Design of Switching Circuits* - D. Van Nostrand, 1962.
11. ROTH J.P. & KARP R.M.: *Minimization over Bolean Graphs* - IBM Journal of Research and Development - Vol. 6, n. 2, Aprile 1962.
12. COBHAM A. & altri: *An application of Linear Programming to the Minimization of Boolean Functions* - Symposium AIEE, Settembre 1961.
13. Mc CLUSKEY E.J.: *Minimization of Boolean Functions* - Bell System Technical Journal - Vol. 35, Novembre 1956.
14. Mc CLUSKEY E.J. & BARTEE T.C.: *A Survey of Switching Theory* - Mc Graw Hill, 1962.
15. FALZONE V.: *Il progetto dei circuiti di Commutazione* - Siderea, 1966.
16. NELSON R.J.: *Simplest normal truth function* - J.Symbolic Logic, Vol.20, n.2, 1955.

*

CAPITOLO III

CIRCUITI ELEMENTARI DI LOGICA E DI MEMORIA

III.1 - Generalità.

Le complesse operazioni logiche ed aritmetiche che avvengono in un calcolatore elettronico o in un qualsiasi sistema numerico (intendendo con questa dizione ogni apparecchiatura che elabora o immagazzina informazioni binarie), vengono realizzate nei circuiti di commutazione. Questi circuiti sono composti da un elevato numero di componenti elementari, appartenenti, però, a pochi tipi fondamentali.

I primi circuiti furono realizzati a contatti e relè, nel modo descritto nel precedente capitolo: erano circuiti lenti, ingombranti e di difficile manutenzione, che furono subito sostituiti da altri, con tubi a vuoto. I tubi permisero la costruzione di circuiti assai più veloci, ma presentavano, oltre all'ingombro e alla scarsa affidabilità, l'inconveniente di una grande dissipazione di potenza.

Gli unici componenti attualmente usati sono quelli a stato solido, di piccole dimensioni, elevata velocità di funzionamento, alto grado di affidabilità. Ai primi circuiti di questo tipo, realizzati soltanto con diodi e transistor, vanno anzi gradatamente sostituendosi i circuiti integrati, comprendenti un intero circuito elementare o più circuiti elementari nello stesso contenitore. Questi ultimi componenti, chiamati, in generale, *microcircuiti* o *micrologici*, hanno elevatissime prestazioni, soprattutto nei riguardi del grado di affidamento, della potenza dissipata e della varietà e complessità delle funzioni realizzate.

In questo capitolo descriveremo le proprietà logiche dei circuiti elementari a diodi e transistor, supponendo note le proprietà fisiche di

tali elementi; accenneremo rapidamente a quelle dei circuiti micrologici e introdurremo, infine, elementi, simboli e caratteristiche delle funzioni di memoria, il cui uso e significato verrà trattato negli ultimi capitoli.

III.2 - Definizioni.

Una funzione logica si realizza circuitalmente usando dei componenti – i diodi e i transistor – capaci di assumere l'uno o l'altro di due stati elettrici diversi, a seconda che su un loro terminale di ingresso sia applicato l'uno o l'altro di due determinati valori di tensione. Nei circuiti elettronici, infatti, i valori 0 e 1 dell'algebra logica sono rappresentati da due livelli caratteristici di tensione, detti livello alto (h) e basso (b). La effettiva corrispondenza tra h e b da una parte e 0 e 1 dalla altra è convenzionale, e va precisata di volta in volta.

Si chiama *logica positiva* la convenzione secondo cui il valore 1 si assegna al livello h; *logica negativa*, quella in cui il valore 1 si assegna al livello b.

Si chiama *circuito logico elementare* o *circuito porta* ogni circuito elettronico ad n ingressi ed una uscita il cui valore, secondo una delle convenzioni precedenti, è 1 in corrispondenza alle configurazioni degli ingressi descritte dalle funzioni somma, prodotto, inversione (per n = = 1), NAND e NOR.

I circuiti logici elementari permettono di realizzare qualsiasi funzione logica, nel modo che vedremo nei prossimi capitoli.

III.3 - Circuiti OR e AND a diodi.

Nella fig.III.1 è riportata la curva caratteristica tensione-corrente di un diodo: si ha una piccola resistenza (qualche Ω) quando la tensione di polarizzazione è tale da rendere l'anodo positivo rispetto al catodo, e una resistenza molto elevata (dell'ordine dei MΩ) quando la tensione di polarizzazione ha senso contrario.

Il comportamento di un diodo, nel montaggio della fig.III.2a), sotto i due valori caratteristici della tensione di alimentazione di +10 V (livello h) e 0 V (livello b), è il seguente: quando il terminale A d'ingresso è a 0 V, la tensione di alimentazione di + 10 V polarizza il diodo direttamente e fa passare una corrente praticamente uguale a 10/R, dato il piccolo valore della resistenza diretta del diodo rispetto a R (qualche kΩ); la tensione rispetto a terra del terminale B d'uscita è pressappoco 0 V (fig.III.2b).

Quando il punto A è a +10 V, nessuna corrente attraversa il diodo; sul punto B si ha una tensione di +10 V rispetto a terra (fig.III.2c).

L'uscita del circuito della fig.III.2 segue dunque esattamente lo ingresso, e non compie nessuna funzione logica.

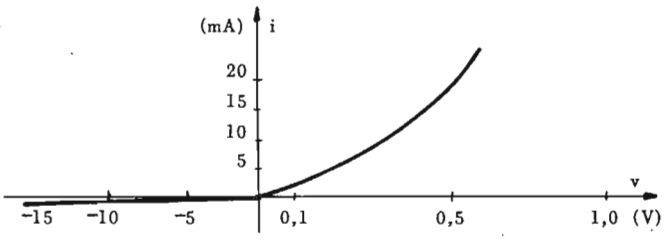


Fig.III.1 - Curva caratteristica di un diodo.

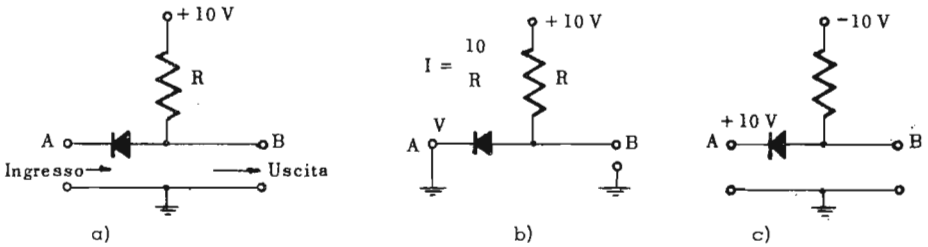
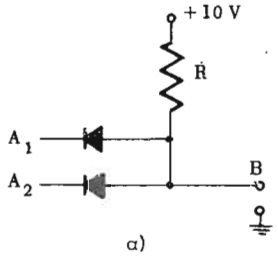


Fig.III.2 - Comportamento di un diodo sotto 2 valori caratteristici di tensione.



A_1	A_2	B
b	b	b
b	h	b
h	b	b
h	h	h

b)

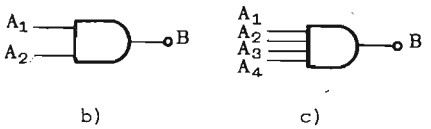
Fig.III.3 - Relazione tra le tensioni di ingresso e d'uscita (b) del circuito a diodi (a).

In un circuito con due diodi in parallelo, disposti come mostrato nella fig.III.3a, se A_1, A_2 o entrambi sono a 0 V, l'uscita è anch'essa a 0 V, perché si ha un passaggio di corrente attraverso R ed il diodo, o i diodi, polarizzati direttamente. L'uscita è invece a +10 V quando A_1

e A_2 sono entrambi a +10 V, cioè i due diodi sono polarizzati inversamente. I valori dell'uscita in corrispondenza alle quattro possibili coppie dei valori d'ingresso sono riportati nella tabella della fig.III.3b, in cui h e b rappresentano, rispettivamente, i valori di +10 V e 0 V.

a)

A_1	A_2	B
0	0	0
0	1	0
1	0	0
1	1	1



b) c)

Fig.III.4 - a) Tabella di verità - in logica positiva - del circuito di fig.III.3a; b) simbolo di una porta AND a 2 ingressi; c) simbolo di una porta AND a 4 ingressi.

La tabella della fig.III.3b, usando la definizione di logica positiva, cioè h = 1, b = 0, può essere trasformata nella tabella di verità della variabile B in funzione delle variabili A_1 e A_2 (fig.III.4a). B vale 1 quando sia A_1 che A_2 valgono 1: il circuito della fig.III.3a, che realizza la funzione prodotto $B = A_1 A_2$, si chiama *circuito porta AND* (AND gate) o, più semplicemente, AND, e si indica col simbolo della fig.III.4b.

I circuiti AND possono, evidentemente, avere più di due ingressi: nella fig.III.4c è disegnato simbolicamente un circuito a 4 ingressi.

a)

A_1	A_2	B
0	0	0
0	1	1
1	0	1
1	1	1

b)

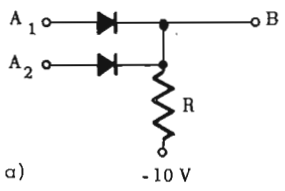
Fig.III.5 - a) Tabella di verità - in logica negativa - del circuito di figura III.3a; b) simbolo di una porta OR a 2 ingressi.

La stessa tabella della fig.III.3b, adottando le definizioni della logica negativa (h = 0, b = 1), si trasforma in una diversa tabella di verità della variabile B in funzione di A_1 e A_2 (fig.III.5a): B ha il valore 1 quando una o entrambe, le variabili A_1 e A_2 valgono 1. In logica negativa, il circuito della fig.III.3a, che realizza la funzione somma $B = A_1 + A_2$, si chiama *circuito porta OR* (OR gate) o più semplicemente OR, e si indica col simbolo della fig.III.5b. Un circuito OR può evidentemente avere più di due ingressi.

Un secondo tipo di porta a diodi è mostrato nella fig.III.6a: gli ingressi sono applicati agli anodi dei diodi e la tensione di polarizzazione è negativa e uguale a -10 V. L'uscita è a -10 V (livello b) solo

quando entrambi gli ingressi sono a -10 V, ed è uguale a 0 V (livello h) in tutti gli altri casi (v. tabella della fig.III.6b).

In logica positiva, il circuito realizza la funzione somma: è cioè un OR, e si indica ancora col simbolo della fig.III.5b; in logica negativa realizza la funzione prodotto; è quindi un AND e si indica sempre col simbolo della fig.III.4b.



a)

A_1	A_2	B
b	b	b
b	h	h
h	b	h
h	h	h

b)

Fig.III.6 - Relazioni tra le tensioni di ingresso e d'uscita (b) del circuito a diodi (a).

Sui circuiti AND-OR a diodi è importante notare che:

- I simboli prescindono dal tipo di logica, quindi di circuito, usato, riferendosi soltanto alla funzione realizzata sul terminale d'uscita.

- Il numero massimo di diodi in un circuito è legato alla resistenza inversa dei diodi stessi, che è grande ma non infinita. Attualmente, deve essere inferiore a 10.

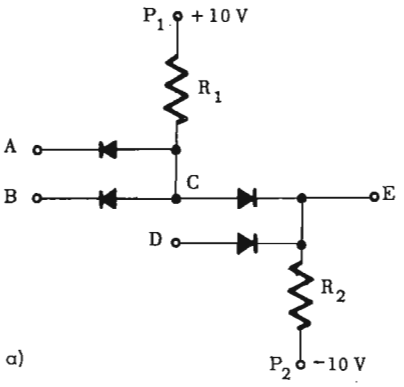
- Un circuito porta può essere usato come ingresso di un secondo circuito dello stesso tipo, calcolando opportunamente le resistenze di carico. I diodi sono comunque elementi passivi, e vanno usati sempre insieme ad elementi attivi. Sull'interconnessione dei circuiti è fatto un cenno nel paragrafo successivo.

- I circuiti a diodi non sono universali, perché non permettono di realizzare l'inversione delle variabili (cioè delle tensioni) d'ingresso. Come invertitori ed elementi attivi, si usano i transistor.

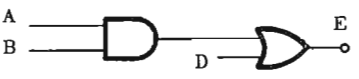
III.4 - Interconnessione dei circuiti AND-OR a diodi.

La necessità di mantenere i livelli delle tensioni all'uscita di un circuito entro valori non eccessivamente distanti da quelli teorici h e b, e di realizzare una elevata velocità nella risposta alle variazioni degli ingressi, limita il numero di AND e OR in serie, e impone precise relazioni tra le resistenze di carico dei circuiti stessi.

Per illustrare il problema, studiamo – tenendo conto delle caratteristiche reali dei diodi – il tipico accoppiamento AND-OR in logica positiva della fig.III.7a, prendendo h= +10 V, b = 0 V, e supponendo dapprima $R_1 = R_2 = 1\ k\Omega$. Nella fig.III.7b è mostrato lo schema simbolico del circuito. Quando gli ingressi A e B all'AND sono entrambi 0 (= 0 V) nel punto C si ha una tensione, pressappoco nulla, eguale alla caduta di tensione (V_D) sulla resistenza dei diodi polarizzati direttamente. Se l'ingresso D dello OR è a 1 (= +10 V), l'uscita del circuito (punto E) si trova $+10 - V_D \simeq +10$ V.



a)



b)

Fig.III.7 - Accoppiamento AND-OR in logica positiva (a) e suo schema logico (b).

Quando, invece, A e B sono entrambi a 1, l'uscita dell'AND è anch'essa a 1, cioè il punto C è a +10 V. Anche il punto E dovrebbe essere a +10 V, ma nel circuito $P_1 R_1 C R_2 P_2$, si ha il passaggio di una corrente i che, trascurando la resistenza diretta del diodo è:

$$i = \frac{V}{R} = \frac{20}{R_1 + R_2} = 10\ mA \quad .$$

La tensione verso massa nei punti C ed E è quindi:

$$V_E = i R_2 - 10 = 10 - 10 = 0\ V \quad .$$

Poiché 0 V corrisponde al valore logico 0, il circuito non realizza la funzione voluta.

Per avere un funzionamento accettabile, deve essere:

$$R_2 \gg R_1 \quad .$$

Ciò si può ottenere aumentando R_2, o diminuendo R_1.

a) Prendendo $R_2 = 10\ R_1 = 10\ k\Omega$, quando A e B sono ad 1, si ha una tensione V_E di uscita:

$$V_E = \left(\frac{R_2}{R_1 + R_2}\right) 20 - 10 = \frac{10}{11} 20 - 10 = 8{,}2\ V \quad .$$

Questo valore è ancora sufficientemente elevato per essere riconosciuto come h, quindi come 1, nella maggior parte delle applicazioni. La situazione, comunque, peggiorerebbe di nuovo in maniera inaccettabile se l'AND fosse collegato in uscita con 2 OR (fig.III.8) . In questo caso il punto C verrebbe a trovarsi su due resistenze R_2 in parallelo, e le uscite dei due OR, ponendo $R = R_2/2 = 5\,k\Omega$, sarebbero:

$$V_{E_1} = V_{E_2} = \frac{R}{R_1 + R}\,20 - 10 =$$

$$= \frac{5}{6}\,20 - 10 = 6{,}6\ V \quad .$$

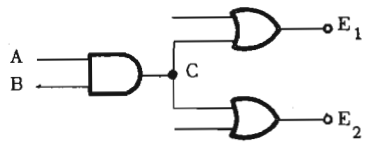


Fig.III.8 - Collegamento dell'uscita AND con 2 OR in parallelo.

Occorrerebbe aumentare ancora R_2: ma non si può andare oltre un certo limite, per non diminuire le velocità di risposta del circuito. Esiste, infatti, una capacità distribuita C_d, dovuta ai collegamenti e ai diodi, che può essere schematizzata come mostrato nella fig.III.9. Quando A e B sono 1, C_d si carica: quando il punto C, per la variazione di A o B va a 0, il diodo D viene a trovarsi polarizzato inversamente, e C_d si scarica su R_2. La costante di tempo C_dR_2 determina la velocità di risposta del circuito, cioè il tempo dopo il quale il punto E va a 0. Se $k_1 = 100\ k\Omega$, con un valore di $C_d = 100$ pF si ha $\tau = R_2C_d = 10\ \mu$sec: l'uscita del circuito non raggiunge un valore accettabile prima di un tempo $\tau = 40\ \mu$sec: la frequenza massima di funzionamento è quindi, di soli 25 kHz.

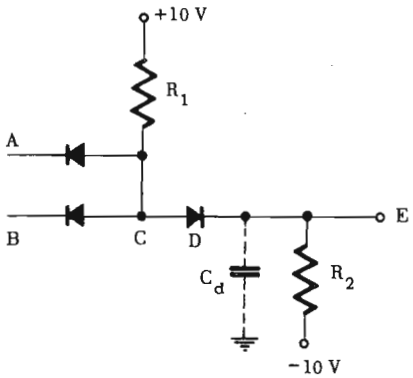


Fig.III.9 - Capacità distribuita in un circuito a diodi.

b) Risultati non migliori si sarebbero ottenuti diminuendo R_1: ci sarebbe stato un buon livello della tensione d'uscita, ma una troppo elevata corrente attraverso i diodi. È invece opportuno che diodi e circuiti operino alla minima potenza possibile, specialmente in quei sistemi, per esempio i calcolatori, dove il loro numero è altissimo.

L'effettiva limitazione al numero degli stadi AND-OR in serie può essere superata usando circuiti attivi, come l'invertitore, per amplificare e riformare il segnale, o ricorrendo ad altri tipi di logica, come vedremo nel seguito.

III.5 - Circuito invertitore a transistor.

Nei circuiti logici, il transistor viene usato principalmente come invertitore; serve, però, anche ad amplificare i segnali in uscita dai circuiti passivi. Poiché devono rappresentare gli stati 0 e 1 in modo che risultino distinguibili senza ambiguità, i transistor vengono fatti lavorare esclusivamente nelle due opposte condizioni di interdizione e saturazione.

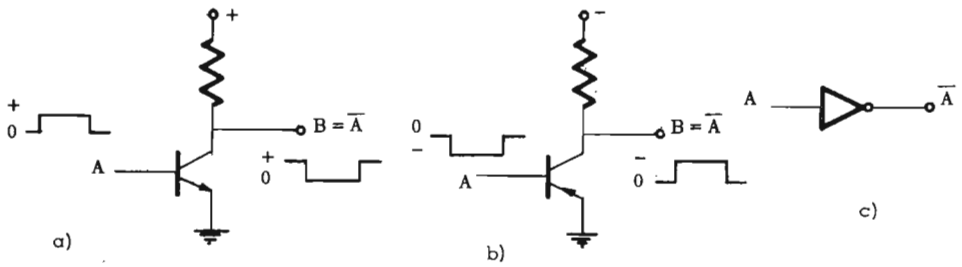


Fig.III.10 - Circuiti invertitori: a transistor n-p-n (a), a transistor p-n-p (b), e loro simbolo logico (c).

In un transistor n-p-n (fig.III.10a), quando la base è a massa, o a un potenziale negativo rispetto a massa, la corrente di base è nulla: il transistor si comporta allora come un contatto aperto, e il suo collettore è al livello positivo. Se la base viene portata ad un potenziale positivo rispetto a massa, il transistor si comporta come un corto circuito e il suo potenziale di collettore va a 0. Se si usa un transistor pnp (fig.III.10b), tutte le polarità vanno invertite. In entrambi i casi, il simbolo di un circuito invertitore (NOT) è quello della fig.III.10c, e prescinde dal tipo di logica e di circuito usati.

In pratica, esistono deviazioni, nel comportamento di un transistor, da quello ideale descritto; come si può vedere esaminando un circuito tipico e una famiglia di curve caratteristiche (fig.III.11). La retta di carico definisce la regione di operazione del transistor: quando il transistor è in saturazione (punto A), si ha una caduta di tensione ai suoi capi che è piccola (qualche decimo di volt) ma non nulla: all'interdizione (punto B) si ha una corrente, I_{C_0}, che provoca una caduta, $I_{C_0}R_L$, per cui la tensione di interdizione è leggermente inferiore a quella di alimentazione. $(V_{CE})_{sat}$ e $(V_{CE})_{int}$ sono comunque tali da poter essere considerati senz'altro rappresentativi dei valori logici 0 e 1 in logica positiva, oppure 1 e 0 in logica negativa.

Il tempo di commutazione di un transistor, per ragioni che qui non interessano, è piccolissimo ma non nullo (da un microsecondo a qualche nanosecondo): la conside-

razione di questo ritardo tra ingresso e uscita è di fondamentale importanza per un tipo di circuiti che vedremo nei prossimi capitoli.

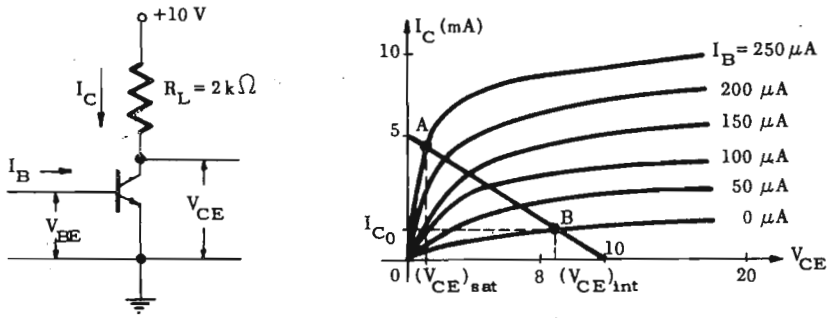


Fig.III.11 - Circuito invertitore reale.

Un semplice circuito invertitore realmente usato è quello della fig.III.12.

La resistenza di ingresso R_B va scelta in modo che, con la minima tensione di ingresso, la corrente di base sia sufficiente a portare il transistor in saturazione: la resistenza di collettore R_C non va presa troppo alta, per non aumentare la caduta $R_C I_{C0}$ all'interdizione: può essere, a titolo d'esempio, $V_{CC} = +10$ V, $R_C = 2$ kΩ, $R_B = 10$ kΩ.

Esistono diverse altre realizzazioni, più o meno complesse, del circuito invertitore, che hanno lo scopo di migliorare la risposta e la velocità del circuito stesso: dal punto di vista logico, tuttavia, esse non si differenziano da quella descritta.

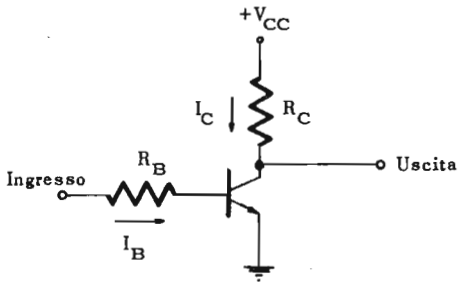


Fig.III.12 - Circuito invertitore con resistenza di ingresso.

III.6 - Circuiti NAND e NOR a transistor e diodi (DTL).

I circuiti NAND e NOR sono stati realizzati per sintetizzare qualsiasi funzione logica senza restrizioni e con l'uso di un solo elemento, dato che gli operatori logici NAND e NOR sono universali.

Un primo circuito è quello della fig.III.13, ottenuto dall'unione di una porta a diodi con un invertitore (tra parentesi sono riportati dei valori indicativi per le tensioni e le resistenze).

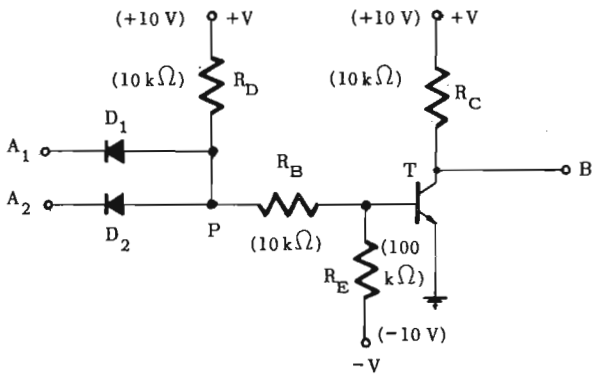


Fig.III.13 - Circuito formato da una porta a diodi e da un invertitore.

Quando A_1 e A_2 sono al potenziale +V, i diodi D_1 e D_2 sono bloccati, passa corrente nel circuito R_D-R_B-R_E e, per essere $R_B \ll R_E$, la base del transistor viene a trovarsi a un potenziale positivo rispetto a massa e sufficiente a mantenere il transistor in saturazione. L'uscita B è, pertanto, praticamente a 0 V.

Se uno degli ingressi (per esempio A_1) viene messo a 0 V, il diodo D_1 conduce: il punto P va a massa, e la base del transistor diventa negativa. Il transistor commuta nello stato di interdizione e l'uscita B si porta al potenziale +V.

Ponendo +V = h, 0 V = b, i valori dell'uscita B in corrispondenza delle quattro possibili coppie di valori d'ingresso sono riportati nella tabella di fig.III.14a. In logica positiva (fig.III.14b) il circuito realizza la funzione NAND ($B = \overline{A_1 A_2}$); in logica negativa (fig.III.14c) la funzione NOR ($B = \overline{A_1 + A_2}$).

a)

A_1	A_2	B
b	b	h
b	h	h
h	b	h
h	h	b

b)

A_1	A_2	B
0	0	1
0	1	1
1	0	1
1	1	0

c)

A_1	A_2	B
0	0	1
0	1	0
1	0	0
1	1	0

Fig.III.14 - Relazioni tra le tensioni d'ingresso e d'uscita del circuito di figura III.13a; loro interpretazioni booleane in logica positiva (b) e negativa (c).

Un secondo circuito si ottiene invertendo i diodi e le polarizzazioni, e sostituendo il transistor npn con un pnp (fig. III.15). Ponendo 0 V = h e -V = b, i valori dell'uscita in corrispondenza delle quattro possibili coppie dei valori d'ingresso sono riportati nella tabella della figura III.16a. In logica positiva (fig.III.16b), il circuito realizza la funzione NOR ($B = \overline{A_1 + A_2}$); in logica negativa (fig. III.16c), la funzione NAND ($B = \overline{A_1 A_2}$).

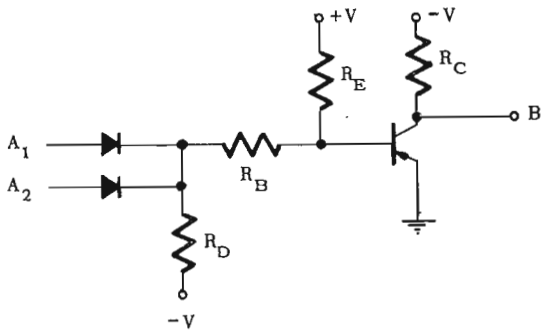


Fig.III.15 - Circuito formato da una porta a diodi e da un invertitore.

a)

A_1	A_2	B
b	b	h
b	h	b
h	b	b
h	h	b

b)

A_1	A_2	B
0	0	1
0	1	0
1	0	0
1	1	0

c)

A_1	A_2	B
0	0	1
0	1	1
1	0	1
1	1	0

Fig.III.16 - Relazioni tra le tensioni d'ingresso e d'uscita del circuito di fig.III.15a; loro interpretazioni booleane in logica positiva (b) e negativa (c).

I circuiti delle figg. III.13 e III.15 si chiamano circuiti logici a diodi e transistor (DTL: Diode Transistor Logic). Indipendentemente dal circuito, quindi dalla logica usata, NAND e NOR si indicano rispettivamente, coi simboli della fig.III.17.

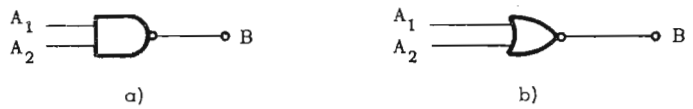


Fig.III.17 - Simboli di un circuito NAND (a) e NOR (b) a 2 ingressi.

I circuiti delle figg. III.13 e III.15 sono oggi praticamente sostituiti da quello mostrato nella fig.III.18 che elimina, tra l'altro, la necessità di una doppia tensione di alimentazione.

In questo circuito il transistor T serve a dare il guadagno e la potenza per mantenere i livelli logici e pilotare altri circuiti; i diodi D_1 e D_2 realizzano la porta d'ingresso; i diodi D_3 e D_4 creano una soglia alla commutazione di T, utile ad una discriminazione di eventuali segnali di disturbo. Il terminale A serve ad aumentare il numero degli ingressi (il cosiddetto *fan-in*) aggiungendo semplicemente altri diodi.

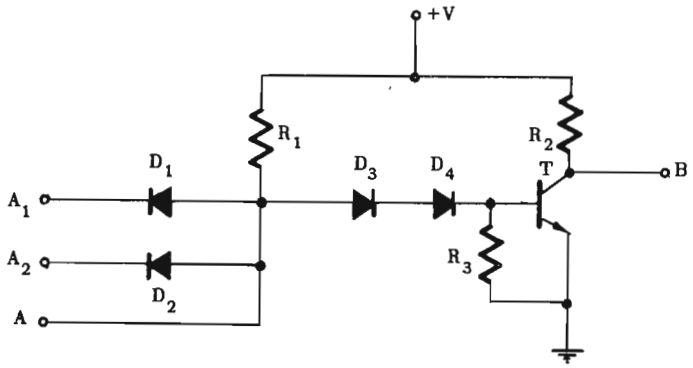


Fig.III.18 - Circuito DTL a una sola tensione di alimentazione.

Nel circuito di fig.III.18 si ha una commutazione di corrente attraverso R_1 che, quando un ingresso (p.es. A_1) è a 0 V, attraversa D_1 e non riesce ad entrare nella base di T e, quando tutti gli ingressi sono a +V, percorre i diodi D_3 e D_4 e satura il transistor, in maniera analoga a quella vista per i circuiti precedenti.

III.7 - Circuiti NAND e NOR con transistor e resistenze (RTL) e interamente a transistor (TTL).

Dai circuiti DTL possono considerarsi derivate due altre famiglie logiche: la RTL (Resistor Transistor Logic) e la TTL (Transistor Transistor Logic).

I circuiti RTL si ottengono sostituendo semplicemente delle resistenze ai diodi d'ingresso, nel modo illustrato nella fig. III.19 (valori indicativi delle resistenze sono: $R_L = 10\ k\Omega$, $R_1 = 2\ k\Omega$, $R_2 = 100\ k\Omega$). I simboli usati per questi circuiti, attualmente poco diffusi per il loro basso *fan-in* e la scarsa affidabilità, sono sempre quelli della fig.III.17.

I circuiti TTL si ottengono invece sostituendo i diodi D_1 e D_2 del circuito di fig.III.18 con i diodi base-emettitore di un transistor multiemettitore TME, e i diodi D_3 e D_4 con il diodo base-collettore dello stesso TME (fig.III.20).

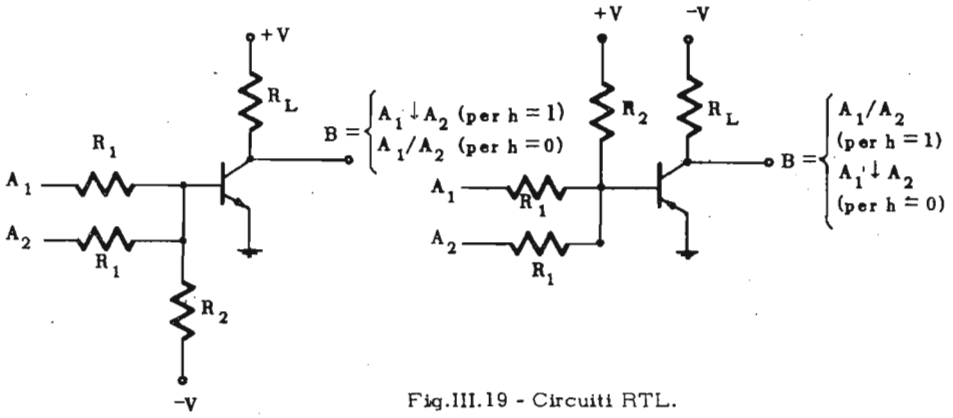


Fig.III.19 - Circuiti RTL.

Questi circuiti funzionano a velocità molto maggiori dei DTL, ma presentano – nei riguardi di questi ultimi – lo svantaggio di un più elevato margine di rumore.

I simboli logici, come per qualsiasi altra realizzazione circuitale dei NAND e NOR sono sempre quelli della figura III.17.

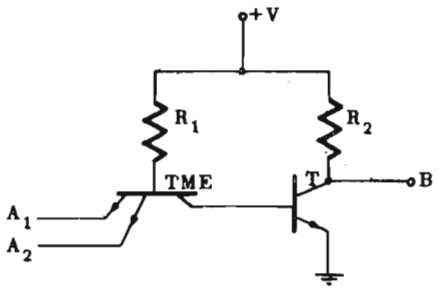


Fig.III.20 - Circuito TTL.

III.8 - Circuiti NAND e NOR con transistor ad accoppiamento diretto (DCTL).

Si possono costruire dei circuiti logici usando i transistor come elementi di ingresso al circuito, ed eliminando diodi e resistenze. I circuiti così ottenuti, chiamati DCTL (Direct Coupled Transistor Logic), sono più rapidi di quelli visti precedentemente, operano a tensioni più basse, quindi con minor dissipazione di potenza, ma richiedono uniformità nelle caratteristiche dei transistor usati.

La fig.III.21 mostra un classico circuito DCTL, costruito con transistor npn, che si comporta come due contatti in serie, ha cioè l'uscita B al livello basso solo se i due transistor conducono; in ogni altra condizione, nella resistenza R_L non passa corrente, e su B si ha il livello alto (fig.III.22a). A seconda che si usi la logica positiva (fig.III.22b) o quella negativa (fig.III.22c), il circuito si comporta quindi come un NAND o come un NOR.

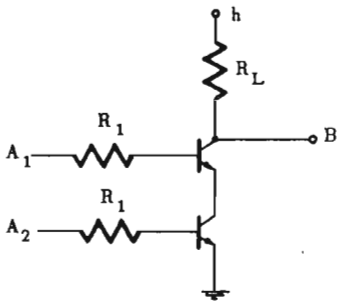


Fig.III.21 - Circuito DCTL con transistor in serie.

Un altro circuito DCTL, sempre con transistor npn è quello della fig.III.23, che si comporta come se fosse costituito da due contatti in parallelo: quando i due transistor sono interdetti, cioè A_1 e A_2 sono entrambi al livello b, non passa corrente in R_L, quindi sull'uscita B si ha il livello h; quando almeno uno dei transistor conduce, su R_L si ha una caduta di tensione praticamente uguale ad h, e l'uscita è al livello b (fig.III.24a). A seconda che si usi la logica positiva (fig.III.24b) o quella negativa (fig.III.24c), il circuito si comporta quindi come un NOR o come un NAND.

a)

A_1	A_2	B
b	b	h
b	h	h
h	b	h
h	h	b

b)

A_1	A_2	B
0	0	1
0	1	1
1	0	1
1	1	0

c)

A_1	A_2	B
0	0	1
0	1	0
1	0	0
1	1	0

Fig.III.22 - Relazioni tra le tensioni d'ingresso e d'uscita del circuito di fig.III.20a; loro interpretazioni booleane in logica positiva (b) e negativa (c).

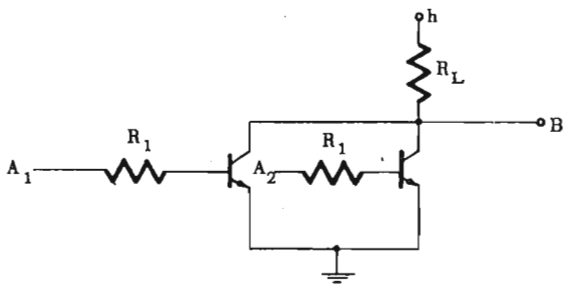


Fig.III.23 - Circuito DCTL con transistor in parallelo.

I circuiti delle figg.III.21 e III.23 possono essere realizzati, ovviamente, anche con transistor p-n-p. A titolo d'esempio, nella fig.III.25 è mostrato un circuito NOR a tre ingressi, in logica positiva.

Una particolarità interessante della logica DCTL, nei confronti della RTL o DTL, è che si possono costruire circuiti complessi usando insieme elementi NAND e NOR: in logica negativa o in logica positiva, infatti, NAND e NOR hanno lo stesso tipo di transistor e gli stessi livelli di ingresso; nella logica DTL, invece, sia in logica negativa che in logica positiva, NAND e NOR sono realizzati con transistor di tipo opposto, e usano valori differenti delle tensioni d'ingresso.

a)

A_1	A_2	B
b	b	h
b	h	b
h	b	b
h	h	b

b)

A_1	A_2	B
0	0	1
0	1	0
1	0	0
1	1	0

c)

A_1	A_2	B
0	0	1
0	1	1
1	0	1
1	1	0

Fig.III.24 - Relazioni tra le tensioni d'ingresso e d'uscita del circuito di fig.II.22a; loro interpretazioni booleane in logica positiva (b) e negativa (c).

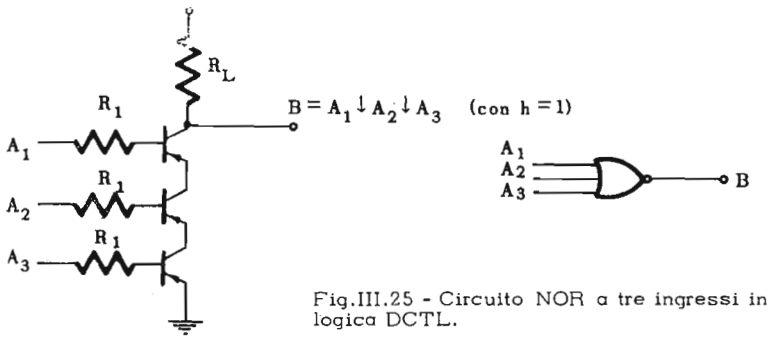


Fig.III.25 - Circuito NOR a tre ingressi in logica DCTL.

III.9 - Il montaggio «WIRED-OR».

La possibilità di realizzare il NOR mettendo in parallelo i collettori di due transistor permette un montaggio particolare dei circuiti realizzati secondo la logica DTL, il cosiddetto *wired-OR* (OR di collettore), mostrato nella fig.III.26a.

In logica positiva, sull'uscita B si ha la funzione NOR degli ingressi (H e K) dei transistor T_1 e T_2; ma H e K sono le uscite degli AND A_1A_2 e A_3A_4, quindi:

$$B = \overline{H+K} = \overline{A_1A_2 + A_3A_4} = (\overline{A}_1 + \overline{A}_2)(\overline{A}_3 + \overline{A}_4) \quad .$$

Il simbolo logico del *wired-OR* in logica positiva è rappresentato nella fig.III.26b.

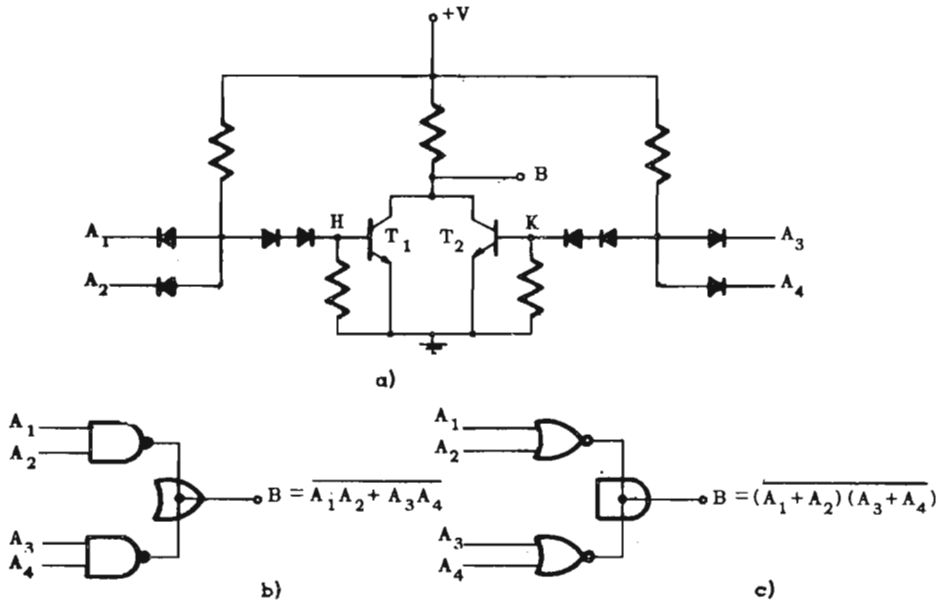


Fig.III.26 - Montaggio WIRED-OR (a) e sue rappresentazioni simboliche in logica positiva (b) e negativa (c).

Lo stesso circuito di fig.III.26a, in logica negativa è un NAND formato con le uscite dei 2 OR $(A_1 + A_2)$ e $(A_3 + A_4)$. Si ha quindi:

$$B = \overline{H \cdot K} = \overline{(A_1 + A_2)(A_3 + A_4)} = \overline{A}_1\overline{A}_2 + \overline{A}_3\overline{A}_4 \quad .$$

Il simbolo logico dell'*OR di collettore* in logica negativa è rappresentato nella fig.III.26c.

III.10 - Confronto fra i vari tipi di circuiti logici.

Un confronto significativo fra i vari schemi circuitali che realizzano le funzioni logiche, soprattutto NAND e NOR, è molto difficile. I fattori che occorre prendere in considerazione sono infatti molti,e riguardano:

– la velocità del circuito, cioè il tempo che intercorre tra l'applicazione di una certa configurazione di valori delle tensioni sugli ingressi e lo stabilirsi della corrispondente tensione sull'uscita;

– l'immunità al rumore, cioè la capacità del circuito di non reagire a variazioni casuali e logicamente non significative delle tensioni d'ingresso;

– il *fan-in* e il *fan-out* (numero di elementi collegabili all'uscita) dell'elemento;

– i requisiti di alimentazione, cioè il numero e il valore delle tensioni necessarie per l'alimentazione;

– la potenza dissipata;

– il grado di affidamento;

– il costo.

Per una completa valutazione dei fattori predetti, si rimanda ai testi di elettronica citati in bibliografia (particolarmente [5] e [12]), avvertendo che la scelta di un particolare tipo di logica è sempre legata alle caratteristiche ed alle prestazioni richieste al circuito.

Molto meno significativa, invece, è l'adozione della logica positiva o negativa, che dipende – al più – dalla maggiore o minore disponibilità di certi componenti. A questo riguardo, non sarà inutile osservare che un circuito realizzato in logica negativa può essere trasformato in uno in logica positiva la cui funzione d'uscita ha la stessa tabella di verità, e viceversa, invertendo diodi e polarità e sostituendo i transistor n-p-n coi p-n-p.

In conclusione, è importante ricordare che esistono cinque funzioni logiche: AND, OR, NOT, NAND, NOR, realizzate le prime due a diodi, le altre a transistor. A seconda dei componenti che si intende usare per costruire un determinato circuito logico, si dovranno risolvere i seguenti problemi:

– esprimere una funzione con le operazioni AND, OR, NOT;

– esprimere una funzione con l'operazione NAND;

– esprimere una funzione con l'operazione NOR.

Altre combinazioni, anche se teoricamente possibile, come le forme NAND-NOR, sono di scarso uso pratico.

III.11 - Elementi di memoria.

Per il funzionamento della maggior parte dei circuiti numerici, è essenziale la possibilità di conservare indefinitamente delle informazioni codificate come insiemi di bit, e di usarle in istanti diversi da quelli in cui vengono prodotte.

Elemento di memoria o *memoria elementare*, è un dispositivo capace di immagazzinare un bit; *memoria* è un insieme di tanti elementi organizzati in modo che sia possibile accedere ad ognuno di essi. *Scrittura* di un gruppo di bit in memoria è l'operazione di immagazzinamento dei bit stessi; *lettura* è l'operazione inversa, cioè il prelievo dei bit immagazzinati; *cancellazione* è la possibilità di eliminare un'informazione immagazzinata per sostituirla con un'altra. A seconda del modo con cui i dati vengono prelevati, le memorie si dividono in *spaziali* e *temporali*. Le prime contengono le informazioni in tante locazioni distinte, in modo che, per leggere il contenuto di una determinata posizione, occorre connetterla fisicamente ad un dispositivo di lettura. Le seconde contengono le informazioni in modo continuo; il dispositivo di lettura è fisso, e i bit vengono letti l'uno dopo l'altro, come su un normale disco. Entrambi questi tipi di memorie sono *statiche*, nel senso che le informazioni immagazzinate rimangono fisse nell'elemento che le contiene.

Esistono anche memorie *dinamiche*, in cui le informazioni si muovono dall'uno all'altro elemento della memoria. Ogni bit va letto mentre esce dall'ultimo elemento, cui è collegato il dispositivo di lettura, e non può essere conservato se non riscrivendolo all'estremità di ingresso.

Le memorie più usate sono le memorie magnetiche, quelle a linee di ritardo e quelle a flip-flop.

Le *memorie magnetiche* sono di tipo statico e nascono dalla possibilità di magnetizzare completamente, per l'esistenza dell'isteresi, una o più particelle magnetiche in una direzione, o in quella opposta. È tecnicamente possibile realizzare memorie formate da nuclei comandati mediante correnti lanciate su fili che li attraversano, o da particelle magnetiche che, troppo piccole per essere magnetizzate separatamente, vengono fissate su un unico supporto (memorie a nastri, tamburi, dischi, carte).

Le memorie a *linee di ritardo* di tipo dinamico, sono quei dispositivi la cui uscita assume il valore dell'ingresso con un ritardo r, dovuto al tempo di propagazione del segnale nel mezzo che costituisce la linea stessa. Le linee di ritardo sono elettriche (es.: uno spezzone di li-

nea a costanti concentrate) o acustiche (quando un segnale elettrico o magnetico deforma un materiale, generando un'onda sonora che si ritrasforma nel segnale originario all'estremità opposta).

Nel testo useremo prevalentemente come memorie elementari, i *flip-flop* che, pur essendo troppo costosi per grosse memorie, sono gli elementi più usati per memorie temporanee e registri. I flip-flop saranno pertanto descritti dettagliatamente nei paragrafi seguenti.

III.12 - Il multivibratore bistabile o flip-flop.

Il multivibratore bistabile o *flip-flop* è l'elemento di memoria che costituisce il componente fondamentale sia dei *registri a scorrimento*, cioè delle apparecchiature che permettano di spostare ordinatamente delle stringhe di bit di una o più posizioni, sia dei *contatori*, cioè degli organi che contano il numero degli impulsi di una certa sequenza.

Il flip-flop – come è noto dall'elettronica generale – è un dispositivo capace di esistere in due stati diversi, manifestantisi come tensioni opposte su due terminali d'uscita, e di rimanere nell'uno o nello altro stato finché un comando esterno non ne provochi la *commutazione* cioè il passaggio allo stato opposto.

Esistono molti tipi di flip-flop che si distinguono dai componenti che li costituiscono, dal numero degli ingressi e dalle caratteristiche dei segnali che ne provocano la commutazione. Quelli di seguito descritti, a solo titolo d'esempio, hanno in comune un circuito (figura III.27a) che ha due sole condizioni di equilibrio stabile: quelle in cui un transistor è interdetto e l'altro è in saturazione.

Le tensioni di collettore dei transistor T_1 e T_2 hanno sempre due valori ben determinati (nel nostro caso -10 V per il transistor interdetto e 0 V per quello che conduce).

Queste tensioni, indicate con y e $\overline{y}$ in quanto hanno valori complementari, possono essere usate per alimentare dei circuiti logici AND, OR, NOT, NAND, NOR, ai quali circuiti forniscono le variabili d'ingresso sia in forma diretta che in forma negata.

Il circuito della fig.III.27a è adatto a funzionare in logica negativa; per convenzione, si assume, come tensione caratteristica dello stato del flip-flop, la tensione y del transistor T_2; il flip-flop è nello stato 0 quando T_2 conduce ($y = 0\,V = 0$), ed è nello stato 1 quando T_2 è bloccato ($y = -10\,V = 1$).

Quando il circuito si trova nello stato 0, o nello stato 1, vi rimane indefinitamente, realizzando così la funzione di memoria. La sua utilità pratica è però legata alla possibilità di portarlo nell'uno o nell'altro stato, e di farlo commutare, con segnali di comando esterni. Questi segnali vengono mandati alla base di uno, o di entrambi, i transistor attraverso dei particolari *circuiti di comando* dai quali dipendono, in ultima analisi, le proprietà dei diversi tipi di flip-flop.

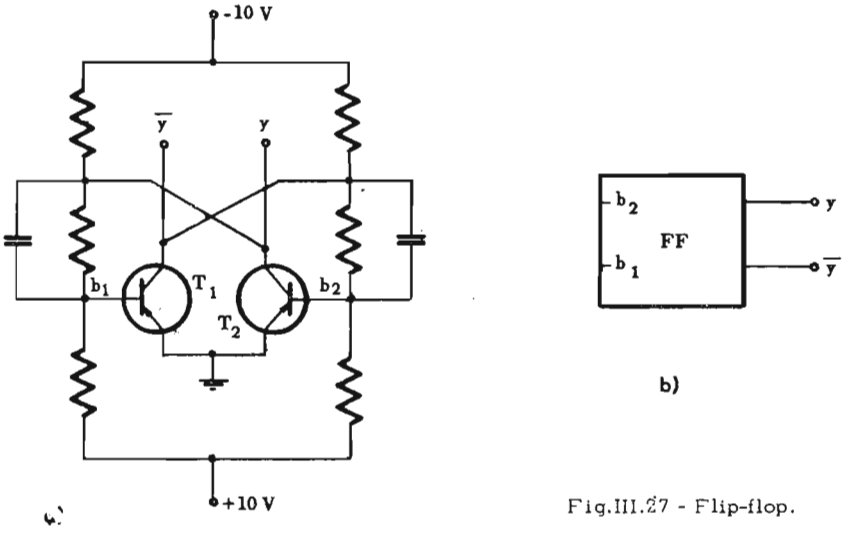


Fig.III.27 - Flip-flop.

Il passaggio di un flip-flop dallo stato 0 allo stato 1 si chiama *setting*; quello dallo stato 1 allo stato 0, *resetting*. Il comando che provoca la commutazione può essere un livello di tensione (comando in continua) o un impulso di tensione (comando in alternata).

Il circuito di fig.III.27a si rappresenta con il simbolo della figura III.27b; b_1 e b_2 sono i punti d'ingresso alle basi dei transistor T_1 e T_2 (ingressi in continua).

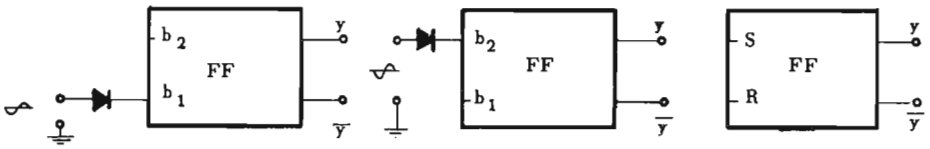
III.12.1 - Comando in continua dei flip-flop.

Supponiamo che il circuito di fig.III.27a si trovi nello stato 1 in cui si ha:

$y = 1$ (tensione al collettore di $T_2 = -10$ V)

$\bar{y} = 0$ (tensione al collettore di $T_1 = 0$ V) .

Mandando alla base di T_1, che sta conducendo ed è in saturazione, un segnale (*set*) capace di assumere valori positivi (fig.III.28a), nel momento in cui la base b_1 diventa positiva, T_1 si blocca e T_2 conduce: il flip-flop è così passato nello stato 0. La presenza del diodo serve a impedire che la base di T_1 segua gli eventuali valori negativi del segnale d'ingresso.



a) b) c)

Fig.III.28 - Comando in continua di un flip-flop.

Per mettere il flip-flop di nuovo nello stato 1, si invia in maniera analoga un segnale positivo sulla base b_2 (fig.III.28b). Il flip-flop ricorda, così su quale dei due ingressi è arrivato l'ultimo segnale positivo. Ovviamente, non debbono essere presenti – contemporaneamente – segnali sui due ingressi.

Il flip-flop comandato dal livello di una tensione si chiama *flip-flop con ingresso a livelli*, e si indica negli schemi col simbolo della figura III.28c; S ed R sono gli ingressi attraverso i quali il flip-flop viene messo negli stati 0 e 1.

III.12.2 - Comando in alternata dei flip-flop.

Un flip-flop può essere comandato, oltre che da livelli di tensione, anche da impulsi inviati a due circuiti differenziatori RC collegati alle basi di T_1 e T_2 mediante diodi (fig.III.29a).

Nella fig.III.29b è mostrato l'andamento delle tensioni nei punti A_1, C_1, R in seguito a un comando di *reset*, cioè un impulso positivo di + 10 V, su A_1. Se il flip-flop si trova nello stato 1 (y = 1), il transistor T_1, la cui base viene bruscamente portata a + 10 V, si blocca, e il flip-flop passa nello stato 0.

Se T_1 è bloccato, l'impulso di *reset* non ha alcun effetto.

Un impulso su A_2, invece, lascia sempre il flip-flop nello stato 1; non è ammesso inviare, contemporaneamente, impulsi sui due ingressi.

Un impulso positivo di 10 V di ampiezza riesce a far commutare il flip-flop in quanto i punti C_1 e C_2 (fig.III.29a) si trovano alla tensio-

ne di 0 V. Se C_1 e C_2 fossero collegati ad una tensione negativa di -10 V, l'impulso stesso non avrebbe alcun effetto. Per mezzo delle tensioni V_c dei punti C_1 e C_2 è così possibile controllare gli impulsi di comando, annullandone l'azione quando $V_c \leq -10$ V. Su questo principio è basato il flip-flop JK, ottenuto dallo SR collegando i terminali C_1 e C_2 ai collettori di T_1 e T_2 invece che a massa (fig.III.30).

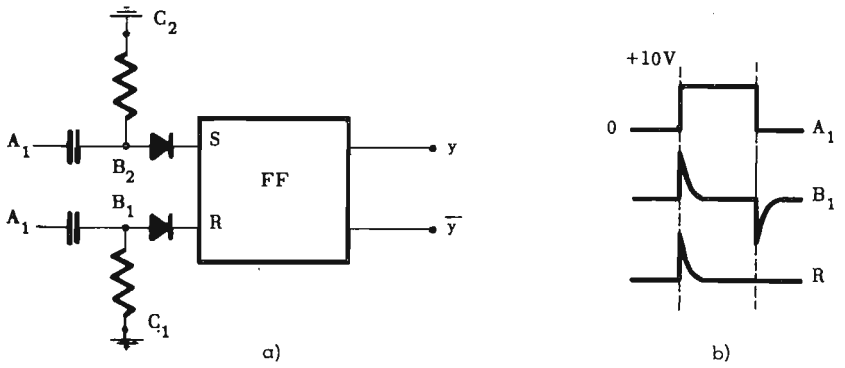


Fig.III.29 - Comando in alternata di un flip-flop.

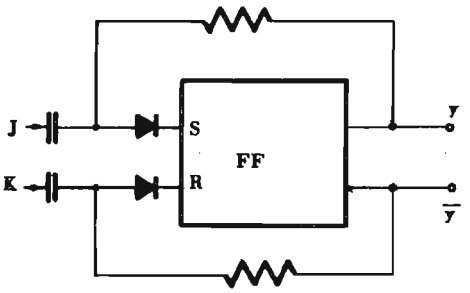


Fig.III.30 - Flip-flop JK.

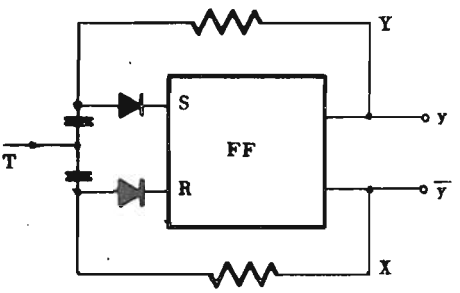


Fig.III.31 - Flip-flop T.

Il flip-flop ottenuto è messo a 1 da un segnale positivo sull'ingresso J ed a 0 da un segnale sull'ingresso K. Se arrivano, contemporaneamente, segnali positivi sui due ingressi, il flip-flop commuta.

Collegando fra loro gli ingressi di un flip-flop JK si ottiene il flip-flop T (fig.III.31). Ogni impulso di comando sull'unico ingresso viene indirizzato ad ambedue i transistor, ma ha effetto soltanto su quello che conduce, e provoca la commutazione del flip-flop. In definitiva, il flip-flop T, cambiando di stato a ogni impulso, ricorda se ha ricevuto un numero pari o dispari di impulsi.

Appare chiaramente, nella fig.III.31, che le caratteristiche del flip-flop T dipendono dai collegamenti tra i punti X e Y e le uscite $\overline{y}$ e y. Collegando i punti X e Y alle uscite di un secondo flip-flop (*flip-flop di controllo*) in modo che le tensioni V_X e V_Y abbiano sempre valori opposti, si ottiene il flip-flop T *a ingressi condizionati*, che si comporta in modo diverso a seconda che i collegamenti vengano fatti nell'uno o nell'altro dei modi illustrati nelle figg.III.32a e III.32b.

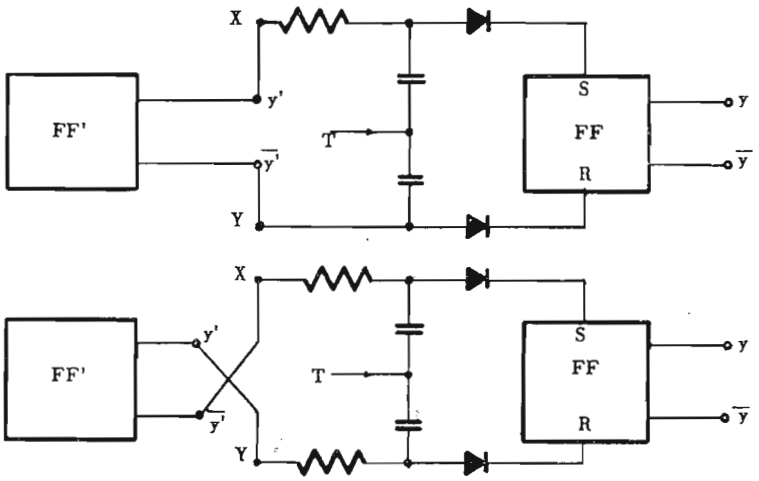


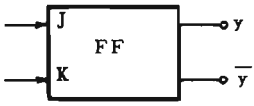
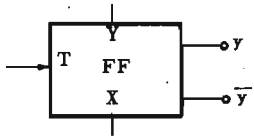
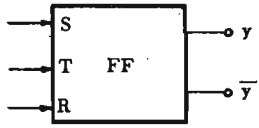
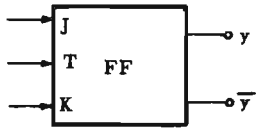
Fig.III.32 - Collegamenti di un flip-flop T con un flip-flop di controllo.

Nel primo caso (fig.III.32a), il flip-flop T a ingressi condizionati assume, ad ogni impulso T, lo stato opposto a quello del flip-flop di controllo. Se i collegamenti sono fatti come indicato nella fig.III.32b, ogni impulso T fa assumere al flip-flop lo stesso stato del flip-flop di controllo.

III.12.3 - Caratteristiche e simboli dei flip-flop.

I circuiti visti finora rappresentano, come già detto, alcune delle molte realizzazioni circuitali dei flip-flop. Non abbiamo considerato i problemi di natura elettronica che si presentano nella realizzazione, nella scelta e nell'impiego dei flip-flop stessi, e che sono problemi legati, principalmente, alla velocità di commutazione, alla forma e durata degli impulsi di comando, alla immunità contro variazioni accidentali delle tensioni sui terminali d'ingresso. Per tali problemi, che esulano dal progetto logico, rimandiamo ai testi in bibliografia, particolarmente a [5].

TABELLA III.1

Flip-flop	Simbolo	Caratteristiche
SR		Un segnale su S mette, o lascia, il ff nello stato 1: un segnale su R mette, o lascia, il ff a 0. Non sono ammessi segnali contemporanei su S ed R.
JK		Si comporta come uno SR (J è l'ingresso di *set*). Cambia di stato se riceve due segnali contemporanei su J e K.
T		Cambio di stato ad ogni impulso sullo ingresso T.
T a ingressi condizionati		y assume, dopo l'impulso T, il valore che Y aveva durante l'impulso.
PQ		Si comporta come uno SR (P è l'ingresso di *set*). Non commuta se riceve 2 segnali contemporanei.
STR		Si comporta come uno SR o come un T, a seconda dell'ingresso su cui arriva il segnale. Può ricevere un solo segnale alla volta. (Esiste anche con ingressi condizionati).
JTK		Si comporta come un JK o come un T. Se c'è un segnale su T non può esserci un segnale su JK, e viceversa. (Esiste anche con ingressi condizionati).

Dal nostro punto di vista, importa sapere che esistono certi tipi di flip-flop, ognuno dei quali commuta con determinati segnali, e si rappresenta con un particolare simbolo.

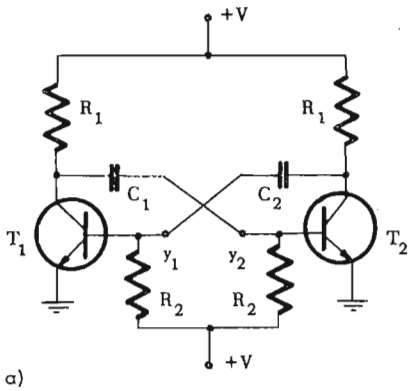
A questo proposito facciamo notare che quanto detto per i flip-flop in logica negativa visti nel paragrafo precedente rimane valido per quelli in logica positiva (ottenibili invertendo le tensioni di alimentazione e usando transistor n-p-n). In questi ultimi, ovviamente, la commutazione è provocata dai fronti di discesa dei segnali impulsivi di comando; ma tali fronti corrispondono ancora al passaggio da 1 a 0 dei segnali stessi.

I più importanti tipi di flip-flop, i loro simboli e le loro caratteristiche sono riportati nella tab.III.1. Si noti che, descrivendo le caratteristiche dei flip-flop si parla genericamente di *segnali sugli ingressi*. Sulla natura di tali segnali, vale quanto detto all'inizio del paragrafo.

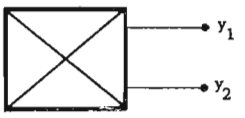
Dal punto di vista logico, la presenza di un segnale significa il valore 1 dell'ingresso su cui il segnale stesso è applicato. Dal punto di vista circuitale, l'eventuale commutazione del flip-flop (che non è istantanea) inizia quando il segnale torna a 0, e si compie in un tempo Δ, brevissimo ma non nullo (l'importanza di questo ritardo apparirà in seguito).

III.13 - Il multivibratore astabile.

In molti e importanti circuiti numerici è essenziale il controllo della velocità di esecuzione delle operazioni e la esatta determinazione delle relazioni temporali fra le operazioni stesse. Per la temporizzazione si usa un segnale *cadenzatore* (clock signal) che è una forma d'onda



a)



b)

Fig.III.33 - Multivibratore astabile.

quadrata a frequenza fissa ottenuta da un *multivibratore astabile*, cioè da un circuito senza stati stabili di equilibrio, in cui due stati semistabili si alternano continuamente e separatamente.

Nella fig.III.33a è mostrato lo schema circuitale di un multivibratore astabile: esso può trovarsi in due stati, ma i condensatori C_1 e C_2 di accoppiamento non permettono che rimanga in nessuno dei due se non per un determinato periodo di tempo, che può andare da frazioni di microsecondi a qualche secondo.

Per analizzare il comportamento del circuito, supponiamo che T_1 conduca e T_2 sia bloccato, cioè $y_1 = (V_{CE})_{sat}$ e $y_2 = +V$. Il condensatore C_1 è connesso a massa attraverso T_1, per cui R_2 lo carica con una costante di tempo R_2C_1: quando V_{B2} raggiunge la tensione $(V_{BE})_{ON}$ ($\cong 0{,}7$ V), T_2 comincia a condurre: C_2 si scarica attraverso il transistor, e alla base di T_1 compare una caduta di tensione negativa, che interdice T_1: T_1 e T_2 si sono così scambiati gli stati. Subito dopo, C_2 viene caricato da R_2 fino a raggiungere $(V_{BE})_{ON}$, con la costante di tempo R_2C_2; per tutto il tempo in cui C_2 è sotto carica, le uscite rimangono fisse; dopo, T_1 e T_2 si alternano di nuovo, perché la caduta di tensione derivante dalla scarica di C_1 attraverso y_1 interdice T_2.

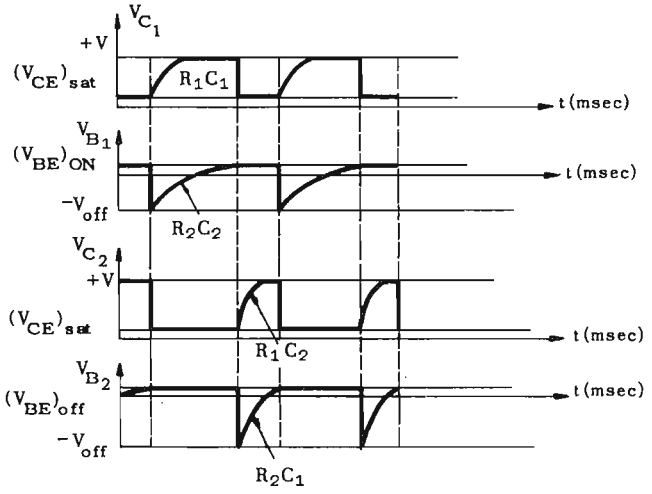


Fig.III.34 - Forme d'onda delle tensioni di base e di collettore dei transistor T_1 e T_2 del multivibratore astabile.

Nella fig.III.34 sono mostrate le forme d'onda delle tensioni di base e di collettore dei due transistor. Le tensioni di base sono le uscite y_1 e y_2 ($= \overline{y_1}$) da cui vengono prelevati i segnali di clock. Se $C_1 = C_2$, la forma d'onda è simmetrica e l'uscita è un segnale rettangolare con valori 0 e 1 di egual durata. Un aumento di C si traduce in un abbassamento della frequenza, e viceversa.

Un multivibratore astabile si disegna, negli schemi, col simbolo della fig.III.33b.

III.14 - Il multivibratore di Schmitt.

Il multivibratore di Schmitt (fig.III.35) è un circuito riformatore d'impulsi, che converte una forma d'onda a variazioni lente in una con fronti di salita e di discesa assai ripidi. Esso serve per ottenere impulsi quadrati di frequenza costante, se sincronizzato con la tensione sinu-

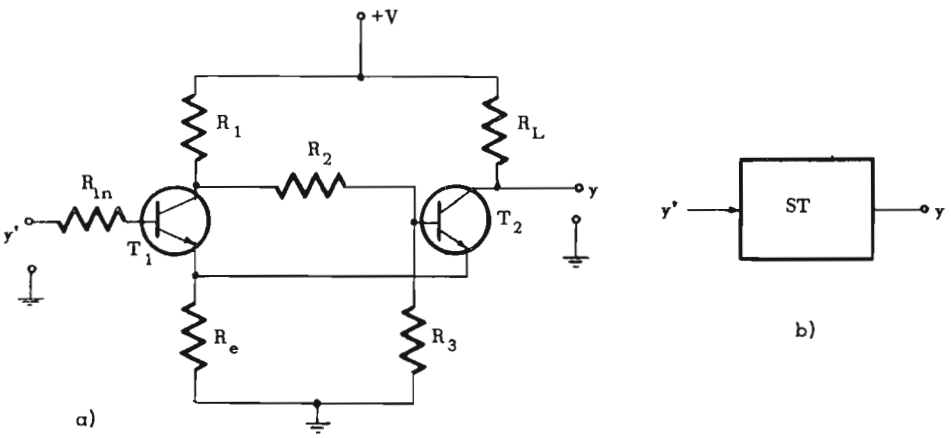


Fig.III.35 - Multivibratore di Schmitt.

soidale di un oscillatore, oppure a riformare impulsi di tensione deformati dal passaggio attraverso circuiti passivi. È realizzato in modo che, quando la tensione di ingresso supera un livello di soglia, il transistor T_1 conduce e interdice T_2; quando la tensione scende sotto un secondo livello di soglia, T_1 si blocca e T_2 conduce.

Analizziamo, ora, il comportamento del circuito, che è formato da due stati amplificatori in continua con la resistenza di emettitore (R_e) in comune. Quando l'ingresso (y') è nullo, T_1 è interdetto e T_2 conduce. In prima approssimazione, la tensione di base del transistor T_2 è:

$$V_{B_2} = \frac{R_3}{R_1 + R_2} V \quad .$$

Trascurando la caduta tra base ed emettitore di T_2, questa tensione è anche quella di emettitore del transistor T_1. Appena la tensione d'ingresso y' supera $E_{ON} = V_{B_2} + (V_{BE})_{ON}$, T_1 comincia a condurre, abbassando la sua tensione di collettore V_{C_1}; T_2 allora si blocca: diminuisce V_{E_2}, quindi V_{E_1}, e T_1 continua a condurre. D'altra parte, V_{C_1}, finché T_1 è in saturazione, mantiene T_2 interdetto.

La tensione alla base di T_2, comunque, deve essere talmente bassa da mantenere T_2 interdetto. Poiché $V_{E_1} = V_{E_2}$, V_{CE_1} deve essere ottenuta da R_3 ed R_2, in modo da risultare minore di $(V_{BE})_{ON}$. Generalmente, $(V_{CE})_{sat}$ è $< (V_{BE})_{ON}$, cosicché un valore d'attenuazione tra 5 e 10 manterrà sicuramente T_2 interdetto. La tensione di soglia che riporta T_2 in conduzione si trova calcolando $V_{E_1} = I_{E1} \cdot R_e$, ed è, in prima approssimazione:

$$V_{E_1} = \frac{R_3}{R_1 + R_3} V \quad .$$

Se la tensione d'ingresso scende sotto V_{E_1}, T_1 conduce di meno: aumentando V_{C_1} per la diminuita corrente attraverso R_1, la tensione V_{B_2} aumenta fino a portare T_2 in conduzione. La corrente attraverso T_2 fa poi aumentare V_{E_2} fino a interdire T_1 e a saturare T_2.

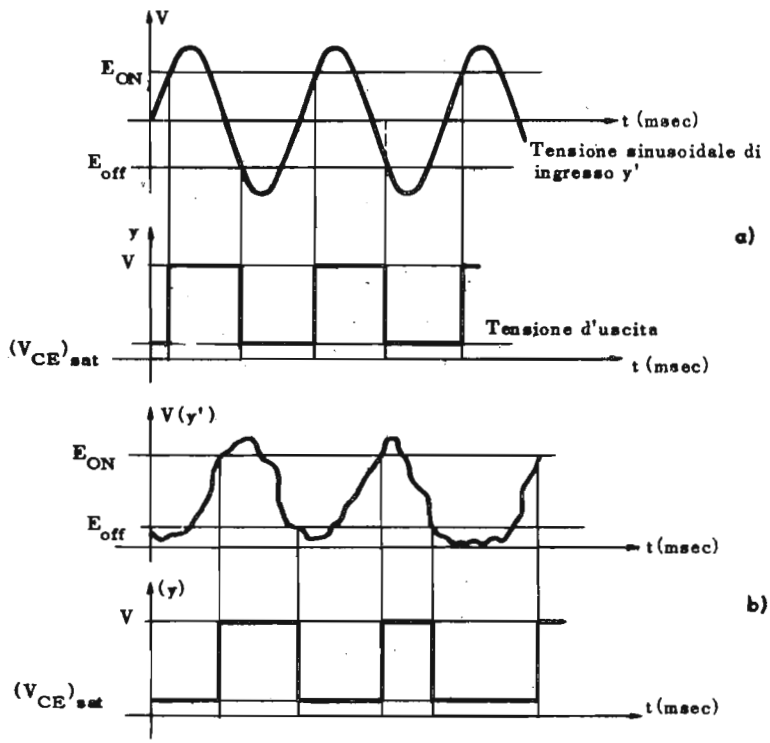


Fig.III.36 - Impieghi del multivibratore di Schmitt: generazione di un segnale di clock da una tensione sinusoidale (a) e rigenerazione di un segnale impulsivo (b).

Negli schemi, il multivibratore di Schmitt è indicato col simbolo della fig.III.35b; nella fig.III.36 sono mostrati i due usi più comuni di questo circuito: la generazione di un clock a ugual frequenza di una tensione sinusoidale e la rigenerazione di un segnale impulsivo.

III.15 - Il multivibratore monostabile.

Il *multivibratore monostabile* (one-shot) e un circuito a due transistor (fig.III.37) con un solo stato stabile: quello in cui un transistor (ad esempio T_2) conduce e l'altro (T_1) è interdetto. Finché non ci sono segnali in ingresso, il circuito rimane stabilmente in questo stato; ap-

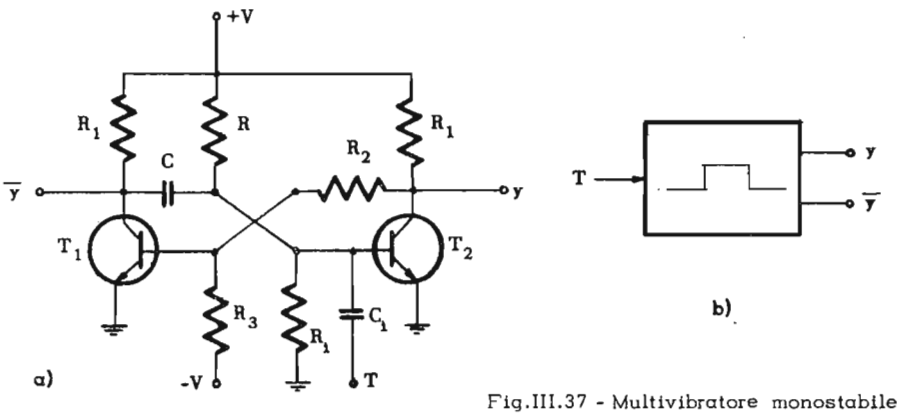


Fig.III.37 - Multivibratore monostabile.

plicando un impulso, passa invece nello stato opposto, in cui T_1 conduce e T_2 è bloccato, ma vi rimane soltanto per un breve periodo di tempo (da una frazione di microsecondo a qualche secondo), dopo il quale ritorna spontaneamente nello stato originario.

Il diodo collegato al condensatore C_1 fa arrivare alla base di T_2 (che conduce nello stato stabile), soltanto il fronte di discesa dell'impulso T, che interdice T_2. Appena T_2 cessa di condurre, l'accoppiamento tra il collettore di T_2 e la base di T_1 manda quest'ultimo in conduzione. T_1, conducendo, scarica C, e provoca un gradiente negativo di tensione che va ad abbassare la tensione di base di T_2, non potendo variare istantaneamente la tensione ai capi del condensatore. Successivamente, C si ricarica sotto la tensione V, con una costante di tempo RC; quando raggiunge $(V_{BE})_{ON}$, T_2 diventa di nuovo conduttore. T_1 allora si interdice, perché il divisore di tensione R_3-R_2 provoca una tensione negativa alla base di T_1 quando l'estremità y di R_2 è vicina a massa (T_2 in saturazione). L'andamento nel tempo delle tensioni nei vari punti del multivibratore monostabile è mostrato nella fig.III.38. La durata dello stato instabile dipende da RC: se +V e -V hanno grandezze eguali, è approssimativamente $t = 0{,}7$ RC secondi.

Il multivibratore monostabile viene impiegato per fornire ritardi di lunghezza determinata e per generare impulsi di durata maggiore di

quella a disposizione o, comunque, di lunghezza fissa (serve, così, per riformare gli impulsi).

I multivibratori monostabili, astabili e di Schmitt sono stati disegnati con circuiti n-p-n. Sono ovvie le modifiche da apportare ai circuiti, agli ingressi e alle uscite nel caso si voglia adottare, come per il flip-flop della fig.III.27, i circuiti p-n-p.

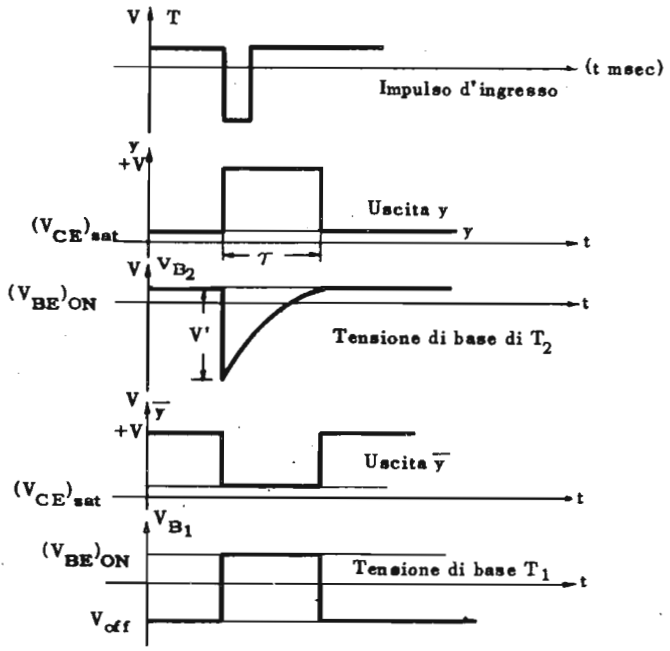


Fig.III.38 - Tensioni nei vari punti del multivibratore monostabile.

III.16 - Circuiti integrati.

I Circuiti Integrati Digitali sono circuiti logici completi riuniti in un unico contenitore, e realizzati, generalmente, secondo le logiche DTL, DCTL o TTL.

Per le loro particolari caratteristiche, i circuiti logici integrati sono oggi avviati a sostituire – in ogni applicazione – i circuiti a componenti discreti; la loro tecnologia è in tale rapida evoluzione che è impossibile darne, in questa sede, un panorama sufficientemente completo. A solo titolo d'esempio, riportiamo di seguito dei tipici circuiti DTL e TTL tratti dal catalogo Philips.

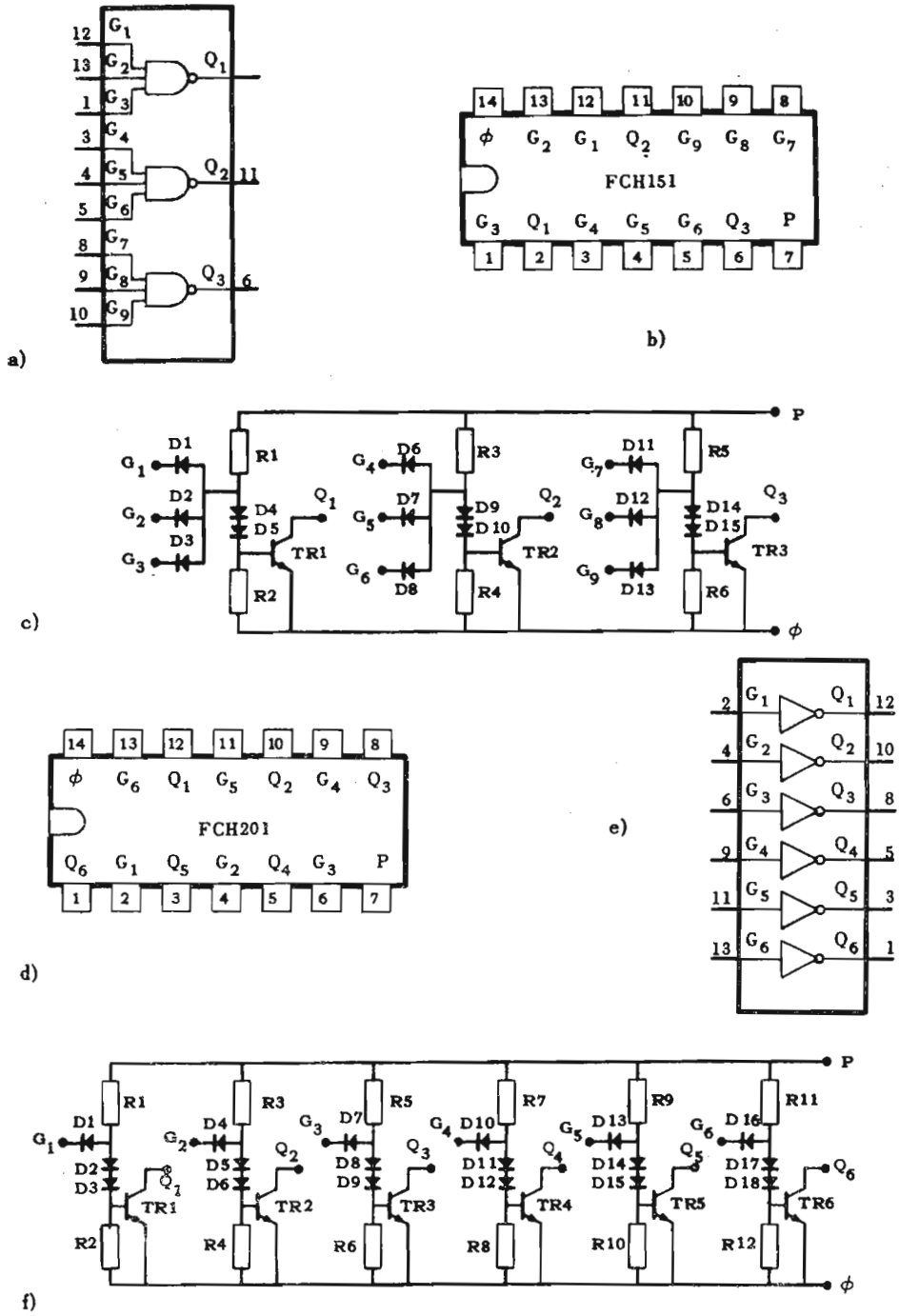


Fig.III.39 - Circuiti integrati: a), b), c): schema logico, rappresentazione esterna, e circuito del triplo NAND a 3 ingressi FCH151; d), e), f): schema logico, rappresentazione esterna e circuito del NOT sestuplo FCH201.

TABELLA III.2 - Esempio di circuiti integrati: FJH151 ed FJH161 dual AND-OR-NOR-gates. «Dati Tecnici Philips» pagg.227÷229.

FJ family
standard temperature range

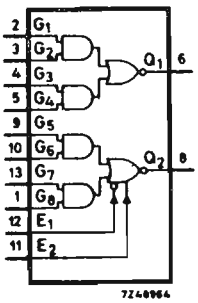
FJH151; FJH161
dual AND-OR-NOR gates

The FJ family of TTL silicon monolithic integrated circuits is designed for medium speed digital equipment in computing, telecommunication, instrumentation and control.

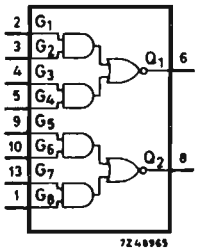
Features of the FJ family: * high-fan-out * low power consumption (typ. 10 mW for standard gates) * high logic swing * low output impedance * short circuit protection * high capacitance drive capability * high noise margin (typ. 1.0 V for standard gates) * comprehensive range of circuits, including NAND gates, AND-OR-NOT gates, gate expanders, flip-flops and complex-function devices * it corresponds to the 74Nseries TTL.

DUAL AND-OR-NOT GATES

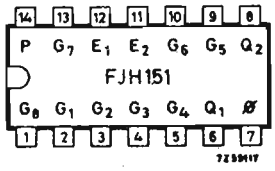
Dual expandable	FJ type	Corresponding SN type
2+2 input AND-OR-NOT gate	FJH151	SN7450N
Dual 2+2 input AND-OR-NOT gate	FJH161	SN7451N

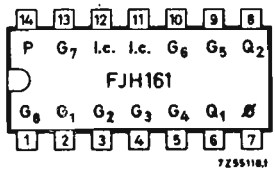


FJH151



FJH161





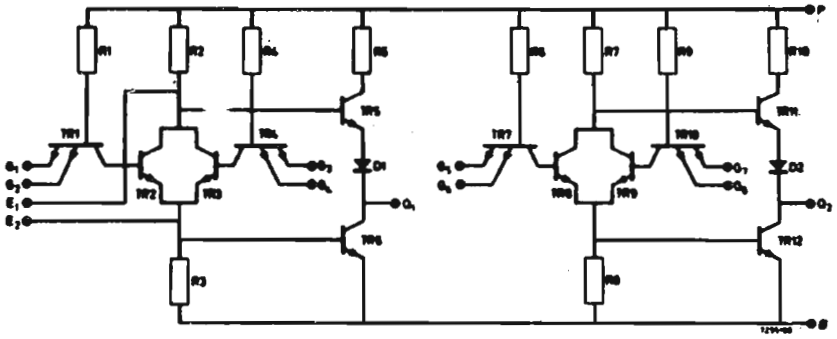
QUICK REFERENCE DATA			
Supply voltage	V_P	5.0 ± 5%	V
Operating ambient temperature range	T_{amb}	0 to +70	°C
Average propagation delay time N = fan-out = 10; T_{amb} = 25 °C	t_{pd}	typ. 13	ns
Available d.c. fan-out (full temperature range)	N_a	≥ 10	
D.C. noise margin (full temperature range)	M_L	> 0.4 typ. 1.0	V V
Average power consumption (per gate) T_{amb} = 25 °C	P_{av}	typ. 14.25	mW

The FJH151 is a dual 2+2 input AND-OR-NOT gate, one of the two gates having additional inputs for up to four FJY101 expander circuits. It can be arranged to perform the exclusive OR function.
The FJH161 is similar to the FJH151, except for being non-expandable.

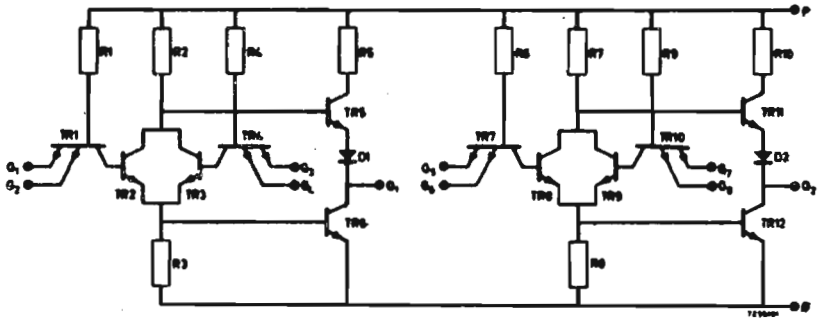
PACKAGE OUTLINE 14 lead dual in-line (See General Section)

CIRCUIT DIAGRAMS

FJH151



FJH161



RATINGS Limiting values in accordance with the Absolute Maximum System (IEC134)

Supply voltage	V_P	max.	7.0	V
Output voltage	V_Q	max.	5.5	V
G input voltage	V_G	max.	5.5	V [1]
Storage temperature	T_{stg}	-55 to	+150	°C
Operating ambient temperature	T_{amb}	0 to	+70	°C

[1] In addition, peak voltage difference between any two inputs = max. 5.5 V.

CHARACTERISTICS

		T_{amb} (°C)				Conditions and references	
		0	25	70		V_P (V)	
STATIC DATA							
Voltages							
Input threshold LOW	V_{GLmax}	0.8	0.8	0.8	V		
Input threshold HIGH	V_{GHmin}	2.0	2.0	2.0	V		
Output LOW	V_{QLmax}	0.4	0.4	0.4	V	4.75	$I_Q=I_{QLmax}$; $V_G=V_{GHmin}$
Output HIGH	V_{QHmin}	2.4	2.4	2.4	V	4.75	$-I_Q=-I_{QHmax}$; $V_G=V_{GLmax}$
Currents							
Input LOW	$-I_{GLmax}$	1.6	1.6	1.6	mA	5.25	$V_G=V_{QLmax}$; $I_Q=0$
Input HIGH	I_{GHmax}	40	40	40	μA	5.25	$V_G=V_{QHmin}$; $I_Q=0$ other inputs 0V
Output LOW	I_{QLmax}	16	16	16	mA		
Output HIGH	$-I_{QHmax}$	0.4	0.4	0.4	mA		
Output short-circuited	$-I_{Qsc\,min}$	18	18	18	mA	5.25	$V_Q=0$; $V_G=0$
	$-I_{Qsc\,max}$	55	55	55	mA		
SUPPLY DATA (each gate)							
Supply current							
Output LOW	I_{PL} typ.	-	3.7	-	mA	5.0	$V_G=5.0$ V; $I_Q=0$
Output HIGH	I_{PH} typ.	-	2.0	-	mA	5.0	$V_G=0$; $I_Q=0$
DYNAMIC DATA							
Rise propagation delay time	t_{pdr} typ.	-	18	-	ns	5.0	N = 10, one pair of AND inputs at V_G = 0.4 V
	t_{pdr} <	-	29	-	ns		
Fall propagation delay time	t_{pdf} typ.	-	8	-	ns	5.0	
	t_{pdf} <	-	15	-	ns		

Ulteriori notizie, particolarmente per quanto riguarda le tecnologie costruttive dei circuiti integrati, possono essere trovate sui testi citati in bibliografia.

III.16.1 - Circuiti logici elementari.

I micrologici, dal punto di vista funzionale, non si differenziano in nulla dalle porte AND, OR, NOT, NAND, NOR già viste, e vengono rappresentati con gli stessi simboli.

La fig.III.39 illustra, a titolo d'esempio, un triplo NAND a 3 ingressi, e un sestuplo NOT, della famiglia FC (logica DTL). Il catalogo fornisce anche tutte le caratteristiche operative.

Nella tab.III.2 vengono riportate, per una migliore comprensione dell'argomento, 3 pagine di un catalogo illustranti le caratteristiche di una doppia porta AND-OR-NOT.

III.16.2 - Flip-Flop.

Analogamente a quanto avviene per le porte AND, OR, NAND, NOR, anche i flip-flop vengono realizzati con la tecnica dei circuiti integrati.

Mentre, però, l'adozione delle porte realizzate con circuiti integrati non porta cambiamenti sostanziali nelle tecniche di progetto logico dei relativi circuiti (tecniche che saranno descritte nel prossimo capitolo), l'uso dei flip-flop a circuiti integrati non può prescindere dalla conoscenza delle diverse proprietà di questi elementi nei confronti di quelli definiti nella tab.III.1.

Di seguito, descriviamo rapidamente le proprietà logiche dei flip-flop a circuiti integrati, rimandando per i vincoli imposti ai segnali di comando e, in genere, per tutte le caratteristiche di tipo elettronico, ai manuali forniti dai costruttori.

I flip-flop realizzati a circuiti integrati più comunemente usati sono:

- il flip-flop D;
- il flip-flop JK;
- il flip-flop JK Master-Slave.

Il flip-flop D (fig.III.40) ha 4 ingressi: S, R, T, D. Gli ingressi S ed R servono a mettere a 1 e a 0 il flip-flop mediante comandi in continua; il flip-flop stesso assume poi lo stato corrispondente al valore logico della tensione presente in D al momento in cui il fronte di salita[1] dell'impulso T raggiunge un determinato valore *di soglia*. Non ha effetto sullo stato del flip-flop né il fronte di discesa né la durata dell'impulso T.

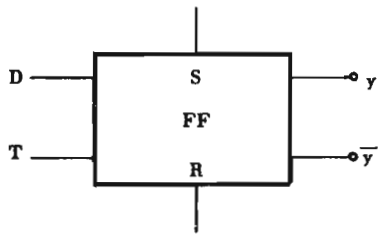


Fig.III.40 - Circuiti integrati: il flip-flop D.

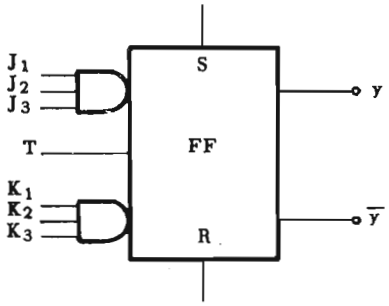


Fig.III.41 - Circuiti integrati: il flip-flop JK.

Il flip-flop JK (fig.III.41) ha 3 ingressi J e 3 ingressi K, riuniti in 2 AND, nonché 2 ingressi (S ed R) a livelli, per mettere a 1 e a 0 il flip-flop.

Dal punto di vista logico, il flip-flop si comporta come un JK in cui l'ingresso J(K) è l'uscita dell'AND $J_1 - J_2 - J_3 (K_1 - K_2 - K_3)$. La commutazione avviene, però, soltanto quando è presente un impulso T, e precisamente sul fronte di salita[1] di tale impulso. Non ha effetto sullo stato del flip-flop né il fronte negativo né la durata dell'impulso T. Lo stato dell'elemento dipende, così, soltanto dai valori di J e K all'istante del fronte di salita di T. Questo flip-flop può funzionare con 1 o 2 ingressi J o K, mettendo a 1 gli ingressi non utilizzati. Mettendo a 1 tutti gli ingressi J e K funziona – infine – come un flip-flop T.

Il flip-flop Master-Slave JK (fig.III.42a) si presenta, all'esterno, come un normale JK, ma è in realtà costituito da 2 flip-flop JK in serie chiamati, appunto il *Master* e lo *Slave*.

Durante il fronte di salita dell'impulso T, i valori di J e K determinano lo stato del *Master*; durante il fronte di discesa di T, lo *Slave* assume lo stato del *Master*.

---

(1) - o di discesa, a seconda delle caratteristiche costruttive del flip-flop usato.

La sequenza delle operazioni (che dipende solo dai livelli dei segnali, nel senso che non è influenzata dai tempi di salita e discesa di J, K, T) è illustrata nella fig.III.42b, ed è la seguente:

1) viene isolato il *Master* dallo *Slave*;

2) il *Master* assume lo stato determinato dagli ingressi J e K;

3) viene isolato il *Master* dagli ingressi J e K;

4) lo *Slave* assume lo stato del *Master*.

(Il flip-flop Master-Slave viene messo a 0 e a 1 dagli ingressi S ed R e funziona con 1 o 2 ingressi J, K, o come un T, nel modo descritto per il flip-flop JK normale).

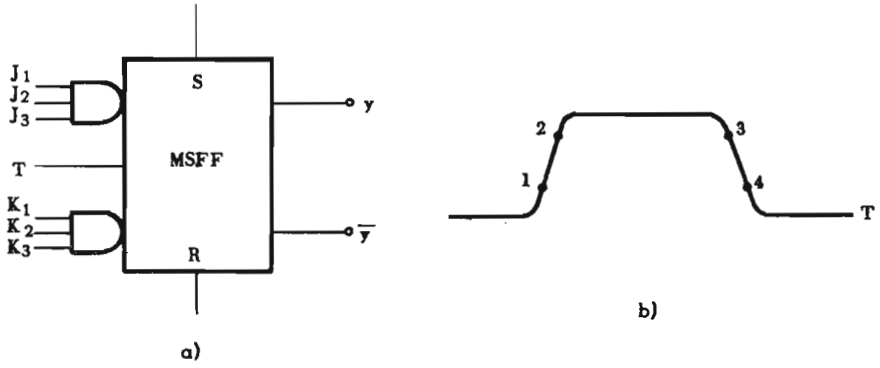


Fig.III.42 - Circuiti integrati: il flip-flop JK Master-Slave (a) e suo funzionamento (b).

A titolo d'esempio, nella tab.III.3 è riportato un tipo di flip-flop JK Master-Slave.

Sono realizzati con circuiti integrati anche i multivibratori astabili, monostabili e di Schmitt; esistono anche circuiti integrati realizzanti funzioni molto complesse (addizionatori, moltiplicatori registri,...) e impieganti tecniche particolari (MOS, LSI, ...): a questi ultimi sarà dedicato un rapido cenno nel capitolo riguardante le tecniche digitali.

III.16.3 - Vantaggi e svantaggi dei circuiti integrati.

I vantaggi che derivano dall'uso dei circuiti integrati nelle apparecchiature digitali sono molti, e tutti piuttosto ovvi; basterà citare:

– le piccole dimensioni (un circuito integrato può contenere, nello stesso spazio occupato da un transistor, 10 transistor convenzionali, 30 resistori, 6 diodi);

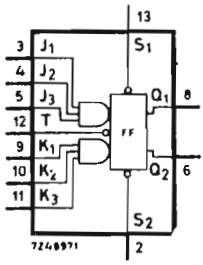
TABELLA III.3 - Flip-flop Master-Slave
(dal catalogo: «Dati Tecnici Philips» pagg.275 ÷ 276).

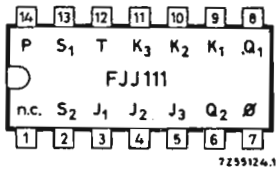
FJ family	**FJJ111**
standard temperature range	JK flip-flop

The FJ family of TTL silicon monolithic integrated circuits is designed for medium speed digital equipment in computing, telecommunication, instrumentation and control.

Features of the FJ family: * high-fan-out * low power consumption (typ. 10 mW for standard gates) * high logic swing * low output impedance * short circuit protection * high capacitance drive capability * high noise margin (typ. 1.0 V for standard gates) * comprehensive range of circuits, including NAND gates, AND-OR-NOT gates, gate expanders, flip-flops and complex-function devices * it corresponds to the 74Nseries TTL.

SINGLE JK MASTER-SLAVE FLIP-FLOP (AND INPUTS)



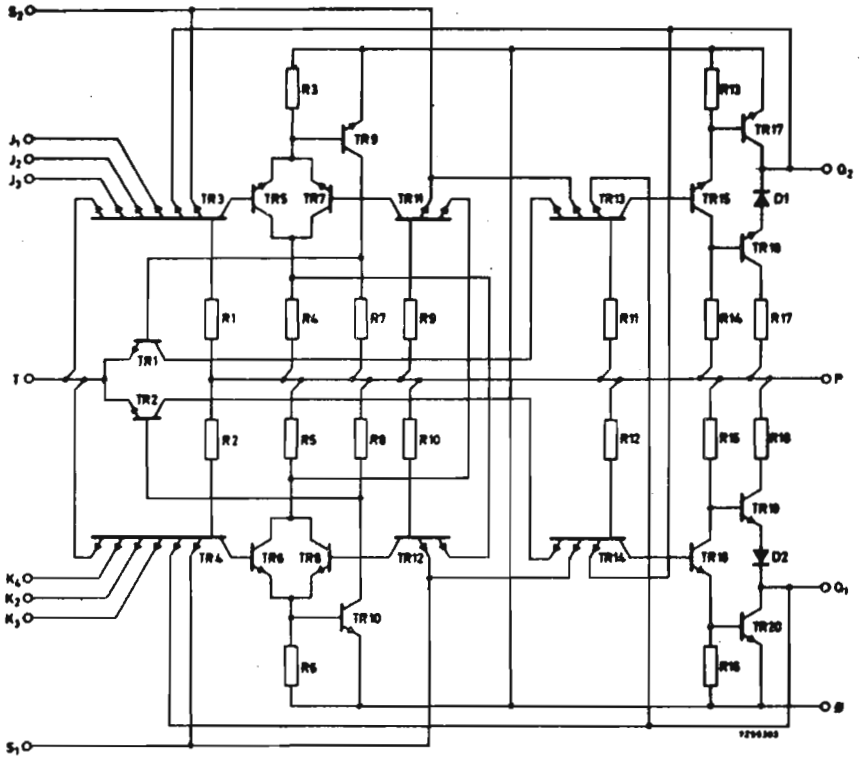


QUICK REFERENCE DATA				
Supply voltage	V_P	5.0 ± 5%		V
Operating ambient temperature range	T_{amb}	0 to +70		°C
Available d.c. fan-out (full temperature range)	N_a	≥	10	
Max. operating frequency; T_{amb} = 25 °C	f	>	10	MHz
Average power consumption; T_{amb} = 25 °C	P_{av}	typ.	40	mW

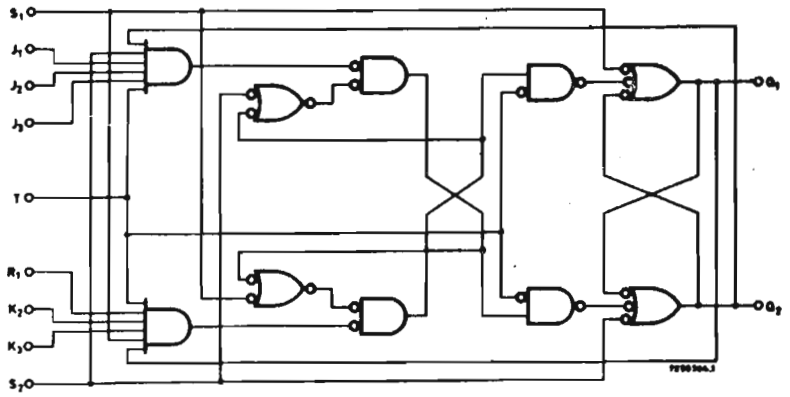
The FJJ111 is a master-slave flip-flop having three J and three K input (AND function). The circuit operates at a frequency up to 15 MHz (typ.). The information at the J and K input enters the master when T is HIGH. Afterwards, when T is LOW, the information is transferred from the master to the slave and appears at the outputs. The FJJ111 corresponds to the SN7472N.

PACKAGE OUTLINE: 14 lead dual in-line (See General Section)

CIRCUIT DIAGRAM



LOGIC DIAGRAM



- la bassa potenza dissipata;
- l'affidabilità (le connessioni tra i componenti sono accuratamente controllabili e assolutamente sicure);
- la facilità di assemblaggio;
- la compattezza dell'apparecchiatura (si realizzano riduzioni di volume dell'ordine di $1/5 \div 1/10$).

L'unico vero svantaggio è che gli utenti debbono accettare i circuiti con tutte le caratteristiche determinate dalle case costruttrici.

*

BIBLIOGRAFIA

1. HIGONNET R.A. & GREA R.H.: *Logical Design of Electrical Circuits* - Mc Graw Hill, 1958.
2. HUMPHREY W.S.: *Switching Circuits with Computer applications* - Mc Graw Hill, 1958.
3. MARCUS M.P.: *Switching Circuits for Engineers* - Prentice-Hall, 1967.
4. PARTEET T.C.: *Digital Computer Fundamentals* - Mc Graw Hill, 1960.
5. MILMAN J.-TAUB H.: *Pulse, Digital and Switching Waveforms* - Mc Graw Hill, 1965.
6. PRESSMAN A.J.: *Design of Transistorized Circuits for Digital Computers* - John Rider, 1959.
7. LEDLEY R.S.: *Digital Computer and Control Engineering* - Mc Graw Hill, 1960.
8. RICHARDS R.R.: *Arithmetic Operations in Digital Computer* - D. Van Nostrand, 1955.
9. KHAMBATA A.J.: *Integrated Semiconductor Circuits* - John Wiley & Sons, 1963.
10. NASHELSKY L.: *Digital Computer Teory* - John Wiley & Sons, 1966.
11. HAWKINS J.K.: *Circuit Design of Digital Computers* - John Wiley & Sons, 1963.
12. EADIE D.: *Introduction to the Basic Computer* - Prentice-Hall, 1968.
13. *Circuiti integrati digitali bipolari e MOS* - Philips, 1971.
14. *Elementi logici in apparecchiature digitali* - Informazioni tecniche Philips, n.22.
15. SACCHI P.F.: *Circuiti Integrati Digitali serie FC* - Biblioteca Tecnica Philips, 1971.

*

CAPITOLO IV

CIRCUITI COMBINATORI

IV.1 - Generalità.

Si chiamano combinatori quei circuiti la cui uscita è, in ogni istante, funzione degli ingressi. Questa definizione non significa che una variazione degli ingressi si ripercuote istantaneamente sull'uscita, ma soltanto che ogni configurazione degli ingressi, comunque raggiunta, dà luogo a un determinato valore dell'uscita; eventuali transitori possono ritardare ma non mutare questo valore, e vanno pertanto trascurati, come logicamente non essenziali.

Per i circuiti combinatori, come per qualsiasi tipo di circuito, si presentano due problemi opposti: quello di descrivere il funzionamento di un circuito di cui è nota la configurazione (analisi), e quello di progettare il circuito che realizza una certa funzione logica, comunque descritta (sintesi). L'analisi è evidentemente più semplice della sintesi, e pertanto inizieremo da essa il nostro studio.

IV.2 - Itinerari e livelli.

Nel trattare i problemi di analisi e di sintesi ci serviremo frequentemente dei termini *itinerario* e *livello*, termini che definiamo ora in maniera del tutto generale.

Ogni circuito di commutazione è formato da un certo numero (n_i) di terminali d'ingresso A_i, da un numero n di elementi, di cui non specificheremo il tipo di (NAND, OR, ...) collegati tra loro in un modo qualsiasi, e da n_u terminali di uscita B_j.

Si chiama *itinerario* tra due elementi X e Y ogni percorso tra X e Y. Itinerario tra un ingresso A_i e un'uscita B_j è ogni percorso che unisce A_i a B_j.

Livello di un elemento X rispetto all'uscita B_j e a un determinato itinerario I^* è il numero di elementi, compreso X, disposti lungo lo itinerario I^*. Livello di una variabile sull'ingresso A_i rispetto all'elemento X (di cui A_i è un ingresso), all'uscita B_j e all'itinerario I^* è il numero di elementi compresi fra A_i e B_j, lungo I^*. Livello di un circuito è il più grande dei numeri che esprimono i livelli della variabili sui suoi ingressi.

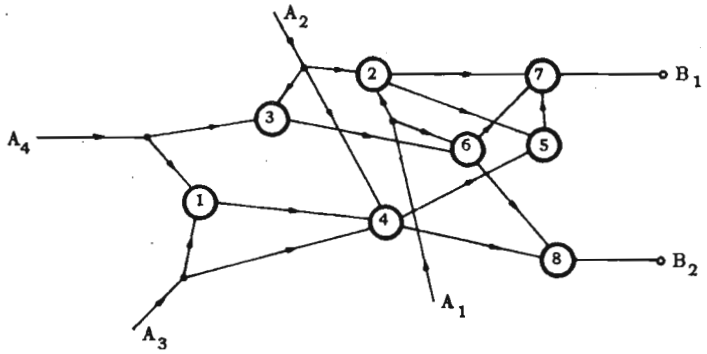
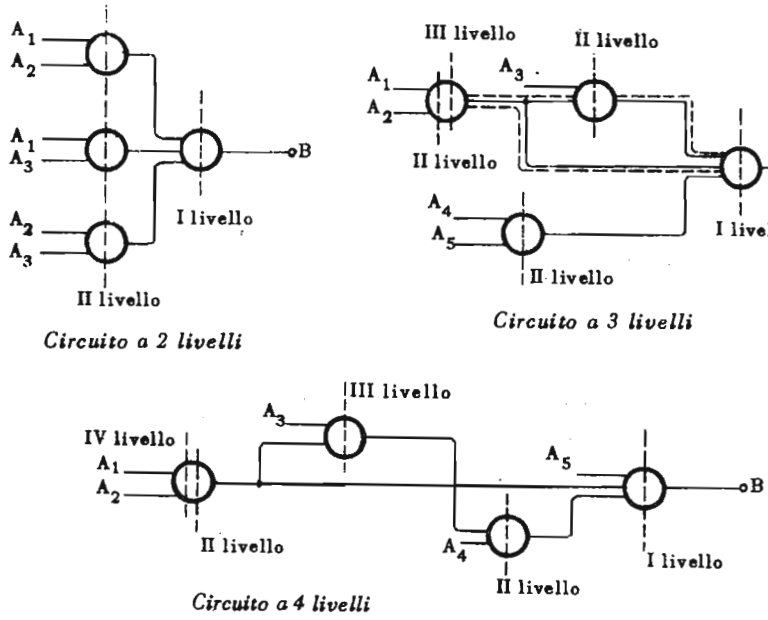


Fig.IV.1 - Circuito di commutazione a 4 ingressi, 2 uscite, 8 elementi.

Nella fig.IV.1 è mostrato un *circuito* a 4 ingressi, 8 elementi e 2 uscite. Gli itinerari tra gli elementi 2 e 7 sono due: 2-7 e 2-5-7, e coincidono con gli itinerari fra l'elemento 2 e l'uscita B_1. Gli itinerari tra A_1 e B_1 sono tre: 2-7, 2-5-7, 6-7. I livelli dell'elemento 2 rispetto alla uscita B e agli itinerari 2-7 e 2-5-7 sono, rispettivamente, 2 e 3. Il livello della variabile A_1 rispetto all'elemento 6 e all'uscita B è 2. Il livello del circuito è 4 (livello dell'elemento 1 rispetto a B_1 e all'itinerario 1-4-5-7).

Per convenzione, i livelli si contano, lungo ogni itinerario, a partire dall'elemento su cui è posto il terminale d'uscita, che è quindi sul primo livello. Ovviamente, ogni elemento può trovarsi su livelli diversi, a seconda dell'itinerario su cui viene considerato. Sono importanti, ai fini pratici, il livello del circuito e la divisione dei livelli in pari e dispari.

Nella fig.IV.2 sono illustrate alcune semplici applicazioni delle definizioni date.



Circuito a 2 livelli

Circuito a 3 livelli

Circuito a 4 livelli

Fig.IV.2 - Livelli e itinerari in 3 circuiti di commutazione.

IV.3 - Analisi dei circuiti AND-OR-NOT.

L'analisi di un circuito mira ad ottenere una qualsiasi rappresentazione della sua funzione d'uscita y. In pratica, si usa dare della y la tabella di verità o una espressione analitica in forma di somme di prodotti, da cui si ricava facilmente la tabella stessa. Poiché i circuiti sono disegnati in forma simbolica, l'analisi prescinde da ogni considerazione di logica positiva o negativa, che si suppone del resto intervenuta nel passaggio dallo schema elettrico a quello logico.

Dalla definizione di analisi e dalle proprietà degli elementi AND-OR-NOT, si intuisce che l'analisi dei circuiti, costruiti con questi elementi, è particolarmente semplice: basta, infatti, partire dagli elementi sui quali entrano le variabili e procedere verso il terminale d'uscita, lungo tutti i possibili itinerari, usando la funzione di uscita di ogni elemento come ingresso dell'elemento successivo. Le y ottenute in forma di prodotto di somme vengono, in genere, trasformate in somma di prodotti.

Gli esempi che seguono chiariranno il procedimento.

Esempio 1: *Analisi del circuito della fig.IV.3.*

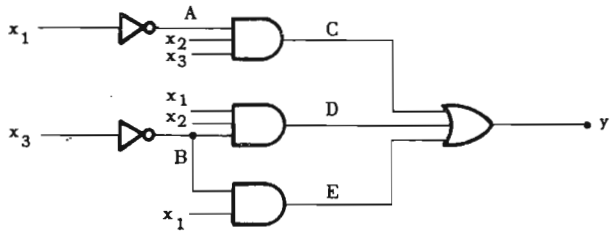


Fig.IV.3 - Circuito AND-OR-NOT.

Nel punto A, all'uscita del primo invertitore, si ha $\overline{x}_1$; all'uscita del primo AND (punto C) si ha $\overline{x}_1 x_2 x_3$. Nel punto B si ha $\overline{x}_3$; all'uscita del secondo AND (punto D) si ha: $x_1 x_2 \overline{x}_3$. Nel punto E si ha infine $x_1 \overline{x}_3$. In definitiva, all'uscita dell'OR si ha la funzione:

$$y = \overline{x}_1 x_2 x_3 + x_1 x_2 \overline{x}_3 + x_1 \overline{x}_3 \quad .$$

La tabella di verità della y (fig.IV.4) si costruisce scrivendo tutte le 8 configurazioni delle variabili $x_1 x_2 x_3$, e ponendo $y = 1$ per $x_1 x_2 x_3 = 011$ e 110, nonché per le due configurazioni in cui $x_1 = 1$ e $x_3 = 0$ (una di queste – la $x_1 x_2 x_3 = 110$ – è però già stata segnata).

Data la semplicità di costruzione della tabella di verità, una volta ottenuta l'equazione della y, tralasceremo d'ora in poi questo ulteriore passo, del resto non sempre richiesto.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Fig.IV.4 - Tabella di verità del circuito di fig.IV.3.

Esempio 2: *Analisi del circuito della fig.IV.5.*

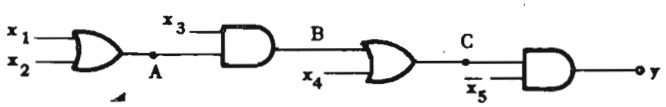


Fig.IV.5 - Circuito AND-OR.

All'uscita del primo OR (punto A) si ha: $y_A = x_1 + x_2$.
All'uscita del primo AND: $y_B = x_3 y_A = x_3 (x_1 + x_2)$.
All'uscita del secondo OR: $y_C = y_B + x_4 = x_3 (x_1 + x_2) + x_4$.
All'uscita del secondo AND: $y = \overline{x}_5 \cdot y_C = \overline{x}_5 \left[x_3 (x_1 + x_2) + x_4 \right]$.

Eliminando le parentesi, si può scrivere la y come somma di prodotti:

$$y = \overline{x}_5 [x_1 x_3 + x_2 x_3 + x_4] = x_1 x_3 \overline{x}_5 + x_4 \overline{x}_5 + x_2 x_3 \overline{x}_5 \quad .$$

Esempio 3: *Analisi del circuito della fig.IV.6.*

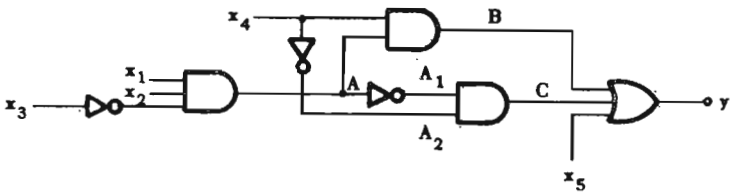


Fig.IV.6 - Circuito AND-OR-NOT.

All'uscita del primo AND si ha: $y_A = x_1 x_2 \overline{x}_3$.

All'uscita del secondo AND: $y_B = x_4 \cdot y_A = x_1 x_2 \overline{x}_3 x_4$.

All'ingresso del terzo AND: $y_{A_1} = \overline{y}_A = \overline{x_1 x_2 \overline{x}_3} = \overline{x}_1 + \overline{x}_2 + x_3$.

$$y_{A_2} = \overline{x}_4.$$

All'uscita del terzo AND: $y_C = y_{A_1} \cdot y_{A_2} = \overline{x}_4 (\overline{x}_1 + \overline{x}_2 + x_3)$.

All'uscita dell'OR finale: $y = y_B + y_C + x_5 = x_1 x_2 \overline{x}_3 x_4 + \overline{x}_4 (\overline{x}_1 + \overline{x}_2 + x_3) + x_5$.

Dagli schemi logici è possibile ricavare il numero di diodi e transistor necessari per realizzare i relativi circuiti: occorrono tanti transistor quanti sono gli invertitori e tanti diodi quanti sono gli ingressi agli AND e agli OR. Così gli schemi delle figg.IV.3 ÷ IV.5 sono realizzabili, rispettivamente, con:

- 2 transistor e 11 diodi;
- 8 diodi;
- 3 transistor e 10 diodi.

Questa osservazione riuscirà utile nel problema di sintesi.

IV.4 - Analisi dei circuiti NAND.

Sfruttando la definizione dell'operatore NAND si può ottenere una espressione analitica della funzione d'uscita di un circuito realizzato con elementi NAND. Ad esempio, quello della fig.IV.7 realizza la funzione:

(IV.1) $$y = [(x_1/x_2)/x_3/x_4]/(x_3/x_5)/x_6 .$$

Le espressioni ottenute in questo modo sono però complicate, soprattutto per la non associatività dell'operatore / che costringe ad usare, anche per le funzioni derivanti dai circuiti più semplici, un gran numero di parentesi. Sono, inoltre, scarsamente utili, in quanto per ricavare la tabella di verità della y occorre passare prima alla forma AND-OR-NOT. La (IV.1) andrebbe, infatti, così trasformata:

(IV.2) $$y = [(x_1/x_2)/x_3/x_4]/(x_3/x_5)/x_6 =$$

$$= \overline{\overline{x_1 x_2} \cdot x_3 x_4} \cdot \overline{\overline{x_3 x_5} \cdot x_6} =$$

$$= \overline{(\overline{x}_1 + \overline{x}_2)\, x_3 x_4}\ (\overline{x}_3 + \overline{x}_5)\, x_6 =$$

$$= (\overline{x}_1 + \overline{x}_2)\, x_3 x_4 + x_3 x_5 + \overline{x}_6 .$$

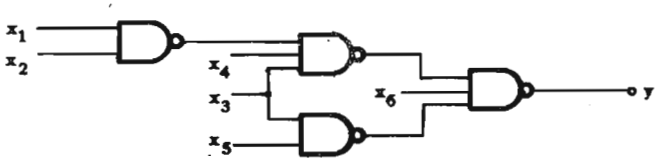


Fig.IV.7 - Circuito NAND.

Per arrivare all'espressione finale della y sono dunque necessari molti passaggi, ognuno dei quali comporta un'applicazione del teorema di De Morgan; né la situazione migliora se si scrive la y direttamente nella forma (IV.2).

Per vedere se è possibile ricavare la y, in forma di somma di prodotti, direttamente dall'esame del circuito, proviamo ora a scrivere le funzioni d'uscita, coi segni di somma-prodotto-inversione, di alcuni tipici circuiti NAND.

La funzione d'uscita y dell'elemento a due ingressi della figura IV.8a è $\overline{x}_1 + \overline{x}_2$. Nelle figg.IV.8b ÷ IV.8c sono riportate le y di altri quattro circuiti, costruite proseguendo passo-passo lungo i relativi itinerari.

Considerando le relazioni tra le variabili d'ingresso ai NAND, le posizioni dei NAND stessi sugli itinerari dei relativi circuiti e la forma delle funzioni d'uscita $y_a \ldots y_e$, si ricavano le seguenti regole:

– La funzione d'uscita, nella forma *somma di prodotti* è la somma di tutte le variabili che entrano sul primo livello, col segno cambiato, più tanti termini quanti sono gli itinerari $I_{1,2}$ verso gli elementi sul secondo livello.

– Ogni tinerario $I_{1,2}$ origina il prodotto di tutte le variabili che entrano sul secondo livello e di tanti fattori quanti sono gli itinerari $I_{2,3}$ verso gli elementi del terzo livello.

– Ogni itinerario $I_{2,3}$ origina la somma di tutte le variabili che entrano sul terzo livello, col segno cambiato, più tanti termini quanti sono gli itinerari verso gli elementi del quarto livello.

Il procedimento va continuato fino ad esaurire tutti gli itinerari verso i terminali d'ingresso, itinerari il cui numero eguaglia pertanto il numero delle lettere che compaiono nell'espressione della funzione d'uscita.

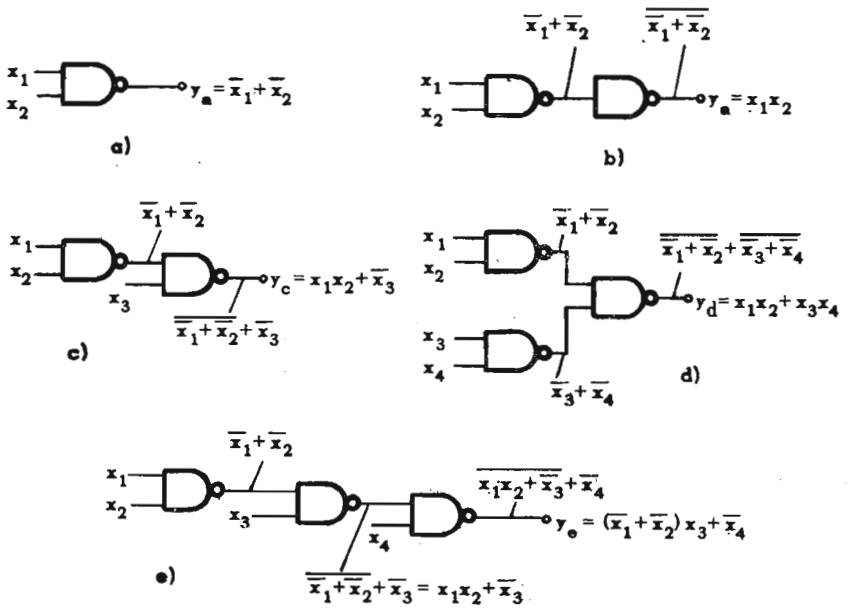


Fig.IV.8 - Funzioni d'uscita di tipici circuiti NAND.

Le regole dell'analisi possono essere espresse più concisamente così:

– i livelli di posto dispari originano le somme;

– i livelli di posto pari originano i prodotti;

– le variabili che entrano sui livelli dispari vanno complementate.

Illustriamo, con qualche esempio, l'applicazione di queste regole.

Esempio 1: *Analisi del circuito della fig.IV.7.*

Sul primo livello entra soltanto la variabile x_6, che deve comparire complementata nella somma finale y; esistono due itinerari $I_{1\cdot2}$ verso il secondo livello, che compariranno come altrettanti termini di y.

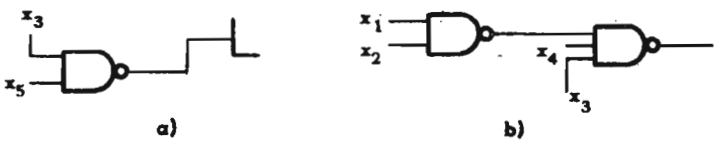


Fig.IV.9 - Itinerari del circuito di fig.IV.7.

Seguiamo il primo $I_{1\cdot2}$ (fig.IV.9a): ci sono due variabili sul secondo livello, che originano il prodotto x_3x_5; mancano itinerari del tipo $I_{2\cdot3}$. Il secondo itinerario $I_{1\cdot2}$, rappresentato nella fig.IV.9b, origina il prodotto delle due variabili che entrano sul secondo livello (x_3 e x_4) e di un fattore originato dall'unico itinerario del tipo $I_{2\cdot3}$, quello che porta al NAND di ingressi x_1 e x_2. Quest'ultimo NAND è su un livello dispari, quindi origina la somma $\overline{x}_1+\overline{x}_2$. In conclusione, il contributo degli itinerari $I_{1\cdot2}$ e $I_{2\cdot3}$ è:

$$x_3x_4\,(\overline{x}_1+\overline{x}_2)\ .$$

La funzione d'uscita y è quindi:

(IV.3) $$y=\overline{x}_6+x_3x_5+x_3x_4\,(\overline{x}_1+\overline{x}_2)\ .$$

Esempio 2: *Analisi del circuito della fig.IV.10.*

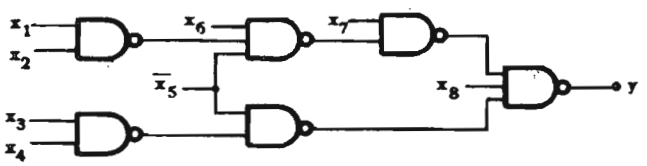


Fig.IV.10 - Circuito NAND.

Sul primo livello entra la variabile x_8, insieme a due ingressi provenienti dai livelli superiori: $\overline{x}_8$ è perciò uno dei tre termini della somma che costituisce l'espressione della y.

Gli itinerari $I_{1\cdot 2}$ verso il secondo livello sono mostrati nelle figg.IV.11a e IV.11b, dove sono messi in evidenza anche gli itinerari di ordine superiore.

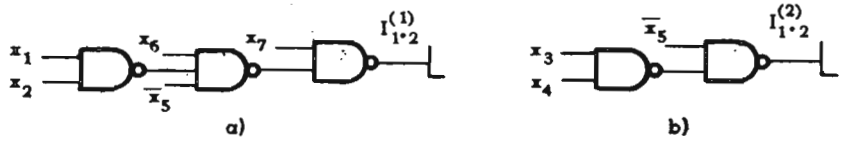


Fig.IV.11 - Itinerari del circuito di fig.IV.9.

L'itinerario $I^{(1)}_{1\cdot 2}$ origina il prodotto di x_7 (secondo livello) per $\overline{x}_6 + x_5$ (sul terzo livello, quindi col segno cambiato) $+ x_1 x_2$ (prodotto delle variabili sul quarto livello). In definitiva:

$$x_7 (\overline{x}_6 + x_5 + x_1 x_2) \quad .$$

L'itinerario $I^{(2)}_{1\cdot 2}$ origina il prodotto della variabile sul secondo livello $(\overline{x}_5)$ per la somma delle due variabili (x_3 ed x_4) sul terzo livello, cioè:

$$\overline{x}_5 (\overline{x}_3 + \overline{x}_4) \quad .$$

La funzione d'uscita y del circuito è quindi:

$$y = \overline{x}_8 + x_7 (\overline{x}_6 + x_5 + x_1 x_2) + \overline{x}_5 (\overline{x}_3 + \overline{x}_4) \quad .$$

Esempio 3: *Analisi del circuito della fig.IV.12.*

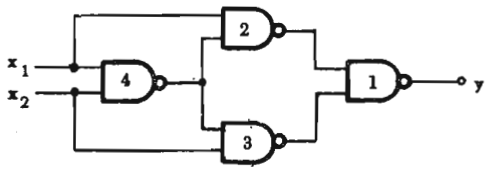


Fig.IV.12 - Circuito NAND.

Nessuna variabile entra sul primo livello. Ci sono due itinerari verso i livelli superiori, attraverso i NAND 1·3·4 e 1·2·4.

Il primo itinerario origina il termine:

$$x_2 (\overline{x}_1 + \overline{x}_2) \; ;$$

il secondo origina:

$$x_1 (\overline{x}_1 + \overline{x}_2) \quad .$$

L'espressione della funzione d'uscita y è perciò:

$$y = x_2 (\overline{x}_1 + \overline{x}_2) + x_1 (\overline{x}_1 + \overline{x}_2) = \overline{x}_1 x_2 + x_1 \overline{x}_2 \quad .$$

Si noti che:

– il NAND 4, sul terzo livello, entra a far parte di due itinerari diversi;

– nell'espressione della y compaiono 4 lettere; i quattro itinerari corrispondenti sono: x_1-2-1-y, x_1-4-2-1-y, x_2-3-1-y.

La funzione $y = \overline{x}_1 x_2 + x_1 \overline{x}_2$, di particolare importanza, per ragioni che vedremo in seguito, viene chiamata *somma modulo 2*, ed indicata col simbolo:

$$y = x_1 \oplus x_2 \ .$$

Esempio 4: *Analisi del circuito della fig.IV.13.*

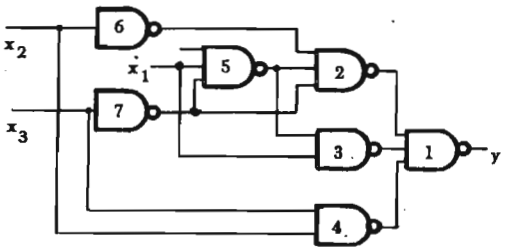


Fig.IV.13 - Circuito NAND.

Nessuna variabile entra sul primo livello. Esistono tre itinerari del tipo $I_{1\cdot 2}$, attraverso i NAND 2, 3, 4. L'itinerario $I^{(2)}_{1\cdot 2}$ origina il prodotto di tre termini, corrispondenti ai tre itinerari del tipo $I_{2\cdot 3}$ attraverso i NAND 5, 6, 7, $I_{2\cdot 3}$ origina il termine $\overline{x}_2$ (x_2 entra su un livello dispari). $I^{(7)}_{2\cdot 3}$ il termine $\overline{x}_3$; $I^{(5)}_{2\cdot 3}$, infine, la somma di $\overline{x}_1$ (su un livello dispari), x_2 (su un livello pari) e x_3 (su un livello pari). In definitiva, l'itinerario $I^{(2)}_{1\cdot 2}$ dà luogo a:

$$y_{1\cdot 2} = \overline{x}_2 \overline{x}_3 (\overline{x}_1 + x_2 + x_3) = \overline{x}_1 \overline{x}_2 \overline{x}_3 \ .$$

L'itinerario $I^{(3)}_{1\cdot 2}$ origina il prodotto di x_1 (su un livello pari) per la somma di $\overline{x}_1$ (su un livello dispari) con $x_2 + x_3$ (su un livello pari). Complessivamente dall'itinerario $I^{(3)}_{1\cdot 2}$ si ha:

$$y_{1\cdot 3} = x_1 (\overline{x}_1 + x_2 + x_3) = x_1 x_2 + x_1 x_3 \ .$$

L'itinerario $I^{(4)}_{1\cdot 2}$ origina il prodotto delle variabili x_2 e x_3, entrambi su un livello pari:

$$y_{1\cdot 4} = x_2 x_3 \ .$$

La funzione d'uscita del circuito è pertanto:

$$y = y_{1\cdot 2} + y_{1\cdot 3} + y_{1\cdot 4} = \overline{x}_1 \overline{x}_2 \overline{x}_3 + x_1 x_2 + x_1 x_3 + x_2 x_3 \ .$$

Si noti che gli ingressi x_2 e x_3 non solo fanno parte di quattro itinerari diversi (1-4, 1-2-6, 1-2-5-6, 1-3-5-6 per x_2 e 1-4, 1-2-7, 1-2-5-7, 1-3-5-7 per x_3) ma si trovano, nei diversi itinerari, sui livelli pari e dispari.

Esempio 5: *Analisi del circuito della fig.IV.14.*

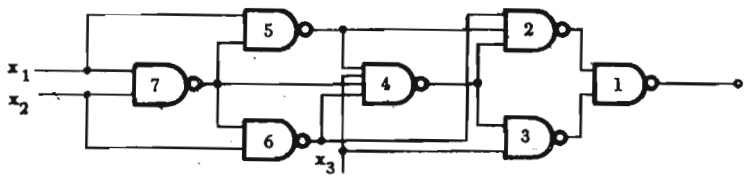


Fig.IV.14 - Circuito NAND.

Esistono due itinerari $I_{1\cdot2}$, attraverso i NAND 2 e 3.

$I^{(2)}_{1\cdot2}$ è formato dal prodotto di $y_{2\cdot6}$, $y_{2\cdot5}$, $y_{2\cdot4}$. La $y_{2\cdot6}$ è formata dalla somma di $y_{6\cdot7}$ e x_2 (livello dispari); per essere $y_{6\cdot7} = x_1\overline{x}_2$ (livello pari), $y_{2\cdot6} = \overline{x}_2 + x_1x_2 = \overline{x}_2 + x_1$. La $y_{2\cdot5}$ è formata dalla somma di $\overline{x}_1$ (livello dispari) e di $y_{5\cdot7} = x_1x_2$ (livello pari), quindi $y_{2\cdot5} = \overline{x}_1 + x_1x_2 = \overline{x}_1 + x_2$. La $y_{2\cdot4}$ è la somma di $\overline{x}_3$ (livello dispari), $y_{4\cdot5}$, $y_{4\cdot7}$, $y_{4\cdot6}$; $y_{45} = x_1(\overline{x}_1 + \overline{x}_2) = x_1\overline{x}_2$; $y_{4\cdot7} = x_1x_2$; $y_{4\cdot6} = x_2(\overline{x}_1 + \overline{x}_2) = \overline{x}_1x_2$, quindi $y_{2\cdot4} = \overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2$.

Il termine di y derivante dall'itinerario considerato è quindi:

$$y_{1\cdot2} = y_{2\cdot6} \cdot y_{2\cdot5} \cdot y_{2\cdot4} = (\overline{x}_2 + x_1)(\overline{x}_1 + x_2)(\overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2) \ .$$

L'itinerario $I^{(3)}_{1\cdot2}$ è formato dal prodotto di $y_{3\cdot6}$ per x_3 (livello pari).

La $y_{3\cdot4}$ è uguale alla $y_{2\cdot4}$, perché i NAND 2 e 3 si trovano entrambi sul secondo livello, quindi:

$$y_{1\cdot3} = x_3 \cdot y_{3\cdot4} = x_3(\overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2) \ .$$

La funzione d'uscita y del circuito è:

$$y = y_{1\cdot2} + y_{1\cdot3} = (\overline{x}_2 + x_1)(\overline{x}_1 + x_2)(\overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2) + x_3(\overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2) =$$

$$= [(\overline{x}_2 + x_1)(\overline{x}_1 + x_2) + x_3](\overline{x}_3 + x_1\overline{x}_2 + x_1x_2 + \overline{x}_1x_2) = (x_3 + x_1x_2 + \overline{x}_1\overline{x}_2)(\overline{x}_3 + x_1 + x_2) =$$

$$= \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_1x_3 + x_2x_3 .$$

IV.5 - Analisi dei circuiti NOR.

Sfruttando, analogamente a quanto fatto per i circuiti NAND, la definizione dell'operatore NOR, si può ottenere direttamente un'espressione analitica della funzione d'uscita di un circuito realizzato con ele-

menti NOR. Ad esempio, il circuito della fig.IV.15 realizza la funzione:

$$y = [(x_1 \downarrow x_2) \downarrow x_3 \downarrow x_4] \downarrow (x_3 \downarrow x_5) \downarrow x_6 \,. \tag{IV.4}$$

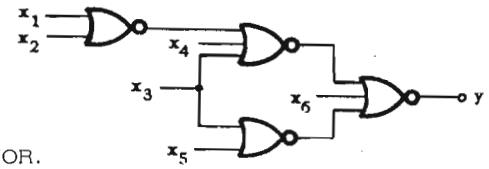


Fig.IV.15 - Circuito NOR.

L'espressione (IV.4), per essere utilizzata nella costruzione di una tabella di verità, va trasformata in somma di prodotti:

$$\begin{aligned}
y &= [(x_1 \downarrow x_2) \downarrow x_3 \downarrow x_4] \downarrow (x_3 \downarrow x_5) \downarrow x_6 = \\
&= \overline{\overline{x_1 + x_2} + x_3 + x_4} + \overline{x_3 + x_5} + x_6 = \\
&= \overline{\overline{x}_1 \overline{x}_2 + x_3 + x_4} + \overline{x}_3 \overline{x}_5 + x_6 = \\
&= \overline{(x_1 + x_2)\overline{x}_3 \overline{x}_4 + \overline{x}_3 \overline{x}_5} + x_6 = \\
&= (\overline{x}_1 \overline{x}_2 + x_3 + x_4)(x_3 + x_5)\,\overline{x}_6 = \\
&= \overline{x}_1 \overline{x}_2 x_3 \overline{x}_6 + \overline{x}_1 \overline{x}_2 x_5 \overline{x}_6 + x_3 \overline{x}_6 + x_4 x_5 \overline{x}_6 \,.
\end{aligned}$$

Per giungere più brevemente alla forma finale della y procediamo, analogamente a quanto fatto per i circuiti NAND, all'esame di alcuni circuiti tipici, a partire dalla definizione di elemento NOR (fig.IV.16a). Le relative espressioni delle y sono state ottenute passo-passo, procedendo dagli ingressi verso il terminale d'uscita (figg. IV.16b ÷ IV.16e).

Considerando le relazioni fra le variabili d'ingresso ai vari NOR, le posizioni dei NOR stessi sui relativi circuiti e le espressioni delle funzioni $y_a \ldots y_e$, si ricava che:

– la funzione y si ottiene come un *prodotto di somme* essendo il prodotto formato da tutte le variabili sul primo livello, col segno cambiato, e da tanti termini quanti sono gli itinerari $I_{1\cdot 2}$ verso i NOR sul secondo livello;

– ogni itinerario $I_{1\cdot 2}$ origina la somma di tutte le variabili che entrano sul secondo livello più tanti termini quanti sono gli itinerari $I_{2\cdot 3}$ verso i NOR del terzo livello;

– ogni itinerario $I_{2,3}$ origina il prodotto di tutte le variabili che entrano sul terzo livello, col segno cambiato, e di tanti termini quanti sono gli itinerari verso i NOR del quarto livello;

– il procedimento va continuato fino ad esaurire tutti gli itinerari verso i terminali d'ingresso, il cui numero eguaglia quindi il numero delle lettere che compaiono nell'espressione della y.

Le regole dell'analisi possono essere espresse più concisamente così:

– i livelli dispari originano i prodotti;

– i livelli pari originano le somme;

– le variabili che entrano sui livelli dispari vanno complementate.

Illustriamo, con qualche esempio, l'applicazione di queste regole.

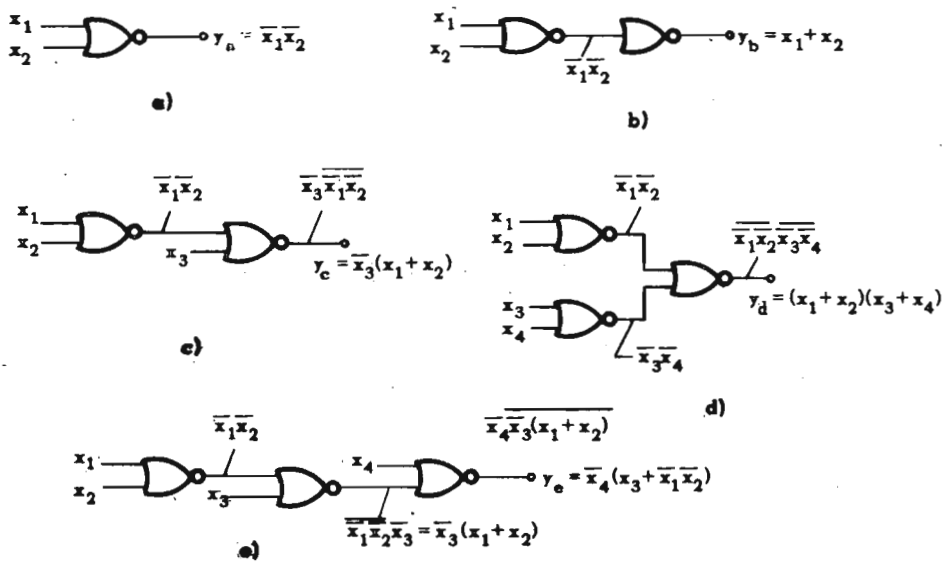


Fig.IV.16 - Funzioni d'uscita di tipici circuiti NOR.

Esempio 1: *Analisi del circuito della fig.IV.17.*

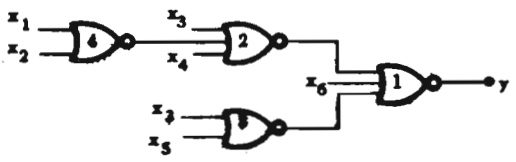


Fig.IV.17 - Circuito NOR.

La y d'uscita è il prodotto della variabile $\overline{x}_6$ per le due funzioni $y_{1\cdot2}$ e $y_{1\cdot3}$, corrispondenti ai due itinerari del tipo $I_{7\cdot2}$.

La $y_{1\cdot3}$ è la somma di x_3 ed x_5 (su un livello pari); la $y_{1\cdot2}$ è la somma di x_3, x_4 e $y_{2\cdot4}=\overline{x}_1\overline{x}_2$ (su un livello dispari). In definitiva:

$$y = \overline{x}_6 \cdot y_{1\cdot3} \cdot y_{1\cdot2} = \overline{x}_6 (x_3 + x_5)(x_3 + x_4 + \overline{x}_1\overline{x}_2) \quad . \tag{IV.5}$$

Confrontando la (IV.5) con la (IV.3), relativa al circuito della fig.IV.7 che ha la stessa configurazione di quello in esame ma è realizzato con elementi NAND, si vede che le due funzioni sono una la duale dell'altra. Questa proprietà, del tutto generale perché deriva dalla dualità tra le funzioni NAND e NOR, può essere così formulata:

– se in un circuito costruito con elementi NOR (NAND) e con funzione d'uscita y, si sostituisce ogni NOR (NAND) con un NAND (NOR), si ottiene un circuito la cui funzione d'uscita y' è duale della y, cioè si ottiene da questa scambiando le operazioni di somma e prodotto.

Esempio 2: *Analisi del circuito della fig.IV.18.*

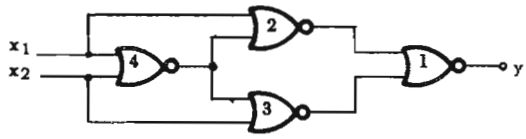


Fig.IV.18 - Circuito NOR.

La funzione d'uscita y è il prodotto di $y_{1\cdot2}$ per $y_{1\cdot3}$, derivanti dai due itinerari del tipo $I_{1\cdot2}$. La $y_{1\cdot2}$ è la somma di x_1 (che entra su un livello pari) e di $y_{2\cdot4}$; $y_{2\cdot4}$ è il prodotto di $\overline{x}_1$ con $\overline{x}_2$ (che entrano su un livello dispari); quindi: $y_{1\cdot2} = x_1 + \overline{x}_1\overline{x}_2$.

La $y_{1\cdot3}$ è la somma di x_2 (che entra su un livello pari) e di $y_{3\cdot4}$; $y_{3\cdot4}$ è il prodotto di $\overline{x}_1$ e $\overline{x}_2$; in definitiva $y_{1\cdot3} = x_2 + \overline{x}_1\overline{x}_2$.

Risulta:

$$y = y_{1\cdot2} \cdot y_{1\cdot3} = (x_1 + \overline{x}_1\overline{x}_2)(x_2 + \overline{x}_1\overline{x}_2) = (x_1 + \overline{x}_2)(x_2 + \overline{x}_1) = x_1x_2 + \overline{x}_1\overline{x}_2 \quad . \tag{IV.6}$$

Questa funzione, duale della *somma modulo 2*, come si può vedere confrontando i circuiti delle figg.IV.12 e IV.13, è anche essa di grande utilità pratica. La (IV.6) vale solo quando x_1 e x_2 hanno entrambi lo stesso valore: il relativo circuito si chiama, per questo, *circuito di coincidenza*. Il circuito della fig.IV.12, che ha invece uscita 1 solo quando x_1 e x_2 hanno valori opposti, viene anche chiamato *circuito di anticoincidenza*.

Esempio 3: *Analisi del circuito della fig.IV.19.*

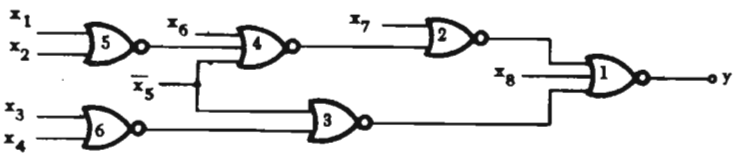


Fig.IV.19 - Circuito NOR.

La funzione d'uscita y è il prodotto di $\overline{x}_8$ (su un livello dispari), $y_{1\cdot2}$ e $y_{1\cdot3}$, provenienti dai due itinerari del tipo $I_{1\cdot2}$.

La $y_{1\cdot3}$ è la somma di $\overline{x}_5$ (su un livello pari) e di $y_{3\cdot6} = \overline{x}_3\overline{x}_4$ (entrambi su un livello dispari); pertanto:

$$y_{1\cdot3} = \overline{x}_5 + \overline{x}_3\overline{x}_4 \quad .$$

La $y_{1\cdot2}$ è la somma di x_7 (su un livello pari) e di $y_{2\cdot4}$; $y_{2\cdot4}$ è il prodotto di $\overline{x}_6 x_5$ (entrambi su un livello dispari) e di $y_{4\cdot5} = x_1 + x_2$ (su un livello pari); pertanto: $y_{1\cdot2} = x_7 + \overline{x}_6 x_5 (x_1 + x_2)$.

L'espressione della funzione d'uscita:

$$y = \overline{x}_8 \cdot y_{1\cdot3} \cdot y_{1\cdot2} = \overline{x}_8(\overline{x}_5 + \overline{x}_3\overline{x}_4)\left[x_7 + \overline{x}_6 x_5 (x_1 + x_2)\right]$$

è la duale di quella del circuito della fig.IV.10.

Esempio 4: *Analisi del circuito della fig.IV.20.*

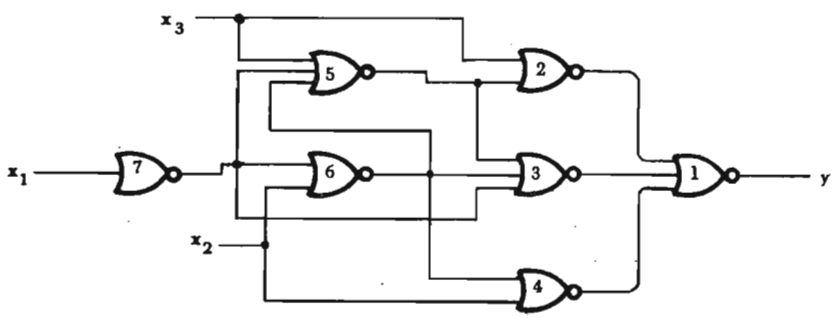


Fig.IV.20 - Circuito NOR.

Sul primo livello non entra nessuna variabile. Esistono tre itinerari del tipo $I_{1\cdot2}$; la y è pertanto il prodotto di $y_{1\cdot2}$, $y_{1\cdot3}$, $y_{1\cdot4}$.

La $y_{1\cdot2}$ è la somma di x_3 (su un livello pari) e di $y_{2\cdot5}$; $y_{2\cdot5}$ è il prodotto di $\overline{x}_3$ (su un livello dispari), x_1 (su un livello pari) e $y_{5\cdot6}$.

La $y_{5\cdot6}$ è la somma di x_2 (su un livello pari) e $\overline{x}_1$ (su un livello dispari). In definitiva:

$$y_{1\cdot2} = x_3 + \overline{x}_3 x_1 (x_2 + \overline{x}_1) = x_3 + x_1 x_2 \ .$$

La $y_{1\cdot3}$ è la somma di $\overline{x}_1$ (su un livello dispari), $y_{3\cdot5}$ e $y_{3\cdot6}$; $y_{3\cdot5}$ è identica a $y_{2\cdot5}$, perché i NOR 2 e 3 si trovano sullo stesso livello; $y_{3\cdot6}$ è il prodotto di $\overline{x}_2$ (su un livello dispari) con x_1 (su un livello pari):

$$y_{1\cdot3} = \overline{x}_1 + \overline{x}_3 x_1 (x_2 + \overline{x}_1) + \overline{x}_2 x_1 = x_1 + x_2 \overline{x}_3 + \overline{x}_2 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3 \ .$$

La $y_{1\cdot4}$ è la somma di x_2 (su un livello pari) e di $y_{4\cdot6}$; $y_{4\cdot6}$ è uguale a $y_{3\cdot6}$, per essere i NOR 3 e 4 sullo stesso livello; si ha così:

$$y_{1\cdot4} = x_1 + x_2 \ .$$

L'espressione della y, nelle due forme *prodotto di somme* e *somma di prodotto* è:

$$y = y_{1\cdot2} \cdot y_{1\cdot3} \cdot y_{1\cdot4} = (x_3 + x_1 x_2)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3)(x_1 + x_2) =$$

$$= (x_1 x_2 + x_1 x_3 + x_2 x_3)(\overline{x}_1 + \overline{x}_2 + \overline{x}_3) =$$

$$= x_1 x_2 \overline{x}_3 + x_1 \overline{x}_2 x_3 + \overline{x}_1 x_2 x_3 \ .$$

IV.6 - Sintesi.

Eseguire la sintesi di un circuito logico combinatorio significa progettare un circuito ad n ingressi avente una determinata funzione di uscita y.

In pratica, il problema della sintesi si presenta sotto forme assai diverse, a seconda di come viene data la funzione y, della libertà lasciata nella scelta dei componenti elettronici, delle limitazioni imposte al numero dei livelli, ad esempio per ragioni di velocità, della disponibilità o meno delle variabili negate agli ingressi del circuito.

I fattori elencati sono soltanto in parte indipendenti, esistendo complesse relazioni tra alcuni di essi, che soltanto una buona pratica può mettere in evidenza.

Prescindendo dai criteri di scelta dei componenti elettronici (diodi transistor o micrologici) e del tipo di logica (AND-OR-NOT, NAND, NOR), sia perché la scelta può avvenire per ragioni imposte dal costruttore o dettate da esigenze di omogeneità con circuiti già costruiti, sia perché riteniamo sufficiente quanto esposto al cap.III, tratteremo i me-

todi di sintesi prima per i circuiti AND-OR-NOT, poi per i circuiti NAND, infine per i NOR. L'esistenza o meno delle variabili d'ingresso in forma negata sarà considerata in ognuno dei detti casi.

IV.6.1 - Descrizione del funzionamento del circuito.

La funzione logica y, che il circuito da progettare deve realizzare al terminale d'uscita, può essere data in quattro forme diverse:

a) verbalmente: è la forma più imprecisa, ma anche la più comune, e consiste in una descrizione a parole del funzionamento del circuito, di eventuali condizioni taciute e di ogni vincolo, di qualsiasi natura, imposto da altri circuiti a monte o a valle.

Dalla descrizione verbale bisogna sempre passare ad una tabella di verità, assegnando il valore implicitamente o esplicitamente espresso nella descrizione stessa ad ognuna delle 2^n possibili configurazioni delle variabili d'ingresso.

Un esempio di descrizione verbale è il seguente: *progettare un circuito logico a tre ingressi, mai contemporaneamente nulli, che dà un segnale d'allarme appena un numero pari d'ingressi assume il valore 1.*

x_1	x_2	x_3	y
0	0	0	–
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fig.IV.21 - Tabella di verità.

b) Con una tabella di verità: è il punto di partenza effettivo della sintesi direttamente traducibile – come già visto – in una forma analitica (canonica) e in una grafica (sulle mappe di Karnaugh).

Alla tabella di verità si perviene anche dalla descrizione verbale della funzione. Ad esempio, nella fig.IV.21 è riportata la tabella di verità della funzione descritta al punto a).

c) Con un'espressione analitica: è questo il modo più conciso, purtroppo non univoco, per descrivere il comportamento di un circuito, con riguardo alle sue proprietà logiche invarianti. L'espressione analitica può essere una forma raffinata di descrizione verbale (ad esempio: progettare un circuito d'allarme per la condizione $x_1 (x_2 \oplus x_3) + \overline{x}_1 x_2 \overline{x}_3$) o derivare dallo schema logico di un circuito, come appresso specificato.

d) Con lo schema logico, quando un certo circuito si vuole riprogettare con componenti diversi, o in forma più economica.

Esempio: *Sintesi di un circuito NAND il cui funzionamento sia identico a quello AND--OR-NOT della fig.IV.22.*

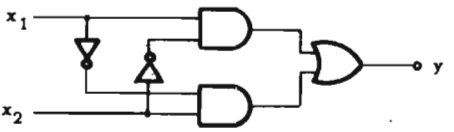


Fig.IV.22 - Schema logico di un circuito.

Dallo schema, bisogna sempre passare all'espressione analitica della funzione d'uscita; per il circuito della fig.IV.22, applicando le regole dell'analisi, si ha.

$$y = x_1\overline{x}_2 + \overline{x}_1 x_2 \ .$$

Il progetto del circuito, partendo dalla tabella di verità o da una espressione analitica della funzione y, si fa applicando i metodi di semplificazione esposti precedentemente per arrivare a una forma conveniente della y stessa.

Abbiamo detto *forma conveniente* e non *forma minima*, perché la forma algebrica minima non sempre può essere realizzata, sia per ragioni tecniche, sia per limitazioni al numero massimo dei livelli.

IV.6.2 - Incidenza del numero massimo di livelli.

Il tempo di commutazione di un componente non è mai nullo, per cui la risposta di un elemento E alla variazione dei valori d'ingresso ha sempre un valore $\Delta \neq 0$. La risposta alla variazione della configurazione degli n ingressi di un circuito a p livelli, formato da elementi E, non potrà quindi aversi prima di un tempo $\Delta \cdot p$. Il numero dei livelli è così proporzionale al ritardo che un segnale incontra nella sua propagazione dall'ingresso all'uscita: la necessità di contenere questo ritardo entro determinati limiti fissa il numero massimo dei livelli del circuito ed incide sulla sintesi. La forma minima algebrica può infatti contenere un numero troppo elevato di parentesi, dando origine ad un circuito con un numero inaccettabile di livelli.

In pratica, poiché i metodi di semplificazione esposti conducono tutti alla forma minima a due livelli, prima di iniziare la ricerca della forma minima assoluta, cui si giunge attraverso una o più fattorizzazione, occorre accettarsi che i ritardi introdotti siano compatibili con la natura del problema. Spesso, il circuito a due livelli è l'unica forma possibile.

Esiste poi un'altra considerazione, sempre di ordine tecnologico, che va tenuta presente: attualmente, il proposito di semplificare al massimo la costruzione e la manutenzione dei calcolatori, o dei sistemi

numerici complessi, porta a preferire, a circuiti minimi ma con connessioni troppo complicate, circuiti con un maggior numero di componenti, ma il cui cablaggio presenti particolari proprietà di simmetria o di semplicità.

I metodi analitici di semplificazione, per i motivi sopra accennati, non vanno in definitiva applicati indiscriminatamente. L'algebra logica, di estrema utilità per le notazioni, non può fornire la soluzione ottima: essa è un indispensabile strumento di lavoro, ma va usato senza perdere di vista l'effettiva utilita' pratica dei risultati che permette di raggiungere.

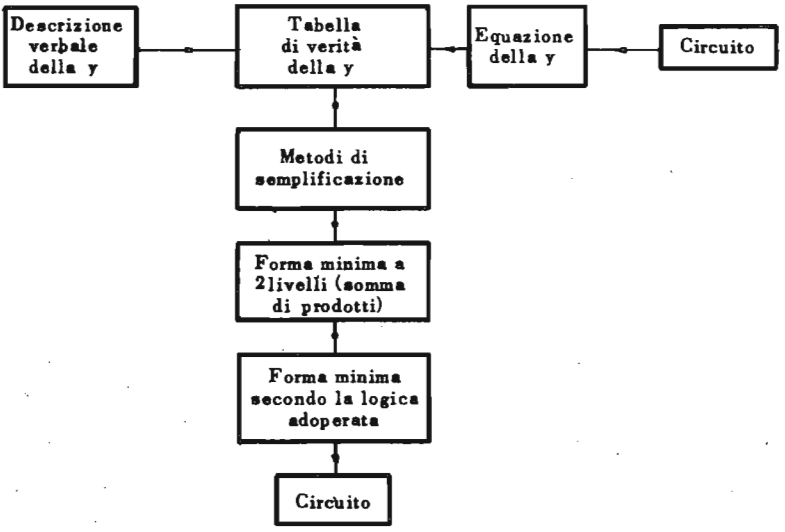


Fig.IV.23 - Passi del procedimento di sintesi.

Ciò premesso, per l'evidente mancanza di criteri generali validi in ogni caso, tratteremo il problema della sintesi secondo il procedimento descritto nello schema della fig.IV.23, dove sono mostrate anche le relazioni tra le quattro forme in cui può essere data la funzione da realizzare.

Per quanto riguarda la sintesi con circuiti integrati, rimandiamo al capitolo dedicato alle tecniche digitali.

IV.7 - Sintesi dei circuiti AND-OR-NOT

La sintesi avviene a partire dalla forma minima della funzione, cioè dalla forma dove compare il minimo numero di lettere. Se, però, il cir-

cuito deve essere realizzato con micrologici, si ricercherà la forma che minimizza il numero dei termini sotto i vincoli imposti dalle caratteristiche del microcircuito adottato.

Daremo qui vari esempi di sintesi, a partire da tutte le forme di rappresentazione della y da realizzare. I metodi di semplificazione adottati sono quelli del capitolo II; esistono altri procedimenti, basati sul costo della funzione e sul concetto di *decomponibilità*, ma sono tutti troppo complessi per essere esposti in questa sede (v. bibliografia, particolarmente [8] e [9]).

Nei riguardi del numero massimo di livelli dei circuiti AND-OR-NOT, alle limitazioni di ordine temporale si aggiungono quelle sul numero massimo di elementi passivi che è possibile disporre in serie; nel valutare la convenienza di un circuito a pochi diodi su più livelli nei confronti di uno a due soli livelli, vanno messi in conto gli elementi attivi necessari a *riformare* il segnale (v. cap.III).

Per il confronto tra espressioni diverse della stessa funzione, riesce utile la seguente *regola* per contare il numero dei diodi e dei livelli senza disegnare il circuito:

Il numero dei diodi è uguale alla somma delle lettere più la somma dei termini (somme o prodotti) con più di una lettera; il numero dei livelli è uguale al massimo numero di segni di prodotto e di somma che si trovano in serie, alternati fra loro.

Esempio: La funzione:

$$y = x_1x_2x_3 + x_1x_2\overline{x}_4 + x_1x_5$$

è realizzata in un circuito a due livelli (i segni alternati sono $\cdot$ e +) e 11 diodi (8 lettere più 3 termini).

La stessa funzione, scritta sotto forma fattorizzata:

$$y = x_1\left[x_2(x_3 + \overline{x}_4) + x_5\right],$$

porta a un circuito a 4 livelli (segno + tra x_3 e $\overline{x}_4$ in serie col segno $\cdot$ tra x_2 e $(x_3 + \overline{x}_4)$, in serie col + davanti x_5, in serie col $\cdot$ tra x_1 e la parentesi quadra) e 8 diodi (uno per ogni lettera, più 3 per i termini $(x_3 + \overline{x}_4)$, $x_2(x_3 + \overline{x}_4)$, $x_2(x_3 + \overline{x}_4) + x_5$; non si deve tener conto di $x_1\left[x_2(x_3 + \overline{x}_4) + x_5\right]$ perché è l'unico termine di una convenzionale somma di prodotti).

Negli esempi che seguono, supporremo di avere a disposizione le variabili d'ingresso in forma diretta e negata. Esamineremo, in seguito, le conseguenze derivanti dal mancato verificarsi di questa ipotesi.

Esempio 1: *Sui tre ingressi $x_1x_2x_3$ di un circuito giunge un numero n compreso tra 0 e 5 ed espresso nel sistema di numerazione binario, essendo x_3 il bit meno significativo. All'uscita del circuito deve essere realizzato il prodotto 3n, espresso sempre in binario.*

Dalla descrizione verbale del circuito occorre ricavare la tabella di verità della sua funzione d'uscita y. Il massimo valore che può assumere la y($5 \times 3 = 15$) è rappresentabile con quattro bit: il circuito avrà perciò 4 uscite. Le 4 tabelle di verità, una per ogni uscita, debbono tradurre in binario le relazioni:

$$0 \times 3 = 0$$
$$1 \times 3 = 3$$
$$2 \times 3 = 6$$
$$3 \times 3 = 9$$
$$4 \times 3 = 12$$
$$5 \times 3 = 15$$

Si terrà conto dalle condizioni che non possono presentarsi per semplificare il circuito.

Le tabelle di verità, essendo tutte le y funzioni delle stesse variabili, sono rappresentate insieme nella fig.IV.24.

Variabili	x_1	x_2	x_3	y_1	y_2	y_3	y_4
Pesi	4	2	1	8	4	2	1
	0	0	0	0	0	0	0
	0	0	1	0	0	1	1
	0	1	0	0	1	1	0
	0	1	1	1	0	0	1
	1	0	0	1	1	0	0
	1	0	1	1	1	1	1
	1	1	0	–	–	–	–
	1	1	1	–	–	–	–

Fig.IV.24 - Tabella di verità.

Dalle tabelle, sono state ricavate le mappe di Karnaugh di $y_1y_2y_3y_4$ (fig.IV.25).

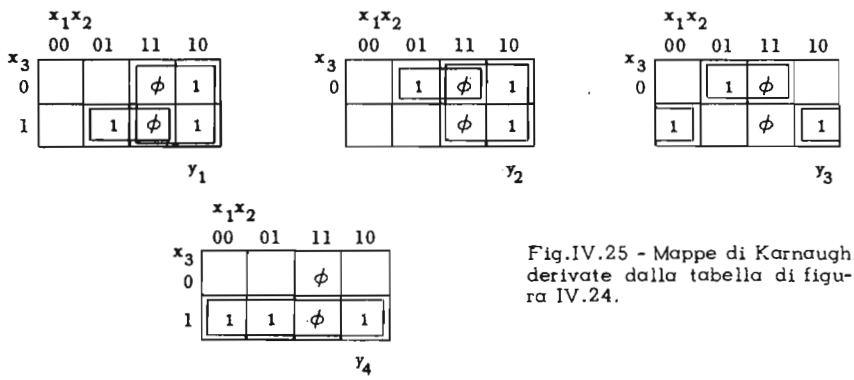


Fig.IV.25 - Mappe di Karnaugh derivate dalla tabella di figura IV.24.

Le forme minime a due livelli delle quattro funzioni sono:

$$\begin{cases} y_1 = x_1 + x_2x_3 \\ y_2 = x_1 + x_2\overline{x}_3 \\ y_3 = x_2\overline{x}_3 + \overline{x}_2x_3 \\ y_4 = x_3 \end{cases}$$

Il circuito completo, ottenuto mettendo in comune il termine $x_2\overline{x}_3$ che compare nelle funzioni y_2 e y_3, è mostrato nella fig.IV.26: è a 2 livelli e impiega 3 AND, 3 NOR e 12 diodi.

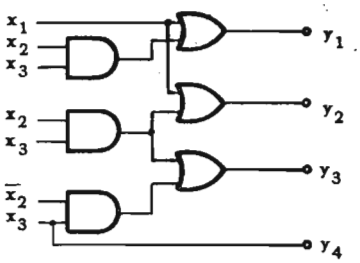


Fig.IV.26 - Circuito minimo derivato dalla tabella di figura IV.24.

Esempio 2: *Sintesi di un circuito a tre variabili avente la tabella di verità della figura IV.27a.*

a)

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	–
1	0	1	0
1	1	0	0
1	1	1	1

b)

x_3 \ x_1x_2	00	01	11	10
0				ϕ
1	1	1	1	

y

Fig.IV.27 - Tabella di verità (a) e relativa mappa di Karnaugh (b).

Dalla mappa di Karnaugh (fig.IV.27b), si ha per la forma minima, l'espressione:

$$y = \overline{x}_1x_3 + x_2x_3$$

realizzabile su due livelli con 2 AND, 1 OR e 6 diodi (fig.IV.28a).

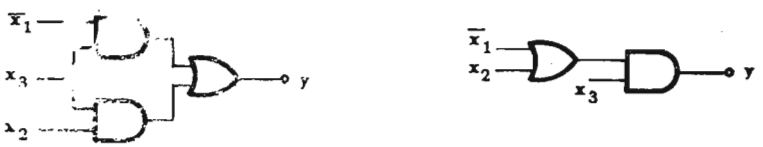


Fig.IV.28 - Realizzazioni circuitali della tabella di verità di fig.IV.27a.

Fattorizzando la variabile x_3, si ottiene l'espressione minima:

$$y = x_3(\overline{x}_1 + x_2)$$

realizzabile ancora su due livelli, con 1 AND, 1 OR e 4 diodi, fig.IV.28b, quindi senza altro più economica della prima.

Esempio 3: *Sintesi di un circuito con la stessa funzione di trasmissione del circuito della fig.IV.29, ma realizzato in maniera più economica.*

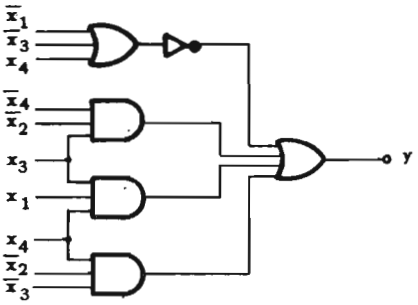


Fig.IV.29 - Circuito AND-OR-NOT.

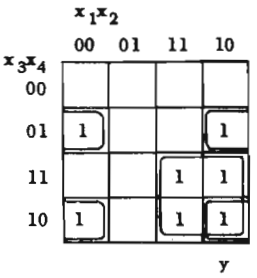


Fig.IV.30 - Mappa di Karnaugh per la funzione d'uscita del circuito della fig.IV.29.

L'espressione analitica della funzione d'uscita y, ricavabile dallo schema del circuito di fig.IV.29, è:

$$y = \overline{\overline{x}_1 + \overline{x}_3 + x_4} + \overline{x}_2x_3\overline{x}_4 + \overline{x}_2\overline{x}_3x_4 + x_1x_3x_4 =$$

$$= x_1x_3\overline{x}_4 + \overline{x}_2x_3\overline{x}_4 + \overline{x}_2\overline{x}_3x_4 + x_1x_3x_4 \ .$$

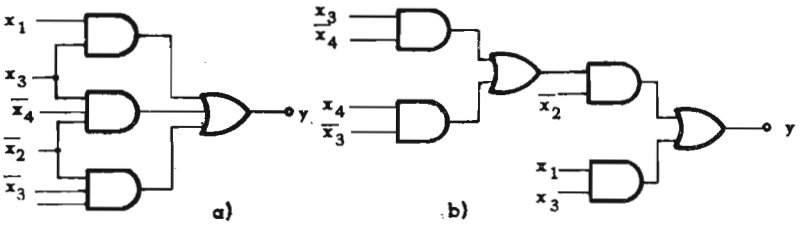


Fig.IV.31 - Circuiti equivalenti a quello della fig.IV.29.

Riportando la y direttamente sulla mappa di Karnaugh (fig.IV.30) si ottiene la espressione minima a due livelli:

$$y = x_1x_3 + \overline{x}_2x_3\overline{x}_4 + \overline{x}_2\overline{x}_3x_4$$

realizzabile con un circuito a 11 diodi (fig.IV.31a). Scrivendo la y in forma fattorizzata:

$$y = x_1x_3 + \overline{x}_2(x_3\overline{x}_4 + x_4\overline{x}_3)$$

si ottiene un circuito a 4 livelli e 12 diodi (fig.IV.31b); quindi senz'altro da scartare.

Si noti, confrontando i circuiti delle figg. IV.28 e IV.31, che non sempre la fattorizzazione aumenta il numero dei livelli, e non sempre diminuisce il numero dei diodi.

Esempio 4: *Sintesi di un circuito per la funzione:*

(IV.7) $$y = x_2x_4(\overline{x}_1 \oplus x_3) + x_1 \oplus x_3 \ .$$

La y, sviluppando le somme modulo 2 e le parentesi, ha l'espressione:

$$y = x_2x_4(\overline{x}_1\overline{x}_3 + x_1x_3) + \overline{x}_1x_3 + x_1\overline{x}_3 =$$
$$= \overline{x}_1x_2\overline{x}_3x_4 + x_1x_2x_3x_4 + \overline{x}_1x_3 + x_1\overline{x}_3 \ .$$

La mappa di Karnaugh, costruibile immediatamente (fig.IV.32) permette di ottenere l'espressione minima a due livelli, non ulteriormente riducibile:

(IV.8) $$y = x_1\overline{x}_3 + x_2x_4 + \overline{x}_1x_3 \ .$$

Il circuito corrispondente ha 3 AND, 1 OR, 9 diodi (fig.IV.33).

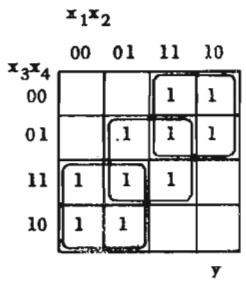


Fig.IV.32 - Mappa di Karnaugh per la funzione: $y = x_2x_4(\overline{x}_1 \oplus x_3) + x_1 \oplus x_3$.

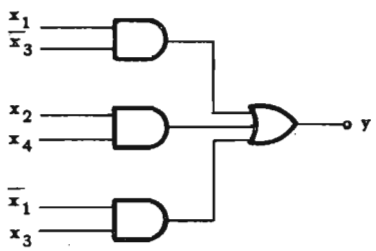


Fig.IV.33 - Circuito per la funzione della fig.IV.32.

Facciamo notare che si sarebbe arrivati immediatamente dalla (IV.7) alla (IV.8) qualora, posto:

$$x_1 \oplus x_3 = \Phi$$

si fosse riconosciuto che:

$$\overline{x}_1 \oplus x_3 = \overline{x_1 \oplus x_3} = \overline{\Phi}$$

per cui:

$$y = x_2x_4(\overline{x}_1 \oplus x_3) + x_1 \oplus x_3 = x_2x_4\overline{\Phi} + \Phi = x_2x_4 + \Phi = x_2x_4 + x_1 \oplus x_3$$

Negli esempi $1 \div 4$ abbiamo supposto di avere a disposizione, all'ingresso del circuito, anche le variabili in forma negata, come generalmente avviene in pratica. Quando si dispone soltanto delle variabili dirette, occorre usare degli invertitori.

Anche gli invertitori vanno minimizzati: non conviene, infatti, usarli semplicemente per invertire le variabili d'ingresso, come mostrano i due esempi seguenti.

Esempio 5: *Circuito AND-OR-NOT per la funzione:*

$$y = x_1/x_2/x_3/x_4 = \overline{x_1 x_2 x_3 x_4} = \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4$$

supponendo di avere la variabili $x_1 x_2 x_3 x_4$ *soltanto in forma diretta.*

La realizzazione della y mediante inversione delle variabili e OR finale porta a un circuito (fig.IV.34a) con 4 diodi e 4 transistor; la realizzazione della y mediante inversione della $y = x_1 x_2 x_3 x_4$, porta invece all'uso di un AND a 4 diodi e un solo invertitore (fig.IV.34b).

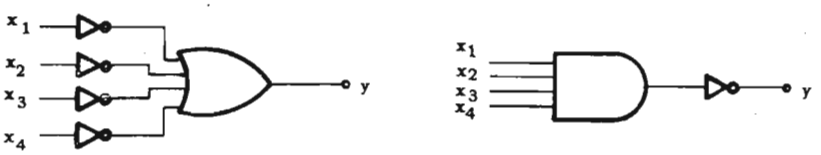


Fig.IV.34 - Realizzazioni della funzione $x_1/x_2/x_3/x_4$.

Esempio 6: *Sintesi di un circuito per la funzione:*

$$y = \Sigma\,(0, 1, 2, 3, 4, 5, 10) \quad .$$

Riportati i minterm della y sulla mappa di Karnaugh (fig.IV.35) si ottiene la forma minima a 2 livelli:

$$y = \overline{x}_1\overline{x}_2 + \overline{x}_1\overline{x}_3 + \overline{x}_2 x_3 \overline{x}_4 =$$
$$= \overline{x}_1(\overline{x}_2 + \overline{x}_3) + \overline{x}_2 x_3 \overline{x}_4 \quad .$$

x_3x_4 \ x_1x_2	00	01	11	10
00	1	1		
01	1	1		
11	1			
10	1			1

y

Fig.IV.35 - Tabella di verità per la funzione $y=\Sigma(0,1,2,3,4,5,10)$.

Se non si dispone degli ingressi negati, la y verrà ottenuta, a 2 o 3 livelli, sempre con 4 invertitori (fig.IV.36).

Realizzando invece la y come negazione della funzione:

$$\overline{y} = (x_1 + x_2)\,(x_1 + x_3)\,(x_2 + \overline{x}_3 + x_4) = (x_1 + x_2 x_3)\,(x_2 + \overline{x}_3 + x_4)$$

si giunge a due circuiti con egual numero di diodi, ma con 2 soli invertitori (fig.IV.37).

Prima di passare alla sintesi con elementi NAND e NOR, osserviamo che, se qualcuno degli AND e OR cui si perviene nel corso di un progetto ha un numero di ingressi superiore a quello ammissibile, lo si può spezzare in due o più elementi dello

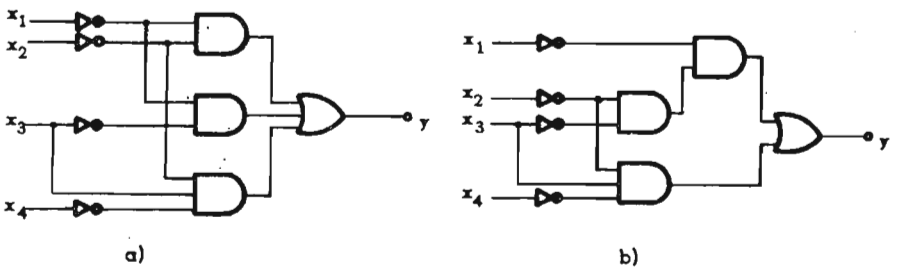


Fig.IV.36 - Realizzazioni circuitali della funzione di fig.IV.35.

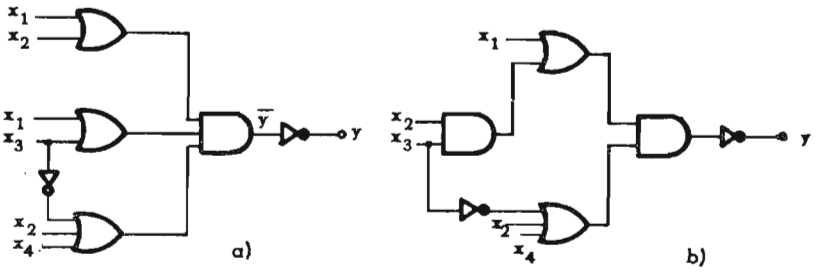


Fig.IV.37 - Altre realizzazioni circuitali della funzione di fig.IV.35.

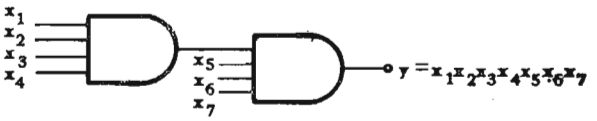


Fig.IV.38 - Realizzazione di un AND di 7 variabili con 2 AND a 4 variabili.

stesso tipo, per l'associatività delle operazioni di somma e prodotto. A titolo d'esempio, nella fig. IV.38 è mostrata la realizzazione della funzione AND delle variabili $x_1 \div \div x_7$ con due AND a 4 ingressi: ovviamente, il numero dei livelli è aumentato di uno, rispetto ad un AND a 7 ingressi.

IV.8 - Sintesi dei circuiti NAND.

La sintesi va fatta partendo ancora dall'espressione minima a due livelli in forma di somma di prodotti; forma minima sarà quella che

minimizza il numero di NAND, scegliendo – tra due circuiti a egual numero di NAND – quello a minor numero di ingressi.

Poiché l'invertitore può essere concepito come un NAND a un solo ingresso, il criterio di minimizzazione deve operare anche sugli invertitori: la sintesi si presenta così nettamente diversa a seconda che si abbiano, o no, a disposizione le variabili negate.

IV.8.1 - Sintesi con le variabili d'ingresso in forma diretta e negata.

Nel caso dell'analisi abbiamo mostrato come un circuito NAND origini una somma di prodotti, comportandosi ogni NAND – a parte il segno delle variabili – come un AND o come un OR, a seconda della sua posizione lungo gli itinerari che lo collegano all'uscita.

Procedendo al contrario, si può costruire un circuito NAND seguendo le stesse regole di un AND-OR, e invertendo le variabili che entrano sui livelli dispari. Gli esempi seguenti mostrano l'applicazione di questa regola, prescindendo da ogni criterio di semplificazione.

Esempio 1: *Disegnare il circuito NAND per la funzione:*

$$y = x_1x_2 + x_3\overline{x}_4\,(x_5 + \overline{x}_6) \quad .$$

Si realizza la somma $x_5 + \overline{x}_6$ nel NAND 1, che funziona da OR; i due prodotti $x_3\overline{x}_4\,(x_5 + \overline{x}_6)$ e x_1x_2 vengono formati, rispettivamente, nei NAND 2 e 3, che funzionano da AND; il NAND 4, funzionante da OR, realizza la somma finale. Il circuito risultante, fatti i dovuti collegamenti, è quello della fig.IV.39a.

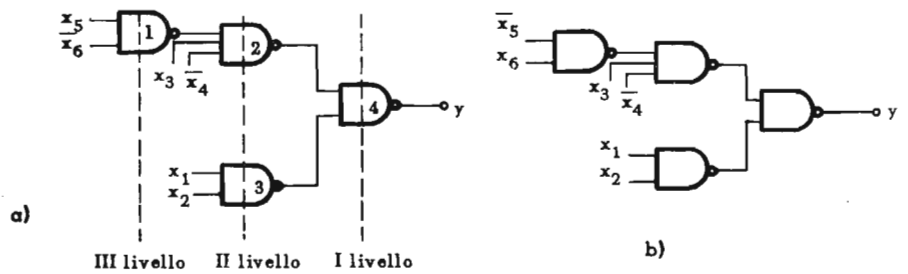


Fig.IV.39 - Sintesi di un circuito NAND.

Applichiamo ora la regola dei livelli: le variabili x_5 e $\overline{x}_6$ entrano su un livello dispari, quindi vanno complementate; il circuito per la y è quello della fig.IV.39b.

Esempio 2: *Circuito NAND per la funzione:*

$$y = x_1\,(x_2x_3x_4 + x_5) + \left[(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)\overline{x}_5 + x_6\right]x_7 \quad .$$

Si costruiscono coi NAND collegati e numerati come nella fig.IV.40a:

- il prodotto $x_2x_3x_4$ (NAND 1);
- la somma $x_5 + x_2x_3x_4$ (NAND 2);
- la somma $\overline{x}_2 + \overline{x}_3 + \overline{x}_4$ (NAND 3);
- il prodotto $(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)\overline{x}_5$ (NAND 4);
- la somma $(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)\overline{x}_5 + x_6$ (NAND 5);
- il prodotto $x_1(x_2x_3x_4 + x_5)$ (NAND 6);
- il prodotto $x_7[(\overline{x}_2 + \overline{x}_3 + \overline{x}_4)\overline{x}_5 + x_6]$ (NAND 7);
- la somma finale (NAND 8).

La complementazione delle variabili sui livelli dispari porta al circuito della figura IV.40b. Questo circuito può essere semplificato osservando che il blocco (figura IV.40c) costituito dai NAND di livello più elevato è contenuto in due itinerari diversi: mettendolo in comune, si ottiene il circuito a 6 NAND della fig.IV.40d.

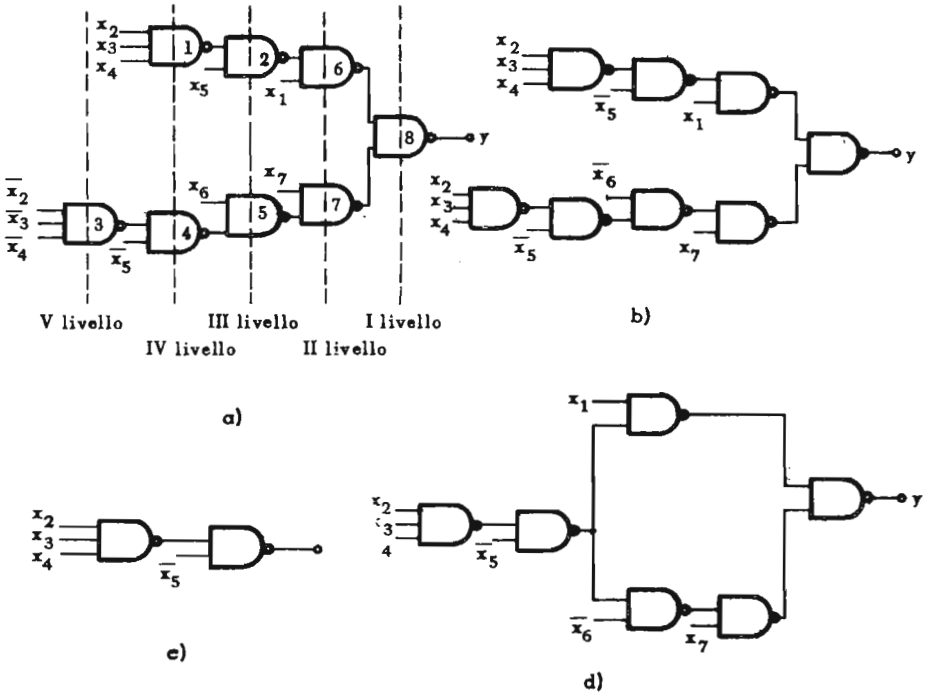


Fig.IV.40 - Sintesi di un circuito NAND.

Il metodo esposto è molto semplice e permette di eliminare i blocchi in comune, senza tener conto dei livelli su cui compaiono. Ricorrendo a un artificio, è anche possibile, senza alcuna trasformazione algebrica, la realizzazione delle funzioni scritte come prodotto di somme.

Esempio 3: *Circuito NAND per la funzione:*

$$y = (\overline{x}_1 x_2 + x_3)(x_4 + \overline{x}_5)\, x_6 \ .$$

Scrivendo la y nella forma:

$$y = (\overline{x}_1 x_2 + x_3)(x_4 + \overline{x}_5)\, x_6 + 0 \ .$$

Si può costruire il circuito della fig.IV.41b, derivato da quello della fig.IV.41a per complementazione delle variabili sui livelli dispari ed eliminazione dell'ingresso 0.

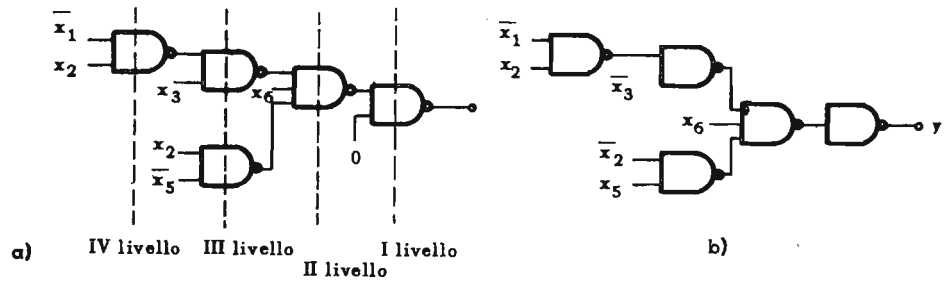


Fig.IV.41 - Sintesi di un circuito NAND.

Poiché, in definitiva, i NAND si comportano esattamente come gli AND e gli OR, rimane valido quanto detto a proposito di questi ultimi elementi. In particolare, il circuito a minor numero di livelli si ottiene sempre nella forma minima come somma di prodotti. I criteri di semplificazione non vengono cambiati dall'aver definitio come forma minima quella a minor numero di elementi perché, almeno per i circuiti a una sola uscita e con le variabili dirette e negate in ingresso, tale forma coincide con quella a minor numero di lettere.

Gli esempi di sintesi dati al par.IV.2 sono, pertanto, tutti applicabili ai circuiti NAND, quando si sostituiscono questi elementi agli OR e agli AND e si complementano le variabili che entrano sui livelli dispari.

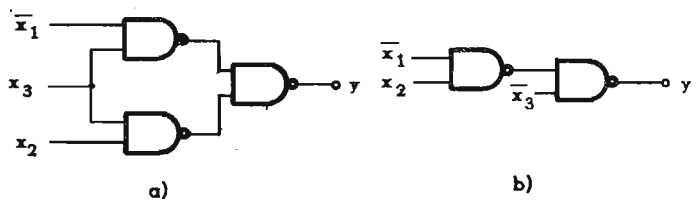


Fig.IV.42 - Due circuiti NAND per le funzioni $\overline{x}_1 x_3 + x_2 x_3$ (a) e $x_3(\overline{x}_1 + x_2)$ (b).

A titolo d'esempio, i due circuiti della fig.IV.28, derivati dalle equazioni:

$$y = \overline{x}_1 x_3 + x_2 x_3 \qquad y = x_3(\overline{x}_1 + x_2)$$

si trasformano, rispettivamente, nei circuiti NAND della fig.IV.42a e IV.42b.

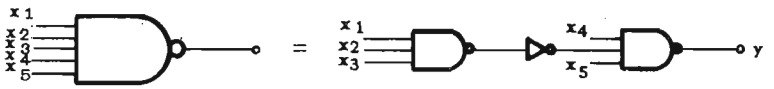


Fig.IV.43 - Conseguenze della non-associatività dell'operatore NAND.

Se qualcuno dei NAND cui si perviene nel corso della sintesi ha un numero di ingressi superiore a quello ammissibile, non lo si può semplicemente spezzare in due, come fatto per gli AND-OR, perché la funzione NAND non è associativa. Se non si possono adottare soluzioni tecniche particolari (ad esempio, nella logica RTL, l'unione dei collettori in parallelo), occorre ricorrere alla proprietà:

$$x_1/x_2/x_3/x_4/\ldots/x_n = \overline{(x_1/x_2/x_3)}/x_4 \ldots x_n$$

per esempio, con lo schema della fig.IV.43.

IV.8.2 - Sintesi dei circuiti NAND quando non si dispone delle variabili negate.

Poiché i metodi analitici di semplificazione trattano allo stesso modo le variabili affermate e negate, mentre i circuiti invertitori hanno un costo paragonabile a quello dei NAND, non conviene realizzare i circuiti NAND semplicemente invertendo le variabili d'ingresso. Non esistendo, d'altra parte, nessun metodo semplice e generale di minimizzazione, non si può che partire dalla forma minima – o da una molto semplificata – e usare degli artifici per far entrare le variabili dirette sui livelli pari e le negate sui livelli dispari, in modo che il cambiamento dei segni elimini gli invertitori. Questi artifici modificano spesso profondamente la forma della funzione, e non di rado comportano aumenti di livelli e di NAND. La scelta del circuito minimo va effettuata – sempre – confrontando un certo numero di circuiti ottenuti per vie diverse, tutte derivanti dai due procedimenti di:

a) separazione delle variabili dirette da quelle negate, per farle entrare su livelli diversi;

b) aggiunta di lettere ridondanti, per creare elementi comuni o stabilire itinerari differenti.

Esempio 1: *Sintesi di un circuito per la funzione*

$$y = x_1(\overline{x}_2 + \overline{x}_3 + x_4) + \overline{x}_2 x_3 x_4 \ .$$

La y è già nella forma minima; nei due termini sono mescolate variabili di segno opposto. Realizzando il circuito normalmente (fig.IV.44a), occorrono 4 NAND e 2 invertitori; scrivendo i due termini in forma diversa:

$$x_1 (\overline{x}_2 + \overline{x}_3 + x_4) = x_1 (\overline{x}_2 + \overline{x}_3) + x_1 x_4$$

$$\overline{x}_2 x_3 x_4 = (\overline{x}_2 + \overline{x}_3)\, x_3 x_4$$

si ottiene invece un circuito (fig.IV.44b) con 5 NAND ma senza invertitori.

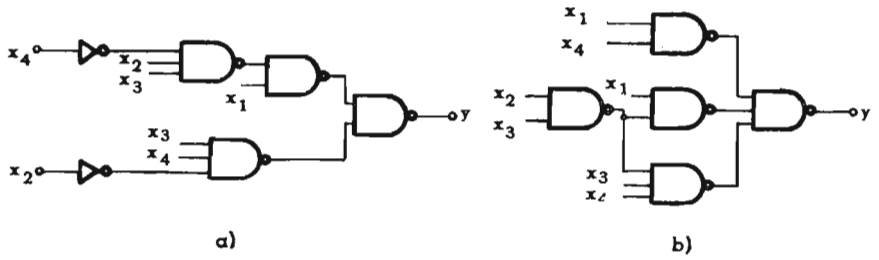


Fig.IV.44 - Circuiti NAND equivalenti.

Esempio 2: *Sintesi di un circuito NAND per la funzione in forma minima:*

$$y = (\overline{x}_1 + \overline{x}_3)\, x_2 + (\overline{x}_2 + \overline{x}_3)\, x_1 \ .$$

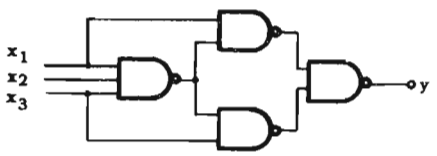
Aggiungendo i termini ridondanti $\overline{x}_2$ e $\overline{x}_1$, rispettivamente nella prima e seconda parentesi, si ha:

$$y = (\overline{x}_1 + \overline{x}_2 + \overline{x}_3)\, x_2 + (\overline{x}_1 + \overline{x}_2 + \overline{x}_3)\, x_1$$

il termine tra parentesi è ripetuto due volte e realizzato su un livello dispari. Il circuito, a quattro NAND, è quello della fig.IV.45.

Fig.IV.45 - Circuito NAND minimo per la funzione:

$(\overline{x}_1 + \overline{x}_3)\, x_2 + (\overline{x}_2 + \overline{x}_3)\, x_1$.



Esempio 3: *Sintesi di un circuito per la funzione:*

$$y = \Sigma\,(0, 1, 4, 6, 7, 9, 12, 15) \ .$$

La mappa di Karnaugh dalla y (fig.IV.46) permette di ricavare, come possibile forma minima a due livelli, la:

$$y = \overline{x}_1 \overline{x}_3 \overline{x}_4 + x_2 \overline{x}_3 \overline{x}_4 + \overline{x}_2 \overline{x}_3 x_4 + x_2 x_3 x_4 + \overline{x}_1 x_2 x_3 \ .$$

realizzabile con 6 NAND e 4 invertitori (fig.IV.47a); una soluzione migliore si ha scrivendo la y nella forma fattorizzata:

$$y = \overline{x}_2\overline{x}_3x_4 + x_2x_3(\overline{x}_1 + x_4) + \overline{x}_3\overline{x}_4(\overline{x}_1 + x_2) \ ,$$

cui corrisponde il circuito della fig.IV.47b, a 6 NAND e 3 invertitori.

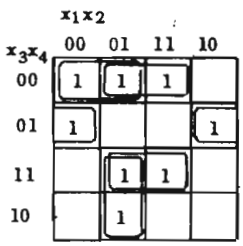


Fig.IV.46 - Mappa di Karnaugh per la funzione $y = \sum(0, 1, 4, 6, 7, 9, 12, 15)$.

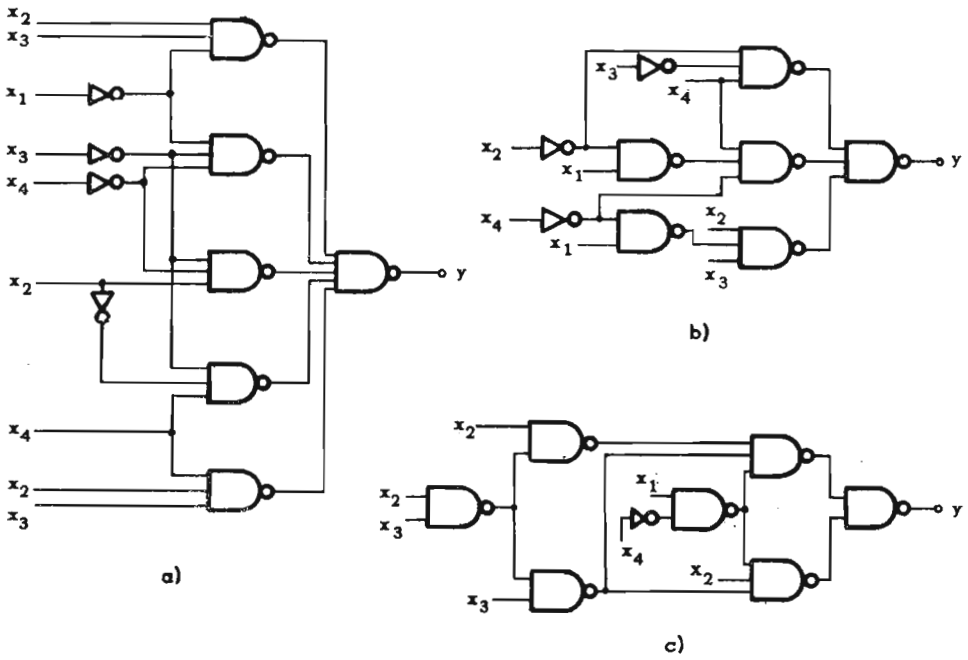


Fig.IV.47 - Circuiti NAND per la funzione di fig.IV.46.

La soluzione minima si ottiene, comunque, a partire da una diversa forma minima a 2 livelli, la:

$$y = \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3 + \overline{x}_2\overline{x}_3x_4 + x_2\overline{x}_3\overline{x}_4 + x_2x_3x_4$$

che si ricava dalla mappa della fig.IV.46, legando il minterm 0000 con 0001 invece che con 0100. Scrivendo la y nella forma fattorizzata:

$$y = \overline{x}_3(\overline{x}_1\overline{x}_2 + \overline{x}_2x_4) + x_2(\overline{x}_1x_3 + x_3x_4) + x_2\overline{x}_3\overline{x}_4$$

e trasformando quest'espressione nel modo seguente:

$$y = (x_2 + \overline{x}_3)(\overline{x}_1\overline{x}_2 + \overline{x}_2x_4 + \overline{x}_1x_3 + x_3x_4) + x_2\overline{x}_3\overline{x}_4 =$$

$$= (x_2x_3 + \overline{x}_3)(\overline{x}_1\overline{x}_2 + \overline{x}_1x_3 + \overline{x}_2x_4 + x_3x_4) + x_2\overline{x}_4(\overline{x}_2 + \overline{x}_3) =$$

$$= (\overline{x}_3 + x_2x_3)(\overline{x}_1\overline{x}_2 + \overline{x}_1x_2x_3 + \overline{x}_2x_4 + x_2x_3x_4) + x_2\overline{x}_4(\overline{x}_2 + \overline{x}_3) =$$

$$= (\overline{x}_3 + x_2x_3)(\overline{x}_1 + x_4)(\overline{x}_2 + x_2x_3) + x_2\overline{x}_4(\overline{x}_2 + \overline{x}_3) \ ,$$

si ottiene il circuito a 7 NAND e un invertitore della fig.IV.47c.

Come si vede, gli artifici da applicare sono talvolta piuttosto complessi; non sempre, tuttavia, si riesce a eliminare del tutto gli invertitori. Qualsiasi circuito per la funzione:

$$y = x_1x_2\overline{x}_3 + \overline{x}_1\overline{x}_2\overline{x}_3$$

ad esempio, avrà sempre bisogno degli invertitori per tutte le variabili d'ingresso (figura IV.48).

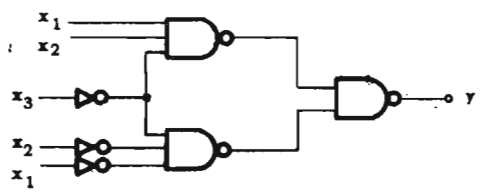


Fig.IV.48 - Circuito NOT--NAND per la funzione:

$y = x_1x_2\overline{x}_3 + \overline{x}_1\overline{x}_2\overline{x}_3$.

Prima di concludere la sintesi a NAND, vogliamo accennare a una semplificazione di tipo particolare, consistente nell'ottenere la funzione da realizzare su più di un terminale (*bundling*).

Supponiamo di dover realizzare un circuito la cui uscita:

$$y = (\overline{x}_1 + \overline{x}_2)(\overline{x}_3 + \overline{x}_4)$$

sia da collegare, insieme con quella y' di un secondo circuito, all'ingresso di un ulteriore NAND, per realizzare (fig.IV.49a) la funzione:

$$Y = \overline{yy'} = \overline{y} + \overline{y}' = x_1x_2 + x_3x_4 + \overline{y}' \ .$$

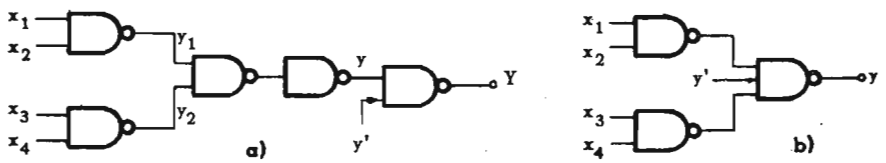


Fig.IV.49 - Circuiti NAND equivalenti.

In questo caso, la Y si ottiene più semplicemente con lo schema della fig.IV.49b; prendendo le uscite del circuito sui terminali y_1 e y_2, invece che su y si eliminano così 2 NAND e 2 livelli.

IV.9 - Sintesi dei circuiti con elementi NOR.

Un circuito NOR origina una funzione in forma di prodotto di somme, comportandosi ogni NOR – a parte il segno delle variabili – come un OR o come un AND, a seconda della posizione del NOR stesso lungo gli itinerari che lo collegano all'uscita.

Si può quindi costruire un circuito NOR seguendo le stesse regole di un AND-OR e invertendo le variabili che entrano nei livelli dispari, purché la funzione che il circuito deve realizzare sia stata scritta come prodotto di somme (v. cap.II).

Esempio 1: *Sintesi di un circuito NOR per la funzione rappresentata nella fig.IV.50.*

Per ricavare la forma minima della y come prodotto di somme, riportiamo su una mappa di Karnaugh (fig.IV.51) gli zeri della y stessa.

Dall'espressione minima della funzione $\overline{y}$:

$$\overline{y} = \overline{x}_1\overline{x}_3 + \overline{x}_1 x_2 + x_1\overline{x}_2 = x_1\overline{x}_2 + \overline{x}_1(x_2 + \overline{x}_3) \quad .$$

Si ottiene la y come prodotto di somme:

$$y = \overline{\overline{y}} = (\overline{x}_1 + x_2)(x_1 + \overline{x}_2 x_3) \quad .$$

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Fig.IV.50 - Tabella di verità di una funzione di 3 variabili.

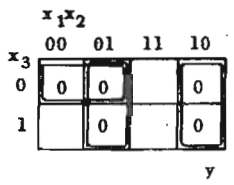


Fig.IV.51 - Rappresentazione sulla mappa di Karnaugh degli zeri della funzione di fig.IV.50.

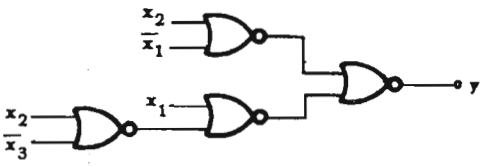


Fig.IV.52 - Circuito NOR per la funzione di figura IV.50.

Nella fig.IV.52 è mostrato il circuito NOR per la y : le variabili $\overline{x}_2$ e x_3, che entrano su un livello dispari, sono state invertite.

In certi casi, conviene effettuare la sintesi tenendo presente che un circuito NOR ha una funzione d'uscita duale di quella di un circuito in cui ogni NOR è sostituito da un NAND.

Esempio 2: *Costruire un circuito NOR più economico di quello della fig.IV.53.*

Se il circuito fosse realizzato con NAND, invece che con NOR, avrebbe una funzione d'uscita y', duale della y, esprimibile nella forma:

$$y' = (\overline{x}_4 + x_1 x_2)(\overline{x}_1 + \overline{x}_5) + x_3(\overline{x}_1 + \overline{x}_2) + (\overline{x}_1 + \overline{x}_5)(\overline{x}_3 + \overline{x}_4) \quad .$$

La y', sviluppata e semplificata con semplici passaggi, diventa:

$$y' = \overline{x}_1 + \overline{x}_5 (x_2 + \overline{x}_3 + \overline{x}_4) + \overline{x}_2 x_3 \ .$$

Questa funzione può essere ottenuta con soli 4 NAND (fig. IV.54); pertanto la sua duale y, che non è necessario nemmeno scrivere, è realizzata col circuito a 4 NOR della fig.IV.54b.

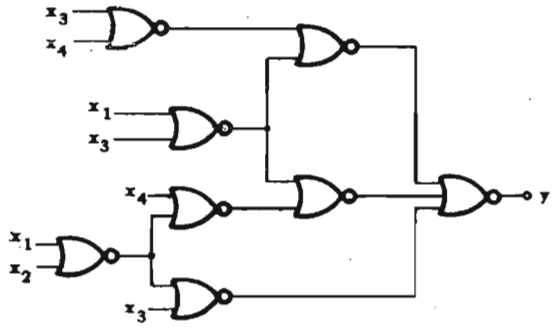


Fig.IV.53 - Circuito NOR a 4 livelli e 8 componenti.

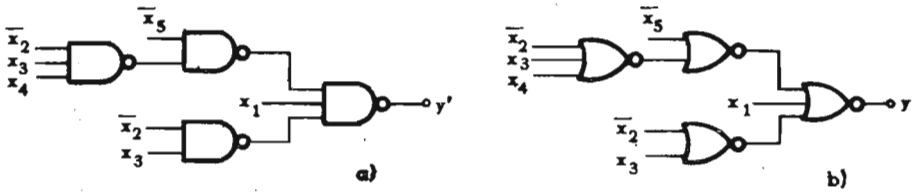


Fig.IV.54 - Circuito NOR equivalente al circuito della fig.IV.53.

IV.10 - Studio di un problema di sintesi.

Per riassumere e applicare a un problema concreto tutte le tecniche esposte nel presente capitolo, effettueremo la sintesi di un circuito (l'addizionatore) particolarmente importante nei calcolatori, usando tutti i possibili componenti.

Nella fig.IV.55 è mostrato il simbolo di un circuito per sommare i 2 bit A e B, secondo le regole esposte nel cap.I. S e C sono le due uscite del circuito, rispettivamente di peso 1 e 2, rappresentanti la somma S e il riporto C (iniziale del termine *carry*). Il circuito si chiama semi-addizionatore (Half-Adder), perché può essere usato per i soli bit meno significativi di due numeri da sommare.

Nella fig.IV.56 è mostrata la tabella di verità delle funzioni S e C.

Fig.IV.55 - Schema a blocchi di un Half-Adder.

Le equazioni delle due uscite del circuito sono:

$$S = A\bar{B} + \bar{A}B = A \oplus B \qquad \text{(IV.10)}$$
$$C = AB \ .$$

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Fig.IV.56 - Tabella di verità della somma (S) e del riporto (C) in funzione degli ingressi A e B dell'HA.

Le (IV.10), se si hanno a disposizione i bit A e B anche in forma negata, danno luogo ai circuiti AND-OR, NAND e wired-OR in DTL positiva, rispettivamente mostrati nelle figg. IV.57a, b, c.

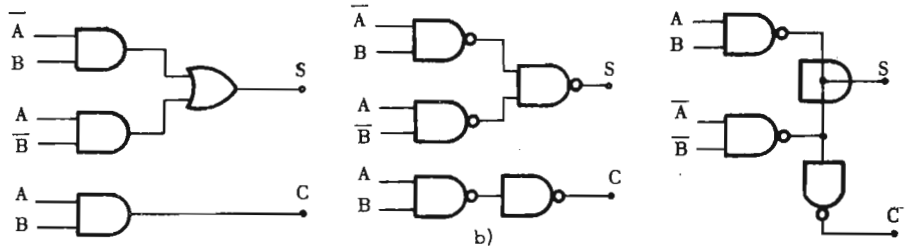


Fig.IV.57 - Circuiti HA.

Per un circuito NOR, conviene scrivere la prima delle (IV.10) nella forma:

$$S = (\bar{A} + \bar{B})(A + B) \qquad \text{(IV.11)}$$

si ottiene così un circuito NOR con tre soli elementi (fig.IV.58); una diversa versione con lo stesso numero di elementi è quella wired-OR in DTL negativa della fig.IV.58b.

Se non si dispone di $\bar{A}$ e $\bar{B}$, il circuito più economico AND-OR-NOT (fig.IV.59) si ottiene, scrivendo la (IV.11) nella forma:

$$S = (\bar{A} + \bar{B})(A + B) = \overline{AB}\,(A + B) \ .$$

Il circuito minimo a NAND (fig.IV.60) si ottiene invece trasformando la (IV.11) in:

$$S = (\bar{A} + \bar{B})\,A + (\bar{A} + \bar{B})\,B \ .$$

Il circuito NOR più conveniente è ancora quello della fig.IV.58, con l'aggiunta di 2 invertitori agli ingressi.

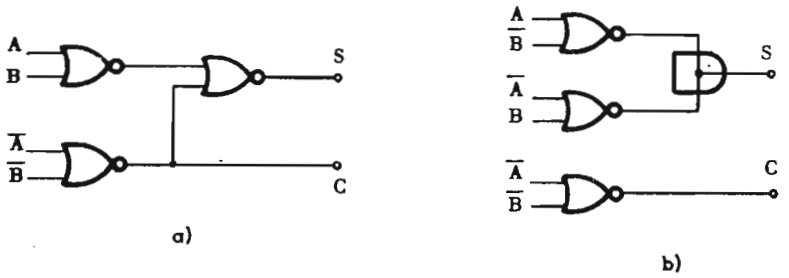


Fig.IV.58 - Circuiti HA.

Come già detto, il semiaddizionatore può sommare solo i due bit meno significativi di due numeri a n bit; per le restanti (n - 1) posizioni occorre un circuito a tre ingressi, che tenga conto del riporto dalla posizione precedente (addizionatore completo o Full Adder). Nella fig.IV.61a è mostrato il simbolo di un FA e, nella fig.IV.61b lo schema di un addi-

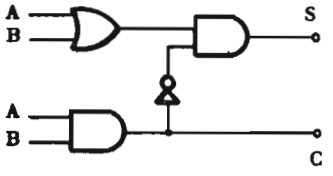


Fig.IV.59 - Circuito HA.

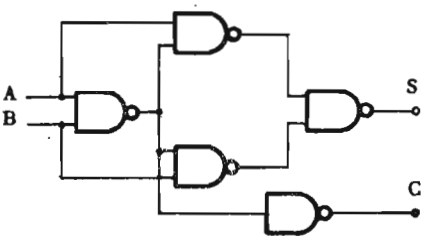


Fig.IV.60 - Circuito HA.

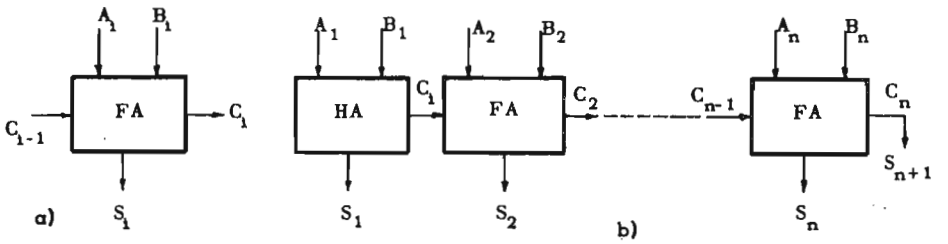


Fig.IV.61 - Schema a blocchi di un Full Adder (a) e di un addizionatore parallelo (b).

zionatore binario in parallelo, costruito con un HA e (n - 1) FA. Nella figura IV.61b, A_i e B_i sono i bit i^{mi} dei numeri A e B da addizionare; A_1 e B_1 quelli meno significativi; C_1 è il riporto dell'addizione $A_1 + B_1$;

C_i ($i \neq 1$) è il riporto di $A_i + B_i + C_{i-1}$; S_1 è il risultato di $A_1 + B_1$; S_i ($i \neq 1$) è il risultato di $A_i + B_i + C_{i-1}$. I due numeri da sommare si presentano insieme all'ingresso del circuito; l'uscita, che costituisce il risultato dell'addizione, ha $n+1$ bit essendo l'$(n+1)^{mo}$ bit il riporto (eventualmente 0) dell'ultimo FA.

La tabella di verità delle funzioni d'uscita C_i ed S_i, in funzione di A_i, B_i e C_{i-1} è riportata nella fig.IV.62; il suo significato è ovvio se si tiene conto che $A_iB_iC_{i-1}$ hanno tutti peso 1, mentre S_i e C_i hanno – – rispettivamente – pesi 1 e 2.

A_1	B_1	C_{1-1}	S_1	C_1
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig.IV.62 - Tabella di verità della somma (S_1) e del riporto C_1 di uno stadio del FA, in funzione di A_1, B_1 e C_{1-1}.

Le espressioni minime di C_i ed S_i, ricavate dalle mappe di Karnaugh della fig.IV.63, sono:

$$(IV.12)\quad S_i = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1} \quad (= A_i \oplus B_i \oplus C_{i-1})$$

$$(IV.13)\quad C_i = A_iB_i + A_iC_{i-1} + B_iC_{i-1} \ .$$

Se si dispone anche delle variabili negate, un buon circuito AND-OR-NOT (fig.IV.64) si ottiene scrivendo la (IV.12) e la (IV.13) nella forma:

$$S_i = C_{i-1}(\overline{A}_i\overline{B}_i + A_iB_i) + \overline{C}_{i-1}(\overline{A}_iB_i + A_i\overline{B}_i) =$$

$$= C_{i-1}(\overline{A}_i\overline{B}_i + A_iB_i) + \overline{C}_{i-1}\,\overline{(\overline{A}_i\overline{B}_i + A_iB_i)}$$

$$C_i = A_iB_i + C_{i-1}(A_i + B_i) \ .$$

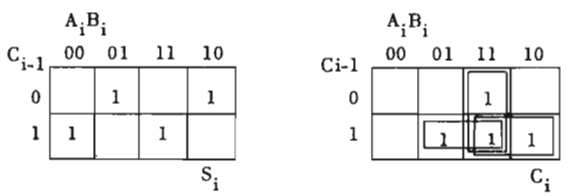


Fig.IV.63 - Mappe di Karnaugh delle funzioni S_1 e C_1.

Il circuito è però a 5 livelli; quello minimo a 2 livelli deriva direttamente dalla (IV.12) e dalla (IV.13), ed è formato da 7 AND, 2 OR e 25 diodi.

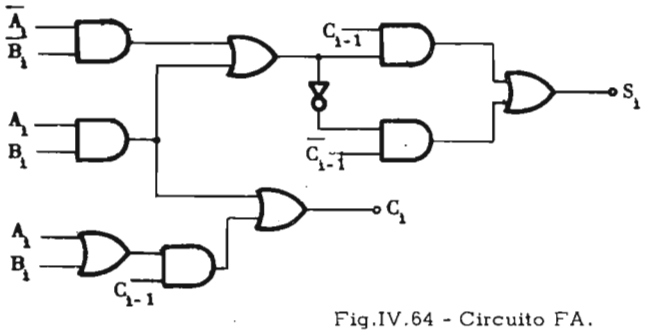


Fig.IV.64 - Circuito FA.

Per il circuito a NAND, conviene scrivere la (IV.12) e la (IV.13) nella forma:

$$S_i = C_{i-1}(A_iB_i + \overline{A}_i\overline{B}_i) + \overline{C}_{i-1}(\overline{A}_i + \overline{B}_i)(A_i + B_i)$$

$$C_i = A_iB_i + C_{i-1}(A_i + B_i) \ .$$

Mettendo in comune i NAND con gli stessi ingressi, si ottiene il circuito della fig.IV.65, a 4 livelli e 8 NAND.

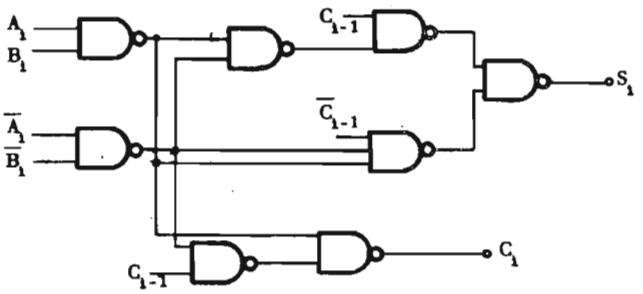


Fig.IV.65 - Circuito FA.

Per ricavare il circuito minimo a NOR, scriviamo le funzioni duali di S_i e C_i:

$$S_i' = (\overline{A}_i + \overline{B}_i + C_{i-1})(\overline{A}_i + B_i + \overline{C}_{i-1})(A_i + B_i + C_{i-1})(A_i + \overline{B}_i + \overline{C}_{i-1}) =$$

$$= (\overline{A}_i + \overline{B}_i\overline{C}_{i-1} + B_iC_{i-1})(A_i + B_i\overline{C}_{i-1} + \overline{B}_iC_{i-1}) =$$

$$= \overline{A}_iB_i\overline{C}_{i-1} + \overline{A}_i\overline{B}_iC_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} \equiv S_i$$

$$C_i' = (A_i + B_i)(A_i + C_{i-1})(B_i + C_{i-1}) =$$

$$= (A_i + B_iC_{i-1})(B_i + C_{i-1}) = A_iB_i + A_iC_{i-1} + B_iC_{i-1} \equiv C_i \ .$$

Per essere $S_i' = S_i$ e $C_i' = C_i$, le due funzioni S_i e C_i sono autoduali, hanno cioè la proprietà di trasformarsi in se stesse quando si scambiano i segni di prodotto e di somma. Il circuito NOR ha, dunque, la stessa configurazione di quello NAND.

Se non si dispone di A_i, B_i, C_{i-1}, il circuito minimo AND-OR-NOT è ancora quello della fig.IV.64, con tre invertitori in più. Il circuito minimo a NAND si ottiene invece scrivendo le equazioni (IV.12) e (IV.13) nelle forme di seguito riportate, miranti a far entrare le variabili dirette sui livelli pari e le negate sui dispari.

(IV.14)
$$\begin{aligned} S_i &= \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_iB_iC_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} = \\ &= A_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + C_{i-1}(\overline{A}_i\overline{B}_i + A_iB_i) = \\ &= A_i\overline{B}_i\overline{C}_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + C_{i-1}(\overline{A}_i\overline{B}_i + A_iB_i + \overline{C}_{i-1}) = \\ &= A_i\overline{B}_i(\overline{A}_i + B + \overline{C}_{i-1}) + \overline{A}_iB_i(A_i + \overline{B}_i + \overline{C}_{i-1}) + \\ &+ C_{i-1}(A_i + \overline{B}_i + \overline{C}_{i-1})(\overline{A}_i + B_i + \overline{C}_{i-1}) = \\ &= A_i(\overline{A}_i + \overline{B}_i)(\overline{A}_i + A_iB_i + \overline{C}_{i-1}) + B_i(\overline{A}_i + \overline{B}_i)(A_iB_i + \\ &+ \overline{B}_i + \overline{C}_{i-1}) + C_{i-1}(A_iB_i + \overline{B}_i + \overline{C}_{i-1})(\overline{A}_i + A_iB_i + \overline{C}_{i-1}) \end{aligned}$$

(IV.15)
$$\begin{aligned} C_i &= A_iB_i + \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} = \\ &= A_iB_i + B_iC_{i-1}(\overline{A}_i + \overline{B}_i) + A_iC_{i-1}(\overline{A}_i + \overline{B}_i) \quad . \end{aligned}$$

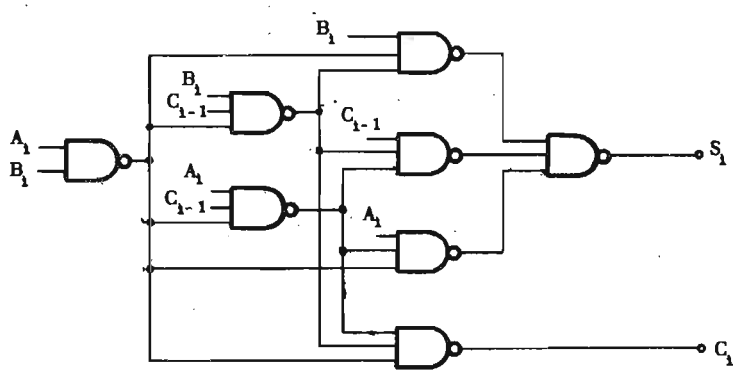


Fig.IV.66 - Circuito FA.

Le espressioni (IV.14) e (IV.15) conducono al circuito minimo della fig.IV.66, realizzato con 8 NAND. Il circuito a NOR si ottiene semplicemente sostituendo i NOR ai NAND.

IV.11 - Circuiti multiterminali.

Un circuito multiterminale MT (fig.IV.67) è un circuito a n ingressi ed m uscite, su ognuna delle quali si realizza una diversa funzione y_i delle variabili $x_1 x_2 \ldots x_n$.

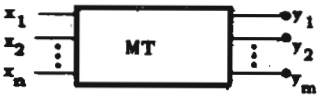


Fig.IV.67 - Schema a blocchi di un Circuito Multiterminale (MT).

Nell'esempio 1 del par.IV.6 abbiamo effettuato la sintesi di un semplice circuito multiterminale a 3 ingressi e 4 uscite, trattando separatamente le funzioni y_1, y_2, y_3 e y_4 e mettendo in comune gli elementi uguali. Questo procedimento può essere, ovviamente, usato in ogni caso, come mostra l'esempio seguente.

Esempio 1: *Circuito MT a NAND per la conversione di un numero a 4 bit espresso in codice binario normale nel corrispondente numero nel codice di Gray.*

Numero decimale	B_1	B_2	B_3	B_4	G_1	G_2	G_3	G_4
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Fig.IV.68 - Tabella di verità per la conversione di un numero a 4 bit in codice binario normale nel corrispondente numero in codice di Gray.

Siano B_i ($i = 1 \ldots 4$) i bit, in codice binario, del numero da convertire, con B_1 bit più significativo, e G_i i bit del numero espresso nel codice di Gray. La tabella di verità (fig.IV.68) del circuito di ingressi B_i e uscita G_i è ricavata dalle definizioni stesse dei due codici (per maggior chiarezza, sono stati riportati i valori decimali corrispondenti a ogni numero binario). Dalle tabelle stesse, è possibile ricavare subito la prima funzione d'uscita:

$$G_1 = B_1 \ .$$

Le mappe di Karnaugh delle altre uscite (fig.IV.69) danno poi le equazioni di G_2, G_3, G_4:

$$G_2 = \overline{B}_1 B_2 + B_2 \overline{B}_1 = B_1 \oplus B_2$$

$$G_3 = B_2 \overline{B}_3 + B_3 \overline{B}_2 = B_2 \oplus B_3$$

$$G_4 = B_3 \overline{B}_4 + \overline{B}_3 B_4 = B_3 \oplus B_4 \ .$$

Nella fig.IV.70 è mostrato il circuito completo, in due versioni: la prima a 12 NAND, la seconda a 6 NAND e 4 invertitori, realizzata con il wired-or delle uscite.

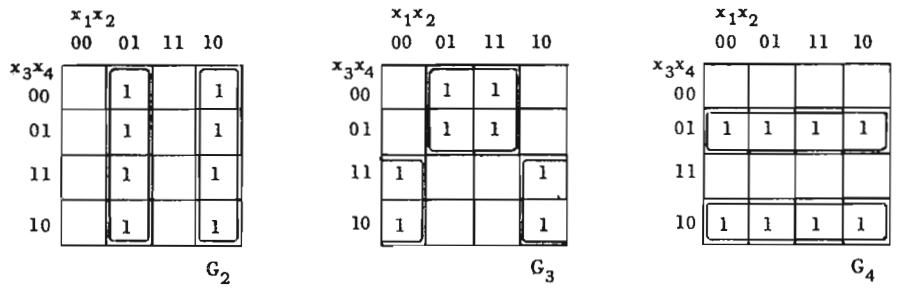


Fig.IV.69 - Mappe di Karnaugh derivate dalle tabelle di verità di fig.IV.68.

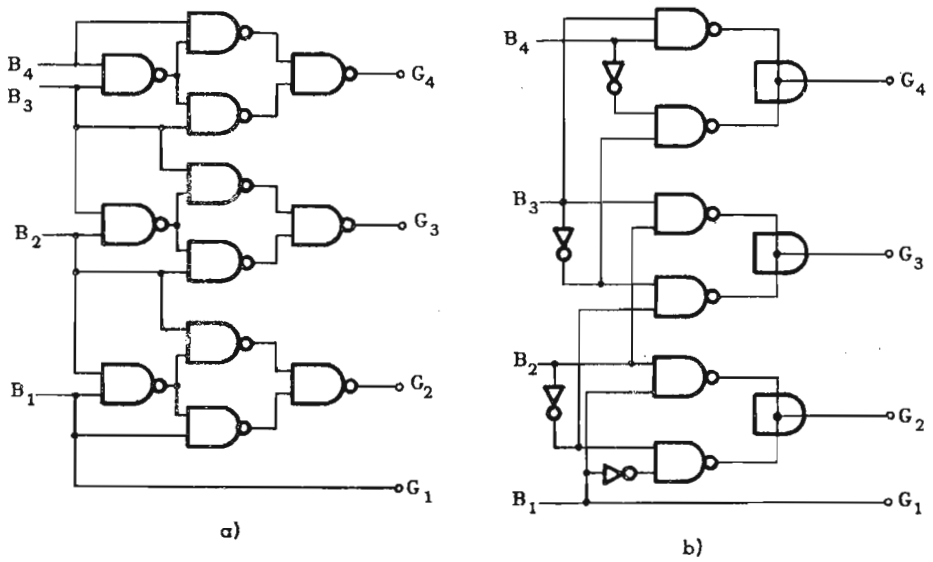


Fig.IV.70 - Circuiti MT per la conversione binario normale-Gray.

È possibile, procedendo in questo modo, progettare circuiti abbastanza economici: talvolta, però, nelle semplificazioni per ottenere le singole funzioni, vengono eliminati dei termini, la cui considerazione avrebbe portato a circuiti migliori.

Esempio 2: *Circuito MT AND-OR a due livelli per le funzioni:*

$$y_1 = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + x_1x_2x_3$$
$$y_2 = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2\overline{x}_3 + \overline{x}_1x_2x_3$$
$$y_3 = x_1\overline{x}_2\overline{x}_3 + x_1\overline{x}_2x_3 + x_1x_2\overline{x}_3 + x_1x_2x_3 + \overline{x}_1x_2x_3 \ .$$

La forma minima delle y_1, come mostra la fig.IV.71, è:

$$y_1 = x_1x_2x_3 + \overline{x}_2\overline{x}_3$$
$$y_2 = \overline{x}_1x_2x_3 + \overline{x}_2\overline{x}_3$$
$$y_3 = x_1 + x_2x_3 \ .$$

Le y_1 contengono un solo termine in comune ($\overline{x}_2\overline{x}_3$ in y_1 e y_2); il circuito MT (figura IV.72) è realizzato con 7 elementi e 16 diodi.

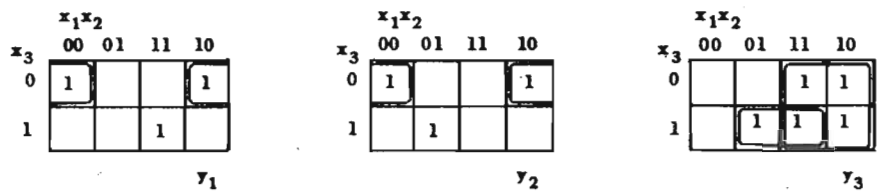


Fig.IV.71 - Mappe di Karnaugh di un circuito MT.

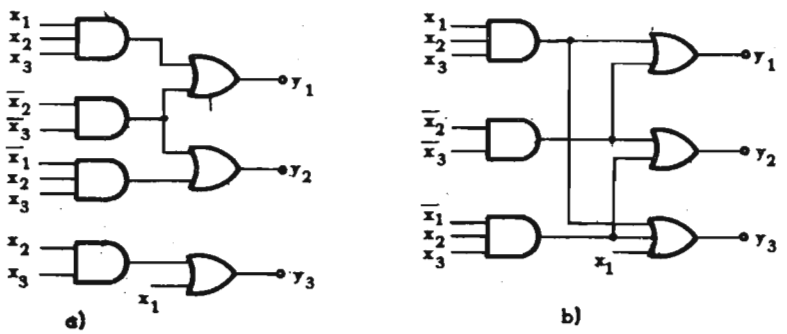


Fig.IV.72 - Circuiti MT derivati dalle mappe di fig.IV.71.

Un circuito più economico, a 6 elementi e 15 diodi (fig.IV.72b) si sarebbe ottenuto se non si fossero semplificati i minterm $x_1x_2x_3$ e $\overline{x}_1x_2x_3$ in y_3:

$$y_3 = x_1 + x_1x_2x_3 + \overline{x}_1x_2x_3 \ .$$

Il problema nuovo dei circuiti MT è appunto l'individuazione di questi termini comuni: nei prossimi due paragrafi mostreremo come tale problema modifichi i metodi di Karnaugh e Mc Cluskey. Illustreremo infine dei circuiti MT di configurazione particolare e di larghissimo impiego pratico (le matrici) progettati con metodi del tutto diversi.

IV.12 - Il metodo di Karnaugh per la semplificazione dei circuiti MT.

Nella semplificazione dei circuiti MT AND-OR occorre tener conto che le forme delle y_i corrispondenti al circuito col minor numero di elementi non sempre coincidono con quelle che minimizzano il numero totale dei diodi. I due casi vanno, pertanto, trattati separatamente. La sintesi dei circuiti NAND o NOR è, ovviamente, deducibile dal procedimento che minimizza il numero degli elementi.

Fino a 5 variabili, il metodo più rapido, anche se non del tutto rigoroso, è quello di Karnaugh così modificato:

a) si riporta ognuna delle y_i su una mappa;

b) si segnano, su ogni mappa, gli insiemi che portano alla forma minima;

c) si confrontano tutti gli insiemi segnati, mettendo in comune quelli che compaiono su almeno due mappe, e spezzando quelli che possono essere ottenuti come somma di più insiemi esistenti in altre mappe;

d) si scrivono, per ogni uscita, le due espressioni fattorizzate derivate da quelle a due livelli cui conducono gli insiemi considerati ai passi b) e c), ricavando per confronto il circuito più economico.

Nell'effettuare i confronti tra le espressioni del tipo b) e c), è utile la seguente regola pratica: *il numero totale dei diodi di un circuito MT è uguale alla somma di tutte le lettere che compaiono nelle espressioni delle varie funzioni, contando una sola volta le lettere dei termini in comune, più la somma di tutti i termini aventi almeno due lettere.*

Ad esempio, per realizzare il circuito MT per le funzioni $y_1 y_2 y_3$:

$$\begin{cases} y_1 = x_2x_3 + \bar{x}_2\bar{x}_3x_4 + x_1x_3x_4 \\ y_2 = x_2 + x_1x_3\bar{x}_4 \\ y_3 = x_1x_3x_4 + x_2\bar{x}_3 + x_1x_3\bar{x}_4 + \bar{x}_2\bar{x}_3x_4 \end{cases}$$

occorrono tanti diodi quante sono le lettere di tutti i termini, contati una sola volta, in y_1, y_2, y_3 (14), più il numero di termini di almeno due lettere (8): in tutto 22 diodi.

Esempio 1: *Sintesi di un circuito MT a 4 ingressi e 3 uscite per le funzioni:*

$$y_1 = \Sigma\,(0, 4, 6, 7, 10, 14, 15)$$
$$y_2 = \Sigma\,(4, 5, 6, 7, 11, 12, 13, 14, 15)$$
$$y_3 = \Sigma\,(0, 1, 4, 5, 9, 10, 11, 13, 14, 15)\ .$$

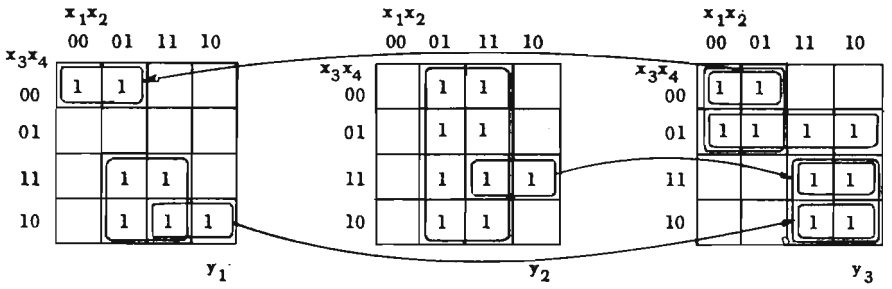


Fig.IV.73 - Mappe di Karnaugh per un circuito MT.

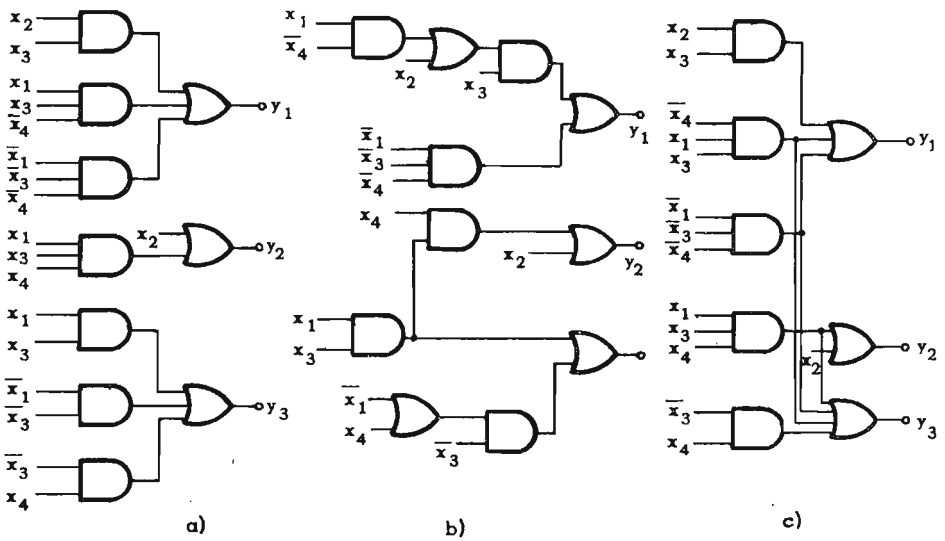


Fig.IV.74 - Circuiti MT derivati dalle mappe di fig.IV.73.

Nella fig.IV.73 sono riportate le mappe di Karnaugh per le y_i; le espressioni minime a due livelli, ottenute considerando separatamente ognuna delle tre funzioni sono:

(IV.16)
$$\begin{cases} y_1 = x_2x_3 + x_1x_3\overline{x}_4 + \overline{x}_1\overline{x}_3\overline{x}_4 \\ y_2 = x_2 + x_1x_3x_4 \\ y_3 = x_1x_3 + \overline{x}_1\overline{x}_3 + \overline{x}_3x_4\ . \end{cases}$$

Il relativo circuito a 2 livelli (fig.IV.74a) impiega 10 elementi (7 AND e 3 OR) e 25 diodi.

Le (IV.16) in forma fattorizzata:

$$y_1 = x_3 (x_2 + x_1\overline{x}_4) + \overline{x}_1\overline{x}_3\overline{x}_4$$
$$y_2 = x_2 + (x_1x_3)\, x_4$$
$$y_3 = \overline{x}_3 (\overline{x}_1 + x_4) + x_1x_3$$

danno invece origine al circuito della fig.IV.74b), a 4 livelli, 11 elementi e 22 diodi.

Spezzando, infine, gli insiemi $\overline{x}_1\overline{x}_3$ e x_1x_3 di y_3 nel modo indicato dalle frecce nella fig.IV.73, le y_1 si scrivono nella forma non fattorizzabile:

$$\text{(IV.17)} \qquad \begin{cases} y_1 = x_2x_3 + x_1x_3\overline{x}_4 + \overline{x}_1\overline{x}_3\overline{x}_4 \\ y_2 = x_2 + x_1x_3x_4 \\ y_3 = \overline{x}_3x_4 + x_1x_3x_4 + x_1x_3\overline{x}_4 + \overline{x}_1\overline{x}_3\overline{x}_4 \end{cases}$$

il relativo circuito (fig.IV.74c) ha 2 livelli, 8 elementi e 22 diodi, quindi è il più economico in ogni senso.

Esempio 2: *Sintesi di un circuito MT a 4 ingressi e 3 uscite per le funzioni:*

$$y_1 = \Sigma\,(3, 4, 7, 11, 12, 13, 15)$$
$$y_2 = \Sigma\,(2, 3, 6, 7, 9, 10, 11, 14, 15)$$
$$y_3 = \Sigma\,(2, 4, 6, 9, 10, 11, 12, 13, 14, 15)\ .$$

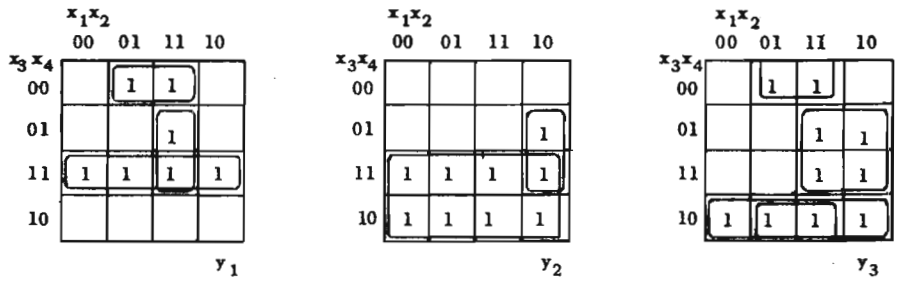


Fig.IV.75 - Mappe di Karnaugh di un circuito MT.

Nella fig.IV.75 sono riportate le mappe di Karnaugh per le y_i; le espressioni minime a due livelli, ottenute considerando separatamente ognuna delle tre funzioni, sono:

$$\text{(IV.18)} \qquad \begin{cases} y_1 = x_3x_4 + x_1x_2x_4 + x_2\overline{x}_3\overline{x}_4 \\ y_2 = x_3 + x_1\overline{x}_2x_4 \\ y_3 = x_3\overline{x}_4 + x_1x_4 + x_2\overline{x}_4\ . \end{cases}$$

Il relativo circuito a 2 livelli comprende 10 elementi e 25 diodi: un circuito più economico, a 9 elementi e 23 diodi (fig.IV.76a), si ottiene spezzando l'insieme $x_2\overline{x}_4$ di y_3 in $x_2\overline{x}_3\overline{x}_4$ (contenuto in y_1), e in $x_2x_3\overline{x}_4$ (contenuto in $x_3\overline{x}_4$ di y_3).

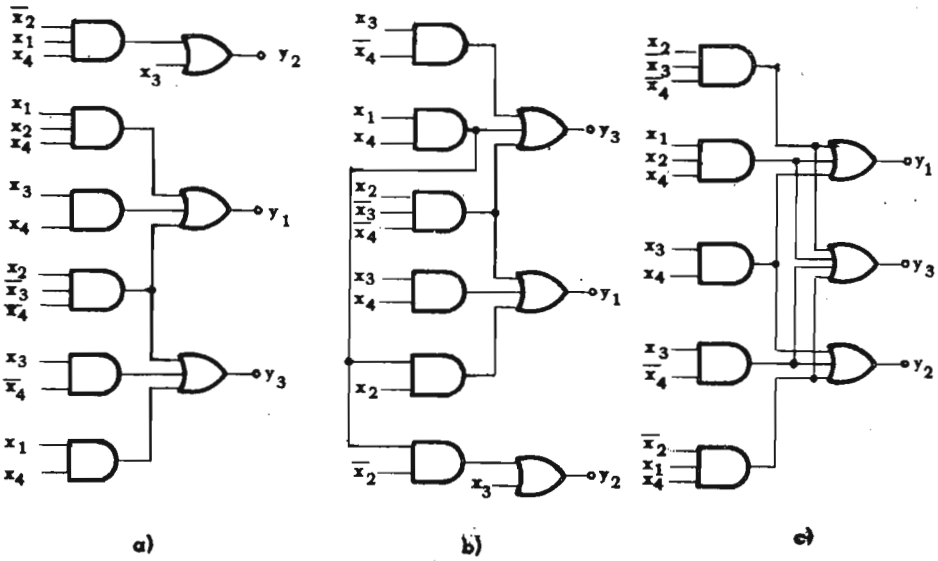


Fig.IV.76 - Circuiti MT derivati dalle mappe di Karnaugh di fig.IV.75.

Il circuito a minor numero di diodi è però quello della fig. IV.76 b, a 3 livelli, 9 elementi e 21 diodi, ottenuto mettendo in comune il termine x_1x_4, cioè scrivendo le (IV.18) nella forma:

$$\text{(IV.19)} \qquad \begin{cases} y_1 = x_2x_3 + (x_1x_4)\,x_2 + x_2\overline{x}_3\overline{x}_4 \\ y_2 = x_3 + (x_1x_4)\,\overline{x}_2 \\ y_3 = x_3\overline{x}_4 + (x_1x_4) + x_2\overline{x}_3\overline{x}_4 \end{cases}.$$

Il circuito a minor numero di elementi, invece, è quello della fig. IV.76 c, a 2 livelli, 8 elementi e 23 diodi, ottenuto dalle (IV.18) spezzando x_3 di y_2 in x_3x_4 e $x_3\overline{x}_4$, termini contenuti rispettivamente in y_1 ed y_3.

IV.13 - Metodo analitico per la semplificazione dei circuiti MT.

Il metodo è una variante di quello di Quine-Mc Cluskey, e permette di ottenere circuiti MT minimi a due livelli. Per semplicità di trattazione, lo esporremo riferendoci ad un circuito particolare, che verrà mi-

nimizzato prima rispetto al numero di elementi, poi di diodi. Tralasceremo eventuali fattorizzazioni che, del resto, intervengono in uno stadio successivo alla semplificazione secondo Quine-Mc Cluskey.

Sia, dunque, da progettare un circuito MT a 4 ingressi e 3 uscite per le funzioni y_1, y_2, y_3, secondo la tabella di verità della fig.IV.77.

x_1	x_2	x_3	x_4	y_1	y_2	y_3
0	0	0	0	0	0	0
0	0	0	1	1	0	1
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	1	1
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Fig.IV.77 - Tabelle di verità di un circuito MT.

La semplificazione avviene in 3 fasi:

a) costruzione della tabella su cui effettuare la ricerca dei primi implicanti.

Si considerano le configurazioni delle x per cui una almeno delle y ha valore 1, e si formano altrettante espressioni aventi come parte numerica le configurazioni stesse e come parte letterale le y eguali ad 1; le y eguali a 0 si segnano con un trattino (la parte letterale indica in quale funzione il minterm corrispondente alla parte numerica è presente).

Ad esempio, l'espressione da considerare per $x_1x_2x_3x_4=0001$, avendosi $y_1y_2y_3 = 101$ sarà:

$$0001\, y_1 - y_3 \ ;$$

e, per $x_1x_2x_3x_4 = 1010$, avendosi $y_1y_2y_3 = 011$:

$$1010 - y_2\, y_3 \ .$$

Dalla tabella di verità della fig.IV.77, si hanno le seguenti espressioni:

$$0001\, y_1 - y_3$$
$$0100 - y_2\, y_3$$

$$
\begin{array}{l}
0101- \; y_2\, y_3 \\
0110\, y_1\, y_2\, - \\
0111\, y_1\, y_2\, - \\
1001\, y_1\, -\, y_3 \\
1010- \; y_2\, y_3 \\
1011\, y_1\, -\, y_3 \\
1100- \; y_2\, y_3 \\
1101- \; y_2\, y_3 \\
1110\, y_1\, y_2\, y_3 \\
1111\, y_1\, y_2\, y_3
\end{array}
$$

b) Ricerca dei primi implicanti.

Sono semplificabili i minterm che differiscono per un solo bit, purché compaiano tutti almeno in una stessa y_i.

Ad esempio, sono semplificabili le due espressioni:

$$
\begin{array}{l}
0011\, y_1\, -\, - \\
0001\, y_1\, y_2\, -
\end{array}
$$

perché i minterm $x_1x_2x_3x_4 = 0001$ e 0011 compaiono entrambi in y_1. La espressione risultante dalla semplificazione deve tener conto che l'implicante trovato è presente solo in y_1:

$$00-1\, y_1\, -\, - \; .$$

Nelle successive fasi del procedimento, restano ancora da confrontare le espressioni:

$$
\begin{array}{l}
0001\, y_1\, y_2\, - \\
00-1\, y_1\, -\, - \; .
\end{array}
$$

Non sono, invece, semplificabili espressioni come:

$$
\begin{array}{l}
0011\, y_1\, -\, - \\
0001- \; y_2\, y_3
\end{array}
$$

perché il minterm 0011 compare solo in y_1 e il minterm 0001 solo in y_2 ed y_3.

Nella pagina seguente è riportata la ricerca dei primi implicanti delle y_i secondo i criteri esposti.

1) Espressioni ordinate secondo i livelli dei relativi minterm, e risultati del primo confronto:

1	$0001\, y_1 - y_3$	✓
4	$0100 - y_2 y_3$	✓
6	$0110\, y_1 y_2 -$	✓
5	$0101 - y_2 y_3$	✓
12	$1100 - y_2 y_3$	✓
10	$1010 - y_2 y_3$	✓
9	$1001\, y_1 - y_3$	✓
7	$0111\, y_1 y_2 -$	✓
13	$1101 - y_2 y_3$	✓
14	$1110\, y_1 y_2 y_3$	✓
11	$1011\, y_1 - y_3$	✓
15	$1111\, y_1 y_2 y_3$	✓

2) Secondo confronto:

1·5	$0-01- - y_3$	✓
1·9	$-001\, y_1 - y_3$	A
4·6	$01-0 - y_2 -$	✓
4·5	$010- - y_2 y_3$	✓
4·12	$-100 - y_2 y_3$	✓
6·7	$011-\, y_1 y_2 -$	✓
6·14	$-110\, y_1 y_2 -$	✓
5·7	$01-1 - y_2 -$	✓
5·13	$-101 - y_2 y_3$	✓
12·13	$110- - y_2 y_3$	✓
12·14	$11-0 - y_2 y_3$	✓
10·14	$1-10 - y_2 y_3$	B
10·11	$101- - - y_3$	✓
9·13	$1-01- - y_3$	✓
9·11	$10-1\, y_1 - y_3$	C
7·15	$-111\, y_1 y_2 -$	✓
13·15	$11-1 - y_2 y_3$	✓
14·15	$111-\, y_1 y_2 y_3$	D
11·15	$1-11\, y_1 - y_3$	E

3) Terzo confronto:

1·5·9·13	$--01-\ -\ y_3$	F
4·5·6·7	$01---\ y_2\ -$	✓
4·5·12·13	$-10--\ y_2\ y_3$	G
4·6·12·14	$-1-0-\ y_2\ -$	✓
6·7·14·15	$-11-\ y_1\ y_2\ -$	H
5·7·13·15	$-1-1-\ y_2\ -$	✓
12·13·14·15	$11---\ y_2\ y_3$	I
10·11·14·15	$1-1--\ -\ y_3$	L
9·11·13·15	$1--1-\ -\ y_3$	M

4) Ultimo confronto:

$$4\cdot5\cdot6\cdot7\cdot12\cdot13\cdot14\cdot15 \qquad -1---\ y_2\ -N\ .$$

I primi implicanti di $y_1 y_2 y_3$ sono quindi:

A	-001	$y_1\ -\ y_3$	(1·9)
B	1-10	$-\ y_2\ y_3$	(10·14)
C	10-1	$y_1\ -\ y_3$	(9·11)
D	111-	$y_1\ y_2\ y_3$	(14·15)
E	1-11	$y_1\ -\ y_3$	(11·15)
F	--01	$-\ -\ y_3$	(1·5·9·13)
G	-10-	$-\ y_2\ y_3$	(4·5·12·13)
H	-11-	$y_1\ y_2\ -$	(6·7·14·15)
I	11--	$-\ y_2\ y_3$	(12·13·14·15)
L	1-1-	$-\ -\ y_3$	(10·11·14·15)
M	1--1	$-\ -\ y_3$	(9·11·13·15)
N	-1--	$-\ y_2\ -$	(4·5·6·7·12·13·14·15)

c) Determinazione della copertura minima.

La scelta dei primi implicanti che realizzano la copertura minima avviene in un reticolo costruito come se le funzioni dovessero essere realizzate l'una indipendentemente dall'altra.

Tratteremo prima la minimizzazione del numero dei termini (o delle lettere), che porta al circuito a 2 livelli col minimo numero di elementi, poi la minimizzazione che porta al circuito col minimo numero di diodi.

IV.13.1 - Minimizzazione del numero dei termini e delle lettere.

Il reticolo (fig.IV.78) ha sulle ascisse i primi implicanti, e sulle ordinate i minterm, delle y_i. Esistono quattro primi implicanti essenziali:

- A, che copre i minterm 1 e 9 in y_1 e in y_3;
- B, che copre i minterm 10 e 14 in y_2 e y_3;
- G, che copre i minterm 4, 5, 12 e 13 in y_2 e y_3;
- H, che copre i minterm 6, 7, 14 e 15 in y_1 e y_2.

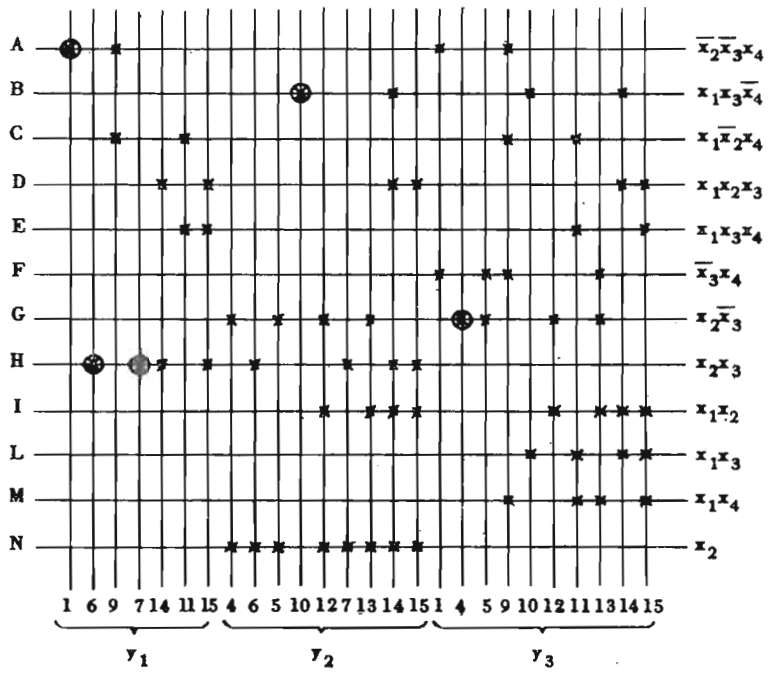


Fig.IV.78 - Reticolo per la ricerca della copertura minima di un circuito MT.

Rimangono scoperti il minterm 11 in y_1 ed y_3 e il minterm 15 in y_3. La copertura minima rispetto al numero dei termini e delle lettere si ottiene scegliendo il primo implicante E, che li copre entrambi, e porta alle espressioni:

$$(IV.20)\qquad \begin{cases} y_1 = H + A + E = x_2x_3 + \overline{x}_2\overline{x}_3x_4 + x_1x_3x_4 \\ y_2 = H + G + B = x_2x_3 + x_2\overline{x}_3 + x_1x_3\overline{x}_4 \\ y_3 = E + G + B + A = x_1x_3x_4 + x_2\overline{x}_3 + x_1x_3\overline{x}_4 + \overline{x}_2\overline{x}_3x_4 \end{cases} .$$

Il relativo circuito (fig.IV.79) si realizza con 8 elementi e 23 diodi.

IV.13.2 - Minimizzazione del numero dei diodi.

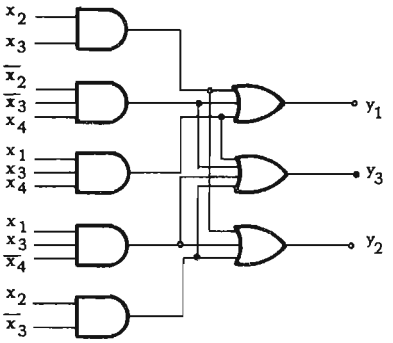


Fig.IV.79 - Circuito MT derivato dalle tabelle di verità di fig.IV.75).

L'espressione che porta al minor numero di diodi può essere diversa da quella per il minor numero di elementi, se diventa conveniente rinunciare all'uso di un termine comune a più funzioni. Ne deriva che il reticolo su cui va cercata la copertura minima deve contenere tutti i termini del precedente, più quelli provenienti dalla scomposizione dei termini con più di una lettera (fig.IV.80). (Ad esempio, insieme con «$-1\,1-y_1\,y_2-$» vanno introdotti i due termini «$-1\,1-y_1--$» e «$-1\,1--y_2-$»).

Dai termini aggiunti vanno, però, eliminati quelli compresi in altri esistenti (ad esempio «$-1\,1--y_2-$» è assorbito da «$-1---y_2-$»).

Nel reticolo della fig.IV.80, i termini sulle ordinate sono stati numerati da n_1 a n_{20}: ad ognuno è stato associato un peso, p_i, che rappresenta il numero di diodi con cui il primo implicante corrispondente verrà realizzato in un circuito a due livelli. Questo peso è dato dalla somma del numero delle y e delle lettere del primo implicante (se questo non è formato da una sola lettera). Ad esempio «$-10-y_1\,y_2-$» ha peso 4, «$-1---y_2-$», peso 1. Il peso di una copertura formata con gli implicanti dalle espressioni $n_jn_kn_m\ldots$ è così eguale alla somma dei pesi p_j, p_k, p_m (questa regola non vale quando una delle y_i è riducibile ad una sola lettera: ma questo caso, particolarissimo, è facilmente individuabile).

Nel nostro esempio, il problema è quello di ottenere $y_1y_2y_3$ con un insieme di primi implicanti contenuti in un certo numero di termini n_i, di pesi p_i, in modo da minimizzare la somma dei pesi stessi.

Nel reticolo non ci sono primi implicanti essenziali, per cui occorre considerare tutte le possibili scelte per ognuno dei minterm.

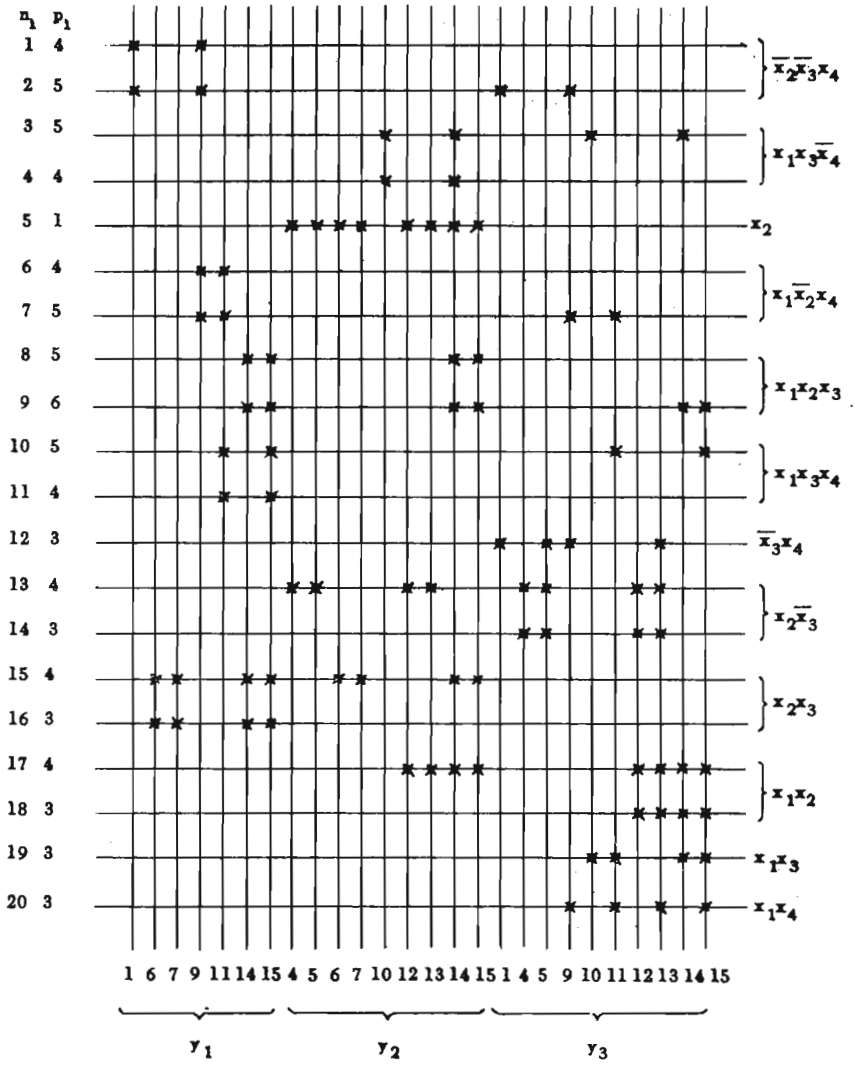


Fig.IV.80 - Reticolo per la ricerca della copertura minima rispetto al numero dei diodi.

Per coprire il minterm 1 di y_1, si può prendere n_1 o n_2; per coprire il minterm 6 di y_1 si può prendere n_{15} o n_{16}: per coprire – insie-

me – i minterm 1 e 6, si deve prendere n_1 o n_2 e n_{15} o n_{16}. Con le notazioni dell'algebra logica, si scriverà:

$$(n_1 + n_2)\,(n_{15} + n_{16})\ .$$

Continuando così per tutti i minterm di $y_1 y_2 y_3$ si arriva a un prodotto di tante somme quanti sono i minterm, prodotto che, semplificato con la legge dell'assorbimento, diventa:

$$\Pi = (n_1+n_2)\,(n_{15}+n_{16})\,(n_6+n_7+n_{10}+n_{11})\,(n_5+n_{13})\,(n_3+n_4)\cdot$$
$$\cdot\ (n_2+n_{12})\,(n_{13}+n_{14})\,(n_5+n_{15})\,(n_3+n_{19})\,(n_7+n_{10}+n_{19}+n_{20})\cdot$$
$$\cdot\ (n_8+n_9+n_{10}+n_{17}+n_{18}+n_{19}+n_{20})\ .$$

Da Π si ottengono tutte le soluzioni del problema prendendo in tutti i modi possibili, una delle n_i da ognuna delle parentisi; il numero dei diodi relativo ad ogni soluzione è la somma dei pesi delle n_i scelte.

Ad esempio, la soluzione:

$$n_1 \cdot n_{15} \cdot n_6 \cdot n_5 \cdot n_3 \cdot n_2 \cdot n_{15} \cdot n_7 \cdot n_8$$

ottenuta coi primi valori di ogni parentesi, ha peso:

$$4+4+4+1+5+5+4+5+5 = 37\ .$$

Per confronto, si trova che la scelta più economica è la:

$$n_2 \cdot n_3 \cdot n_5 \cdot n_{10} \cdot n_{14} \cdot n_{16}$$

di peso:

$$5+5+1+5+3+3 = 22\ .$$

Il relativo circuito a 22 diodi (fig.IV.81), per essere:

$$\begin{aligned} n_2 &= -00\,1\,y_1 - y_3 \\ n_3 &= 1-10-y_2 y_3 \\ n_5 &= -1---y_2- \\ n_{10} &= 1-1\,1\,y_1 - y_3 \\ n_{14} &= -10---y_3 \\ n_{16} &= -1\,1-y_1-- \end{aligned}$$

si costruisce dalle equazioni:

$$(IV.21)\qquad \begin{cases} y_1 = \bar{x}_2\bar{x}_3x_4 + x_2x_3 + x_1x_3x_4 \\ y_2 = x_2 + x_1x_3\bar{x}_4 \\ y_3 = \bar{x}_2\bar{x}_3x_4 + x_2\bar{x}_3 + x_1x_3\bar{x}_4 + x_1x_3x_4 \end{cases}.$$

Si noti che:

a) Il metodo esposto, rigoroso dal punto di vista teorico, non è molto utile in pratica, perché richiede calcoli eccessivamente lunghi, specialmente nella seconda variante. La sua applicazione, anzi, non è semplice nemmeno disponendo di un grosso calcolatore. Tranne casi particolari, conviene quindi procedere per tentativi, applicando il metodo di Karnaugh.

b) Le (IV.21) si ottengono immediatamente dalle (IV.20), semplificando il più possibile l'espressione di y_2: questo suggerisce una via da seguire nella ricerca della forma minima per tentativi.

c) Il circuito della fig.IV.81 usa lo stesso numero di elementi di quello della fig.IV.79: ciò è una particolarità dovuta all'esistenza di un termine (x_2) in y_2 che non richiede alcun AND per essere realizzato.

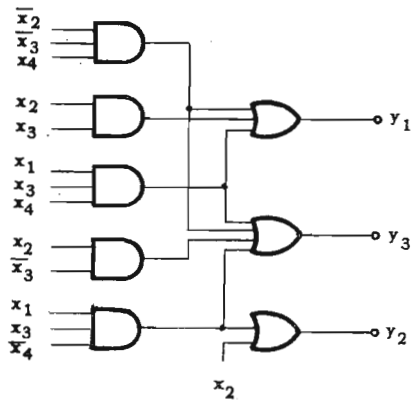


Fig.IV.81 - Circuito MT derivato dalle tabelle di verità di fig.IV.75.

Nei problemi a più alto numero di variabili, non esiste – in genere – un circuito che minimizza insieme elementi e diodi.

IV.14 - Selettori e matrici.

I selettori sono circuiti MT a diodi di uso assai frequente in ogni tipo di apparecchiature numeriche: hanno n ingressi e 2^n uscite, ognuna delle quali assume il valore 1 per una sola delle 2^n possibili configurazioni degli ingressi.

Un selettore si può realizzare con 2^n circuiti AND, ognuno dei quali ha per uscita uno dei 2^n minterm delle variabili $x_1x_2\ldots x_n$. La figura IV.82 rappresenta un selettore per 2 variabili, con 4 uscite, 4 AND e 8 diodi.

I selettori prendono il nome di *matrici* quando, come di solito avviene, si realizzano con un montaggio particolare, che ricorda l'aspetto delle matrici matematiche; nelle figg.IV.83a e IV.83b sono mostrate due matrici per 2 variabili funzionanti, secondo lo schema della fig. IV.82, rispettivamente in logica positiva e negativa.

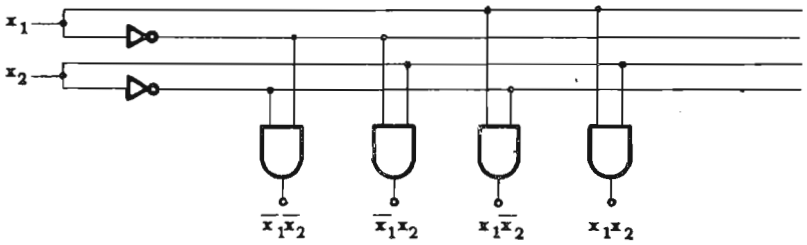


Fig.IV.82 - Selettore a 2 variabili.

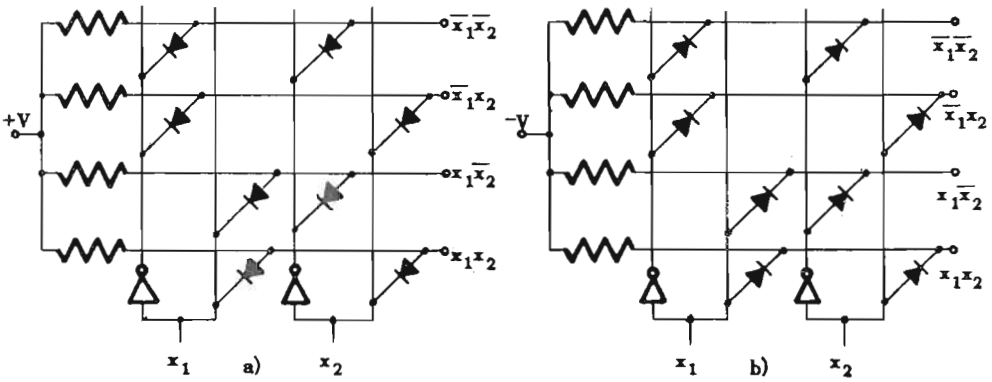


Fig.IV.83 - Matrici per 2 variabili in logica positiva (a) e negativa (b).

Una semplice rappresentazione simbolica delle matrici della figura IV.83, indipendente dal tipo di logica; è quella della fig.IV.84: sulle verticali sono riportati gli ingressi del circuito, sulle orizzontali le uscite; ogni punto rappresenta un diodo, ogni riga un AND i cui ingressi si leggono sulle colonne segnate coi puntini.

Dalle matrici a due ingressi deriva immediatamente la struttura di quelle più complesse: per ogni variabile occorrono tanti diodi quante sono le uscite; i 2^n diodi relativi alla variabile x_i vanno disposti su 2^i gruppi, alternati tra $\bar{x}_i$ e x_i ($i = 1, 2, \ldots, n$).

A titolo d'esempio, nella fig.IV.85 è mostrata la matrice a 4 variabili, 16 uscite e 64 diodi.

Le matrici sono circuiti molto semplici ma, all'aumentare del numero degli ingressi, richiedono un numero eccessivo di diodi ($n \cdot 2^n$, per n variabili).

Selettori più economici, rispettivamente per $n \geq 3$ ed $n \geq 4$, si ottengono con i montaggi *ad albero* e *a matrici multiple*.

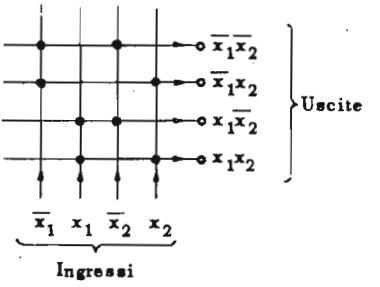


Fig.IV.84 - Rappresentazione simbolica delle matrici di fig.IV.83.

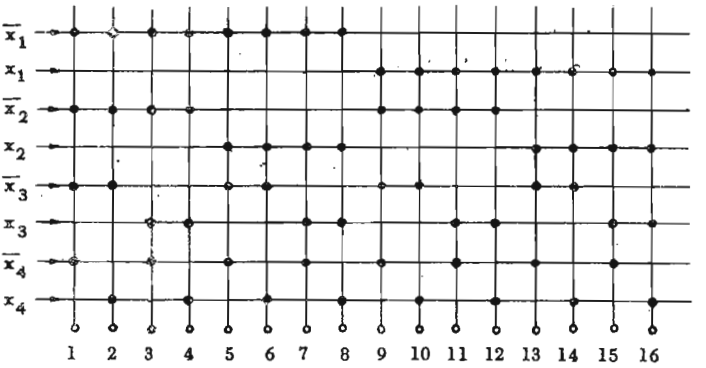


Fig.IV.85 - Matrice a 4 variabili.

IV.14.1 - Selettori ad albero.

La matrice a due variabili della fig.IV.83a può essere modificata adoperando un solo diodo tra la colonna $\overline{x}_1$ (x_1) e le righe $\overline{x}_1\overline{x}_2$ e $\overline{x}_1 x_2$ ($x_1\overline{x}_2$ e $x_1 x_2$), nel modo illustrato dalla fig.IV.86a.

Il circuito si chiama selettore ad albero perché rappresentabile con lo schema della fig.IV.86b, dove ogni ramo corrisponde a un diodo. Un albero a n variabili adopera 2 diodi per la variabile x_1, 4 per $x_2 \ldots 2^n$ per la x_n. In totale soltanto $\sum_1^n 2^i$ diodi contro gli $n \cdot 2^n$ di una matrice. L'albero però presenta lo svantaggio che i suoi diodi sul primo stadio debbono portare una corrente piuttosto forte, pari a 2^{n-1} volte quella portata dagli ultimi diodi. Questo montaggio pertanto, si usa quando il numero delle variabili non è eccessivamente elevato.

A titolo d'esempio, nella fig.IV.87a e IV.87b sono mostrati gli schemi di due selettori a matrice e ad albero per trasformare – o decodificare – un numero binario a tre cifre in un numero decimale da 0 a 7; per il selettore ad albero è mostrato anche lo schema circuitale (fig.IV.87c). Entrambi i circuiti hanno 3 ingressi, dove si presentano in parallelo i bit $x_1 x_2 x_3$ di pesi 4-2-1, e 8 uscite: è al valore 1 quell'uscita corrispondente al numero decimale rappresentato dagli ingressi. Si noti che, mentre la matrice adopera 24 ($= 3 \cdot 2^3$) diodi, l'albero ne adopera soltanto 14 ($= 2 + 4 + 8$).

Per le ragioni illustrate, quando $n \geq 4$, il montaggio più usato è quello a matrici multiple.

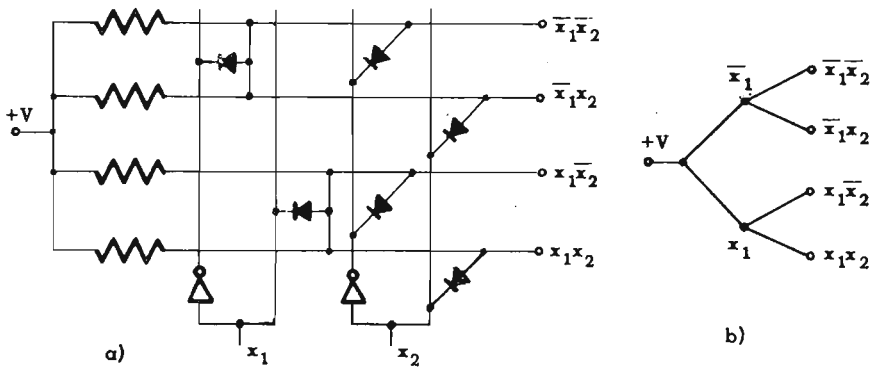


Fig.IV.86 - Selettore ad albero a 2 variabili.

IV.14.2 - Matrici multiple.

Si chiamano matrici multiple quei selettori in cui la scelta di una uscita viene effettuata attraverso un certo numero di matrici parziali (o sottomatrici) a 2 o 3 variabili, le cui uscite vengono accoppiate in tutti i modi possibili con 2^n AND disposti su un successivo livello. Ad esempio, una matrice multipla a 4 variabili è formata da 2 matrici M_1 ed M_2 a 2 variabili, in serie con una matrice M che combina tutte le uscite di M_1 con tutte quelle di M_2 (fig.IV.88a). Per maggior chiarezza, nella figura IV.88b è mostrato lo schema AND del selettore stesso; il circuito, che può servire (come indicato in figura) per convertire un numero binario a 4 bit $x_1 \cdot x_2 \cdot x_3 \cdot x_4$, di pesi $8 \cdot 4 \cdot 2 \cdot 1$, nella corrispondente cifra esadecimale, impiega 48 diodi contro i 64 ($= 4 \cdot 2^4$) necessari per una matrice completa.

La ripartizione ottimale delle n variabili di un selettore a matrici parziali (per $n \geq 5$) si ottiene dividendo le variabili stesse in due gruppi G_1 e G_2, il più possibile eguali tra loro, ed ognuno dei due gruppi G_1 e G_2 in due altri (G_{11}, G_{12} e G_{21}, G_{22}) il più possibile eguali, e così via fino ad arrivare a gruppi di non più di 3 variabili.

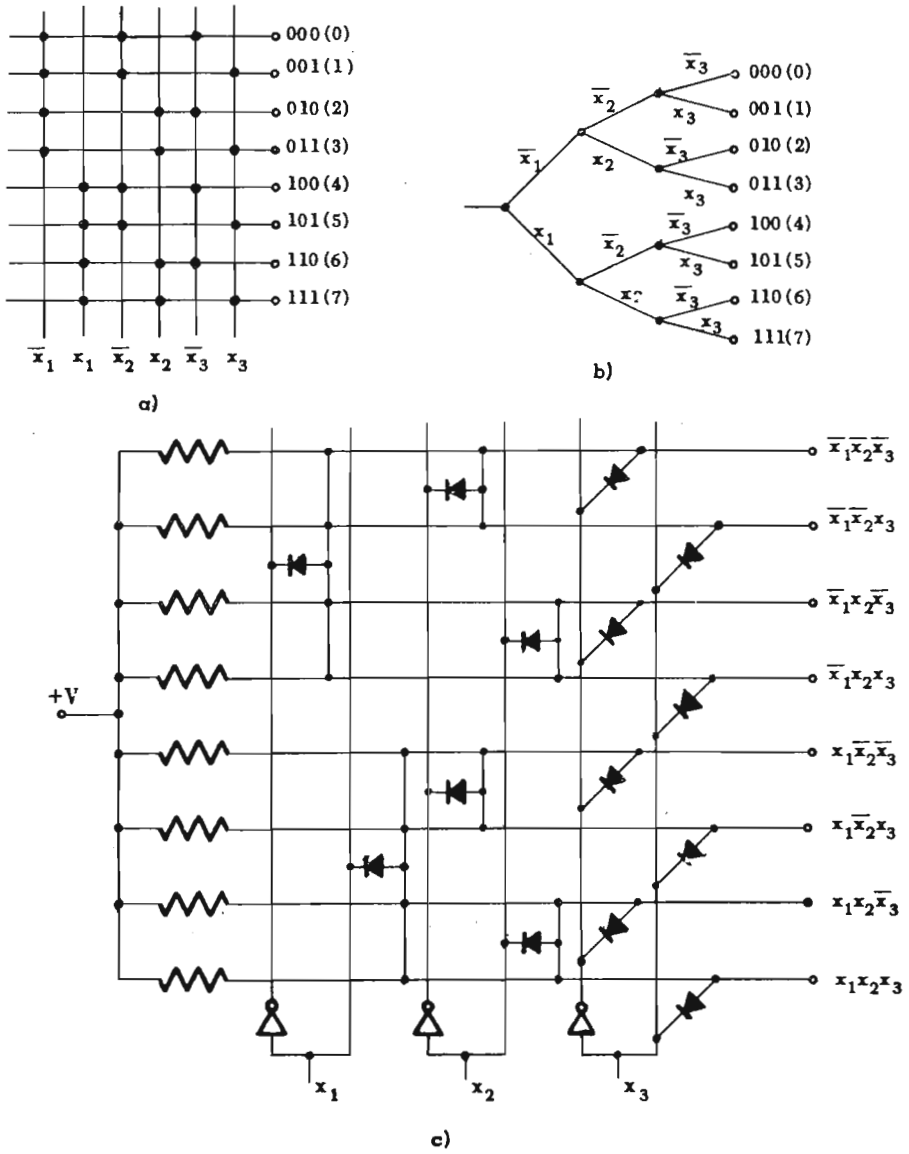


Fig.IV.87 - Selettore ad albero a 3 variabili.

A titolo d'esempio, nella fig.IV.89 è riportato il diagramma di ripartizione e lo schema a blocchi di un selettore a matrici multiple per 7 variabili.

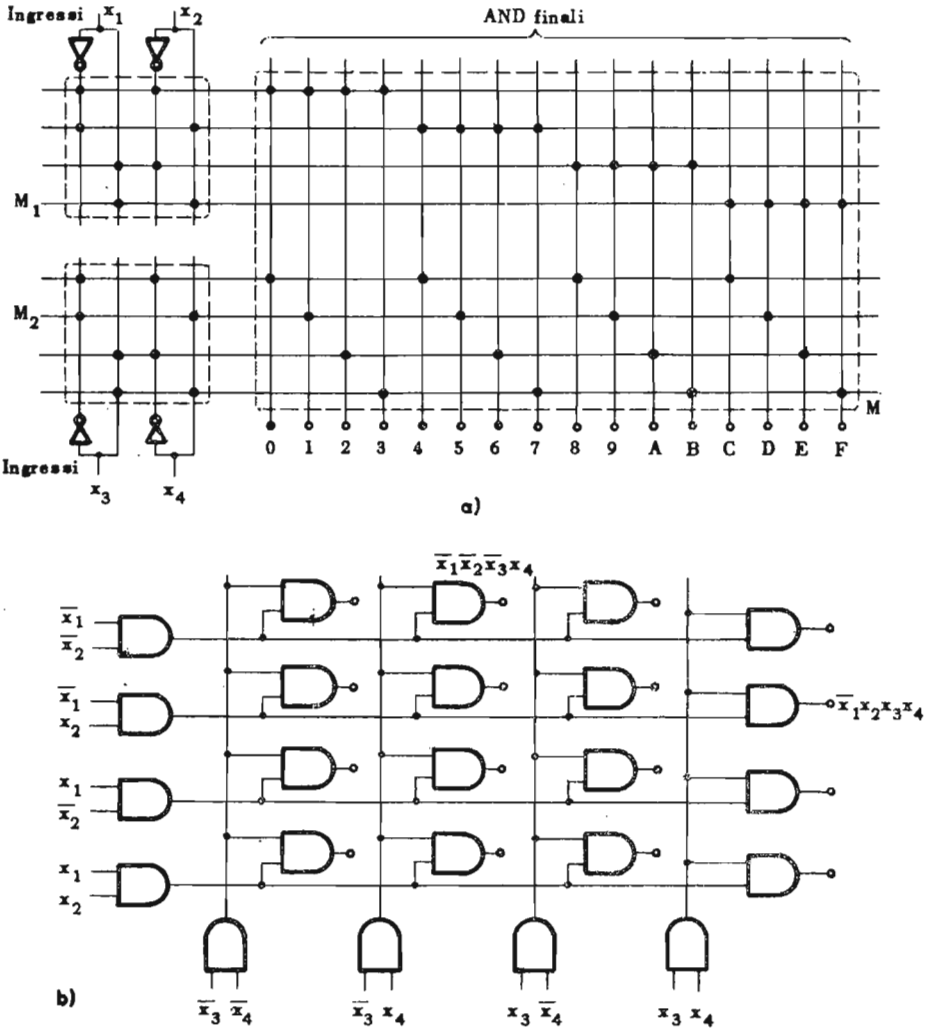


Fig.IV.88 - Matrice multipla a 4 variabili.

In pratica, quando gli stadi superano 3, bisogna tener conto nel bilancio dei componenti degli organi attivi necessari per evitare una eccessiva degradazione del segnale nel passaggio attraverso tre stadi in serie.

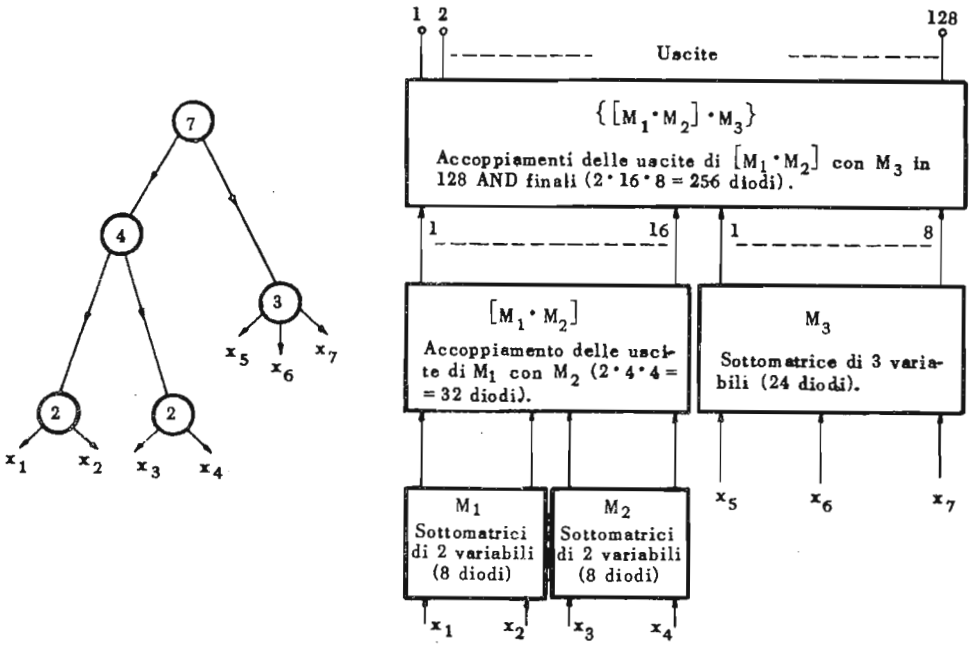


Fig.89 - Matrice multipla a 7 variabili.

IV.14.3 - Matrici incomplete.

Se qualcuna delle 2^n uscite di un albero non viene utilizzata, il numero dei diodi diminuisce, non solo nella matrice dell'ultimo stadio, ma anche in quelle degli stadi precedenti. La diminuzione dipende anche dalla ripartizione delle variabili sugli ingressi delle varie matrici, nel senso che ripartizioni diverse danno luogo a semplificazioni di diversa entità, come mostra l'esempio seguente, che descrive il metodo di eliminazione degli elementi ridondanti.

Esempio 1: *Realizzare un selettore a 5 variabili in cui siano utilizzate tutte e sole le uscite corrispondenti ai minterm che non fanno parte della funzione:*

$$y = \overline{x}_1 (x_2 x_5 + x_3) \quad .$$

Il diagramma di ripartizione per una matrice a 2 stadi è quello della fig.IV.89a. Il termine $\overline{x}_1 x_3$ di y apporta una riduzione in M_3 ed M_5: per tenerne conto, scriviamo $\overline{x}_1 x_3$ accanto ad M_3 ed M_5 (fig.IV.90b). Il termine $\overline{x}_1 x_2 x_5$ influisce invece soltanto su M_5, perché $\overline{x}_1 x_2$ appartiene a M_3, ed x_5 ad M_2: lo scriviamo accanto a M_5. A parte, scriviamo poi il prodotto dei termini comuni $(\overline{x}_1 x_2 x_3 x_5)$.

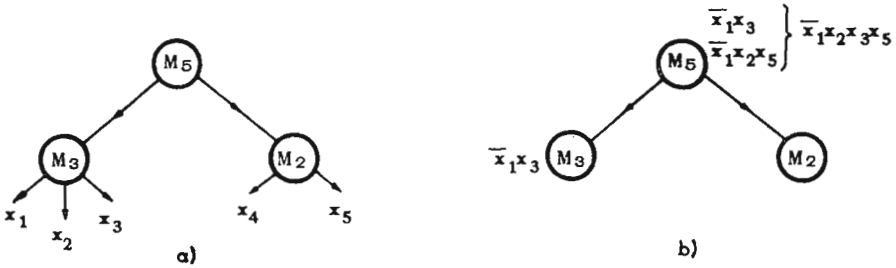


Fig.IV.90 - Matrice multipla minima incompleta a 5 variabili.

Possiamo ora calcolare le uscite da eliminare: in M_3, $2^{3-2} = 2$ uscite contengono il termine $\overline{x}_1 x_3$: poiché ogni uscita di M_3 è associata a 3 diodi, si eliminano $3 \times 2 = 6$ diodi. In M_5, $2^{5-2} = 8$ uscite contengono $\overline{x}_1 x_3$; $2^{5-3} = 4$ contengono $\overline{x}_1 x_2 x_5$, e $2^{5-4} = 2$ contengono $\overline{x}_1 x_2 x_3 x_5$: le uscite da eliminare sono complessivamente: $8 + 4 - 2 = 10$ (le uscite del termine comune vanno sottratte, per non contarle due volte). Poiché ogni uscita di M_5 è associata a 2 diodi, si risparmiano 20 diodi. Il numero dei diodi necessari si ottiene sottraendo $20 + 6 = 26$ al numero di quelli occorrenti per la matrice multipla:

$$3 \cdot 2^3 + 2 \cdot 2^2 + 2 \cdot 2^5 - 26 = 96 - 26 = 70 \ .$$

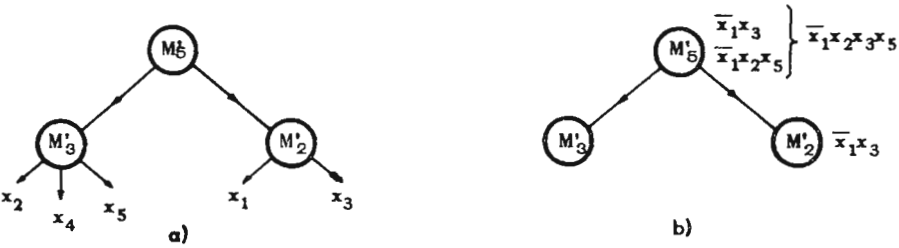


Fig.IV.91 - Matrice multipla incompleta non minima a 5 variabili.

Se si fosse adottato il diagramma di ripartizione della fig.IV.91a, si sarebbe ottenuta una minore economia, perché $\overline{x}_1 x_3$ avrebbe eliminato due soli diodi in M_2, invece che 6 in M_3 (fig.IV.91b).

Se, infine, si fosse adoperata una matrice unica, sarebbero occorsi:

$$5 \times 22 = 110 \text{ diodi}$$

essendo 5 il numero dei diodi per stadio e 22 $(= 32 - 10)$ le uscite da utilizzare.

*

BIBLIOGRAFIA

1. KEISTER W. & RITCHIE A.E. & WASHBURU G.H.: *The Design of Switching Circuits* - J. Van Nostrand, 1951.

2. GREA R. & HIGONNET R.: *Etude logique des circuits des contacts* - Revue Generale de l'èlectricitè - Gennaio 1954.

3. NASLIN P.: *Circuites à relais* - Dunod, 1958.

4. MALEY G.A. & EARLE J.: *The logic design of transistor digital Computers* - Prentice-Hall. 1963.

5. MARCUS M.P.: *Switching Circuits for Engineers* - Prentice Hall, 1962.

6. BARTEE T.C.: *Computer Design of Multiple Output Logical Networks* - IRE Transactions on Electronic Computers - Vol. 10 - Marzo 1961.

7. POLANSKY R.B.: *Minimization of Multiple Output Switching Circuits* - AIEE - Conference Paper - Ottobre 1960.

8. CURTIS H.A.: *A New Approach to the Design of Switching Circuits* - Van Nostrand 1962.

9. ROTH J.P.: *Minimization over Boolean Trees* - IBM JRD - Vol. 4, n. 5 - Novembre 1960.

10. CHU J.T.: *A Generalization of a Theorem of Quine for Simplifying Truth Functions* - IRE Transaction on EC - Vol. 10, n. 2 - Giugno 1961.

11. RIGHI R.: *Algebra Booleana* - Vol.II, Siderea 1971.

*

CAPITOLO V

INTRODUZIONE AI CIRCUITI SEQUENZIALI

V.1 - Generalità.

Si chiamano *sequenziali* quei circuiti le cui uscite, ad ogni istante t, dipendono dai valori assunti dagli ingressi sia all'istante t, che in tutti gli istanti precedenti.

Esiste una teoria matematica, quella delle *macchine sequenziali* che interpreta e descrive il comportamento di questi circuiti, comportamento che è assai diverso da quello dei circuiti combinatori. Intenderemo, appunto, per *macchina sequenziale* $\mathcal{M}$ un modello matematico del circuito sequenziale C, modello caratterizzato da un insieme di regole che trasformano una sequenza di valori delle variabili $x_1 x_2 \ldots x_n$ in una delle variabili $z_1 z_2 \ldots z_m$, in modo tale che le x e le z coincidano con gli ingressi e le uscite del circuito C. La teoria delle macchine sequenziali, d'altra parte, si può applicare ai circuiti concepiti secondo un certo schema, chiamato *modello fondamentale* dei circuiti sequenziali, che appresso descriveremo.

V.2 - Equazioni e modello fondamentale dei circuiti sequenziali.

Visto dall'esterno, un circuito sequenziale (fig.V.1) si presenta proprio come un circuito combinatorio a n ingressi ed m uscite; ognuna delle uscite, però, dipende dal tempo, oltre che dagli ingressi $x_1 x_2 \ldots x_n$, secondo una equazione che è del tipo:

$$z_j = z_j(x_1 x_2 \ldots x_n, t) \qquad (j = 1, 2, \ldots, m) \quad . \tag{V.1}$$

Per trattare analiticamente i circuiti sequenziali, data la natura atemporale dell'algebra booleana, occorre eliminare, con qualche artificio, la variabile t dalla (V.1).

Per definizione, un circuito sequenziale funziona correttamente se è in grado di conservare, in qualche organo di memoria, le informazioni relative agli ingressi passati. Chiameremo *stato* di un circuito sequenziale in un certo istante t, la configurazione della sua memoria in quell'istante. Due identiche configurazioni d'ingresso, appartenenti alla stessa sequenza o a due sequenze diverse, daranno uscite diverse se e solo se coesisteranno con due diversi stati del circuito. Le uscite del circuito diventano così funzioni degli ingressi x e dello stato del circuito.

Fig.V.1 - Schema a blocchi di un circuito sequenziale visto dall'esterno.

Lo *stato* immagazzina le informazioni sugli ingressi precedenti sotto forma di segnali interni al circuito, nel senso che stati diversi sono rappresentati da diversi insiemi di segnali interni. Se un circuito esiste in s stati distinti, occorreranno per riconoscerli tutti, k segnali ($2^{k-1} < s \leq 2^k$) che si indicano con $y_1 y_2 \ldots y_k$ e si chiamano *variabili interne*. Le uscite di un circuito sequenziale, concepite come funzioni degli ingressi x e delle variabili interne y (cioè degli stati) possono allora essere espresse nella forma:

$$z_j = z_j(x_1 x_2 \ldots x_n\, y_1 y_2 \ldots y_k) \qquad (j = 1, 2, \ldots, m) \quad . \tag{V.2}$$

Le (V.2) si chiamano *equazioni di uscita del circuito sequenziale.*

L'introduzione delle variabili interne y e l'espressione delle uscite z in funzione delle y non basta a descrivere completamente il funzionamento del circuito: occorre ancora specificarne il comportamento interno, cioè il modo in cui cambiano le y sotto l'azione di una generica configurazione d'ingressi. Supponendo il comportamento del circuito deterministico, nel senso che il nuovo stato q' raggiunto a partire dallo stato q, per effetto degli ingressi $x_1 x_2 \ldots x_n$, è univocamente determinato, e indicando con y_i' e y_i i valori delle y corrispondenti agli stati q' e q, si potrà scrivere:

$$y_i' = y_i'(x_1 x_2 \ldots x_n\, y_1 y_2 \ldots y_k) \qquad (i = 1, 2, \ldots, k) \quad . \tag{V.3}$$

Le (V.3) si chiamano *equazioni interne* del circuito; insieme alle (V.2), sostituiscono le (V.1).

Le equazioni (V.2) e (V.3) si possono interpretare come le funzioni di uscita di un circuito combinatorio multiterminale ad (n + k) ingressi ed (m + k) uscite (fig.V.2); ma in questo modo si perde ogni riferimento al legame temporale tra le y' e le y: le y', infatti, sono i valori che le y assumeranno nell'istante $t + \Delta$, per conservare il ricordo della configurazione di ingresso $x_1 x_2 \ldots x_n$ intervenuta all'istante t sullo stato $y_1 y_2 \ldots y_k$, le (V.3) specificano quindi i valori delle y', da usarsi come valori delle y con il prossimo insieme di ingressi, per formare un altro insieme di uscite, secondo le (V.2). Si noti che Δ non può essere nullo: se, infatti, ogni y assumesse immediatamente il valore della corrispondente y', ogni configurazione d'ingresso intervenuta su uno stato provocherebbe istantaneamente una variazione dello stato stesso; questa variazione ne provocherebbe una seconda, questa una terza, e così via, fino a rendere del tutto imprevedibile il comportamento del circuito. Sull'effettivo valore di Δ si dirà, comunque, in seguito.

Fig.V.2 - Schema a blocchi di un circuito sequenziale con ingressi e uscite interne ed esterne.

Per legare le y alle y', conviene introdurre un elemento di ritardo Δ (fig.V.3), definito come un dispositivo a un ingresso ed un'uscita le cui variazioni seguono quelle dell'ingresso con un ritardo Δ secondo l'equazione:

$$y(t + \Delta) = y(t)$$

che, indicando con y l'uscita $y(t + \Delta)$ e con y' l'ingresso y(t), può scriversi:

$$y = y'$$

eliminando ogni esplicito riferimento al tempo.

Fig.V.3 - Elemento di ritardo.

Legando ogni y_i al corrispondente y_i' attraverso un elemento Δ_i, possiamo ora tener conto della relazione tra gli stati all'istante t e quelli all'istante $t+\Delta_i$, e disegnare il modello fondamentale di un circuito sequenziale (fig.V.4).

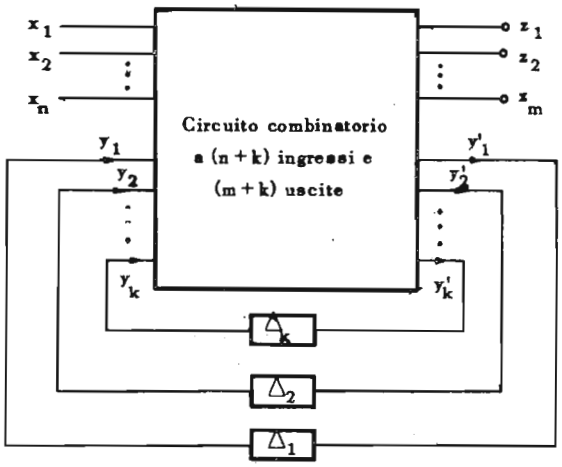


Fig.V.4 - Schema a blocchi di un circuito sequenziale.

V.3 - Funzionamento sincrono del modello fondamentale.

Il circuito di fig.V.4 interpreta le equazioni (V.2) e (V.3), ma funziona correttamente solo se i segnali interni y e quelli esterni x relativi all'istante t si presentano simultaneamente ai corrispondenti ingressi. Il metodo più semplice per ottenere la sincronizzazione tra le due classi di segnali consiste nel controllare il circuito con un oscillatore esterno, che alimenta un segnale rettangolare a frequenza fissa, per indicare intervalli eguali di tempo: t, $t+\Delta$, $t+2\Delta$, ... (clock). In questi circuiti, chiamati *sincroni*, i ritardi Δ_i dello schema della fig. V.4 sono tutti eguali al periodo Δ di clock. Le equazioni (V.2) e (V.3) vanno interpretate nel senso che, dati i segnali $x_1 x_2 \ldots x_n$ ed $y_1 y_2 \ldots y_k$ all'istante t, le $z_1 z_2 \ldots z_m$ sono le uscite al tempo t e le $y_i' y_2' \ldots y_k'$ gli ingressi agli elementi di ritardo al tempo t. Le $y_i' y_2' \ldots y_k'$, inoltre, sono i valori che $y_1 y_2 \ldots$ $\ldots y_k$ assumeranno all'istante $t+\Delta$, quando il prossimo segnale di clock trasformerà appunto ogni y_i nella corrispondente y_i' e metterà sugli ingressi i nuovi valori $x_1(t+\Delta)$, $x_2(t+\Delta)$, ... $x_n(t+\Delta)$. Per questa ragione, le y_i' si chiamano anche *stati futuri* del circuito.

I circuiti sequenziali sincroni si realizzano, in pratica, sostituendo dei flip-flop agli elementi di ritardo, per ragioni che vedremo in seguito. Per quanto riguarda il modello fondamentale di circuito sequenziale sincrono, comunque, la realizzazione della rete di memoria non ha importanza: interessa solo che i Δ siano tutti eguali, e che i valori $x_1 x_2 \ldots \ldots x_n$ vengano alimentati all'ingresso del circuito a intervalli di tempo Δ.

Per fissare le idee e mostrare come si comporta un circuito di questo tipo, esaminiamo il funzionamento di quello ad un ingresso e una uscita dell'esempio seguente.

Esempio 1: *Studio del circuito sequenziale sincrono della fig. V.5, nell'ipotesi che gli elementi logici AND-OR siano privi di ritardo.*

Supponiamo che si abbia $\Delta_1 = \Delta_2 = \Delta$ e che l'ingresso x, 0 all'istante t_0, assuma i valori, non necessariamente diversi, di una sequenza $\{I_x\}$ agli istanti Δ, 2Δ, $3\Delta \ldots$. Disegnato il circuito, per mostrarne la rispondenza allo schema ideale, nella forma della fig. V.6, si ottengono le equazioni dell'uscita z e degli stati futuri y_1' e y_2':

$$(V.4) \qquad \begin{cases} z = y_2 \\ y_1' = y_1 y_2 + x y_2 \\ y_2' = x \overline{y_1} \quad . \end{cases}$$

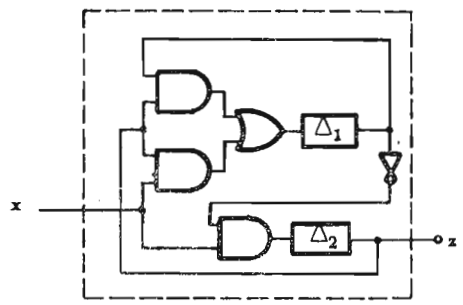


Fig. V.5 - Circuito sequenziale sincrono.

Se si pone $y_1(t_0) = y_2(t_0) = 0$, dalle equazioni (V.4) si ricava:

$$y_1'(t_0) = y_2'(t_0) = z(t_0) = 0 \quad ;$$

i valori delle y_i' diventeranno, all'istante $t_0 + \Delta$, i nuovi valori di y_1 e y_2. Se, in $t_0 + \Delta$, la x passa da 0 a 1, si ha:

$$\begin{aligned} y_1'(t_0 + \Delta) &= 0 \\ y_2'(t_0 + \Delta) &= 1 \\ z(t_0 + \Delta) &= 0 \quad . \end{aligned}$$

I valori delle y_i' diventeranno, all'istante $t_0 + 2\Delta$, i nuovi valori di y_1 e y_2. Se, in questo istante, x torna a 0, si ha: $y_1' = y_2' = 0$, $z = 1$. Se, invece, $x(t + 2\Delta) = 1$, si ha: $y_1' = y_2' = z = 1$.

All'istante $t + 3\Delta$, l'ingresso x può quindi trovare il circuito nello stato $y_1 = y_2 = 0$ oppure nello stato $y_1 = y_2 = 1$, a seconda del valore di x nell'istante preceden-

te. L'evoluzione del circuito nel primo caso, è stata già esaminata; nel secondo, indipendentemente dal valore di x, si ha: $y_1' = y_2' = 0$, $z = 1$.

Resta ancora da esaminare il comportamento del circuito a partire dallo stato $y_1 y_2 = 10$, in corrispondenza al quale si ha: $y_1 = y_2 = z = 0$.

I risultati dell'analisi compiuta, riassunti nella fig.V.7, possono essere chiaramente visualizzati con un grafo, detto *diagramma degli stati*, che ha tanti nodi quanti sono i possibili stati del circuito (4 nel caso in esame), distinti dai rispettivi valori delle variabili secondarie y. Il valore dell'ingresso che porta dallo stato a allo stato b è scritto sul segmento orientato che unisce a con b, a sinistra del segno /. A destra, è scritto il valore della uscita. Ad esempio, poiché l'ingresso $x = 0$ applicato allo stato $y_1 = y_2 = 1$ manda il circuito nello stato $y_1 y_2 = 10$ con uscita 1, esiste un segmento orientato 0/1 dal nodo 11 al nodo 10.

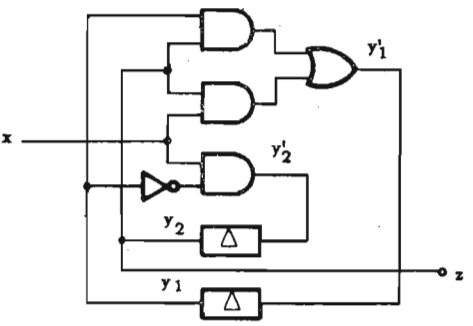


Fig.V.6 - Circuito sequenziale di fig.V.5 disegnato secondo lo schema di fig.V.4.

Ingresso	Stato del circuito all'istante t		Stato del circuito all'istante t+Δ		Uscita
x	y_1	y_2	y_1'	y_2'	z
0	0	0	0	0	0
1	0	0	0	1	0
0	0	1	0	0	1
1	0	1	1	1	1
0	1	1	1	0	1
1	1	1	1	0	1
0	1	0	0	0	0
1	1	0	0	0	0

Fig.V.7 - Analisi del comportamento del circuito di fig.V.5.

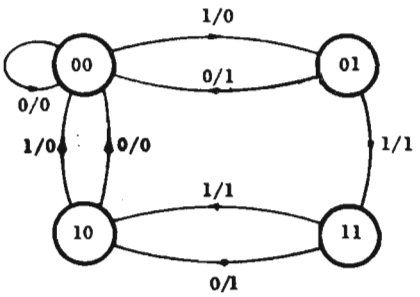


Fig.V.8 - Diagramma di stato del circuito di fig.V.5.

Il diagramma di stato del circuito è mostrato nella fig.V.8. Nei prossimi capitoli verrà esposto un metodo, molto più semplice, per analizzare i circuiti sequenziali ed ottenere il relativo diagramma degli stati.

V.4 - Funzionamento asincrono del modello fondamentale.

Il funzionamento sincrono impone dei vincoli alla costituzione del circuito e alla velocità dei segnali esterni. Il modello fondamentale può funzionare, però, secondo le equazioni (V.2) e (V.3) senza nessun clock esterno, quindi senza sincronizzare le x con le y, se qualsiasi variazione degli ingressi esterni x avviene soltanto negli istanti in cui tutti gli elementi di ritardo Δ_i sono inattivi, cioè quando si ha $y_i' = y_i$. I circuiti che funzionano in questo modo si chiamano asincroni. Per mostrare come funziona un circuito sequenziale asincrono, esaminiamo il modello ideale dell'esempio seguente. Si noti che soltanto i modelli ideali dei circuiti sincroni sono eguali a quelli asincroni: i due tipi di circuiti sono, in pratica, realizzati in maniera molto differente.

Esempio 2: *Studio del circuito sequenziale asincrono della fig.V.9.*

Per ipotesi, supponiamo che gli elementi AND-OR siano privi di ritardo e che, a partire dall'istante iniziale t_0, in cui $x = y_1 = y_2 = y_1' = y_2' = z = 0$, tutte le variazioni dell'ingresso avvengano a intervalli di tempo non necessariamente uguali, ma sempre superiori a Δ, essendo Δ il più grande dei 2 valori Δ_1 e Δ_2. La x, inoltre, rimanga costante ogni volta che qualche y_i è diversa dalla corrispondente y_i'.

Le equazioni del circuito sono:

$$(V.5) \qquad \begin{cases} z = y_1' \\ y_1' = xy_1 + \bar{x}y_2 \\ y_2' = x\bar{y_1} + \bar{x}y_2 \end{cases} .$$

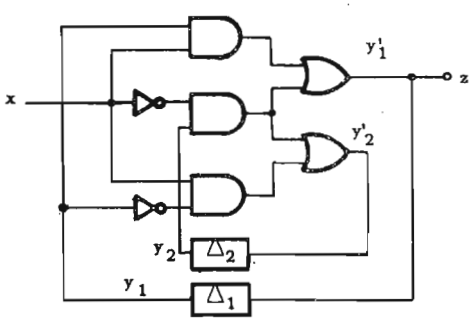


Fig.V.9 - Circuito sequenziale asincrono.

Variando l'ingresso da 0 a 1, si ha $y_1' = 0$, $y_2' = 1$, $z = 0$. Dopo un tempo Δ, y_1 e y_2 assumono i valori di y_1' e y_2', e si ha: $x = 1$, $y_1 = y_1' = 0$, $y_2 = y_2' = 1$, $z = 0$. Soltanto a partire da questo istante si può variare il valore di x: d'altra parte, soltanto la variazione di x può cambiare lo stato del circuito. Per $x = 0$, dopo un transitorio di durata $\delta \leq \Delta$, si ha: $y_1' = y_1 = 1$, $y_2' = y_2 = 1$, $z = 1$. Ponendo ancora $x = 1$, dopo un nuovo tran-

sitorio $\delta' \leq \Delta$ si ottiene: $y_1' = y_1 = 1$, $y_2' = y_2 = 0$, $z = 1$. Finalmente, rimettendo x a 0, si torna allo stato iniziale. L'evoluzione del circuito è mostrata nella fig.V.10; le righe della tabella possono essere divise in due classi: quelle in cui $y_1' \neq y_1$, corrispondenti ai periodi transitori e quelli, segnati con l'asterisco, in cui $y_1' = y_1$, corrispondenti agli stati in cui il circuito permane stabilmente finché non viene cambiato il valore dell'ingresso. Anche per i circuiti asincroni si può costruire il diagramma degli stati (figura V.11) la cui forma è leggermente diversa dal grafo della fig.V.8, in quanto il valore

	Ingresso	Stato all'istante t		Stato all'istante t + Δ		Uscita
	x	y_1	y_2	y_1'	y_2'	z
*	0	0	0	0	0	0
	1	0	0	0	1	0
*	1	0	1	0	1	0
	0	0	1	1	1	1
*	0	1	1	1	1	1
	1	1	1	1	0	1
*	1	1	0	1	0	1
	0	1	0	0	0	0

Fig.V.10 - Analisi del comportamento del circuito di fig.V.9.

dell'uscita, associato agli stati del circuito, è scritto nei nodi che rappresentano gli stati, non sulle transizioni. Ad esempio, poiché l'ingresso x = 1 applicato allo stato in cui $y_1 = y_2 = z = 1$ manda il circuito, dopo un transitorio δ, nello stato in cui $y_1 = z = 1$ e $y_2 = 0$, è disegnato un segmento orientato 1 dal nodo 11/1 al nodo 10/1.

La presenza, nel diagramma, di percorsi chiusi sullo stesso nodo, indica che se l'ingresso x = 0 (x = 1) porta allo stato q^*, il circuito rimane in q^* per tutto il tempo in cui la x rimane al valore 0 (1). Questa proprietà formale caratterizza tutti i diagrammi dei circuiti sequenziali asincroni, e traduce una delle ipotesi fondamentali del loro funzionamento.

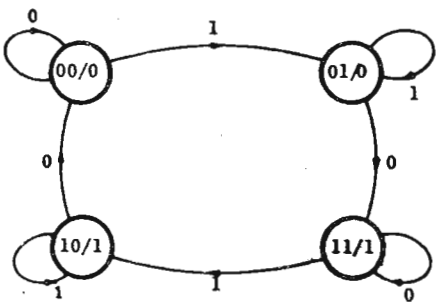


Fig.V.11 - Diagramma di stato del circuito di fig.V.9.

V.5 - Macchine e circuiti sequenziali.

Partendo dalla definizione di circuito sequenziale, che fa riferimento alla corrispondenza tra le sequenze di ingresso e di uscita di un

circuito, abbiamo trovato un insieme di equazioni che possono generare la corrispondenza stessa, almeno da un punto di vista matematico.

Abbiamo poi cercato di interpretare queste equazioni con un circuito ideale, e siamo pervenuti a un *modello fondamentale* dei circuiti sequenziali. Nell'esaminare questo modello, abbiamo visto che la possibilità che esso funzioni correttamente è legata a certe relazioni temporali tra l'entità dei ritardi con cui le y_i prendono il valore delle y_i' e la velocità di variazione degli ingressi, ed abbiamo trovato due modi diversi per soddisfare queste relazioni, modi che conducono ai circuiti ideali sincroni e asincroni.

Siamo però sempre rimasti nell'ambito dei circuiti ideali, di modelli cioè ricavati dalle equazioni matematiche. Per avere dei circuiti reali, realizzabili ad esempio con diodi e transistor, è necessario precisare il significato circuitale degli elementi di ritardo e delle relazioni temporali fra i predetti elementi e gli ingressi.

In definitiva, per risolvere i problemi relativi ai circuiti sequenziali reali, occorre:

– *per l'analisi:*

a) trovare le regole per passare dal circuito reale al modello ideale;

b) scrivere le equazioni relative al modello ottenuto;

c) costruire il diagramma degli stati delle equazioni;

– *per la sintesi:*

d) decidere se il circuito sarà sincrono o asincrono;

e) costruire il diagramma degli stati del circuito;

f) ricavare un modello ideale, possibilmente minimo, dal diagramma;

g) trasformare il modello ideale in un circuito reale.

I passi indicati con le lettere a) ... g) non offrono tutti la stessa difficoltà né hanno la stessa importanza. In particolare:

– i passi a), d) e g) sono di natura elettronica, quindi legati alla scelta dei componenti adoperati. Il passo b) è molto semplice, e si esaurisce nello scrivere le $(m + k)$ funzioni d'uscita di un normale circuito combinatorio multiterminale a $(n + k)$ ingressi. I problemi relativi ai passi c) ed f) sono invece molto complessi, e si risolvono applicando la teoria delle macchine sequenziali. Il passo e), infine, è di natura logica, e consiste nella descrizione del comportamento del circuito a partire dalla relazione fra le sue sequenze di ingresso e di uscita.

Abbiamo già accennato, nel par.V.1, all'esistenza delle macchine sequenziali e alle loro relazioni con i circuiti. Preme qui mettere in evidenza che le macchine sequenziali non vanno identificate con i circuiti, reali o ideali. Quella delle macchine sequenziali è una complessa teoria matematica che, pur avendo molte altre interpretazioni, viene applicata ai modelli ideali dei circuiti sequenziali, e la cui comprensione richiede una buona conoscenza della teoria dei gruppi.

Ci limiteremo, nel seguito, a dare le definizioni fondamentali e i metodi di rappresentazione delle macchine sequenziali e poiché, in definitiva, esistono teorie diverse per le macchine sincrone e asincrone, ne esporremo le conclusioni trattando – separatamente – l'analisi e la sintesi dei corrispondenti circuiti.

V.6 - Macchine sequenziali.

Una macchina sequenziale, o a stati finiti, $\mathfrak{M}$ è un automatismo astratto ad n ingressi ed m uscite, definito da:

– un insieme finito $q_1 q_2 \dots q_s = Q$ di stati interni;

– un insieme finito $i_1 i_2 \dots i_e = I$ di valori degli ingressi $x_1 x_2 \dots x_n$;

– un insieme finito $w_1 w_2 \dots w_u \equiv W$ di valori delle uscite $z_1 z_2 \dots z_m$;

– un insieme di regole definite in una *mappa di transizione* τ, che specifica lo stato q' raggiunto dalla macchina per effetto del valore i degli ingressi applicato allo stato q;

– un insieme di regole definite in una *mappa delle uscite* ω, che specifica i valori w assunti dalle uscite $z_1 z_2 \dots z_m$ per effetto del valore i degli ingressi applicato allo stato q.

La macchina $\mathfrak{M}$ viene indicata con l'insieme delle grandezze che la definiscono:

$$\mathfrak{M} = (Q, I, W, \tau, \omega) \ .$$

Si dicono *complete* o completamente specificate, le macchine che – a partire da ogni stato – ammettono tutti i valori degli ingressi specificando, per ognuno di essi, i valori di q e di w. Si chiamano *incomplete*, o incompletamente specificate, le macchine che non sono complete.

Si chiama *sequenza* degli ingressi, delle uscite o degli stati, ogni successione ordinata di elementi di I, di W o di Q (es.: $i_2 i_5 i_3 i_1 i_1 i_1 i_3$; $w_1 w_5$; $q_1 q_2 q_3 q_4 q_1$).

Una macchina $\mathcal{M}$ si rappresenta, generalmente, con un grafo, il *diagramma degli stati*, che può essere dato in due forme diverse, dette *modello di Moore* e *modello di Mealy*.

Nel *modello di Moore* si fanno dipendere le uscite della $\mathcal{M}$ unicamente dagli stati: ogni stato q_j della macchina è rappresentato da un nodo, che contiene il numero j e i valori delle uscite ad esso relativi. Dei segmenti orientati uniscono i nodi secondo le indicazioni della mappa di transizione τ.

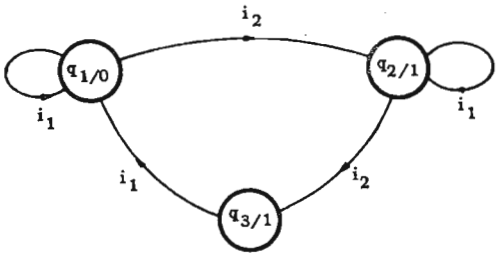


Fig.V.12 - Diagramma di stato secondo Moore.

Nella fig.V.12 è riportato un diagramma degli stati di una macchina a tre stati, incompleta, secondo questo modello: i_1 e i_2 possono indicare, per esempio, i valori 0 e 1 di un unico ingresso x.

Nel *modello di Mealy* le uscite dipendono dagli stati e dagli ingressi: ogni stato q_j è rappresentato da un nodo che contiene soltanto il numero j; le mappe ω e τ sono riportate sui segmenti orientati che uniscono i vari nodi. La fig.V.13 riporta, a titolo d'esempio, la rappresentazione, secondo Mealy, della stessa macchina sequenziale di fig.V.12.

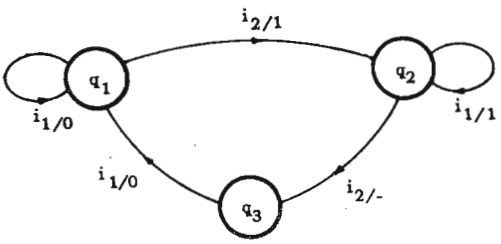


Fig.V.13 - Diagramma di stato secondo Mealy.

Molto usata è anche una descrizione tabellare delle macchine sequenziali, la *tavola di flusso* di Huffman, derivata direttamente dai dia-

grammi degli stati. Le righe di questa tabella contengono gli stati q della macchina, le colonne i valori i degli ingressi, e la generica casella di coordinate (q, i) i valori di q' e z ricavati dal diagramma degli stati. Nella fig.V.14 è mostrata la tavola di flusso corrispondente al diagramma della fig.V.13.

Le rappresentazioni di Moore e Mealy sono perfettamente equivalenti, nel senso che è possibile descrivere la stessa macchina senza ambiguità, nell'uno o nell'altro modello. È anzi possibile trasformare direttamente un diagramma di Moore in uno di Mealy e viceversa, in modo che alle stesse sequenze d'ingresso corrispondano le stesse sequenze d'uscita, con le seguenti regole:

Stati	Ingressi i_1	Ingressi i_2
q_1	$q_1/0$	$q_2/1$
q_2	$q_2/1$	$q_3/-$
q_3	$q_1/0$	$-/-$

Fig.V.14 - Tavola di flusso secondo Huffman.

a) per passare dal diagramma degli stati secondo Moore a quello secondo Mealy, basta eliminare le uscite da ogni nodo, e scriverle su tutti i segmenti orientati che portano al nodo stesso. La trasformazione conserva il numero dei nodi e la forma del grafo. Nelle figg.V.15a e V.15b è mostrato un esempio di tale trasformazione.

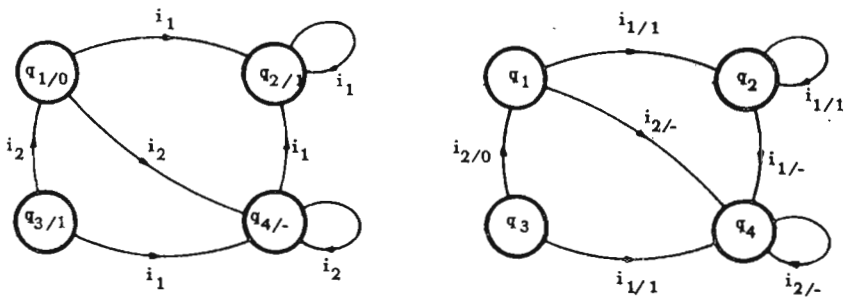
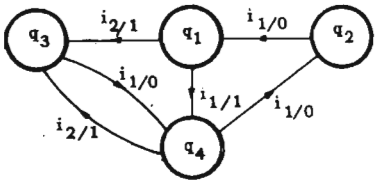


Fig.V.15 - Passaggio da un diagramma di Moore a quello di Mealy.

b) Per passare dal diagramma secondo Mealy a quello secondo Moore, bisogna costruire un grafo con tanti nodi quanti sono gli stati raggiunti dai segmenti orientati con diversi valori delle uscite. In ogni nodo si scriverà il numero di uno stato con l'uscita relativa. I segmenti orientati dovranno rispettare le relazioni τ e ω del grafo originario.

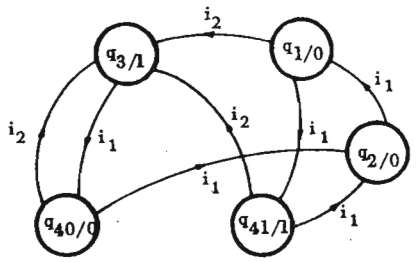
Ad esempio, il diagramma degli stati della fig.V.16a contiene 2 stati (q_3 e q_4) che ricevono due segmenti orientati. Di questi, q_3 non viene sdoppiato, perché entrambi gli itinerari comportano uscita 1, mentre

q_4 origina uno stato q_{40} con uscita 0, che riceve l'ingresso i_1 proveniente da q_3, e uno stato q_{41} con uscita 1, che riceve l'ingresso i_1 proveniente da q_1. Dai due stati q_{40} e q_{41} vanno anche previsti dei collegamenti con lo stato q_3 sotto l'ingresso i_2, e con lo stato q_2 sotto l'ingresso i_1. In definitiva, si ottiene il modello di Moore della fig.V.17a: non vengono conservati né gli stati né la forma del diagramma, sono pertanto diverse anche le tavole di flusso (figg. V.16b e V.17b).



Stato	i_1	i_2
q_1	$q_4/1$	$q_3/1$
q_2	$q_1/0$	-
q_3	$q_4/0$	-
q_4	$q_2/0$	$q_3/1$

Fig.V.16 - Diagramma di Mealy e corrispondente tavola di flusso.



Stato	i_1	i_2
$q_1/0$	q_{41}	q_3
$q_2/0$	q_1	-
$q_3/1$	q_{40}	-
$q_{40}/0$	q_2	q_3
$q_{41}/1$	q_2	q_3

Fig.V.17 - Diagramma di Moore e tavola di flusso per una macchina equivalente a quella di fig.V.16.

Si verifica facilmente che qualsiasi sequenza di ingressi applicata alle macchine delle figg. V.16 e V.17, produce sempre la stessa sequenza di uscite. In pratica, l'uno o l'altro modello danno risultati eguali ma, a seconda delle relazioni tra le sequenze d'ingresso e d'uscita, può convenire usare l'uno piuttosto che l'altro: su questo argomento torneremo in seguito.

I diagrammi degli stati di Moore e Mealy coincidono con quelli introdotti nell'analisi dei circuiti sequenziali e, poiché i primi descrivono una macchina $\mathcal{M}$, i secondi un circuito C, è intuitiva l'identificazione di

$\mathcal{M}$ con C, se i relativi diagrammi coincidono. E' importante notare che i diagrammi degli stati possono essere disegnati dalla descrizione verbale del comportamento della macchina.

Esempio 1: *Diagramma degli stati di una macchina (fig.V.18) avente come ingressi i bit di ordine j di due numeri binari A e B da addizionare ($x_1 = a_j$, $x_2 = b_j$); la macchina può*

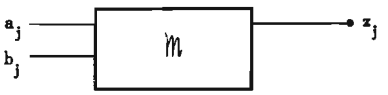


Fig.V.18 - Schema a blocchi di una macchina sequenziale.

esistere in due stati che rappresentano – rispettivamente – il valore 0 e 1 del riporto, proveniente dall'addizionare delle cifre di ordine (j – 1).

I bit a_j e b_j si presentano sugli ingressi ad intervalli di tempo costanti; l'uscita z_j è la somma di a_j, b_j, q_{j-1}. Nella fig.V.19a è mostrato il diagramma degli stati secondo Mealy: q_0 è lo stato che corrisponde al bit di riporto 0; q_1 quello per il bit di ri-

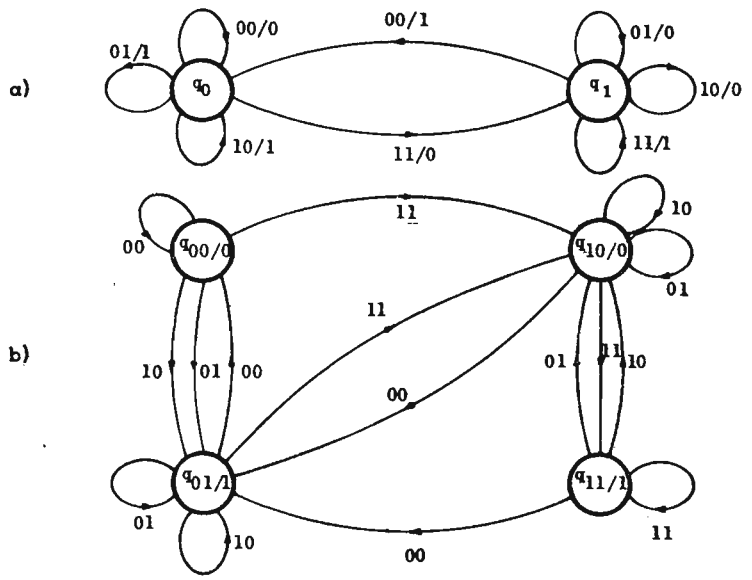


Fig.V.19 - Diagrammi di Mealy (a) e di Moore (b) per la macchina di fig.V.18.

porto 1. Nella fig.V.19b è disegnato il diagramma secondo Moore: è questo un caso evidente della diversa complessità dei 2 diagrammi: il comportamento delle uscite dell'addizionatore binario è, del resto, chiaramente dipendente anche dagli ingressi, oltre che

dal valore del riporto. Le condizioni di ingresso sono quattro, tante quante sono le configurazioni di due variabili, essendo ammessi tutti i valori di a_j e b_j.

Il circuito sequenziale che si comporta come questa macchina si chiama *addizionatore binario in serie*.

Esempio 2: *Diagramma degli stati di una macchina* $\mathfrak{M}$ *a un ingresso e tre uscite, una e una sola delle quali ha valore 1. Se* z_j *è l'uscita 1, al variare dell'ingresso da 0 a 1,* z_j *va a 0, e l'uscita* z_{j+1} *va ad 1* $(j = 1, 2, 3)$.

L'uscita della macchina dipende esclusivamente dal suo stato. I diagrammi secondo Moore e Mealy sono mostrati nelle figg. V.20a e V.20b. Il circuito sequenziale che si comporta come questa macchina si chiama *contatore ad anello*.

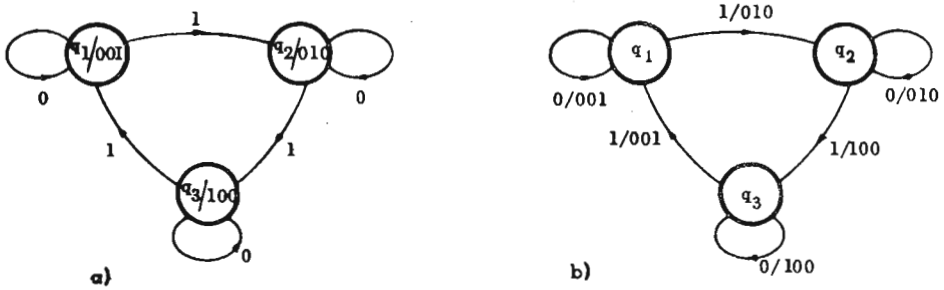


Fig.V.20 - Diagrammi di Moore e Mealy per una stessa macchina sequenziale.

Per le macchine sequenziali sono importanti le seguenti definizioni:

a) uno stato q_j di una macchina $\mathfrak{M}$ si chiama *stabile* se ogni ingresso i che porta a q_j lascia la $\mathfrak{M}$ in q_j. Un esempio di stato stabile è quello della fig.V.21.

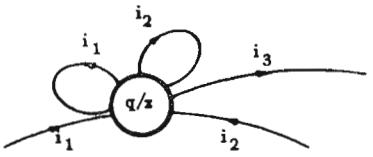


Fig.V.21 - Stato stabile di una macchina sequenziale.

b) Uno stato q_j di una macchina $\mathfrak{M}$ si chiama *instabile* se esiste almeno un ingresso i che porta a q_j e, successivamente, fa evolvere la $\mathfrak{M}$ verso uno stato $q_k \neq q_j$. Un esempio di stato instabile è quello della fig.V.22.

c) Una macchina sequenziale si dice *asincrona* se ogni suo stato è stabile; *sincrona* se contiene almeno uno stato instabile. È asincrona, ad esempio, la macchina della fig.V.19a; sono sincrone quelle delle figg.V.19b

e V.20. Dalla definizione segue che una macchina asincrona può cambiare di stato solo in conseguenza di una variazione degli ingressi; in altre parole, la sequenza di ingresso $i_j i_j i_k i_k i_k i_m i_m$ provoca la stessa sequenza di uscita della sequenza $j_j i_k i_m$.

Si noti, che le definizioni di macchina sincrona e asincrona non coincidono con quelle dei circuiti. Come vedremo in seguito, anzi, qualsiasi diagramma degli stati si può realizzare con un circuito sincrono, ma solo un certo tipo di diagrammi – che si riferiscono a macchine asincrone – si può realizzare con un circuito asincrono. Ciò non significa, tuttavia che alcune relazioni fra sequenze d'ingresso e d'uscita portano necessariamente a circuiti sincroni, ma soltanto che le relazioni stesse vanno interpretate diversamente a seconda del tipo di circuito cui si deve arrivare.

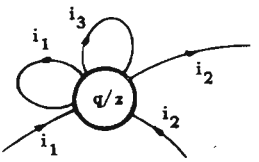


Fig.V.22 - Stato instabile di una macchina sequenziale.

d) Una sequenza d'ingresso $i_1 i_2 \dots i_p$ è *applicabile* alla macchina $\mathcal{M}$ nello stato q se, per ogni i_j su q_j $(1 \leq j \leq p-1)$ esiste lo stato q_{j+1} e se è definita l'uscita *finale* $\omega\,(q_p, i_p)$.

e) Una macchina sequenziale $\mathcal{M}'$ si dice *equivalente* ad una macchina sequenziale $\mathcal{M}$ se tutte le sequenze d'ingresso σ_I applicabili ad uno stato q di $\mathcal{M}$ sono applicabili ad uno stato q' di $\mathcal{M}'$ e producono un'uscita finale $\omega' = \omega$. In altre parole, ogni sequenza σ_I di ingressi produce una sequenza di uscite $\sigma_{\omega'}$ in $\mathcal{M}'$ eguale alla sequenza σ_ω delle uscite di $\mathcal{M}$ almeno in tutti i valori specificati di quest'ultima. Ad esempio, se la macchina $\mathcal{M}_1$, sotto la sequenza $i_1 i_2 i_3 i_4$, a partire dallo stato $q^{(1)}$ produce le uscite 0-10 e la macchina $\mathcal{M}_c$, sotto la stessa sequenza, a partire dallo stato $q^{(2)}$ produce le uscite 0110, si dirà che la $\mathcal{M}_2$ è equivalente alla $\mathcal{M}_1$; il contrario è però falso.

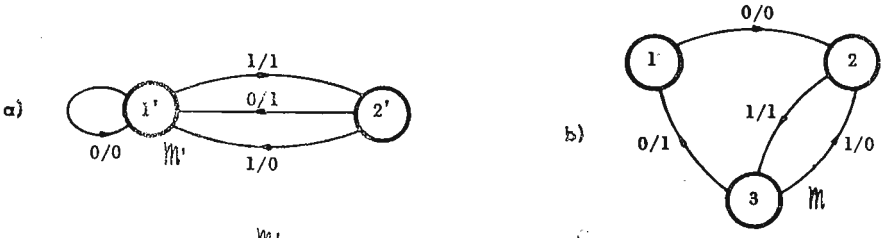


Fig.V.23 - Macchine $\mathcal{M}'$ equivalente alla macchina $\mathcal{M}$.

A titolo d'esempio, nella fig.V.23a è mostrata una macchina $\mathcal{M}'$ equivalente alla macchina $\mathcal{M}$ della fig.V.23b. È facile verificare che tutte le sequenze d'uscita di $\mathcal{M}'$ a partire dallo stato 1' e 2' sono rispetti-

vamente – eguali a quelle ottenibili applicando tutte le condizioni di ingresso alla coppia di stati 1-2 e allo stato 3.

f) Si chiama *macchina minima* equivalente a una macchina $\mathcal{M}$ quella, tra le macchine $\mathcal{M}'$, che ha il minimo numero di stati. Si chiama *minimizzazione degli stati di* $\mathcal{M}$ il procedimento di ricerca della sua macchina equivalente minima.

g) Si chiama *codificazione degli stati di* $\mathcal{M}$ il procedimento che, assegnando i valori di un certo numero di variabili $y_1 y_2 \ldots y_k$ ad ogni stato di $\mathcal{M}$, permette di tradurre in una forma analitica, sostanzialmente equivalente alle equazioni (V.2) e (V.3), le mappe di transizione e di uscita della macchina. Per mezzo della codificazione degli stati, dunque, si passa dalla macchina al circuito sequenziale, o meglio al suo modello ideale.

La minimizzazione degli stati è un problema molto importante per le macchine sequenziali, e lo è per la sintesi dei circuiti nella misura in cui la riduzione degli elementi di ritardo significa una effettiva diminuzione del costo. La minimizzazione degli stati, infatti, essendo il loro numero legato a quello dei segnali interni (y) e degli elementi di ritardo (Δ) porta automaticamente alla minimizzazione della relativa parte di circuito (rete di memoria). Tale minimizzazione può avvenire, però, a spese della rete logica combinatoria, che può risultare molto più complessa e costosa, in particolare nei circuiti asincroni, dove gli elementi delle due reti hanno identico costo.

Pur con le riserve esposte, la minimizzazione degli stati rimane l'unico criterio per la sintesi dei circuiti sequenziali. Grande importanza hanno anche i criteri di assegnazione degli stati secondari che, prescindendo dal costo degli elementi di memoria, conducono ai circuiti combinatori più semplici. Tali criteri sono molto diversi per le macchine – – e i circuiti – sincroni o asincroni: tratteremo quindi separatamente i due casi.

V.6.1 - Minimizzazione degli stati di una macchina sequenziale $\mathcal{M}$.

Il procedimento di minimizzazione consiste nel ricavare, da una macchina $\mathcal{M}$, la macchina equivalente minima $\mathcal{M}'$ riducendo il numero degli stati della $\mathcal{M}$ stessa.

A questo proposito si tenga presente che una macchina sequenziale è caratterizzata non tanto dalla forma del suo diagramma degli stati o dal numero degli stati stessi, quanto dalla corrispondenza tra l'insieme delle sequenze d'ingresso e l'insieme delle sequenze d'uscita.

Supponiamo, ora, che 2 stati q_j e q_k sono tali che:

a) qualsiasi sequenza d'ingresso $\sigma_I = i_1 i_2 \ldots i_p$ applicabile a q_j è applicabile a q_k;

b) l'uscita finale $\omega(q_{p,k}, i_p)$ è sempre uguale all'uscita $\omega(q_{p,j}, i_p)$ (con $q_{p,i}$ è indicato lo stato finale raggiunto per effetto di σ_I su q_i).

Una volta che la macchina $\mathcal{M}$ sarà giunta nello stato q_j, genererà – per ogni σ_I – sequenze d'uscita σ_ω i cui elementi, se specificati, saranno generabili anche dalla stessa σ_I su q_k. L'evoluzione della $\mathcal{M}$ a partire da q_k sarà così *non in contrasto* con l'evoluzione della $\mathcal{M}$ a partire da q_j.

Per chiarire questo concetto è bene, a questo punto, precisare la natura della macchina $\mathcal{M}$, dato che le conseguenze del verificarsi delle condizioni a) e b) sono diverse a seconda che $\mathcal{M}$ sia completa o no.

Per le *macchine complete*, ogni σ_I è applicabile e genera sequenze d'uscita σ_ω completamente specificate.

Per tali macchine, dunque, se le condizioni a) e b) sussistono per q_k nei riguardi di q_j, debbono sussistere anche per q_j nei riguardi di q_k.

Gli stati q_j e q_k si dicono stati *equivalenti* ($q_j = q_k$), nel senso che l'evoluzione della $\mathcal{M}$ a partire da q_j e da q_k è identica.

Se $q_j = q_k$, è possibile considerare q_j e q_k un solo stato, e *fonderli* nello stato q_{jk}, se si fa in modo che tutti gli ingressi che portavano la $\mathcal{M}$ in q_j e/o in q_k la portino nel nuovo stato q_{jk}. In pratica, questo si ottiene cancellando, nella tabella di flusso, la riga q_k e sostituendo, dovunque, q_j a q_k.

Quanto detto per due stati si può ripetere per un numero qualsiasi di stati. Si dimostra anzi (vedi, ad esempio, [9] in bibliografia) che gli stati equivalenti possono essere fusi in modo univoco e che il procedimento di fusione porta ad una *partizione* degli stati stessi in *classi* di stati equivalenti.

Per le *macchine incomplete*, l'esistenza di condizioni non specificate in qualche σ_ω rende possibile che le condizioni a) e b) siano valide per lo stato q_k nei riguardi di q_j ma non viceversa, nel senso che possono esistere sequenze σ_I applicabili a q_k ma non a q_j.

Si parla, infatti, per tali macchine, di stati *compatibili* ($q_j \sigma q_k$), non di stati equivalenti (poiché l'equivalenza è più restrittiva della compatibilità, se $q_j = q_k$ è anche $q_j \sigma q_k$).

Anche se l'evoluzione della $\mathcal{M}$ a partire da q_j non è uguale a quella che parte da q_k (perché quest'ultima ha elementi della sequenza d'uscita che nella prima non sono specificati), q_j e q_k possono ancora essere fusi con le modalità esposte per gli stati equivalenti delle macchine complete, assegnando alle uscite non specificate i valori determinati della σ_ω relativa a q_k.

Se per 3 stati q_j, q_k, q_e si ha:

$$q_j \sigma\, q_k$$
$$q_j \sigma\, q_e$$
$$q_k \not\sigma\, q_e$$

i 3 stati non possono essere fusi insieme, ma q_j va fuso con q_k e/o con q_e. Le 3 possibilità danno luogo a macchine diverse (vedi esempi seguenti). La minimizzazione delle macchine incomplete non porta, dunque, necessariamente ad un'unica soluzione.

*

Il procedimento di minimizzazione che, fra tutti quelli proposti, ci sembra il più semplice, si applica sulla tavola di flusso della macchina $\mathcal{M}$ da minimizzare: è un procedimento esaustivo e fornisce tutte le *coppie* di stati compatibili (o equivalenti, nel caso di macchine complete) della $\mathcal{M}$.

Prima di applicare il metodo conviene esaminare la tavola di flusso per eliminare eventuali stati *doppi* q^*, che hanno cioè eguali valori di q' e z di altri stati q già esistenti.

Se esistono stati di questo tipo, si eliminano dalla tabella sopprimendo la riga q^* e sostituendo, in ogni casella, q a q^*.

Esempio: *Dalla tavola di flusso della fig.V.24a, si può eliminare lo stato 4 che è «doppio» dello stato 1. Si ottiene, così, la tavola di fig.V.24b.*

a)

	i_1	i_2
1	2/1	3/0
2	3/0	4/1
3	2/1	1/0
4	2/1	3/0

b)

	i_1	i_2
1	2/1	3/0
2	3/0	1/1
3	2/1	1/0

Fig.V.24 - Eliminazione di uno stato doppio da una tavola di flusso.

Sulla tavola di flusso, che non contiene stati doppi, si applica il metodo di minimizzazione, di seguito riportato.

a) Le coppie di stati compatibili (o equivalenti) vengono individuate per esclusione delle coppie di stati non compatibili. Si comincia coll'eliminare le coppie di stati q_m e q_j che, almeno per un ingresso i, hanno uscite diverse.

Le coppie di stati equivalenti, se esistono, sono un sottoinsieme delle coppie (q_j, q_k) per cui si ha:

$$\omega(q_j, i) = \omega(q_k, i) \qquad \text{(V.1)}$$

per ogni i.

Gli stati q_j e q_k per i quali vale la (V.1) si dicono *compatibili rispetto alle uscite* $(q_j \alpha q_k)$.

b) Si costruisce una tabella con tante righe quante sono le coppie di stati equivalenti rispetto alle uscite, e tante colonne quanti sono gli ingressi. In ogni casella si scrivono le coppie di stati verso cui evolve la $\mathcal{M}$ per effetto dell'ingresso i sulla coppia (q_j, q_k).

c) Si esamina la tabella. Se la casella di coordinate $(q_j' q_k', i')$ contiene una coppia di stati non compatibili rispetto alle uscite, cioè non compresa tra le coppie delle righe della tabella, la coppia (q_j', q_k') *non* è equivalente, perché gli stati q_j' e q_k' evolvono, sotto l'ingresso i', verso stati con uscite diverse.

Si cancella allora dalla tabella la riga (q_j', q_k').

d) Si riprende l'esame della tabella, cancellando tutte le coppie di stati (q_j'', q_k'') che evolvono verso le coppie cancellate al passo c).

e) Si ripete il procedimento finché non è possibile cancellare altre coppie. Le coppie di stati rimaste sono formate da stati compatibili (o equivalenti). Ognuna di tali coppie, infatti, evolvendo verso un solo stato, o verso una coppia di stati con uscite compatibili (o eguali), realizza sequenze d'uscita compatibili (o eguali) per ogni sequenza di ingresso.

f) Si cerca di formare gruppi di $n(n-1)/2$ coppie composte da n stati, con n massimo. In tali gruppi, ogni stato è legato in coppia con i restanti $(n-1)$ stati del gruppo. Esiste, cioè, una relazione di mutua compatibilità che permette di fondere gli n stati in uno solo.

g) Si distribuiscono gli stati di $\mathcal{M}$ in s sottoinsiemi $S_1, S_2 \ldots S_s$, con s minimo, in modo che:

– ogni S_i contenga solo stati compatibili;

– ogni q_j di $\mathcal{M}$ sia contenuto in almeno uno degli S_i (per le macchine complete, in uno solo degli S_i);

– per ogni ingresso i e per ogni S_i esista uno S_j tale che l'ingresso i faccia evolvere *tutti* gli stati di S_i in stati di S_j.

h) Si passa, infine, alla macchina $\mathcal{M}'$ fondendo tutti gli stati di S_i nello stato q'_i.

Gli esempi che seguono chiariranno l'applicazione del metodo.

Esempio 1: *Minimizzazione degli stati per la macchina completa di fig.V.25.*

Stato	i_1	i_2	i_3
1	4/00	4/11	4/11
2	5/01	4/10	2/01
3	4/00	5/00	6/00
4	5/01	6/10	2/01
5	6/00	6/11	6/11
6	1/01	6/10	2/01

Fig.V.25 - Tavola di flusso di una macchina completa a 6 stati.

– Le coppie compatibili rispetto all'uscita (passo a) sono:

$$(1 \cdot 5),\ (2 \cdot 4),\ (2 \cdot 6),\ (4 \cdot 6)\ .$$

– La tabella che mostra l'evoluzione degli stati delle coppie compatibili (passo b) è riportata nella fig.V.26.

	i_1	i_2	i_3
1 • 5	4 • 6	4 • 6	4 • 6
2 • 4	5 • 5	4 • 6	2 • 2
2 • 6	5 • 1	4 • 6	2 • 2
4 • 6	5 • 1	6 • 6	2 • 2

Fig.V.26 - Tabella di evoluzione degli stati α-compatibili.

– L'esame della tabella (passo c) non porta all'eliminazione di nessuna coppia. Tutte le coppie, pertanto, sono formate da stati equivalenti.

– Esiste una relazione di mutua compatibilità (passo f) fra le 3 coppie:

$$(2 \cdot 4),\ (2 \cdot 6),\ (4 \cdot 6)\ .$$

Gli stati 2, 4, 6 possono quindi essere fusi in un solo stato.

– Gli stati di $\mathcal{M}$ risultano, in definitiva ripartiti in 3 insiemi:

$$\{1 \cdot 5\},\ \{2, 4, 6\},\ \{3\}$$

ad essi si fanno corrispondere, ordinatamente, gli stati 1', 2', 3' della macchina $\mathcal{M}'$ (passo g).

– La tavola di flusso della $\mathcal{M}'$ (passo h) è riportata nella fig.V.27.

Stati equivalenti di $\mathcal{M}$	Stati di $\mathcal{M}'$	i_1	i_2	i_3
1 • 5	1'	2'/00	2'/11	2'/11
2 • 4 • 6	2'	1'/01	2'/10	2'/01
3	3'	2'/00	1'/00	2'/00

Fig.V.27 - Tavola di flusso della macchina minima equivalente alla macchina della fig.V.25.

Esempio 2: *Minimizzazione degli stati per la macchina completa di fig.V.28 (nella tavola, numeri eguali indicano configurazioni d'uscita uguali; ogni configurazione d'uscita è per ipotesi, completamente specificata).*

Stato	i_1	i_2	i_3	i_4
1	1/4	4/1	6/3	6/4
2	6/3	4/3	1/1	7/4
3	3/3	6/2	4/4	8/1
4	6/3	4/3	5/1	3/4
5	1/4	4/1	6/3	6/4
6	3/1	5/3	6/2	3/3
7	7/3	6/2	2/4	8/1
8	1/3	8/2	4/4	6/1

Fig.V.28 - Tavola di flusso di una macchina completa a 8 stati.

La tavola di flusso mostra che lo stato 5 è doppio dello stato 1, essendo le corrispondenti righe della tabella identiche. L'eliminazione dello stato 5 porta alla tabella di fig.V.29.

– Le coppie di stati compatibili rispetto alle uscite sono:

$$(2 \cdot 4),\ (3 \cdot 7),\ (3 \cdot 8),\ (7 \cdot 8)\ .$$

Stato	i_1	i_2	i_3	i_4
1	1/4	4/1	6/3	6/4
2	6/3	4/3	1/1	7/4
3	3/3	6/2	4/4	8/1
4	6/3	4/3	1/1	3/4
6	3/1	1/3	6/2	3/3
7	7/3	6/2	2/4	8/1
8	1/3	8/2	4/4	6/1

Fig.V.29 - Tavola di flusso derivata dall'eliminazione dello stato 5 della tavola di fig.V.28.

– La fig.V.30 mostra l'evoluzione delle coppie di stati compatibili.

– Gli stati 3 e 8, dipendendo dalle compatibilità degli stati 3 e 1, che non sono compatibili, debbono essere cancellati dalla tabella.

– Gli stati 7 e 8 sono incompatibili perché evolvono verso gli stati 7 • 1 che non sono compatibili, quindi vanno cancellati dalla tabella.

– Esistono due sole coppie di stati rimasti, che risultano pertanto equivalenti:

$$2 = 4$$
$$3 = 7 \ .$$

	i_1	i_2	i_3	i_4
2 • 4	6 • 6	4 • 4	1 • 1	7 • 3
3 • 7	3 • 7	6 • 6	4 • 2	8 • 8
3 • 8	3 • 1	6 • 8	4 • 4	8 • 6
7 • 8	7 • 1	6 • 8	2 • 4	8 • 6

Fig.V.30 - Tabella di evoluzione degli stati α-compatibili.

Stati di $\mathcal{M}$	Stati di $\mathcal{M}'$	i_1	i_2	i_3	i_4
1	1'	1'/4	2'/1	4'/3	4'/2
2 • 4	2'	4'/3	4'/3	1'/1	3'/4
3 • 7	3'	3'/3	4'/2	2'/4	5'/1
6	4'	3'/1	1'/3	4'/2	3'/3
8	5'	1'/4	5'/2	2'/4	4'/1

Fig.V.31 - Tavola di flusso della macchina minima $\mathcal{M}'$ equivalente alla macchina della fig.V.28.

– Nella fig.V.31 è riportata la tavola di flusso della macchina equivalente minima $\mathcal{M}'$.

Esempio 3: *Minimizzazione degli stati della macchina incompleta di fig.V.32.*

Stato	i_1	i_2
1	1/-	2/1
2	3/1	1/1
3	2/0	1/1

Fig.V.32 - Tavola di flusso di una macchina incompleta a 3 stati.

– La tavola di flusso mostra che esistono 2 coppie di stati compatibili rispetto alle uscite:

$$1 \alpha 2$$
$$1 \alpha 3 \ .$$

– La tabella di evoluzione delle coppie α-compatibili (fig.V.33) mostra che l'unico modo per avere una macchina minima $\mathcal{M}'$ a 2 stati, rispettando le condizioni esposte nel punto g), è quello di formare i 2 sottoinsiemi non disgiunti:

$$S_1 \equiv \{1 \cdot 2\}$$
$$S_2 \equiv \{1 \cdot 3\}$$

contenenti entrambi lo stato 1.

	i_1	i_2
1 • 2	1 • 3	1 • 2
1 • 3	1 • 2	1 • 2

Fig.V.33 - Tabella di evoluzione degli stati α-compatibili.

– Nella fig.V.34 è mostrata la macchina $\mathcal{M}'$ che, a differenza della $\mathcal{M}$ risulta completamente specificata.

Stati di $\mathcal{M}$	Stati di $\mathcal{M}'$	i_1	i_2
1 • 2	1'	2'/1	1'/1
1 • 3	2'	1'/0	1'/1.

Fig.V.34 - Tavola di flusso della macchina minima $\mathcal{M}'$ equivalente alla macchina di fig.V.32.

Esempio 4: *Minimizzazione degli stati della macchina incompleta di fig.V.35.*

– La tabella di flusso mostra l'esistenza delle seguenti coppie compatibili rispetto alle uscite:

(1 • 4), (1 • 5), (1 • 6), (2 • 3), (2 • 6), (3 • 6), (4 • 5), (4 • 6), (5 • 6) .

– L'evoluzione delle coppie α-compatibili è riportata nella fig.V.36.

– Esistono 2 sole coppie che risultano incompatibili: la (2 • 3) e la (2 • 6), che evolvono verso gli stati α-incompatibili (2 • 5) e (3 • 5).

Stato	i_1	i_2	i_3	i_4
1	–/–	5/0	4/1	3/1
2	3/1	2/1	4/1	1/0
3	6/1	5/1	1/1	3/0
4	–/–	6/0	–/–	3/1
5	5/1	1/0	–/–	3/1
6	5/1	–/–	4/1	–/–

Fig.V.35 - Tavola di flusso di una macchina incompleta a 6 stati.

– Nell'insieme delle coppie compatibili:

(1 • 4), (1 • 5), (1 • 6), (3 • 6), (4 • 5), (4 • 6), (5 • 6)

esiste un gruppo di 6 coppie formato dai 4 stati (1, 4, 5, 6) che soddisfa alla condizione f): gli stati predetti possono pertanto essere fusi in uno solo.

	i_1	i_2	i_3	i_4
1 • 4	– • –	5 • 6	4 • –	3 • 3
1 • 5	– • 5	5 • 1	4 • –	3 • 3
1 • 6	– • 5	5 • –	4 • 4	3 • –
3 • 6	6 • 5	5 • –	1 • 4	3 • –
4 • 5	– • 5	6 • 1	– • –	6 • 3
4 • 6	– • 5	6 • –	– • 4	6 • –
5 • 6	5 • 5	1 • –	– • 4	3 • –

Fig.V.36 - Tabella di evoluzione degli stati α-compatibili.

Stati di $\mathcal{M}$	Stati di $\mathcal{M}'$	i_1	i_2	i_3	i_4
1 • 4 • 5 • 6	1'	1'/1	1'/0	1'/1	3'/1
2	2'	3'/1	2'/1	1'/1	1'/0
3 • 6	3'	1'/1	1'/1	1'/1	3'/0

Fig.V.37 - Tavola di flusso della macchina minima $\mathcal{M}'$ equivalente alla macchina di fig.V.35.

– Nella fig.V.37 è mostrata la macchina equivalente minima $\mathcal{M}'$ che ha 3 soli stati.

V.7 - Circuiti sincroni e asincroni.

Una semplice classificazione dei circuiti sequenziali reali è basata sulle caratteristiche dei loro segnali esterni, che si distinguono in sincroni o asincroni, a seconda che le relative variazioni avvengono in sincronismo con un clock o in istanti qualsiasi, e in impulsivi o a livelli, a seconda delle caratteristiche della tensione che li rappresenta.

Più precisamente, i segnali x d'ingresso, o di comando, di un circuito sequenziale si dividono in quattro tipi:

a) Segnali a livelli asincroni, che possono cambiare di valore ad ogni istante, col solo vincolo che due variazioni successive siano distanziate di un tempo superiore a quello di risposta del circuito, definito come lo intervallo di tempo massimo fra l'applicazione di una condizione d'ingresso su un certo stato stabile e il raggiungimento dello stato stabile corrispondente.

Nella fig.V.38a è riportato il segnale $x = 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0$.

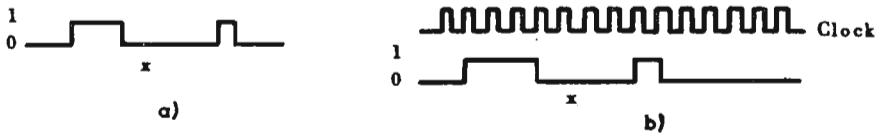


Fig.V.38 - Segnali a livelli asincroni (a) e sincroni (b).

b) Segnali a livelli sincroni, che possono cambiare valore solo in determinati istanti, per esempio in corrispondenza del fronte di salita degli impulsi di un clock. Nella fig.V.38b è riportato il segnale $x = 0 \rightarrow 1 \rightarrow 0 \rightarrow \rightarrow 1 \rightarrow 0$.

Fig.V.39 - Segnali impulsivi asincroni (a) e sincroni (b).

c) Segnali impulsivi asincroni, costituiti da brevi impulsi di tensione, di ampiezza e durata costanti, prodotti in istanti qualsiasi, col solo vinco-

lo che l'intervallo tra due impulsi consecutivi superi il tempo di risposta del circuito. Nella fig.V.39a è mostrato il segnale $x = 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1$, da interpretare come: 3 impulsi x agli istanti t_1, t_2, t_3.

d) Segnali impulsivi sincroni, costituiti da brevi impulsi di tensione che hanno significato solo in corrispondenza con gli impulsi C di un clock. Il segnale x vale 1 o 0 a seconda che $x \cdot C = 1$ o $x \cdot C = 0$.

Nella fig.V.39b è riportato il segnale $x = 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0$, da interpretare come: 4 impulsi x in coincidenza con gli impulsi 2, 3, 7, 9 di clock.

Nei circuiti sequenziali reali:

a) Gli elementi logici comandati dai segnali a livelli cambiano di stato in coincidenza coi fronti di salita e di discesa dei segnali stessi, cioè ad ogni loro variazione. Nei circuiti impulsivi i cambiamenti sono invece provocati dal solo fronte di salita o di discesa, a seconda del tipo di logica usata, cioè ogni impulso viene avvertito come un unico comando.

b) E' sempre possibile cambiare il tipo di comando di un circuito. Nella fig.V.40, a titolo d'esempio, è mostrata la trasformazione del fronte di salita di un segnale a livelli in impulso esponenziale positivo, mediante un semplice circuito differenziatore RC, seguito da un diodo.

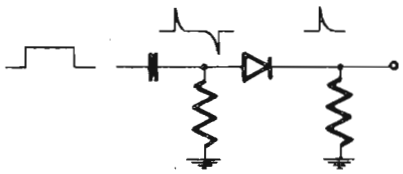


Fig.V.40 - Trasformazione di un segnale a livelli in uno impulsivo.

c) Le uscite, che nel modello ideale abbiamo distinte in interne (y') ed esterne (z), possono essere ad impulsi o a livelli, a seconda del tipo di circuito; gli ingressi interni y sono, invece, sempre a livelli.

La differenza tra circuiti a livelli e a impulsi è molto importante dal punto di vista elettronico, ma lo è scarsamente dal punto di vista logico, nel senso che lo stesso modello di macchina sequenziale può descrivere il primo o il secondo tipo di circuiti, come sarà chiaro dagli esempi che daremo nei prossimi capitoli. Di primaria importanza è invece la distinzione tra circuiti asincroni e sincroni, non solo per i diversi criteri di progetto, ma anche per il diverso tempo di risposta agli stessi comandi.

Supponiamo, ad esempio, di avere due circuiti logicamente identici (fig.V.41a) ed entrambi in uno stato tale che, all'arrivo di un segnale opportuno sull'ingresso x_1, l'uscita z_A passi da 0 a 1, e sia il primo di questi circuiti asincrono, il secondo sincrono.

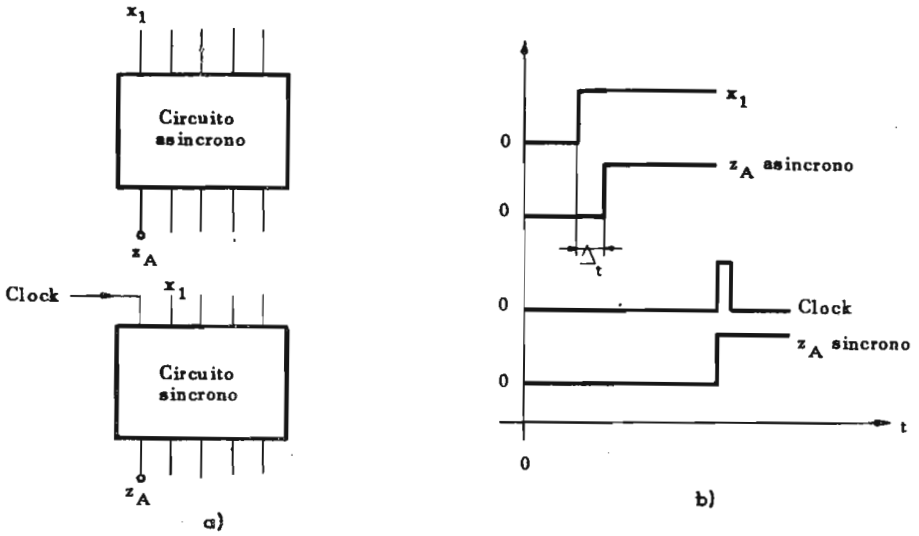


Fig.V.41 - Effetti di una variazione d'ingresso su un circuito asincrono e sincrono.

Se il segnale $x_1 = 1$ viene applicato all'istante $t = 0$ sui terminali x_1 dei due circuiti, l'uscita $z_A = 1$ si avrà nel circuito asincrono dopo un ritardo Δ_t dovuto esclusivamente al tempo di propagazione del segnale nei suoi elementi logici; nel circuito sincrono, invece, soltanto quando sarà comparso l'impulso di clock (fig.V.41b).

*

BIBLIOGRAFIA

1. BARTEE T.C. - LEBOW I.L. - REED I.S.: *Theory and Design of Digital Machines* - Mc Graw-Hill, 1962.

2. CALDWELL S.M.: *Switching circuits and Logical Design* - Wiley & Sons, 1958.

3. GILL A.: *Introduction to Theory of Finite-State Machines* - Mc Graw-Hill, 1962.

4. GINSBURH S.: *An Introduction to Mathematical Machine Theory* - Addison-Wesley, 1902.

5. HUFFMAN D.A.: *The Synthesis of Sequential Switching Circuits* - Journal of the Franklin Institute, 1954.

6. Mc CLUSCKEY E.J. & BARTEE T.C.: *A Survey of Switching Circuit Theory* - Mc Graw-Hill, 1962.

7. MEALY G.H.: *A Method for Synthesizing Sequential Circuits* - Bell System TJ - Volume 34, settembre 1955.

8. MOORE E.F.: *Gedanken-Experiments on Sequential Machines* - Automata Studies - N.34 - Princeton University, 1956.

9. MILLER R.E.: *Switching Theory* - Vol.II - John Wiley & Sons, 1965.

10. GINSBURG: *A Synthesis Technique for Minimal State Sequential Machines* - IRE Transaction on EC - Vol.8, N.1, marzo 1959.

11. MILLER R.E.: *State Reduction for Sequential Machines* - IBM Research Report - giugno 1959.

*

CAPITOLO VI

CIRCUITI SEQUENZIALI ASINCRONI

VI.1 - Generalità.

Nei capitoli precedenti sono stati esposti i concetti fondamentali della teoria dei circuiti sequenziali, definendo tali circuiti dal punto di vista operativo e introducendo i loro modelli ideali e le loro equazioni caratteristiche. Sono stati quindi elencati i problemi relativi alla costruzione di un modello ideale a partire dalla descrizione del suo comportamento, e risolti alcuni di essi con la teoria delle macchine a stati finiti. Si è rimandata al seguito la soluzione dei restanti problemi e il passaggio dal modello ideale al circuito reale che origina le stesse sequenze d'uscita sotto le stesse sequenze di ingresso.

In questo capitolo ci occuperemo dell'analisi e della sintesi dei circuiti sequenziali asincroni, definiti come quei circuiti sequenziali comandati da segnali di tipo asincrono, a impulsi o a livelli. Il metodo di studio adottato è quello proposto da Huffman, basato sul modello ideale già discusso, e riportato nella fig.VI.1, alle cui notazioni ci atterremo sempre. Non faremo alcuna distinzione tra circuiti a livelli o a impulsi, concependo questi ultimi come il risultato di due successive variazioni di livello.

VI.2 - Analisi.

Un circuito sequenziale asincrono, in pratica, si riconosce perché è formato esclusivamente da elementi logici, in generale NAND o NOR, e contiene almeno un collegamento tra l'uscita di un elemento su un livello j e l'ingresso di un elemento su un livello (j + k). Questo collegamento si chiama un *loop di reazione* o semplicemente un *loop*.

Nella fig.VI.2 sono disegnati tre circuiti sequenziali asincroni.

Un circuito asincrono reale è apparentemente assai diverso dal modello sul quale vogliamo basare il nostro studio: la differenza maggiore è la mancanza – nel primo – di qualsiasi elemento di ritardo.

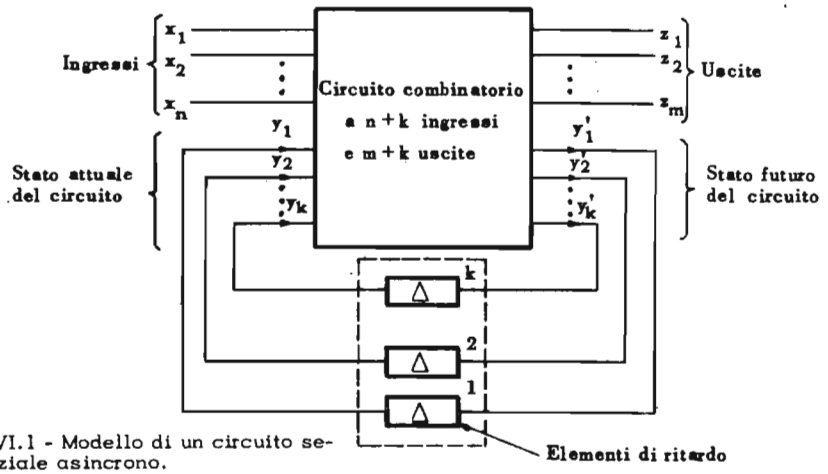


Fig.VI.1 - Modello di un circuito sequenziale asincrono.

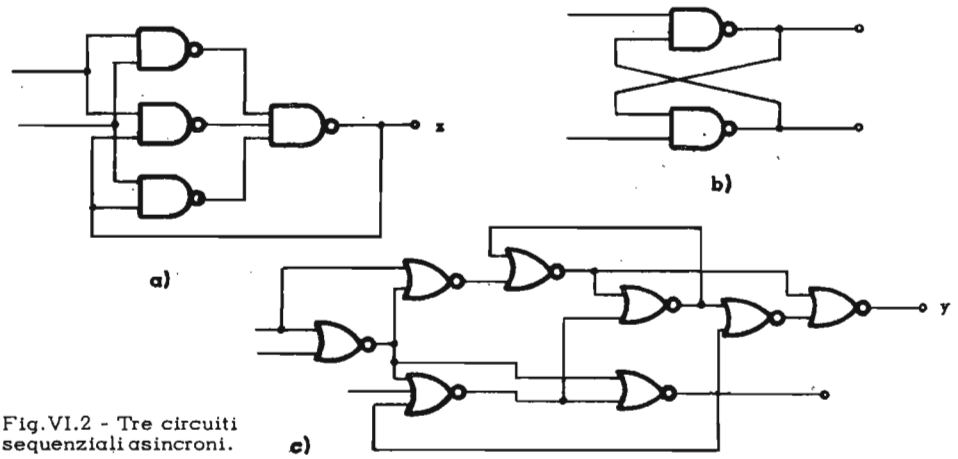


Fig.VI.2 - Tre circuiti sequenziali asincroni.

Nello studio dei circuiti combinatori abbiamo tacitamente supposta istantanea la risposta di ogni elemento logico: nel senso, per esempio, che l'uscita $z = 0$ di un NAND a 2 ingressi x_1 e x_2 entrambi a 1, per effetto della variazione di uno degli ingressi all'istante t, passava nello stesso istante al valore 1.

In realtà, come già accennato nel cap.III, esiste sempre un ritardo tra l'applicazione di una condizione d'ingresso che provoca una variazione d'uscita e l'istante in cui detta variazione si verifica, ritardo

che dipende in parte dalle caratteristiche e dal carico dei transistor, in parte dai collegamenti tra le uscite di un livello e gli ingressi del seguente.

La considerazione di questi ritardi, la cui entità è sconosciuta, risulta priva di significato per i circuiti combinatori ma è essenziale per i sequenziali, in quanto permette di introdurre gli elementi di ritardo negli schemi reali. Detti schemi possono infatti essere riportati al modello della fig.VI.1 scomponendo ogni elemento logico reale in un elemento logico istantaneo più un elemento di ritardo Δ, che tiene conto sia del ritardo proprio dei componenti che dei ritardi distribuiti nei collegamenti. Ad esempio, il circuito della fig.V.29, riportato nella fig. VI.3a, può essere scomposto come mostrato nella fig.VI.3b.

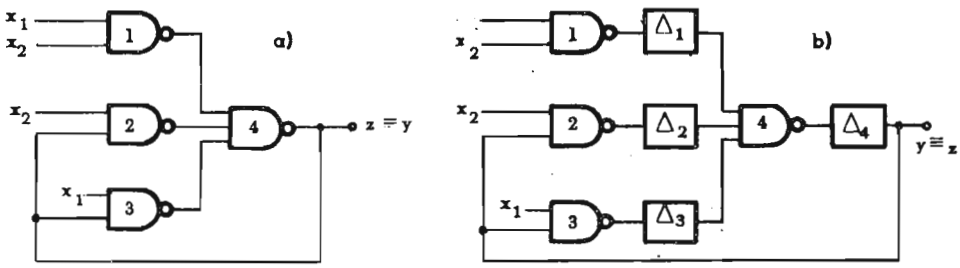


Fig.VI.3 - Separazione degli elementi di ritardo dagli elementi logici in un circuito sequenziale asincrono.

Supponendo poi che tutti i ritardi Δ_i siano eguali tra loro, il circuito può essere ridisegnato sostituendo i quattro elementi di ritardo con uno solo, di grandezza $\Delta = 2\Delta_i$, posto all'uscita del NAND 4. Chiamando $y'(t)$ la funzione di eccitazione, o di ingresso, dell'elemento Δ, e $y(t)$ la sua funzione di risposta o d'uscita, in modo da avere $y(t+\Delta) = y'(t)$ si ottiene finalmente l'analogia formale cercata (fig.VI.4).

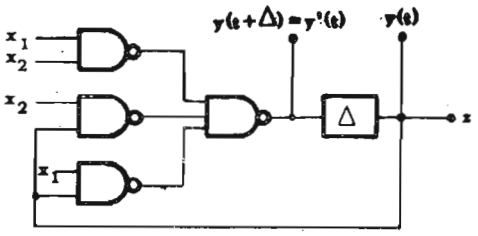


Fig.VI.4 - Circuito sequenziale asincrono di figura VI.3a ripotato allo schema di fig.VI.1.

Le ipotesi fatte, che conducono nel caso generale a inserire tanti elementi di ritardo quanti sono i loop di reazione del circuito, sono valide solo in prima approssimazione, ma sono necessarie per l'applicazione della teoria. Quando occorre, si deve tener conto, nei modi che vedremo, del mancato verificarsi delle ipotesi stesse.

Nel seguito, conformemente alla nomenclatura introdotta nel capitolo precedente, chiameremo le x e le z, ingressi e uscite del circuito; le y e le y', rispettivamente, ingressi e uscite esterne, oppure funzione di risposta e di eccitazione dell'elemento di ritardo Δ. Per lo studio dei circuiti utilizzeremo le equazioni delle z e delle y' nella forma:

$$z_i = z_i(x_1 x_2 \dots x_n \; y_1 y_2 \dots y_k) \qquad (i = 1, 2, \dots, m) \tag{VI.1}$$

$$y'_j = y'_j(x_1 x_2 \dots x_n \; y_1 y_2 \dots y_k) \qquad (j = 1, 2, \dots, k) \; . \tag{VI.2}$$

Ad esempio, per il circuito della fig.VI.4 si hanno le equazioni:

$$\begin{cases} z = y \\ y' = x_1 x_2 + x_2 y + x_1 y \; , \end{cases} \tag{VI.3}$$

dove è scomparso qualsiasi riferimento al tempo. In realtà, il carattere temporale di queste equazioni è implicito nella relazione tra y e y', in quanto le (VI.3) possono essere interpretate così:

$$\begin{cases} y'(t) = y(t+\Delta) = x_1 x_2 + x_2 y(t) + x_1 y(t) \\ z(t) = y(t) \; . \end{cases} \tag{VI.4}$$

L'analisi si effettua a partire dalle equazioni (VI.1) e (VI.2) del circuito, e mira alla costruzione della tabella di flusso o del diagramma degli stati. Per facilitare l'analisi stessa, è opportuna la divisione degli stati in stabili e instabili, a seconda che si abbia, rispettivamente:

– per tutti gli j: $y'_j(t) = y_j(t)$;

– oppure, almeno per uno degli j: $y'_j(t) \neq y_j(t)$;

in corrispondenza ad una data configurazione di $(x_1 x_2 \dots x_n y_1 y_2 \dots y_k)$.

Per l'esistenza tra y e y' dell'elemento di ritardo, un circuito non può rimanere in uno stato instabile ma deve evolvere spontaneamente verso lo stato stabile corrispondente. La definizione data di stato stabile e instabile è sostanzialmente equivalente a quella del capitolo precedente.

Esporremo il metodo d'analisi illustrando alcuni esempi:

Esempio 1: *Analisi del circuito sequenziale della fig.VI.5.*

Il circuito, considerando il NOR di ingressi R e Z_2 sul primo livello, e quello di ingressi S e Z_1 sul secondo (fig.VI.6a) ha 2 uscite e un loop di reazione; l'elemento

di ritardo e le relative funzioni di eccitazione y' e risposta y vanno inseriti come mostrato nella fig.VI.6b.

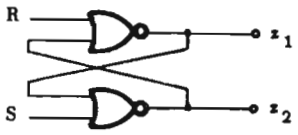


Fig.VI.5 - Circuito sequenziale asincrono a 2 ingressi e 2 uscite.

Dalle equazioni del circuito:

$$\text{(VI.5)} \qquad \begin{cases} y' = \overline{R}(S+y) = \overline{R}S + \overline{R}y \\ z_1 = y \\ z_2 = \overline{S}\,\overline{y} \end{cases}$$

si ricavano due tabelle, molto simili alle mappe di Karnaugh, che hanno tante colonne quante sono le configurazioni degli ingressi (cioè 2^n) e tante righe quanti sono gli stati del circuito (2^k).

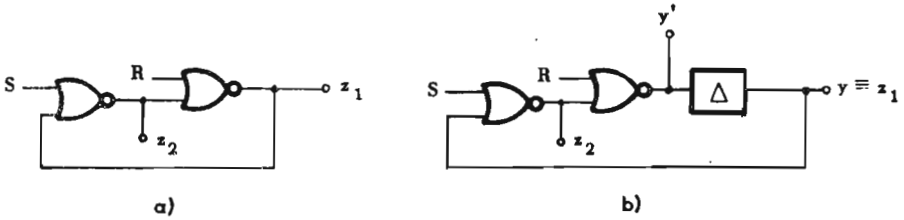


Fig.VI.6 - Schema del circuito di fig.VI.5.

Per il circuito della fig.VI.5, si hanno due ingressi (R, S) ed una variabile secondaria (y); le tabelle avranno quindi (fig.VI.7) 4 colonne, 2 righe e 8 caselle, tante quante sono le configurazioni diverse in cui può trovarsi il circuito, configurazioni ottenibili combinando uno qualsiasi dei 2^k stati con uno dei 2^n valori degli ingressi.

	RS			
y	00	01	11	10
0				
1				

Fig.VI.7 - Tabella per lo studio del circuito di fig.VI.5.

La prima delle due tabelle, dedicata agli stati del circuito, si chiama *matrice di eccitazione* e si costruisce riempiendo ogni casella con i valori che le y' assumono in corrispondenza ai valori delle x e delle y che ne costituiscono le coordinate. Nella figura VI.8 è riportata la matrice di eccitazione del nostro circuito: i valori delle y' sono stati ricavati dalla prima delle (VI.5).

La seconda tabella, dedicata alle uscite del circuito, si chiama *matrice d'uscita*, e si costruisce riempiendo ogni casella con i valori che le z assumono in corrispondenza alle sue coordinate. La matrice d'uscita, ricavata dalla seconda e dalla terza delle (VI.5) è rappresentata nella fig.VI.9: i due valori di ogni casella si riferiscono, rispettivamente, a z_1 e z_2.

y \ RS	00	01	11	10
0	0	1	0	0
1	1	1	0	0

y'

Fig.VI.8 - Matrice di eccitazione del circuito di fig.VI.5.

y \ RS	00	01	11	10
0	01	00	00	01
1	10	10	10	10

z_1z_2

Fig.VI.9 - Matrice d'uscita del circuito di fig.VI.5.

La matrice di eccitazione contiene tutte le informazioni relative all'evoluzione del circuito, e permettere di distinguere immediatamente gli stati stabili dagli instabili. Ogni casella rappresenta infatti il valore y'(t), all'istante t, della funzione di eccitazione dell'elemento di ritardo, in corrispondenza al valore preesistente della sua funzione di risposta y e all'applicazione – all'istante t – delle condizioni d'ingresso RS; per essere y(t) l'ordinata della casella stessa, sono stati stabili quelli i cui valori coincidono con la propria ordinata. Sono, ad esempio, stati stabili quelli delle caselle RSy = 011 e 110; sono instabili quelli delle caselle RSy = 010 e 101.

Nella fig.V.10, gli stati stabili sono racchiusi in un circoletto.

Fig.VI.10 - Stati stabili e instabili del circuito di fig.VI.5.

y \ RS	00	01	11	10
0	(0)	1	(0)	(0)
1	(1)	(1)	0	0

y'

Dire che lo stato y' individuato da RSy = 010 è instabile, significa dire che se il circuito si trova in uno stato in cui y = 0 e i suoi ingressi vengono portati al valore RS = 01, la funzione di eccitazione dell'elemento di ritardo passa ad 1 ed è diversa dalla sua funzione di risposta, ancora uguale a 0. Poiché, per la definizione stessa di circuito asincrono, i comandi RS rimangono a 01 finché non si arriva ad un nuovo stato stabile, il circuito dovrà evolvere verso lo stato individuato dalla colonna RS = 01 e dalla riga y = y' = 1, stato che raggiungerà dopo un tempo Δ, conformemente alla relazione $y(t+\Delta) = y'(t)$.

Il passaggio da uno stato instabile al corrispondente stato stabile è dunque spontaneo. Quando il circuito si trova in uno stato stabile, vi rimane invece finché non si

verificano cambiamenti degli ingressi. E poiché tali cambiamenti hanno effetto istantaneo su y', ma modificano y (se y' è $\neq$ y) soltanto dopo un tempo Δ, il circuito reagisce ad una variazione dalle x da $[x^0]$ a $[x^*]$ passando prima allo stato individuato dalla stessa riga di y e dalla colonna $[x^*]$ in seguito a quello individuato dalla colonna $[x^*]$ e dalla riga y'. Ad esempio, se il circuito si trova nello stato rappresentato dalla casella RSy = 000 e si porta – all'istante t – l'ingresso S al valore 1, si passa nello stato instabile (y' = 1) della colonna RS = 01 e della riga 0; spontaneamente il circuito passa poi, all'istante $t+\Delta$, nello stato stabile (y = 1) della casella RSy = 011.

In conclusione, sulla tabella:

– Gli spostamenti orizzontali sono provocati dall'esterno, variando gli ingressi del circuito.

– Gli spostamenti verticali avvengono spontaneamente, e riportano il circuito, da uno stato instabile in cui si trovava per la variazione degli ingressi, nello stato stabile corrispondente.

La matrice di eccitazione può essere scritta in una forma diversa, chiamata *tabella degli stati*, in cui si assegna, a tutti gli stati stabili di ogni riga, uno stesso numero, racchiuso in un cerchio, e – a tutti gli stati instabili – il numero senza cerchio degli stati stabili verso cui essi evolvono. Nella fig.VI.11 è mostrata la tabella degli stati, derivata dalla matrice della fig.VI.8. Questa rappresentazione è particolarmente usata nel procedimento di sintesi.

Fig.VI.11 - Tabella degli stati del circuito di fig.VI.5.

y \ RS	00	01	11	10
0	①	2	①	①
1	②	②	1	1

y'

Come la matrice di eccitazione descrive l'evoluzione interna, cioè la successione degli stati del circuito, la matrice d'uscita ne descrive il comportamento esterno, cioè la risposta alle variazioni degli ingressi. A differenza della prima, questa seconda matrice è del tutto indipendente dal tempo, perché le z, le x e le y si riferiscono tutte allo stesso istante [vedi le equazioni (VI.2)].

Fig.VI.12 - Tavola di flusso del circuito di fig.VI.5.

y \ RS	00	01	11	10
0	⓪/01	1/00	⓪/00	⓪/01
1	①/10	①/10	0/10	0/10

$y'z_1z_2$

Unendo la matrice di eccitazione, o la tabella degli stati, alla matrice d'uscita, si ha finalmente la tavola di flusso (fig.VI.12), che costituiva lo scopo della nostra a-

nalisi, e che differisce da quella introdotta per le macchine sequenziali unicamente perché in essa si usa mettere in evidenza gli stati stabili. Sfruttando quanto esposto nel capitolo precedente, è immediatamente ricavabile, da questa tavola, un diagramma degli stati del circuito. Nella fig.VI.13 è appunto mostrato il diagramma secondo Moore; nella fig.VI.13b quello secondo Mealy.

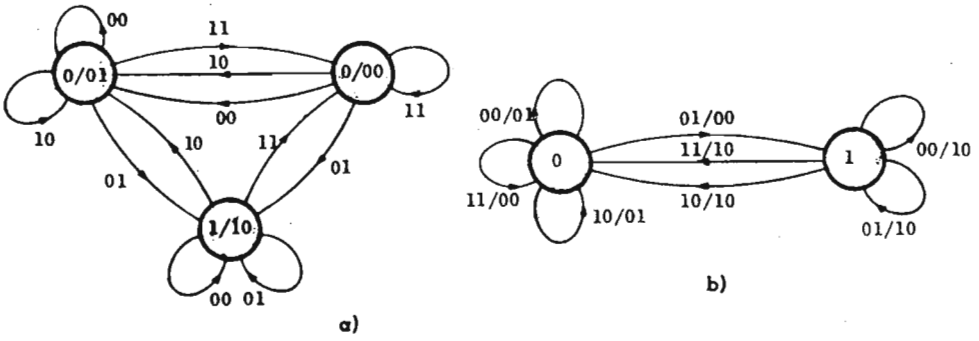


Fig.VI.13 - Diagramma di stato del circuito di fig.VI.5: secondo Moore (a) e Mealy (b).

L'interpretazione della tavola di flusso, o del diagramma degli stati, è immediata. A titolo d'esempio, la sequenza d'uscita provocata dall'applicazione della sequenza d'ingresso $x_1x_2 = 00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 01 \rightarrow 11 \rightarrow 01 \rightarrow 00$ a partire dallo stato $y = 0$, è: $z_1z_2 = 01 \rightarrow 10 \rightarrow 00 \rightarrow 01 \rightarrow 10 \rightarrow 00 \rightarrow 10 \rightarrow 10$.

Nel costruire le matrici, quindi la tavola di flusso e il diagramma degli stati, abbiamo ammesso qualsiasi variazione delle condizioni d'ingresso. In realtà, nei circuiti asincroni, per ragioni che vedremo in seguito, si evita la variazione contemporanea di più ingressi.

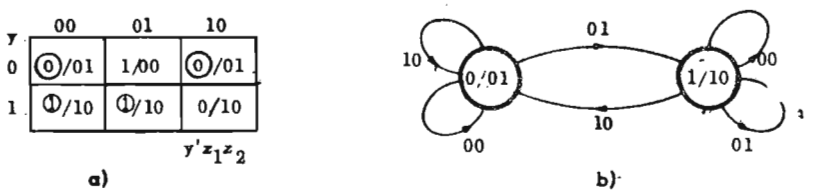


Fig.VI.14 - Tavola di flusso e diagramma di stato del circuito di figura VI.5, se si esclude l'ingresso RS = 11.

Il circuito analizzato è di uso assai comune: i suoi comandi sono impulsi di tensione che possono presentarsi sull'ingresso R o nell'ingresso S, ma mai contemporaneamente, e sempre partendo dalla condizione SR = 00. Scomponendo gli impulsi in variazioni di livello, possono così presentarsi le due sequenze di ingresso:

$$RS = 00 \rightarrow 10 \rightarrow 00 \; ; \qquad RS = 00 \rightarrow 01 \rightarrow 00 \; .$$

Nella tavola di flusso non ha allora più senso considerare la colonna RS = 11 (fig.VI.14a) e, nel diagramma di Moore possono essere eliminati il nodo 0/00 e tutte le transizioni che ad esso fanno capo (fig.VI.14b); questo stato infatti non è mai raggiungibile perché non esistono più gli ingressi RS = 11.

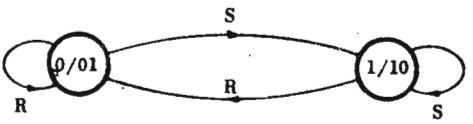


Fig.VI.15 - Rappresentazione semplificata del diagramma di stato della fig.VI.14b.

Il grafo della fig.VI.14b, mettendo in evidenza gli impulsi invece dei livelli, cioè indicando con R l'arrivo di un impulso di tensione sull'ingresso R che riassume le sequenze RS = 00→10→00; e con S l'arrivo di un impulso su S che riassume la sequenza RS = 00→01→00, prende la forma, più evidente, della fig.VI.15, sul cui particolare significato ci soffermeremo a lungo nel prossimo capitolo.

Il circuito a due ingressi della fig.VI.5, pertanto, poiché:

- ha due uscite di valore sempre opposto;
- esiste in due soli stati stabili (y = 0 e y = 1);
- è lasciato, da un impulso sull'ingresso R, nello stato 0;
- è lasciato, da un impulso sull'ingresso S, nello stato 1;

è un flip-flop RS, del tipo già noto, il cui funzionamento è determinato dai livelli, piuttosto che dai fronti di salita o di discesa, come nei tipi visti al cap.III.

VI.2.1 - Situazioni particolari nelle tavole di flusso.

Nel caso dell'analisi possono presentarsi le situazioni, più o meno complesse, che di seguito vengono presentate su alcune tabelle degli stati, senza riferimento a particolari circuiti.

a) Lo stato stabile viene raggiunto attraverso un certo numero di stati instabili diversi (fig.VI.16). Queste transizioni attraverso stati intermedi permettono importanti semplificazioni, senza altri inconvenienti che un maggior ritardo nel tempo necessario per raggiungere lo stato stabile.

Nella fig.VI.16, per passare dallo stato ② allo stato ① occorre attendere il passaggio da 0 a 1 di y_1, il passaggio da 1 a 0 di y_3 e il passaggio da 1 a 0 di y_1: poiché le 3 variazioni sono successive, passerà un tempo 3Δ. Le transizioni multiple sono indicate con delle frecce nella tabella degli stati.

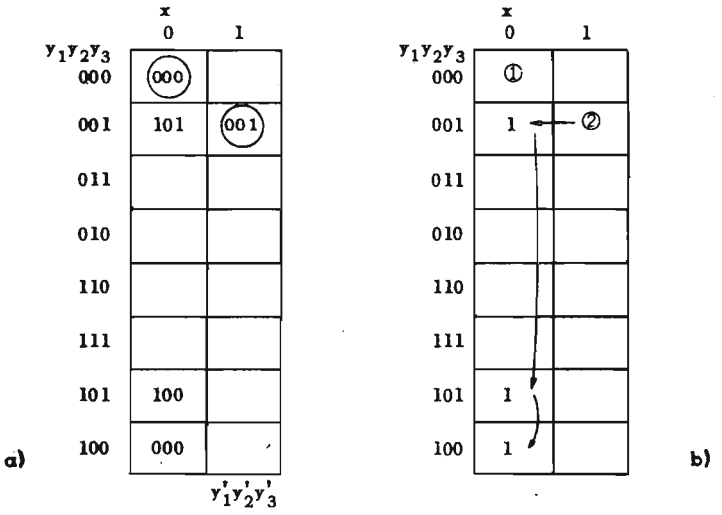


Fig.VI.16 - Transizioni multiple.

b) Non si raggiunge nessuno stato stabile (fig.VI.17), ma viene descritto continuamente un ciclo, finché non vengono cambiati i valori degli ingressi. La condizione deriva, in genere, da una errata impostazione del progetto, tranne casi particolari, in cui si desidera proprio un oscillatore asincrono.

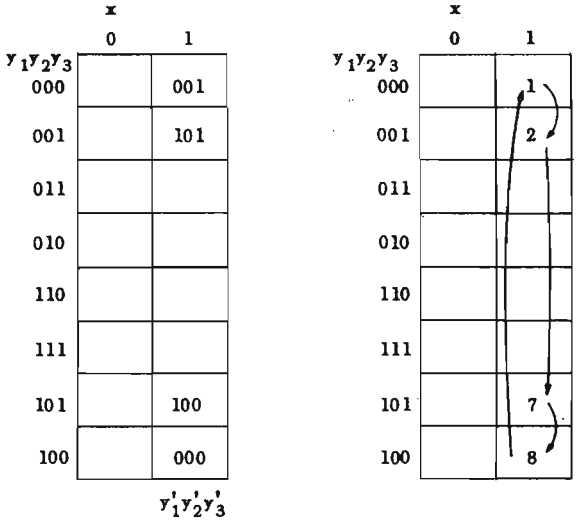


Fig.VI.17 - Cicli.

c) Il passaggio da uno stato instabile al corrispondente stato stabile comporta il cambiamento di più variabili di stato; è il caso, ad esempio, di un circuito a 3 loop di reazione, quindi a 3 elementi di ritardo, che trovandosi nello stato $y_1 y_2 y_3 = 000$, per una variazione degli ingressi da $[x^0]$ a $[x^*]$ assuma la terna di valori $y_1' y_2' y_3' = 111$.

Quando ciò accade, la maggior velocità di risposta di un elemento di ritardo rispetto agli altri, che può verificarsi in pratica, può portare il circuito ad evolvere in modo errato.

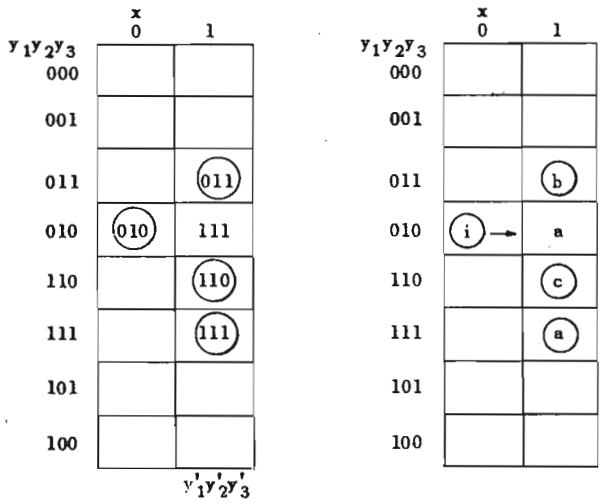


Fig.VI.18 - Corse critiche.

Riferendoci alla fig.VI.18, l'applicazione dell'ingresso x = 1 nello stato stabile (i) in cui $y_1 y_2 y_3 = 010$, porta il circuito nello stato instabile a, in cui $y_1' y_2' y_3' = 111$; l'evoluzione spontanea, che dovrebbe avvenire verso lo stato (a) ($y_1 y_2 y_3 = 111$), è condizionata dal contemporaneo verificarsi del passaggio di y_1 e y_3 da 0 a 1.

Se la y_1 assume il valore della y_3, il circuito raggiunge lo stato stabile non desiderato (c), con $y_1 y_2 y_3 = 110$, da cui potrà muoversi soltanto per effetto di una variazione degli ingressi. Se la y_3 diventa 1 prima della y_1, si raggiunge invece lo stato stabile indesiderato (b), in cui $y_1 y_2 y_3 = 011$. Situazioni del genere vanno sempre evitate, come vedremo nei prossimi paragrafi, e prendono il nome di *corse critiche*.

d) Il passaggio da uno stato instabile al corrispondente stato stabile mediante cambiamento di più variabili interne può essere ammesso quando, qualunque sia l'effettivo valore dei ritardi con cui gli y_i coinvolti nel

passaggio assumono i valori dei rispettivi y_i', si raggiunge sempre lo stesso stato stabile. Si hanno, in questo caso, le cosiddette *corse non critiche*, di cui si ha un esempio nella fig.VI.19: se si pone $x = 0$ a partire dallo stato stabile ②, in cui $y_1 y_2 y_3 = 011$, si ha lo stato instabile in cui $y_1' y_2' y_3' = 000$, e poiché anche gli stati in cui $y_1 y_2 y_3 = 001$ e 010 sono instabili ed evolvono verso lo stato stabile $y_1 y_2 y_3 = 000$, questo viene raggiunto in ogni modo.

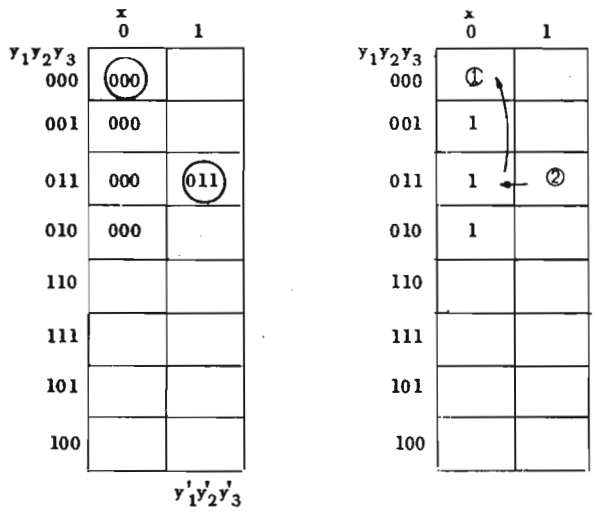


Fig.VI.19 - Corse non critiche.

Prima di passare alla sintesi, riteniamo opportuno effettuare la analisi di un circuito più complesso di quello della fig.VI.5, applicando tutte le considerazioni fin qui esposte.

Esempio: *Analisi del circuito sequenziale asincrono a NAND e invertitori della figura VI.20.*

Esistono due loop di reazione; introducendo gli elementi di ritardo e le denominazioni $y_1 y_2$ e $y_1' y_2'$ per le funzioni di risposta e di eccitazione, si ottiene il circuito (fig.VI.21) nella stessa forma del modello della fig.VI.1. Le equazioni delle y' e delle z sono:

(VI.6)
$$\begin{cases} y_1' = y_1(\overline{x}_1 + \overline{x}_2 + y_2) + \overline{x}_1\overline{x}_2 y_2 = \\ \quad = \overline{x}_1 y_1 + \overline{x}_2 y_1 + y_1 y_2 + \overline{x}_1\overline{x}_2 y_2 \\ y_2' = x_1(y_1 y_2 + x_2\overline{y}_1) + \overline{x}_1\overline{x}_2 y_1 + \overline{x}_1\overline{y}_1 y_2 = \\ \quad = x_1 y_1 y_2 + x_1 x_1\overline{y}_1 + \overline{x}_1\overline{x}_2 y_1 + \overline{x}_1\overline{y}_1 y_2 \\ z = \overline{y}_1 \ . \end{cases}$$

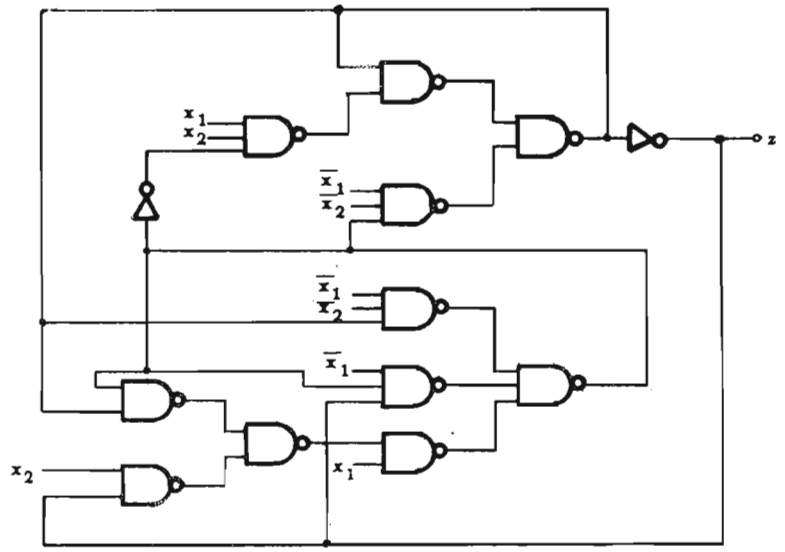


Fig.VI.20 - Circuito sequenziale NAND-NOT.

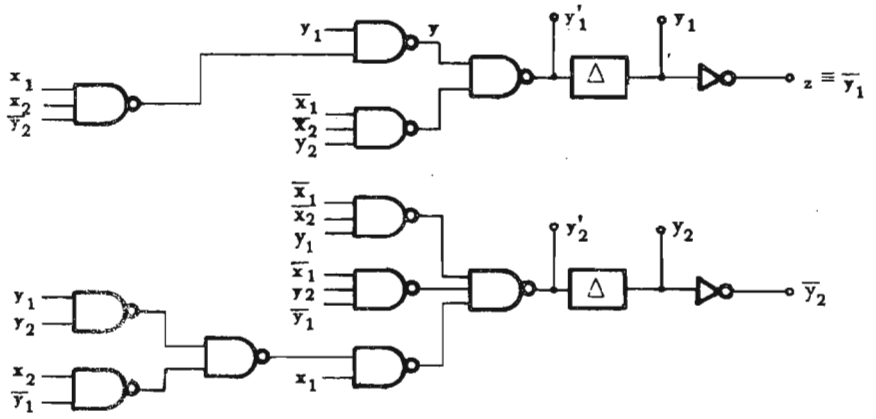


Fig.VI.21 - Rappresentazione del circuito di fig.VI.21 secondo lo schema di fig.VI.1.

a)

y_1y_2 \ x_1x_2	00	01	11	10
00	00	00	01	00
01	11	01	01	00
11	11	10	11	11
10	11	10	00	10

$y'_1y'_2$

b)

y_1y_2 \ x_1x_2	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	0	0	0	0

z

Fig.VI.22 - Matrici d'eccitazione (a) e d'uscita (b) del circuito di fig.VI.20

Le matrici di eccitazione (fig.VI.22a) e d'uscita (VI.22b), a quattro righe e quattro colonne (perché esistono due loop di reazione e due ingressi esterni), si ottengono scrivendo semplicemente in ogni casella i valori di $y_1'y_2'z$ forniti dalle (VI.6).

Nella fig.VI.23a è riportata la tavola di flusso costruita dalle due matrici, e nella fig.VI.23b il relativo diagramma degli stati secondo Moore.

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)/1	(00)/1	01/1	(00)/1
01	11/1	(01)/1	(01)/1	00/1
11	(11)/0	10/0	(11)/0	(11)/0
10	11/0	(10)/0	00/0	(10)/0

a)

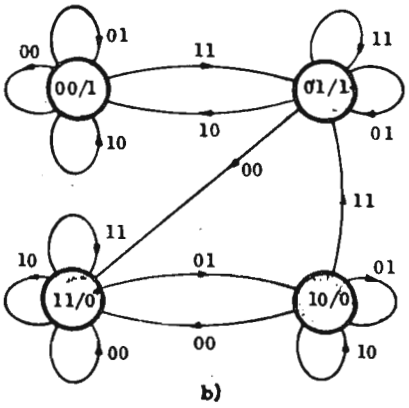


Fig.VI.23 - Tavola di flusso e diagramma di stato del circuito di fig.VI.21.

Esaurita l'analisi dei circuiti, passiamo ora al più complesso problema della sintesi.

VI.3 - Sintesi dei circuiti sequenziali asincroni.

Il problema della sintesi si presenta sempre in questi termini: note le possibili sequenze di valori delle variabili $x_1x_2\ldots x_n$, progettare un circuito a m uscite $z_1z_2\ldots z_m$ la cui successione di valori sia in una certa relazione con quella delle x.

La sintesi si effettua con un procedimento analitico proposto da Huffman, lungo ma rigoroso. Il procedimento in questione è stato diviso, per comodità di trattazione, nelle cinque parti seguenti, che definiremo ed esamineremo separatamente:

1) Costruzione di una tabella degli stati (detta *matrice primitiva delle sequenze*) relativa a una macchina sequenziale asincrona $\mathfrak{M}$, a partire dalla descrizione verbale del comportamento del circuito.

2) Minimizzazione degli stati di $\mathfrak{M}$, e costruzione di una tabella degli stati (detta *matrice delle sequenze*) per la macchina $\mathfrak{M}'$ equivalente minima della $\mathfrak{M}$.

3) Determinazione del numero minimo k di variabili interne necessarie per il funzionamento asincrono del modello ideale di circuito sequenziale derivato dalla $\mathfrak{M}'$, ed assegnazione dei valori di dette variabili alle righe della matrice delle sequenze.

4) Costruzione della tavola di flusso per il modello ideale del circuito.

5) Costruzione del circuito sequenziale asincrono reale, a partire dalla tavola di flusso.

Il modello ideale è sempre quello della fig.VI.1.

Le ipotesi fondamentali poste a base della sintesi sono:

1) tutti gli elementi logici, e l'intera rete, hanno risposta istantanea;

2) i segnali d'ingresso sono del tipo a livello asincroni;

3) ogni variazione degli ingressi avviene mentre il circuito si trova in uno stato stabile;

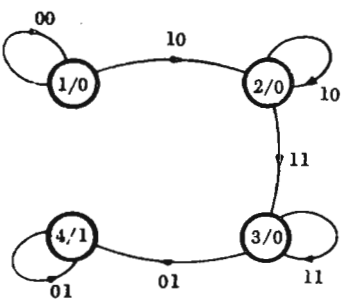
4) i valori Δ relativi agli elementi di ritardo del modello ideale sono tutti eguali.

Queste ipotesi sono soltanto approssimate; nei paragrafi successivi vedremo le conseguenze delle situazioni reali sui circuiti progettati ammettendo le ipotesi predette.

VI.4 - Matrice primitiva delle sequenze.

La matrice primitiva delle sequenze è la tavola di flusso, secondo il modello di Moore, di una macchina sequenziale asincrona $\mathfrak{M}$ costruita col solo scopo di descrivere in maniera completa e precisa il funzionamento del circuito, senza alcuna preoccupazione sul numero degli stati introdotti: va quindi ricavata dal diagramma degli stati, la cui costruzione è stata trattata nel capitolo precedente. Per quanto esposto finora, la matrice deve avere tante colonne quante sono le possibili configurazioni ($\leq 2^n$) degli ingressi, e tante righe quanti sono gli stati stabili della $\mathfrak{M}$; i valori delle uscite corrispondenti allo stato stabile cui la riga è assegnata vanno scritti in una ulteriore colonna; il numero di righe, che non può essere fissato a priori, eguaglia il numero dei nodi del grafo.

Esempio 1: *Costruire la matrice primitiva per un circuito sequenziale a due ingressi* (x_1x_2) *ed un'uscita* (z) *che va ad 1 quando e solo quando, partendo da* $x_1x_2=00$, *si ha prima* $x_1=1$, *poi* $x_2=1$, *infine* $x_1=0$. *L'uscita deve tornare a 0 per ogni cambiamento di* x_1 *o* x_2. *Si supponga che i 2 ingressi non cambino mai di valore contemporaneamente.*



a)

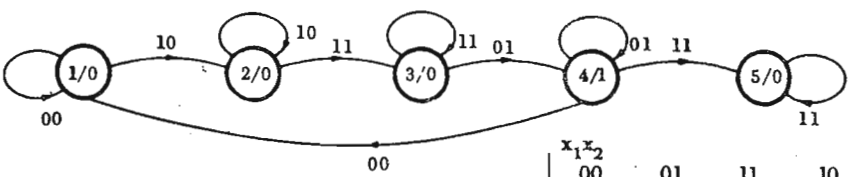
Stati	x_1x_2 00	01	11	10	z
1	①			2	0
2			3	②	0
3		4	③		0
4		④			1

b)

Fig.VI.24 - Procedimento di costruzione della matrice primitiva (1).

La sequenza delle x che porta all'uscita $z=1$ è: $x_1x_2=00\rightarrow10\rightarrow11\rightarrow01$; essa origina i primi 4 nodi del diagramma degli stati (fig.VI.24a) costruiti così: lo stato 1 è quello di riposo; per la variazione $0\rightarrow1$ di x_1, si passa allo stato 2, con uscita 0, e da qui allo stato 3, ancora con uscita 0, quando x varia da 0 a 1. L'uscita assume il valore 1 quando, a partire dallo stato 3, la x_1 torna a 0, portando il sistema nello stato 4. In ogni stato sono stati aggiunti dei loop chiusi sullo stato stesso, ad indicarne la stabilità, derivante dalla definizione di macchina asincrona.

Nella fig.VI.24b è mostrata la parte della matrice corrispondente al grafo.

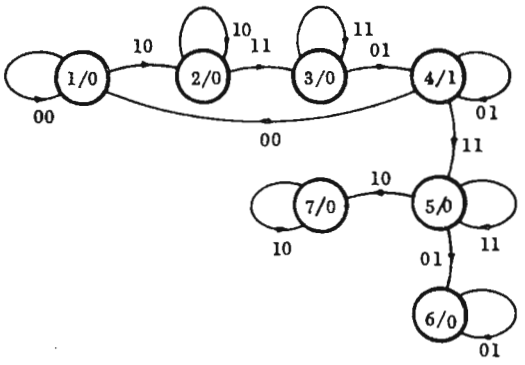


	x_1x_2 00	01	11	10	z
4	1	④	5	-	1

Fig.VI.25 - Procedimento di costruzione della matrice primitiva (2).

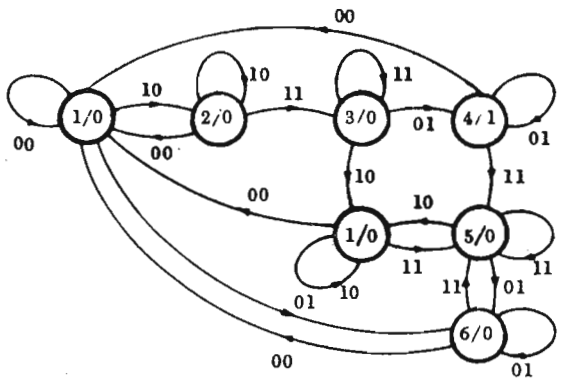
Per completare il diagramma, e la matrice, degli stati, occorre prevedere tutte le altre possibili sequenze degli ingressi. Intanto, stabiliamo che ogni variazione a partire da $x_1x_2=01$ dello stato 4, riporta a 0 l'uscita. Poiché può cambiare un solo ingresso alla volta, si può avere $x_1x_2=01\rightarrow00$ o $x_1x_2=01\rightarrow11$. La prima variazione deve portare a uno stato stabile in cui $x_1x_2=00$, $z=0$: questo stato già esiste (è lo stato 1), quindi ci sarà un collegamento 00 tra 4 ed 1. La variazione $x_1x_2=00\rightarrow11$ deve portare

a uno stato q in cui $x_1x_2 = 11$, $z = 0$; tale stato non può coincidere con il 3: da q_1, infatti, si deve poter tornare all'uscita $z = 1$ solo passando per lo stato 1, mentre da 3 si arriva a $z = 1$ con la sola variazione di x_1 da 1 a 0. Si deve pertanto introdurre un nuovo stato 5, con uscita $z = 0$; poiché la transizione $x_1x_2 = 01 \rightarrow 10$, per ipotesi, non è permessa, si è esaurito lo studio del circuito a partire dallo stato 4 (fig.VI.25).



Stato	x_1x_2 00	01	11	10	z
1	①			2	0
2			3	②	0
3		4	③		0
4	1	④	5	-	1
5	-	6	⑤	7	0
6		⑥			0
7				⑦	0

Fig.VI.26 - Procedimento di costruzione della matrice primitiva (3).



Stato	x_1x_2 00	01	11	10	z
1	①	6	-	2	0
2	1	-	3	②	0
3	-	4	③	7	0
4	1	④	5	-	1
5	-	6	⑤	7	0
6	1	⑥	5	-	0
7	1	-	5	⑦	0

Fig.VI.27 - Procedimento di costruzione della matrice primitiva (4).

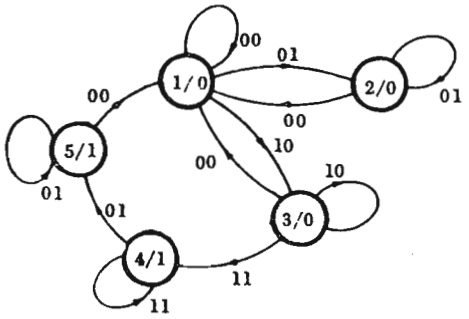
Lo stato 5 è stabile sotto gli ingressi $x_1x_2 = 11$; sono permesse le variazioni $x_1x_2 = 11 \rightarrow 01$ e $11 \rightarrow 10$, che portano a 2 nuovi stati, 6 e 7, entrambi con uscita $z = 0$ (fig.VI.26).

Per completare il diagramma, occorre ancora esaminare tutte le transizioni possibili a partire dagli stati 1, 2, 3, 6, 7.

La sola transizione non prevista a partire dallo stato 1 è la $x_1x_2 = 00 \rightarrow 01$, che deve portare a uno stato q con $z = 0$ da cui non si possa raggiungere l'uscita $z = 1$ se non ripassando per lo stato 1: q coincide dunque con lo stato 6.

Per ragioni analoghe, la $x_1x_2 = 10-00$ porterà dallo stato ② allo stato ① e la $x_1x_2 = 11-10$ dal ④ al ⑦. Dagli stati ⑤ e ⑦, infine, si potrà ritornare allo stato ① o andare al ③. Il diagramma, e la matrice, degli stati (fig.VI.27) sono così completi.

Esempio 2: *Matrice primitiva delle sequenze per un circuito a 2 ingressi (x_1x_2) e una uscita (z) che ripete ogni segnale su x_2 iniziato quando $x_1 = 1$. Si supponga che i segnali su x_1 e x_2 abbiano tutti la stessa durata, che non cambino mai contemporaneamente e che non possa iniziare un segnale su x_1 se esiste un segnale su x_2.*



Stato	x_1x_2 00	01	11	10	z
1	①	2	-	3	0
2	1	②	-	-	0
3	1	-	4	③	0
4	-	5	④	-	1
5	1	⑤	-	-	1

Fig.VI.28 - Matrice primitiva e diagramma di stato di un circuito.

Nella fig.VI.28 sono riportati il diagramma degli stati e la matrice primitiva. Si noti che dagli stati ② e ⑤ è possibile passare solo allo stato ①, perché x_1 non può passare a 1 prima che x_2 torni a 0. Nello stato ④, invece, non può avvenire la variazione $x_1x_2 = 11 \rightarrow 10$ perché i segnali sui 2 ingressi debbono avere eguale durata. Per il resto, la matrice non presenta novità rispetto a quella dell'esempio 1.

La macchina $\mathfrak{M}'$ costruita con i criteri esposti avrà, nella maggioranza dei casi, un numero di stati assolutamente eccessivo (si confrontino, ad esempio, le tavole di flusso delle figg.VI.27 e VI.28 con una qualsiasi di quelle ricavate dall'analisi). Spesso, infatti, una variazione delle x lascia il sistema nello stesso stato; inoltre, molti degli stati sono, a tutti gli effetti, equivalenti. Il secondo passo della sintesi è, appunto, la ricerca della macchina $\mathfrak{M}'$ equivalente minima della $\mathfrak{M}$.

VI.5 - Minimizzazione del numero degli stati.

La minimizzazione degli stati si fa applicando il procedimento descritto nei paragrafi V.6.1 e V.6.2.

Esempio 1: *Minimizzare gli stati della macchina sequenziale asincrona la cui matrice primitiva derivata da un modello di Moore è rappresentata nella fig. VI.29.*

Stati	x_1x_2 00	01	11	10	z_1z_2
1	①	2	7	4	00
2	5	②	3	4	01
3	1	10	③	4	11
4	1	6	7	④	10
5	⑤	6	3	8	00
6	1	⑥	11	4	01
7	9	2	⑦	12	11
8	9	10	11	⑧	10
9	⑨	2	11	8	00
10	9	⑩	7	8	01
11	1	2	⑪	12	11
12	1	6	7	⑫	01

Fig. VI.29 - Matrice primitiva di una macchina sequenziale asincrona.

Le coppie di stati compatibili rispetto alle uscite sono: (1 • 5), (1 • 9), (5 • 9), (2 • 6), (2 • 10), (2 • 12), (6 • 10), (6 • 12), (3 • 7), (3 • 11), (7 • 11), (4 • 8).

La tabella di evoluzione degli stati α-compatibili, mostra che le coppie (2 • 12), (6 • 12), (3 • 7), (3 • 11) sono formate da stati non equivalenti, perché sotto gli ingressi $x_1x_2 = 10$ evolvono verso gli stati 4 e 12 ($4 \not\alpha 12$) (fig. VI.30).

L'esame della tabella costruita con le restanti coppie mostra infine la non-equivalenza degli stati (1 • 5), (5 • 9), (2 • 6) e (2 • 10), che evolvono verso le coppie (7 • 3) e (3 • 11), cancellate precedentemente (fig. VI.31).

	x_1x_2 00	01	11	10
1·5	1·5	2·6	7·3	4·8
1·9	1·9	2	7·11	4·8
5·9	5·9	6·2	3·11	8
2·6	5·1	2·6	3·11	4
2·10	5·9	2·10	3·7	4·8
2·12	5·1	2·6	3·7	(4·12)
6·10	1·9	6·10	11·7	4·8
6·12	1	6	11·7	(4·12)
3·7	1·9	10·2	3·7	(4·12)
3·11	1	10·2	3·11	(4·12)
7·11	9·1	2	7·11	12
4·8	1·9	6·10	7·11	4·8

Fig.VI.30 - Evoluzione degli stati α-compatibili (I).

	x_1x_2 00	01	11	10
1·5	1·5	2·6	(7·3)	4·8
1·9	1·9	2	7·11	4·8
5·9	5·9	6·2	(3·11)	8
2·6	5·1	2·6	(3·11)	4
2·10	5·9	2·10	(3·7)	4·8
6·10	1·9	6·10	11·7	4·8
7·11	9·1	2	7·11	12
4·8	1·9	6·10	7·11	4·8

Fig.VI.31 - Evoluzione degli stati α-compatibili (II).

Stati	x_1x_2 00	01	11	10	z_1z_2
1	①	2	7	4	00
2	5	②	3	4	01
3	1	6	③	4	11
4	1	6	7	④	10
5	⑤	6	3	4	00
6	1	⑥	7	4	01
7	1	2	⑦	8	11
8	1	6	7	⑧	01

M'

Fig.VI.32 - Tavola di flusso della macchina equivalente minima a quella di fig. VI.29.

Le restanti coppie sono formate da stati equivalenti; il processo di minimizzazione porta quindi alla seguente partizione degli stati di $\mathcal{M}$:

$$\{1 \cdot 9\},\ \{2\},\ \{3\},\ \{4 \cdot 8\},\ \{5\},\ \{6 \cdot 10\},\ \{7 \cdot 11\},\ \{12\}\ .$$

La macchina $\mathcal{M}'$ col minimo numero di stati ha la tavola di flusso della figura VI.32: è questa la matrice delle sequenze del circuito da realizzare.

Esempio 2: *Minimizzare gli stati della macchina sequenziale asincrona della fig.VI.33.*

Stati	x_1x_2 00	01	11	10	z
1	①	3	–	7	0
2	②	5	–	6	0
3	1	③	4	–	1
4	–	3	④	6	1
5	1	⑤	4	–	1
6	2	–	4	⑥	0
7	1	–	4	⑦	0

$\mathcal{M}$

Fig.VI.33 - Tavola di flusso di una macchina $\mathcal{M}$.

Gli stati 3 e 5 si trovano nella stessa colonna, hanno la stessa uscità e uguali i valori di tutte le caselle. Si può allora eliminare la riga 5, e sostituire ogni 5 con un 3 (fig.VI.34).

Stati	x_1x_2 00	01	11	10	z
1	①	3	–	7	0
2	②	3	–	6	0
3	1	③	4	–	1
4	–	3	④	6	1
6	2	–	4	⑥	0
7	1	–	4	⑦	0

$\mathcal{M}$

Fig.VI.34 - Eliminazione di uno stato dalla tavola di fig.VI.33.

La macchina $\mathcal{M}$ è incompleta. L'esame delle uscite porta alla relazione di α-incompatibilità tra gli stati dei 2 insiemi:

$$(1 \cdot 2 \cdot 6 \cdot 7) \not\alpha\ (3 \cdot 4) \quad .$$

Coppie	x_1x_2 00	01	11	10
1·2	1·2	3	–	6·7
1·6	1·2	3	4	6·7
1·7	1	3	4	7
2·6	2	3	4	6
2·7	2·1	3	4	6·7
3·4	1	3	4	6
6·7	2·1	–	4	6·7

Fig.VI.35 - Evoluzione delle coppie di stati α-compatibili.

Stati	x_1x_2 00	01	11	10
1·2·6·7	1·2	3	4	6·7
3·4	1	3	4	6

$\mathcal{M}$

Stati	x_1x_2 00	01	11	10
1	①	2	2	①
2	1	②	②	1

$\mathcal{M}'$

Fig.VI.36 - Tavola di flusso della macchina equivalente minima alla macchina $\mathcal{M}$ di fig.VI.34.

Le coppie α-compatibili sono: (1·2), (1·6), (1·7), (2·6), (2·7), (6·7), (3·4). La tabella formata da queste coppie (fig.VI.35) mostra che non esistono coppie tipo $\not\alpha$.

Le relazioni:

$$\frac{n\,(n-1)}{2}$$

con n massimo portano alla fusione degli stati (3·4) e (1·2·6·7). La macchina minima $\mathcal{M}'$ ha la tavola di flusso della fig.VI.36b, costruita a partire da quella della fig.VI.36a.

Esempio 3: *Minimizzazione degli stati della macchina sequenziale asincrona della figura VI.37.*

La $\mathcal{M}$ è incompletamente specificata. Le coppie $\not\alpha$ sono: (1·4), (1·5), (1·6), (1·7), (1·8), (2·4), (2·5), (2·6), (2·7), (2·8), (4·5), (4·8), (5·7), (5·8), (6·8), (7·8). Resta

da verificare la σ-compatibilità tra le coppie (1 • 2), (1 • 3), (2 • 3), (3 • 4), (3 • 5), (3 • 6), (3 • 7), (3 • 8), (4 • 6), (4 • 7), (5 • 6), (6 • 7). La tabella di verifica (fig.VI.38) conduce alle relazioni: $1\sigma 2$, $1\sigma 3$, $2\sigma 3$, $5\sigma 6$.

Stati	x_1x_2 00	01	11	10	z_1z_2
1	①	2	–	3	00
2	1	②	–	3	00
3	1	–	5	③	—
4	–	–	④	3	10
5	7	–	⑤	6	11
6	7	2	–	⑥	1–
7	⑦	8	–	6	10
8	1	⑧	4	6	01

Fig.VI.37 - Tavola di flusso di una macchina $\mathcal{M}$.

	Coppie	x_1x_2 00	01	11	10
	(1 • 2)	1	2	–	3
	(1 • 3)	1	2–	–5	3
	(2 • 3)	1	2–	–5	3
$\not\phi$	(3 • 4)	1–	–	(5 • 4)	3
$\not\phi$	(3 • 5)	(1 • 7)	–	5	3 • 6
$\not\phi$	(3 • 6)	(1 • 7)	–2	5–	3 • 6
$\not\phi$	(3 • 7)	(1 • 7)	–8	5–	3 • 6
$\not\phi$	(3 • 8)	1	–8	(5 • 4)	3 • 6
$\not\phi$	(4 • 6)	–7	–2	4–	(3 • 6)
$\not\phi$	(4 • 7)	–7	–8	4–	(3 • 6)
	(5 • 6)	7	–2	5–	6
$\not\phi$	(6 • 9)	7	(2 • 8)	–	6

Fig.VI.38 - Coppie α-compatibili e σ-compatibili.

La macchina minima $\mathcal{M}'$ ha la tavola di flusso della fig. VI.39, costruita dalla fig.VI.37, ponendo $1' \equiv \{1,2,3\}$, $2' \equiv \{4\}$, $3' \equiv \{5 \cdot 6\}$, $4' \equiv \{7\}$, $5' \equiv \{8\}$. (Gli apici sono stati eliminati dalla figura per maggior chiarezza).

Stati	x_1x_2 00	01	11	10	z_1z_2
1	①	①	3	①	00
2	-	-	②	1	10
3	4	1	③	②	11
4	④	5	-	3	10
5	1	⑤	2	3	01

$\mathcal{M}'$

Fig.VI.39 - Macchina minima equivalente a quella di fig.VI.37.

Il prossimo passo è forse il più delicato e importante della sintesi. Con la codificazione degli stati interni, infatti, si comincia a determinare la struttura del circuito.

Qualitativamente, si può già dire che il numero delle righe della matrice delle sequenze stabilisce il numero minimo delle variabili interne necessarie alla sua codifica: almeno k variabili, infatti, sono necessarie per una matrice a 2^k righe.

VI.6 - Codificazione degli stati.

Per codificazione degli stati si intende la determinazione del numero minimo (k) di variabili y necessarie per il funzionamento asincrono di quel modello ideale di circuito sequenziale che ha come tabella degli stati la matrice delle sequenze, nonché l'assegnazione dei valori di dette variabili ai singoli stati della matrice stessa.

L'assegnazione degli stati secondari, per evitare errate evoluzioni del circuito, deve avvenire in modo da non permettere corse critiche. I valori delle y vengono, di solito, assegnati in modo che a 2 stati fra cui esiste una transizione corrispondano valori adiacenti delle y. Se economicamente vantaggiose nei riguardi del circuito sequenziale, si possono – in alternativa – introdurre corse non critiche e transizioni multiple.

L'effettiva assegnazione degli stati determina a posteriori il numero k. Tale numero non può essere determinato in altro modo, non esistendo relazioni analitiche tra il suo valore e il numero degli stati. Di qualche utilità, in pratica, può essere la tabella della fig.VI.40, ricavata da una formula di Huffman, che mostra i limiti superiori e inferiori di k per matrici fino a 32 righe.

k_m	1	2		3		4		5	
r	2	3	4	5-6	7-8	6 ÷ 12	13 ÷ 16	17 ÷ 24	25 ÷ 32
k_M	1	2	3	4	5	6	7	8	9

Fig.VI.40 - Valore minimo (k_m) e massimo (k_M) di k per matrici a r righe.

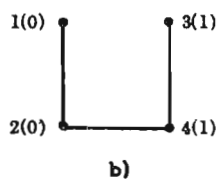
La codificazione degli stati si fa con l'aiuto di un *diagramma delle transizioni* e di una *tavola delle transizioni*, appresso definiti.

Il diagramma delle transizioni, comprende tanti punti quanti sono gli stati della matrice delle sequenze; accanto a ogni punto si scrive il numero dello stato e – tra parentesi – i valori delle relative uscite. Si uniscono, infine, i punti corrispondenti alle righe fra cui esiste una transizione.

Esempio 1: *Nella fig.VI.41b è mostrato il diagramma delle transizioni relativo alla matrice della fig.VI.41a. Il punto A è unito al punto B perché nella colonna $x_1x_2=00$, dallo stato 1 della riga B si passa allo stato ① della riga A. I punti A e C sono separati perché in nessuna colonna è prevista una transizione tra le righe corrispondenti.*

a)

Stati	x_1x_2 00	01	11	10	z
1	①	2	2	–	0
2	1	②	②	4	0
3	4	–	③	4	1
4	④	2	3	④	1



b)

Fig.VI.41 - Matrice di flusso e diagramma delle transizioni.

I numeri degli stati e i valori delle uscite vanno poi disposti su una particolare mappa di Karnaugh, dove non compaiono le coordinate delle singole caselle, in modo che due numeri collegati nel diagramma di transizione occupino due caselle adiacenti. Tutte le transizioni provocano così la variazione di una sola y.

Si comincia col provare a sistemare le r righe su una mappa per k_m variabili. Se questo non è possibile, si cerca di realizzare le transizioni attraverso altri stati secondari o corse non critiche. Se con nessuno di questi artifici si arriva al risultato, si riprova con una mappa a $(k_m + 1)$ variabili, e così via.

Sistemate, comunque, le lettere su una mappa, occorre codificare le righe e le colonne di quest'ultima secondo un codice ciclico, che conviene scegliere in modo da far coincidere il maggior numero di z con altrettante y. La mappa così costruita si chiama *tavola delle transizioni*.

Si ottiene, finalmente, la tabella degli stati del circuito sostituendo ogni stato della matrice primitiva con le coordinate che lo stato stesso ha sulla tavola di transizione. Ripetiamo che dall'assegnazione degli stati interni dipende in maniera determinante la forma del circuito: è sempre consigliabile, pertanto, ricavare più tavole di transizione, per confrontare i circuiti che da esse si ottengono.

Esempio 2: *Assegnare gli stati secondari alla matrice delle sequenze della fig.VI.42.*

	x		
Stato	**0**	**1**	**z**
1	①	2	0
2	3	②	1
3	③	4	1
4	1	④	0

Fig.VI.42 - Matrice delle sequenze.

Il diagramma delle transizioni è riportato nella fig.VI.43a. Gli stati possono essere disposti su una mappa per 2 variabili (fig.VI.43b) in modo che tutti quelli legati da transizioni si trovino entro caselle adiacenti. Una buona codificazione della mappa, che permette di prendere l'uscita direttamente su y_1, è mostrata nella fig.VI.43c. Dalla tavola così codificata, si ricava la matrice degli stati (fig.VI.44).

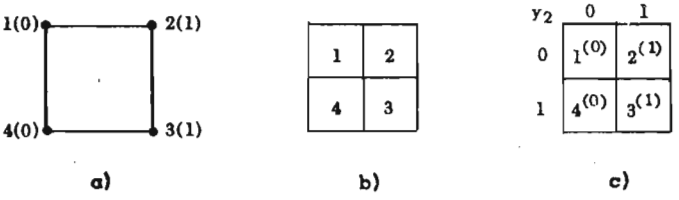


Fig.VI.43 - Assegnazione degli stati secondari alla matrice di fig.VI.42.

Fig.VI.44 - Matrice degli stati derivata dalla matrice di figura VI.42.

y_1y_2	x 0	1	z
00	①	2	0
01	1	④	0
11	③	4	1
10	3	②	1

Esempio 3: *Assegnare gli stati secondari alla matrice delle sequenze della fig.VI.45.*

Stati	$x_1x_2x_3$ 000	001	010	100	z_1z_2
1	①	2	3	4	00
2	②	②	3	4	10
3	③	2	③	4	01
4	④	2	3	④	10

Fig.VI.45 - Matrice delle sequenze.

Il diagramma di transizione è riportato nella fig.VI.46a.

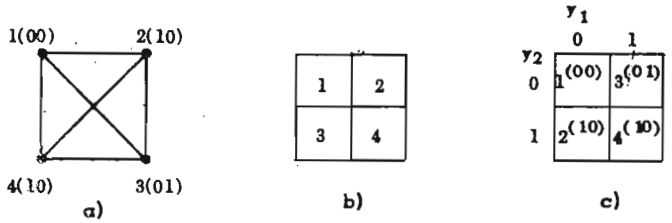


Fig.VI.46 - Assegnazione degli stati secondari alla matrice di fig.VI.45.

Non è possibile sistemare gli stati 1 • 2 • 3 • 4 su una mappa per 2 variabili in modo che tutte le transizioni avvengano con la variazione di una sola y, in quanto ciò richiederebbe l'adiacenza di tutte le coppie di stati, mentre qualsiasi disposizione degli stati (ad esempio quella della fig.VI.46b) non può realizzare l'adiacenza delle caselle opposte 1 • 4 e 2 • 3. La particolare struttura della matrice delle sequenze, che nelle colonne $x_1x_2x_3 = 001$, 010 e 100 presenta un solo stato stabile, suggerisce tuttavia di adottare delle corse non critiche tra gli stati 1 • 4 e 2 • 3. La più conveniente tabella delle transizioni, codificata in modo da minimizzare la rete d'uscita, è quella della figura VI.46c: non si possono ricavare direttamente le due z dalle y, ma si ottiene una buona soluzione ponendo:

$$z_1 = y_2 \; ; \qquad z_2 = y_1\overline{y}_2 \; .$$

La matrice degli stati è mostrata nella fig.VI.47.

y_1y_2	$x_1x_2x_3$ 000	001	010	100	z_1z_2
00	①	2	3	4	00
01	②	②	3	4	10
11	④	2	3	④	10
10	③	2	③	4	01

Fig.VI.47 - Matrice degli stati derivata dalla matrice di fig.VI.45.

Esempio 4: *Assegnare gli stati secondari alla matrice delle sequenze di fig.VI.48.*

	x_1x_2 00	01	11	10	z
1	①	2	4	1	0
2	3	②	3	②	0
3	③	4	③	1	0
4	1	④	④	2	1

Fig.VI.48 - Matrice delle sequenze.

Il diagramma delle transizioni (fig.VI.49) non può essere realizzato su una mappa per 2 variabili, per le stesse ragioni dell'esercizio precedente. La natura della matrice primitiva non permette, inoltre, di ottenere corse non critiche o transizioni multiple. Occorre, quindi, usare 3 variabili interne, i cui 8 valori possono essere assegnati alle righe della matrice in parecchi modi, sostanzialmente appartenenti a 2 categorie:

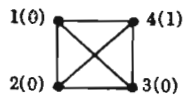


Fig.VI.49 - Diagramma delle transizioni per la matrice di fig.VI.48.

a) quattro valori vengono assegnati ai quattro stati; i rimanenti vengono usati, tutti o in parte, come stati intermedi per le transizioni che non possono avvenire direttamente;

b) vengono assegnati più valori ad uno o più stati.

a) Una delle molte soluzioni del primo tipo è quella mostrata nella tabella della figura VI.50, codificata in modo da essere $z = y_3$. Le transizioni $1 \rightarrow 2$, $2 \rightarrow 3$, $2 \rightarrow 4$ avvengono

direttamente, le altre attraverso uno stadio intermedio ($1 \rightarrow 5 \rightarrow 4$, $3 \rightarrow 6 \rightarrow 4$, $3 \rightarrow 7 \rightarrow 1$). La tabella degli stati è quella della fig.VI.51, che prevede 4 transizioni multiple: esisto-

y_3 \ y_1y_2	00	01	11	10
0	1 (0)	2 (0)	3 (0)	7
1	5	4 (1)	6	–

Fig.VI.50 - Assegnazione degli stati secondari alla matrice di fig.VI.48.

no parecchie caselle vuote, che semplificherebbero la forma del circuito; c'è, di contro, lo svantaggio di un tempo di risposta minore per le transizioni dirette (Δ) che per quelle inverse (2Δ).

$y_1y_2y_2$ \ x_1x_2	00	01	11	10	z
000	①	2	4	①	0
001	1	–	4	–	–
011	1	④	④	2	1
010	3	②	3	②	0
110	③	4	③	1	0
111	–	4	–	–	–
101	–	–	–	–	–
100	–	–	–	1	–

Fig.VI.51 - Tabella degli stati derivata dall'assegnazione di fig.VI.50.

y_3 \ y_1y_2	00	01	11	10
0	1	5	2	6
1	7	3	8	4

a)

$y_1y_2y_3$ \ x_1x_2	00	01	11	10	z
000	①	2	4	①	0
001	1	–	4	–	–
011	③	4	③	1	0
010	–	2	–	1	–
110	3	②	3	②	0
111	3	4	3	–	–
101	1	④	④	2	1
100	–	–	–	2	–

b)

Fig.VI.52 - Altra assegnazione degli stati e relativa tabella per la matrice di fig.VI.48.

Una soluzione dello stesso tipo, che rende però costante il tempo di risposta del circuito, è quella della fig.VI.52b, derivata dalla tabella della fig.VI.52a. Tutte le transizioni dirette sono state eliminate e si hanno soltanto le multiple: $1 \to 5 \to 2$, $1 \to 5 \to 3$, $1 \to 7 \to 4$, $2 \to 8 \to 3$, $2 \to 6 \to 4$, $3 \to 8 \to 4$; la codificazione adottata rende $z = y_1 y_3$.

b) A qualche stato può essere assegnato più di un valore delle y, per aumentare le adiacenze nella tabella delle transizioni. Assegnando m valori delle y a un solo stato, m righe della tabella degli stati contengono stati equivalenti, che conviene indicare con lo stesso numero, e distinguere con indici diversi per le varie transizioni. Fisicamente, non importa quale degli stati equivalenti il circuito occupi ad un dato istante, purché tutte le uscite di questi abbiano lo stesso valore. Una soluzione del genere è indicata nella tabella delle transizioni della fig.VI.53, codificata in modo che $z = y_1 y_3$.

y_3 \ y_1y_2	00	01	11	10
0	$1_1^{(0)}$	$2_1^{(0)}$	$3_1^{(0)}$	$3_2^{(0)}$
1	$1_2^{(0)}$	$2_2^{(0)}$	$4_1^{(1)}$	$4_2^{(1)}$

Fig.VI.53 - Assegnazione degli stati secondari alla matrice di fig.VI.48.

	$y_1y_2y_3$ \ x_1x_2	00	01	11	10	z
	000	(1_1)	2_1	4_2	(1_1)	0
	001	(1_2)	2_2	4_2	(1_2)	0
	011	3_1	(2_2)	3_1	(2_2)	0
	010	3_1	(2_1)	3_1	(2_1)	0
	110	(3_1)	4_1	(3_1)	1_1	0
	111	1_2	(4_1)	(4_1)	2_2	1
	101	1_2	(4_2)	(4_2)	2_2	1
a)	100	(3_2)	4_2	(3_2)	1_1	0

	$y_1y_2y_3$ \ x_1x_2	00	01	11	10	z
	000	①	2_1	4_2	①	0
	001	1	–	4_2	–	–
	011	3	(2_2)	3	(2_2)	0
	010	3	(2_1)	3	(2_1)	0
	110	(3)	4_1	③	1	0
	111	1	(4_1)	(4_1)	2_2	1
	101	1	(4_2)	(4_2)	2_2	1
b)	100	–	–	–		–

Fig.VI.54 - Tabella degli stati derivata dall'assegnazione di fig.VI.53.

Esistono molte transizioni possibili: 1_1, ad esempio, può raggiungere direttamente $2\,(1_1 \to 2_1)$, o passare attraverso $1_2\,(1_1 \to 1_2 \to 2_2)$: la scelta dell'una o dell'altra via deve sempre essere compiuta avendo di mira l'economia o la velocità di operazione del circuito. Nella fig.VI.54a è mostrata una delle possibili matrici degli stati derivabili della fig.VI.53.

Importa notare 2 tipi di semplificazioni, assai particolari, che si possono effettuare sulle matrici costruite con questo metodo:

1) Nella riga $y_1y_2y_3 = 100$, gli stati stabili (3_2) non possono essere raggiunti né in seguito a comandi esterni (nessun altro stato stabile si trova sull'ultima riga), né in seguito a evoluzioni del circuito (non ci sono stati (3_2) nelle colonne 00 e 11). Gli stati (3_2) possono allora essere eliminati, insieme con lo stato instabile 4_2 non più raggiungibile. Va invece conservato lo stato 1_1, perché necessario alla transizione $3_1 \rightarrow 1_1$.

2) La colonna 00 si può modificare sostituendo a (1_1) lo stato ① e a 1_2 lo stato instabile 1: introducendo la transizione $4_1 \rightarrow 4_2 \rightarrow 1_2 \rightarrow 1_1$ si eliminano così gli stati 2_2 e (1_2) della riga 001, non più raggiungibili.

La nuova più semplice forma della matrice è mostrata nella fig.VI.54b.

Abbiamo esposto, sommariamente, i principali metodi di assegnazione delle variabili secondarie nel caso delle matrici a 4 righe. L'estensione a matrici con 5 o più righe non dovrebbe presentare difficoltà. Si tenga presente che esistono molti altri metodi, più o meno complessi, di codificazione degli stati secondari: nessuno, però, permette di ottenere – con sicurezza – il circuito minimo.

Costruita, comunque, la matrice degli stati, si può compiere il passo successivo della sintesi.

VI.7 - Tavola di flusso.

La tavola di flusso si ottiene sostituendo semplicemente i numeri della tabella degli stati con i k valori delle y e gli m valori delle z, secondo le regole seguenti:

– il numero (j) della riga j di coordinate $y_1^j y_2^j \ldots y_k^j$ e uscite $z_1^j z_2^j \ldots z_m^j$ va sostituito con $(y_1^j y_2^j \ldots y_k^j) / z_1^j z_2^j \ldots z_m^j$;

– il numero j della riga r, in tutte le colonne per le quali si passa direttamente da j a (j), va sostituito con $y_1^j y_2^j \ldots y_k^j / z_1^r z_2^r \ldots z_m^r$;

– il numero j della riga r, in tutte le colonne per le quali si passa da j a (j) attraverso lo stato j della riga s (transizioni multiple), va sostituito con $y_1^s y_2^s \ldots y_k^s / z_1^r z_2^r \ldots z_m^r$, con ovvio significato dei simboli;

– dovunque non compaiano esplicitamente, le z assumeranno i valori che sono stati previsti nella codificazione della tabella di transizione.

Esempio 1: *Tavola di flusso relativa alla tabella degli stati della fig.VI.55.*

Lo stato stabile ① si trova nella riga di coordinate $y_1y_2 = 00$ e uscita $z = 0$; la corrispondente casella andrà riempita con $(00)/0$; lo stato ② si trova nella riga $y_1y_2 = 01$

e ha uscita 0, quindi (01)/0. La casella $xy_1y_2 = 100$ contiene lo stato instabile 2; la transizione allo stato ② è diretta, quindi si scriverà 10/0 : 10 sono le coordinate dello stato ②, 0 il valore dell'uscita che porta a $z = y_1$.

y_1y_2	x = 0	1	z
00	①	2	0
01	1	④	0
11	③	4	1
10	3	②	1

Fig.VI.55 - Tabella degli stati.

y_1y_2	x = 0	1
00	(00)/0	10/0
01	00/0	(01)/0
11	(11)/1	01/1
10	11/1	(10)/1

$y_1'y_2'z$

Fig.VI.56 - Tavola di flusso derivata dalla tabella degli stati di fig.VI.55.

La tavola di flusso completa, costruita con gli stessi criteri, è mostrata nella figura VI.56.

Esempio 2: *Tavola di flusso per la tabella degli stati della fig.VI.57.*

y_1y_2	$x_1x_2x_3$ = 000	001	010	100	z_1z_2
00	①	2	3	4	00
01	②	②	3	4	10
11	④	2	3	④	10
10	③	2	③	4	01

Fig.VI.57 - Tabella degli stati.

x_1x_2	$x_1x_2x_3$ = 000	001	010	1000
00	(00)/00	01/00	10/00	11/00
01	(01)/10	(01)/10	10/10	11/10
11	(11)/10	01/10	10/10	(11)/10
10	(10)/01	01/01	(10)/01	11/01

$y_1y_2'z_1z_2$

Fig.VI.58 - Tavola di flusso per la tabella degli stati di fig.VI.57.

Nella tabella non esistono transizioni multiple, ma soltanto transizioni semplici e corse, indicate con frecce, dallo stato q al (q). Ai fini della tavola di flusso (figura VI.58), le corse vanno considerate come transizioni semplici.

Esempio 3: *Tavola di flusso per la tabella degli stati della fig.VI.59.*

$y_1y_2y_3$ \ x_1x_2	00	01	11	10	$z\ (=y_3)$
000	①	2	4	①	0
001	1	–	4	–	–
011	1	④	④	2	1
010	3	②	3	②	0
110	③	4	③	1	0
111	–	4	–	–	–
101	–	–	–	–	–
100	–	–	–	1	–

Fig.VI.59 - Tabella degli stati.

$y_1y_2y_3$ \ x_1x_2	00	01	11	10
000	(000)/0	010/0	001/0	(000)/0
001	000/1	–/1	011/1	–/1
011	001/1	(011)/1	(011)/1	010/1
010	110/0	(010)/0	010/0	(010)/0
110	(110)/0	111/0	(110)/0	100/0
111	–/1	011/1	–/1	–/1
101	–/1	–/1	–/1	–/1
100	–/0	–/0	–/0	000/0

Fig.VI.60 - Tavola di flusso per la tabella degli stati di fig.VI.59.

I valori delle y' da porre nelle caselle contenenti stati stabili o instabili senza frecce non hanno bisogno di ulteriori specificazioni. Nella prima riga e nella colonna 11 esiste lo stato 4 che evolve verso ④ mediante la transizione attraverso la riga 001; i

valori delle y' da porre nella casella $x_1x_2y_1y_2y_3 = 11000$ sono perciò 001. Nella fig.VI.59 esistono 4 valori di z non specificati; tuttavia, poiché deve essere $z = y_3$ (v. fig.VI.50) essi rimangono ben determinati. La matrice di flusso completa è riportata nella fig.VI.60.

Nel par.VI.2 abbiamo visto che dalle equazioni di un circuito sequenziale è possibile ricavare la sua tavola di flusso. Seguiremo ora la strada inversa per ricavare, dalla tavola di flusso, il circuito.

VI.8 - Costruzione del circuito sequenziale.

In ogni casella della tavola di flusso sono scritti (k + m) bit, corrispondenti ai valori delle y e delle z. È così possibile ricavare le (k + + m) mappe di Karnaugh per le funzioni $y_1y_2\ldots y_k$ e $z_1z_2\ldots z_m$. Ottenute, dalle mappe, le equazioni, e scelti gli elementi logici da usare, si costruisce un circuito combinatorio MT a m + k uscite, per le variabili $x_1x_2\ldots x_n y_1y_2\ldots y_k$. Realizzando i collegamenti di reazione tra le y' e le y, si ottiene infine il circuito sequenziale asincrono reale.

Esempio 1: *Circuito sequenziale per la matrice di flusso della fig.VI.61.*

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)/0	10/0	11/0	01/0
01	00/1	-/1	11/1	(01)/1
11	(11)/1	10/1	(11)/1	(11)/1
10	00/0	(10)/0	11/0	-/0

Fig.VI.61 - Matrice di flusso.

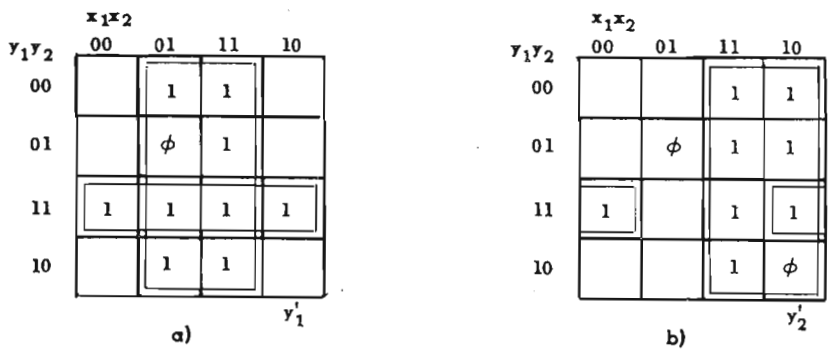


Fig.VI.62 - Mappe di Karnaugh derivate dalla matrice di flusso di fig.VI.61.

Poiché $y_2 = z$, basta ricavare le mappe di Karnaugh per le due funzioni y_1' e y_2' (fig.VI.62).

Si ha così:

$$\begin{cases} y_1' = x_2 + y_1 y_2 = x_2 + \overline{x}_2 y_1 y_2 \\ y_2' = x_1 + \overline{x}_2 y_1 y_2 \end{cases}.$$

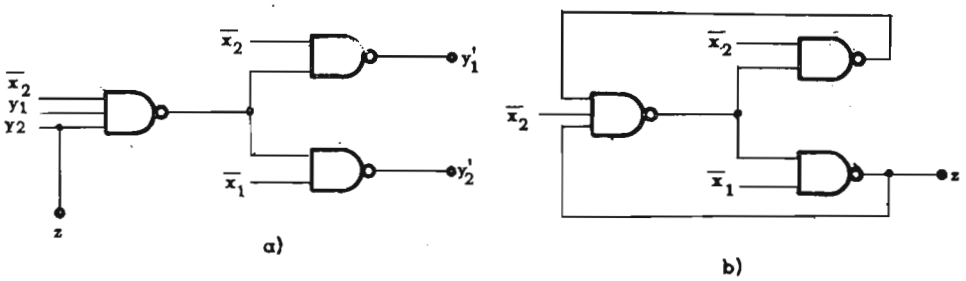


Fig.VI.63 - Circuito sequenziale asincrono per la matrice di flusso di fig.VI.61.

Il circuito NAND costruito con gli ingressi $x_1 x_2 y_1 y_2$ è mostrato nella fig.VI.63a; i collegamenti di reazione lo trasformano nel circuito reale di fig.VI.63b.

VI.8.1 - Scelta degli elementi logici.

Non tutti gli elementi logici si possono usare indifferentemente nei circuiti sequenziali asincroni. Per fissare le idee, esaminiamo un caso pratico: il funzionamento di un circuito a diodi per la funzione di eccitazione $y' = \overline{x}_1 x_2 + x_1 y$ nei due stati stabili opposti $y' = y = 1$ e $y' = y = 0$, sotto la condizione d'ingresso $x_1 x_2 = 11$.

Nelle figg.VI.64a e VI.64b sono mostrati gli schemi logico ed elettrico del circuito; in logica positiva, il valore 1 rappresenti la tensione di +10 V, il valore 0 la tensione di 0 V. Nella fig.VI.65 è riportato la parte del circuito che contiene il loop di reazione, con l'indicazione dei punti che si trovano alle tensioni di 0 V e 10 V. Le tensioni y' e y di ingresso e d'uscita dell'elemento di ritardo dipendono invece dai rapporti tra i valori delle resistenze R_1 ed R_2. Poiché l'uscita della parte sinistra del circuito, cioè l'AND $x_1 y$, è uno degli ingressi dell'OR $y' x_1 x_2$, per avere $y = y'$ quando si è nello stato stabile 1 in cui $y' = y = 1$ (= 10 V) deve essere $R_1 \ll R_2$.

D'altra parte, attraverso il loop di reazione, l'uscita dell'OR è uno degli ingressi dell'AND, quindi per avere $y = y'$, quando si è nello stato stabile 0 in cui $y = y' = 0$ (= 0 V), deve essere $R_1 \gg R_2$. Questa

contraddizione, che non si può eliminare (al limite, facendo $R_1 = R_2$, si avrebbe sempre $y' = y = 5$ V) esiste perché i due circuiti AND e OR so-

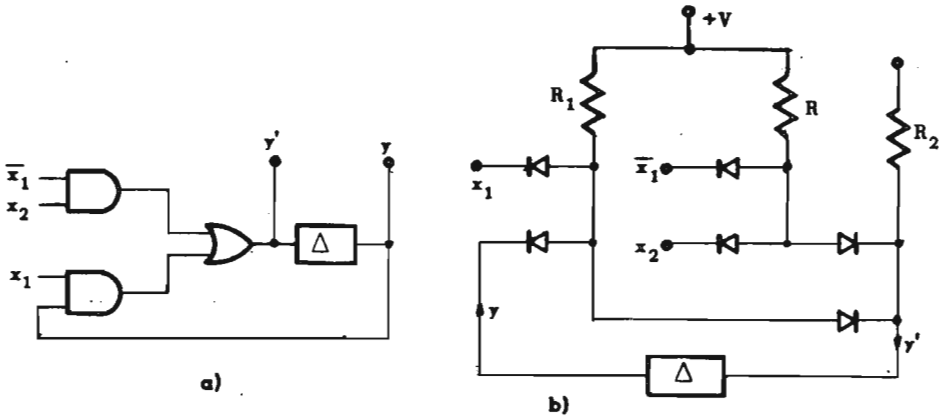


Fig.VI.64 - Circuito a diodi per la funzione $y' = \bar{x}_1 x_2 + x_1 y$.

no realizzati con elementi passivi, quindi l'unico punto d'equilibrio per le stesse condizioni d'ingresso è determinato dal rapporto R_1/R_2. Per avere due punti di equilibrio bisogna introdurre un elemento attivo nel loop. In definitiva, mentre la parte di rete logica di un circuito sequenziale che non comprende loop di reazione può essere realizzata comunque, nei loop stessi vanno impiegati elementi attivi, oppure anche elementi attivi; per semplicità, conviene usare sempre NAND o NOR.

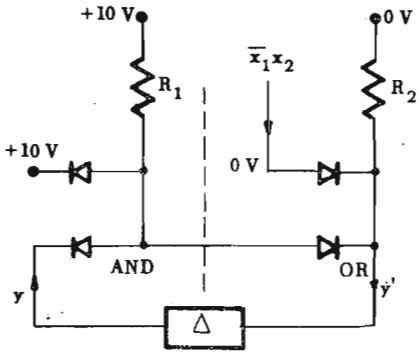


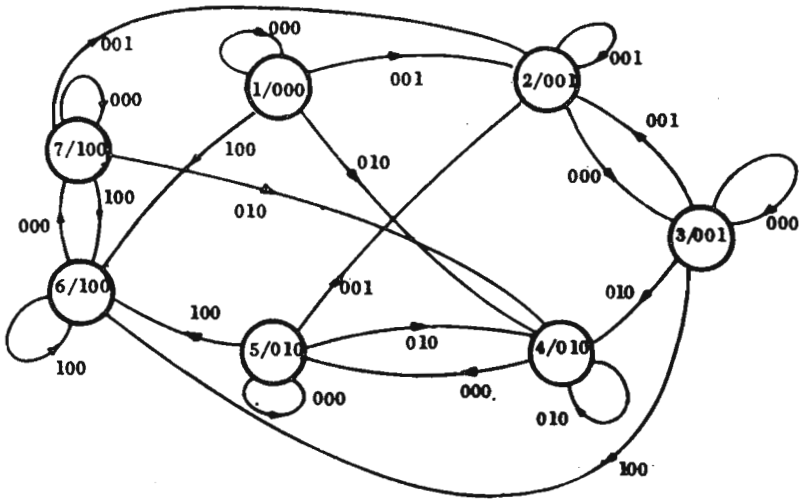
Fig.VI.65 - Tensioni nel circuito di fig.VI.64.

Per meglio legare i concetti finora esposti, riteniamo utile a questo punto, effettuare la sintesi completa di un circuito asincrono.

VI.9 - Un esempio di sintesi.

Un circuito sequenziale ha 3 ingressi ($x_1 x_2 x_3$) sui cui possono presentarsi – ma non contemporaneamente – degli impulsi di tensione, e 3 uscite ($z_1 z_2 z_3$). Il circuito parte da uno stato di riposo, con le tre u-

scite a 0. Contemporaneamente all'arrivo di un impulso su x_i, z_i va ad 1, e vi rimane finché un impulso su x_k manda a 0 z_i, e ad 1 z_k (i, k = 1, 2, 3; $i \neq k$).



$x_1x_2x_3$ 000	001	010	100	$z_1z_2z_3$
①	2	4	6	000
3	②	–	–	001
③	2	4	6	001
5	–	④	–	010
⑤	2	4	6	010
7	–	–	⑥	100
⑦	2	4	6	100

Fig.VI.66 - Diagramma degli stati e matrice primitiva.

Possiamo considerare il circuito a livelli, scomponendo ogni impulso in due variazioni di tensione opposte. Nella matrice primitiva (figura VI.66) compaiono allora le sole colonne di peso 0 e 1.

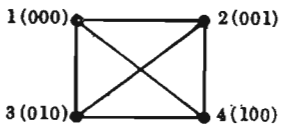
Applicando il metodo del par.V.6.1 per la minimizzazione degli stati (fig.VI.67a), si giunge facilmente alla matrice delle sequenze della fig.VI.68 e al diagramma delle transizioni della fig.VI.67b. Per usare

il minimo numero di variabili secondarie, conviene introdurre delle corse non critiche; una delle possibili tabelle degli stati si ottiene con la codificazione:

$$1 = 00$$
$$2 = 01$$
$$3 = 11$$
$$4 = 10 \ .$$

a)

Coppie	000	001	010	100
2 • 3	3	2	4	6
4 • 5	5	2	4	6
5 • 6	7	2	4	6



b)

Fig.VI.67 - Minimizzazione degli stati e diagramma delle transizioni.

	$x_1x_2x_3$ 000	001	010	100	$z_1z_2z_3$
1	①	2	3	4	000
2	②	②	3	4	001
3	③	2	③	4	010
4	④	2	3	④	100

Fig.VI.68 - Matrice delle sequenze.

y_1y_2	$x_1x_2x_3$ 000	001	010	100
00	(00)/000	01/000	11/000	10/000
01	(01)/001	(01)/001	11/001	10/001
11	(11)/010	01/010	(11)/010	10/010
10	(10)/100	01/100	11/100	(10)/100

$y_1'y_2'z_1z_2z_3$

Fig.VI.69 - Tavola di flusso.

La corrispondente tavola di flusso è quella della fig.VI.69.

Per avere il circuito più economico, una volta definite le espressioni delle uscite:

$$\text{(VI.6)} \qquad \begin{cases} z_1 = y_1 \overline{y_2} \\ z_2 = y_1 y_2 \\ z_3 = \overline{y_1} y_2 \end{cases},$$

si scriverà la matrice di eccitazione con tutte le 8 colonne; assegnando i valori più convenienti alle condizioni non specificate (fig. VI.70), si ottengono le equazioni:

$$\text{(VI.6')} \qquad \begin{cases} y_1' = x_1 + x_2 + \overline{x}_3 y_1 \\ y_2' = x_2 + x_3 + \overline{x}_1 y_2 \end{cases}.$$

y_1y_2 \ $x_1x_2x_3$	000	001	(011)	010	(110)	(111)	(101)	100
00	00	01	11	11	11	11	11	10
01	01	01	11	11	11	11	11	10
11	11	01	11	11	11	11	11	10
10	10	01	11	11	11	11	11	10
								$y_1'y_2'$

Fig.VI.70 - Matrice di eccitazione.

Dalle (VI.6) e (VI.6') si ricava, infine, il circuito a 7 NAND e 5 invertitori della fig.VI.71.

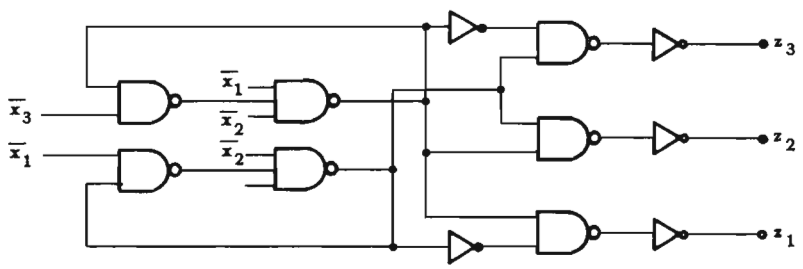


Fig.VI.71 - Circuito NAND derivato dalla matrice di fig.VI.70.

Il circuito della fig.VI.71 è stato ottenuto minimizzando il numero dei loop di reazione, cioè la rete di memoria. Per cercare di ottenere un circuito più economico, si può provare ad eliminare completamente la

rete d'uscita, complicando la rete di memoria con un loop ridondante, e assegnando gli stati interni in modo da far coincidere le y con le z. Si ha così la tavola di flusso della fig.VI.72.

$y_1y_2y_3$ \ $x_1x_2x_3$	000	001	010	100
000	(000)/000	001/000	010/000	100/000
001	(001)/001	(001)/001	010/001	100/001
011	-/011	-/011	-/011	-/011
010	(010)/010	001/010	(010)/010	100/010
110	-/110	-/110	-/110	-/110
111	-/111	-/111	-/111	-/111
101	-/101	-/101	-/101	-/101
100	(100)/100	001/100	010/100	(100)/100
				$y_1' y_2' y_3' z_1 z_2 z_3$

Fig.VI.72 - Una diversa matrice di flusso per la macchina di fig.VI.66.

Da questa, scegliendo opportunamente i valori opzionali di $y_1'y_2'y_3'$, si ottengono le equazioni:

$$\begin{cases} y_1' = x_1 + \overline{x}_2\overline{x}_3 y_1 \\ y_2' = x_2 + \overline{x}_1\overline{x}_3 y_2 \\ y_3' = x_3 + \overline{x}_1\overline{x}_2 y_3 \\ z_1 = y_1 \\ z_2 = y_2 \\ z_3 = y_3 \ . \end{cases}$$

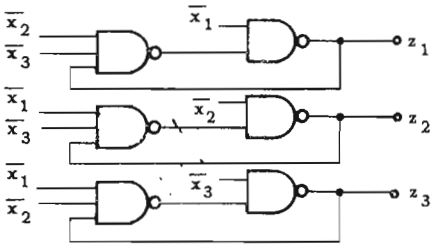


Fig.VI.73 - Circuito NAND derivato dalla matrice di fig.VI.72.

Il relativo circuito (figura (VI.73) è – sotto ogni aspetto – più conveniente di quello della fig.VI.71; ha infatti meno componenti (6 NAND), è più veloce (2 livelli invece di 5) ed è molto più semplice.

VI.10 - Le alee nei circuiti sequenziali.

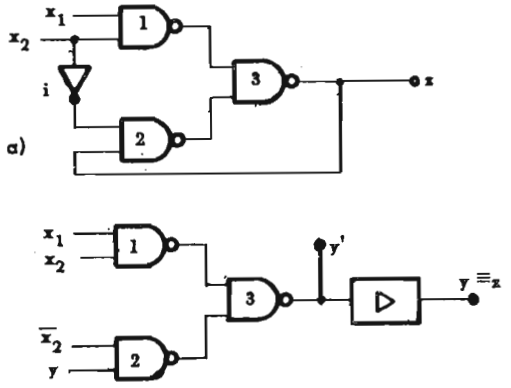
Una delle ipotesi fondamentali su cui è stata basata l'analisi e la sintesi dei circuiti asincroni era l'invarianza di Δ per tutti i loop. Il

mancato verificarsi di questa ipotesi causa un funzionamento dei circuiti reali diverso da quello previsto dalla matrice dei flussi, che si indica col termine *alea* (in inglese *hazard*).

Le alee sono di 4 tipi: i primi tre si verificano nei circuiti in corrispondenza del cambiamento di una variabile secondaria, e possono essere previsti ed eliminati con accorgimenti di natura logica o mediante correzione circuitale dei ritardi. Il quarto tipo di alea accompagna il cambiamento simultaneo di più variabili secondarie e può appartenere ad ognuno dei 3 tipi precedenti. Le alee predette prendono rispettivamente i nomi di alee statiche, dinamiche, essenziali e multiple.

VI.10.1 - Alee statiche.

Le alee statiche sono causate dalla diversa lunghezza delle vie percorse da un segnale per raggiungere l'ingresso di un loop di reazione, e provocano la variazione transitoria di un'uscita che dovrebbe rimanere costante durante il cambiamento di una variabile interna y. Le alee statiche, provocate in genere dal passaggio di un segnale attraverso un invertitore, sono importanti quando la falsa uscita dura abbastanza da causare una errata evoluzione del circuito stesso o di un altro circuito sequenziale a valle; possono essere eliminati logicamente oppure inserendo nel circuito un effettivo elemento di ritardo.



y	x1x2 = 00	01	11	10
0	(0)/0	(0)/0	1/0	(0)/0
1	(1)/1	0/1	(1)/1	(1)/1

y'z

b)

Fig.VI.74 - Alee statiche.

Consideriamo, per lo studio di un'alea statica, il circuito della fig.VI.74a, la cui matrice di flusso è riportata nella fig.VI.74b, di equazioni:

$$\begin{cases} y_1' = x_1 x_2 + \bar{x}_2 y \\ z = y \ . \end{cases}$$

Equazioni e tabelle sono state ottenute considerando, secondo le convenzioni introdotte, la rete logica istantanea e tutto il ritardo provocato dall'elemento Δ (fig.VI.74c).

Studiamo, ora, la risposta del circuito alla sequenza d'ingresso $x_1x_2 = 11 \rightarrow 10$, nelle condizioni ideali e in quelle reali.

Nel circuito di fig.VI.74c, quando $x_1 = x_2 = 1$, le uscite dei NAND 1 e 2 sono 0 e 1, e si ha uno stato stabile $y' = y = z = 1$. Se x_2 va a 0, cambiano – istantaneamente – le uscite dei NAND 1 e 2, per cui l'uscita del NAND 3 rimane invariata.

Diversamente vanno le cose nel circuito reale (fig.VI.74a), dove ogni elemento ha un proprio ritardo ed è $\Delta_1 \neq \Delta_2 \neq \Delta_3 \neq \Delta_i$.

Per $x_1 = x_2 = 1$, l'uscita del NAND 1 è 0 e quella del NAND 2 è 1. Quando x_2 va a 0, cambia dopo un tempo Δ_1 l'uscita del NAND 1, che va a 1; l'uscita del NAND 2 va a 0 dopo un tempo $\Delta_2 + \Delta_i$, generalmente $> \Delta_1$. Per un tempo $\Delta_\tau = \Delta_2 + \Delta_i - \Delta_1$ le uscite dei NAND 1 e 2 sono entrambi 1, quindi l'uscita z e la funzione y assumono il valore 0; passato il tempo Δ_τ, y e z riprendono il valore normale.

L'effetto del transitorio 1-0-1 sulla y può essere quello di mandare il circuito nello stato stabile indesiderato $x_1x_2y = 100$, se il segnale $y = 0$ rimane all'ingresso del NAND 2 per un tempo superiore al suo ritardo alla risposta (Δ_2). L'effetto del transitorio sull'uscita può provocare il cattivo funzionamento di un eventuale circuito sequenziale a valle di quello considerato, che riceva z come ingresso.

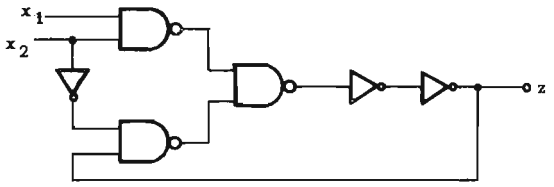


Fig.VI.75 - Eliminazione circuitale di un'alea statica.

Un primo modo di cancellare gli effetti dell'alea statica è l'inserzione, nel loop di reazione, di un elemento di ritardo addizionale δ, sufficientemente elevato per evitare che il transitorio si presenti all'ingresso del NAND 2. Il ritardo $\delta > \Delta_\tau$ può essere, ad esempio, creato con 2 invertitori in serie (fig.VI.75).

Non è però conveniente costruire dei circuiti lenti solo per evitare alee statiche; una soluzione migliore è l'eliminazione delle alee stesse con mezzi di natura logica, una volta individuate le cause che le pro-

vocano. Esaminiamo, per questo, la forma di 2 insiemi che compongono la funzione y' sulla mappa di Karnaugh (fig.VI.76a): essi dipendono entrambi da x_2: uno è, anzi, tutto in x_2, l'altro in $\overline{x}_2$: pertanto nell'inversione di x_2 a partire dalla condizione $x_1 x_2 y = 111$, esistendo un intervallo Δ_τ in cui x_2 è già 0 e $\overline{x}_2$ non è ancora 1, si ha $y' = 0$. Si può allora eliminare l'alea aggiungendo un terzo insieme, indipendente da x_2, che assicuri durante l'inversione di x_2 il valore 1 della funzione.

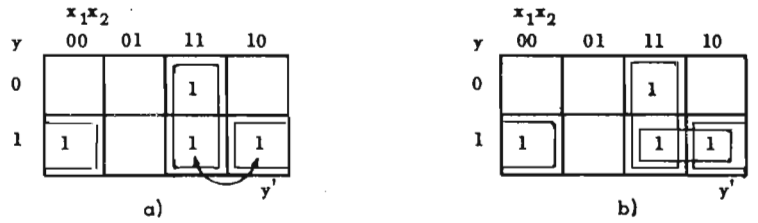


Fig.VI.76 - Eliminazione logica di un'alea statica.

Sulla mappa di Karnaugh, la y' deve contenere i 3 insiemi mostrati nella fig.VI.76b; il circuito privo di alee statiche avrà le equazioni:

$$\begin{cases} y' = x_1 x_2 + \overline{x}_2 y + x_1 y \\ z = y \ , \end{cases}$$

e la forma della fig.VI.77.

Per eliminare le alee statiche, in definitiva, basta realizzare le y_i' con insiemi connessi, nel senso mostrato dalla fig.VI.76b; ciò evita anche i transitori del tipo 0-1-0. I circuiti così progettati si chiamano *privi di alee* (*hazard free*).

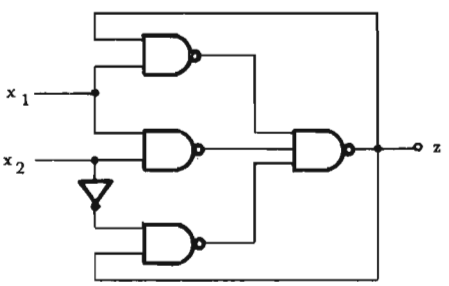


Fig.VI.77 - Circuito senza alee statiche.

E' evidente che le alee statiche sono pericolose solo se le transizioni che le originano possono effettivamente verificarsi: questa informazione può essere ricavata soltanto dalla matrice primitiva, perché viene distrutta nel passaggio alla macchina equivalente minima.

VI.10.2 - Alee dinamiche.

Sono quelle che causano 3 cambiamenti successivi di un'uscita che dovrebbe cambiare soltanto una volta (es.: 0-1-0 invece di 0-1-0-1);

derivano dalle diverse lunghezze delle vie percorse da un segnale per raggiungere l'ingresso del loop di reazione, possono esistere soltanto nei circuiti a più di due livelli, e sono pericolose quando le uscite transitorie durano abbastanza da provocare effetti indesiderati.

Si può dimostrare che le condizioni necessarie e sufficienti per il verificarsi di un'alea dinamica sono: l'esistenza di 2 segnali x e $\overline{x}$ nell'espressione della funzione di eccitazione y' e di due segnali x e $\overline{x}$ (di cui almeno 1 su un percorso diverso dai primi) nell'espressione della funzione inversa $\overline{y}'$.

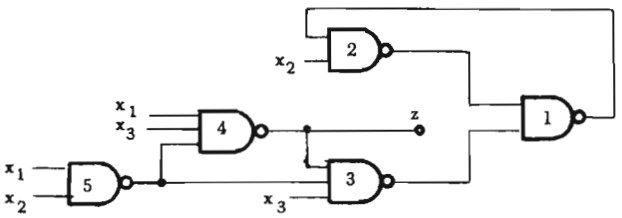


Fig.VI.78 - Circuito con alea dinamica.

In pratica, le condizioni sono verificate quando esistono almeno 3 vie che portano il segnale x all'elemento alla cui uscita si realizza una y, se almeno una delle vie comporta l'inversione della x e almeno una no.

Un'alea dinamica si presenta nel circuito (fig.VI.78) di equazioni:

$$\begin{cases} y' = \overline{x}_1 x_3 + x_2 y \\ z = \overline{x}_1 + x_2 + \overline{x}_3 \end{cases},$$

la cui matrice di flusso è rappresentata nella fig.VI.79.

y \ $x_1x_2x_3$	000	001	011	010	110	111	101	100
0	⓪/1	1/1	1/1	⓪/1	⓪/1	⓪/1	⓪/0	⓪/1
1	0/1	①/1	①/1	①/1	①/1	①/1	0/0	0/1

y'z

Fig.VI.79 - Alea dinamica.

Le condizioni per l'esistenza dell'alea dinamica sono verificate, in quanto il segnale x_2 raggiunge il NAND 1 attraverso 3 itinerari (2-1, 5-3-1, 5-4-3-1), il secondo dei quali comporta la necessaria inversione.

Se teniamo separati i 3 segnali, indicandoli con $x_2^{(1)}$, $x_2^{(2)}$ e $x_2^{(3)}$, a seconda che seguano, rispettivamente, gli itinerari: 2-1, 5-3-1 o 5-4-3-1, possiamo scrivere:

$$y' = x_2^{(1)} y + x_3 (\overline{x}_1 + \overline{x}_2^{(2)}) (x_1 x_2^{(3)} + \overline{x}_1 + \overline{x}_3) \ .$$

Nella transizione dallo stato stabile in cui $x_1 x_2 x_3 y = 1111$ allo stato 1010, provocata dal passaggio di x_2 da 1 a 0, $x_2^{(1)}$, $x_2^{(2)}$, $x_2^{(3)}$ – data la diversa lunghezza dei rispettivi itinerari – cambiano valore nello ordine: $x_2^{(1)} \rightarrow x_2^{(2)} \rightarrow x_2^{(3)}$. Per essere, durante tutto il periodo, y costantemente uguale a 1, si ha così:

1) quando $x_2^{(1)} = x_2^{(2)} = x_2^{(3)} = y = 1$, $y' = 1$ (stato iniziale);
2) quando $x_2^{(1)} = 0$; $x_2^{(2)} = x_2^{(3)} = y = 1$, $y' = 0$
3) quando $x_2^{(1)} = x_2^{(2)} = 0$; $x_2^{(3)} = y = 1$, $y' = 1$
4) quando $x_2^{(1)} = x_2^{(2)} = x_2^{(3)} = 0$; $y = 1$, $y' = 0$ (stato finale).

Le alee dinamiche possono essere rimosse aggiungendo un ritardo concentrato nelle vie più rapide (itinerari $x_2^{(1)}$ e $x_2^{(2)}$) o più semplicemente, eliminando le fattorizzazioni e costruendo il circuito a 2 livelli: anch'esse, infatti, come le alee statiche, nascono dalla struttura fisica del circuito, non dalla natura logica del problema.

VI.10.3 - Alee essenziali.

Le alee essenziali sono inerenti alla struttura logica di alcuni problemi, non solo alla realizzazione fisica dei rispettivi circuiti. Sono le alee più pericolose, sia perché nessun artificio logico può evitarle, sia perché non causano false uscite o brevi transitori, ma provocano sempre un'errato funzionamento del circuito.

x		
0	1	
①	2	
3	②	
③	④	

x	
0	1
①	2
3	②
③	4
	④

Fig.VI.80 - Alee essenziali.

Un'alea essenziale nasce ogni volta che nella matrice di flusso esistono uno stato S_1 ed un ingresso x tali che tre variazioni consecu-

tive di x, a partire da S_1, portino il circuito in uno stato (S_4) diverso da quello (S_2) cui si giunge dopo il primo cambiamento di x. Graficamente, occorre e basta che 2 colonne della matrice degli stati abbiano una delle strutture della fig.VI.80.

Il circuito può, in questi casi, partendo da ①, trovarsi per x = 0→1 nello stato ④ invece che in ②, come mostra l'esempio seguente.

Esempio 1: *Analisi di un circuito realizzato senza alee statiche e dinamiche, secondo le tabelle della fig.VI.81.*

y_1y_2 \ x	0	1
00	①	2
01	3	②
11	③	4
10	-	④

y_1y_2 \ x	0	1
00	00	01
01	11	01
11	11	10
10	-	10

y'

Fig.VI.81 - Esempio di alea essenziale.

Partendo dallo stato ①, se la variazione 0→1 di x viene avvertita contemporaneamente dagli elementi Ey_1 ed Ey_2 da cui vengono prelevate y_1 e y_2, la y_2' assume il valore 1, il sistema passa nello stato instabile 2, e raggiunge poi lo stato stabile desiderato.

Se invece, per una ragione qualsiasi, la variazione di x raggiunge Ey_2 prima di Ey_1, l'evoluzione del circuito è la seguente:

1) partendo da ①, x passa da 0 a 1;

2) Ey_2 risente la variazione di x, porta la sua uscita a 1 e fa evolvere il sistema prima nello stato instabile 2, poi nello stato stabile ②;

3) se Ey_1 si accorge del mutamento di y_2 prima che – dagli stati precedenti – gli sia giunta notizia del cambiamento di x, viene a trovarsi in una condizione in cui vede x = 0 e y_2 = 1, condizione che interpreta come lo stato instabile 3, che ha, appunto, x = 0, y_1 = = 0, y_2 = 1. In risposta a questa sollecitazione, Ey_1 porta y_1 ad 1 e fa evolvere il circuito verso lo stato stabile ③;

4) quando, finalmente, Ey_1 avente il cambiamento di x, il circuito, che si trovava nello stato ③, passa allo stato ④, diverso da quello cui era destinato.

L'alea essenziale nasce, in definitiva, perché il cambiamento di una variabile d'ingresso provoca la variazione di una variabile interna che si propaga nel circuito più rapidamente del cambiamento che l'ha provocata.

Le alee essenziali si presentano, purtroppo, nei circuiti più comuni e più importanti dei calcolatori (contatori, shift register, ...): benché si possano progettare dei sistemi relativamente insensibili a esse; l'unico mezzo sicuro per eliminarle è l'inserimento di ritardi tali che il cambiamento delle y avvenga solo dopo che le variazioni degli ingressi abbiano raggiunto tutti gli elementi del circuito su cui, in qualsiasi modo, influiscono. Una soluzione radicale del problema è l'uso di un impulso di clock che faccia avvenire ogni variazione nel momento più opportuno: ma, così, il circuito non è più asincrono.

VI.10.4 - Alee multiple.

Le alee multiple si hanno quando i 3 tipi di alee già visti coinvolgono più di una variabile. La loro eliminazione complica in modo notevole il problema della sintesi, e non è possibile che in casi particolari. Per questa ragione, si prescrive sempre che gli ingressi dei circuiti asincroni non possano cambiare contemporaneamente.

Lo studio effettuato sulla natura delle alee ci permette, a questo punto, di riassumere le condizioni necessarie per il corretto funzionamento dei circuiti sequenziali asincroni:

1) la rete logica deve essere priva di alee;

2) le variazioni di stato non debbono comportare corse critiche;

3) gli ingressi non debbono variare contemporaneamente;

4) nessuna variazione di stato deve avvenire prima che la variazione di ingresso che la origina sia giunta in ogni parte della rete logica;

5) le condizioni d'ingresso debbono permanere per un tempo sufficiente perché il circuito raggiunga uno stato stabile.

Le prime 2 condizioni si riflettono sul progetto logico del circuito, e servono ad evitare una sua errata evoluzione. La terza condizione vincola i circuiti di comando a monte di quello in esame, cioè l'ambiente esterno, e serve ad evitare alee multiple. La quarta condizione riguarda la velocità del circuito, anzi l'entità del ritardo da inserire nei loop di reazione, e serve ad evitare le conseguenze delle alee essenziali. L'ultima condizione, infine, riguarda la velocità dei circuiti esterni di comando, in relazione alla velocità del circuito sequenziale stesso ed è assolutamente necessaria perché su di essa è basata tutta la sintesi. Se si conosce il ritardo massimo alla risposta dei componenti del circuito, è facile trovare il valore limite della velocità di variazione degli ingressi. Sul problema torneremo, per altre ragioni, alla fine del prossimo capitolo.

Concludiamo la sintesi dei circuiti sequenziali asincroni esponendo infine i criteri di progetto di alcuni circuiti di uso generale nei calcolatori. Il significato, l'importanza e l'uso di questi circuiti sarà chiarito nell'ultimo capitolo, dedicato ai sistemi numerici, dove saranno esposte le versioni sincrone dei circuiti stessi, assai più note e comuni.

VI.10.5 - Contatore binario.

Il contatore binario (binary trigger) è un circuito ad un ingresso e un'uscita il cui valore, inizialmente 0, cambia ogni volta che l'ingresso passa da 0 a 1.

Nella fig.VI.82 è mostrata la matrice primitiva, a 4 righe e 2 colonne, del circuito; non esistono stati equivalenti o pseudo-equivalenti, e non è possibile fondere nessuna coppia di righe. Il diagramma (figura VI.83a) e la tabella (fig.VI.83b) delle transizioni portano alla tabella degli stati e alla tavola di flussi delle figg. VI.83c e VI.83d.

Stati	x = 0	x = 1	z
1	①	2	0
2	3	②	1
3	③	4	1
4	1	④	0

Fig.VI.82 - Matrice primitiva di un contatore binario.

Per ricavare le equazioni del circuito, conviene rappresentare separatamente le mappe delle funzioni y_1' e y_2', alla ricerca di eventuali alee statiche; ottenute le equazioni, verificheremo poi che non vi siano alee dinamiche. Ci occuperemo, infine, delle alee essenziali, per ora inevitabili (v. fig.VI.83c).

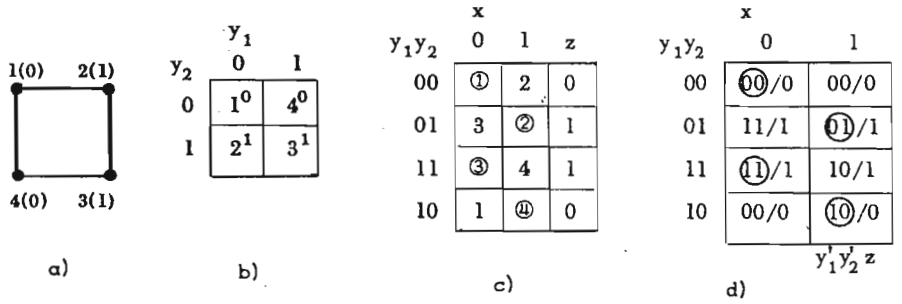


Fig.VI.83 - Costruzione della tabella di flusso per il contatore binario.

Sia y_1' che y_2', sintetizzate nella forma minima a 2 livelli, presentano le alee statiche della fig.VI.84a; per evitarle bisogna coprire i minterm delle 2 funzioni con i 3 insiemi della fig.VI.84b.

Si hanno così le equazioni, prive di alee dinamiche:

$$\begin{cases} y_1' = \bar{x}y_2 + xy_1 + y_1y_2 = y_1\,(x + y_2) + \bar{x}y_2 \\ y_2' = \bar{x}y_2 + x\bar{y}_1 + \bar{y}_1y_2 = \bar{y}_1\,(x + y_2) + \bar{x}y_2 \\ z = y_2\ , \end{cases}$$

da cui deriva il circuito a 6 NAND e 2 invertitori della fig.VI.85.

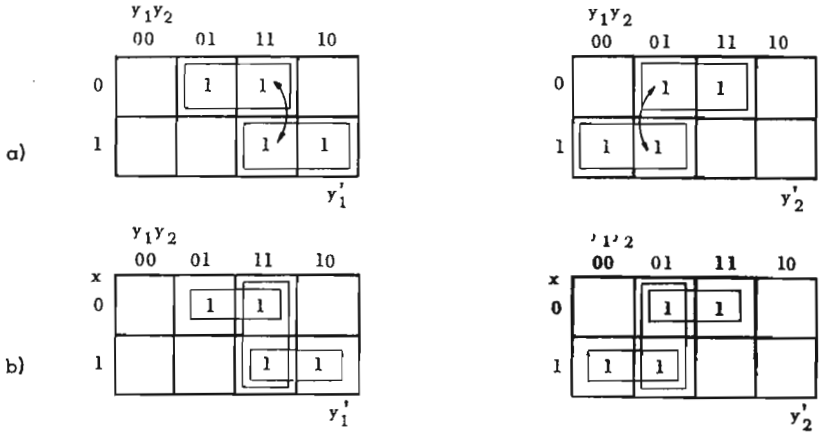


Fig.VI.84 - Sintesi, con le mappe di Karnaugh, di un contatore binario senza alee statiche.

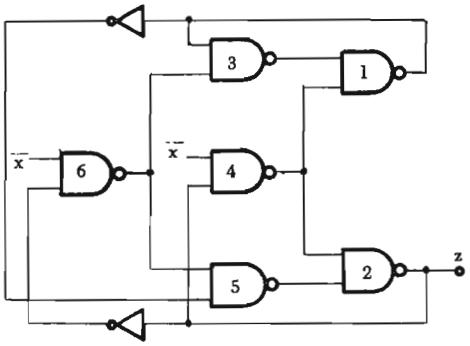


Fig.VI.85 - Circuito NAND-NOT per il contatore binario.

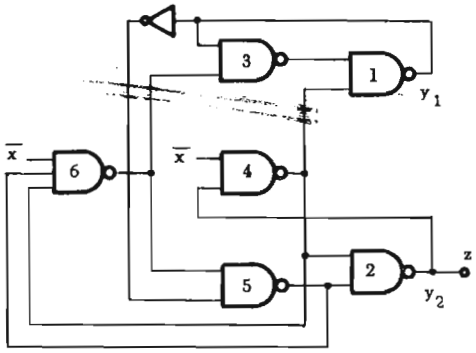


Fig.VI.86 - Contatore binario semplificato.

Applicando il *bundling* si può risparmiare un invertitore: sostituendo y_2 all'ingresso del NAND 6, con gli ingressi del NAND 2, si ottiene il circuito della fig.VI.86.

Osserviamo, poi, che all'uscita dal NAND 5 si ha la funzione:

$$y_5 = \overline{\overline{y_1}\, y_6} = y_1 + \overline{y_6} \quad ,$$

(si è indicato con y_i la funzione d'uscita dal NAND i).

Se si sostituisce l'ingresso y_1 con l'uscita del NAND 3, si ha:

$$y_5' = \overline{\overline{y_3}\, y_6} = \overline{y_3} + \overline{y_6} \quad ,$$

e, per essere:

$$y_3 = \overline{y_1\, y_6} = \overline{y_1} + \overline{y_6} \quad ,$$

risulta:

$$y_5' = \overline{y_3} + \overline{y_6} = \overline{\overline{y_1} + \overline{y_6}} + \overline{y_6} = y_1 y_6 + \overline{y_6} = y_1 + \overline{y_6} = y_5 \quad .$$

Si può allora eliminare anche l'ultimo invertitore, ottenendo il circuito a 6 NAND della fig.VI.87; in esso si è anche sostituito x a $\overline{x}$, essendo il contatore binario (v. figura VI.83c) invariante rispetto alla complementazione della variabile d'ingresso.

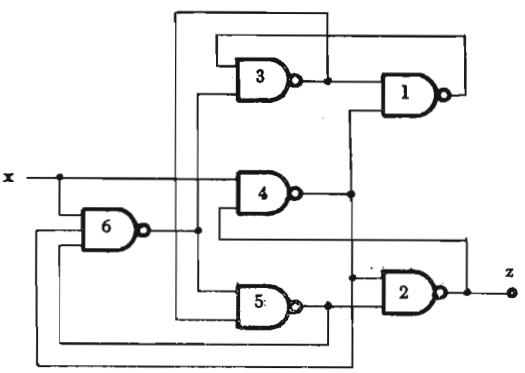


Fig.VI.87 - Versione finale del contatore binario.

Questo circuito non solo è più economico di quello della fig.VI.85, ma è praticamente insensibile alle alee essenziali, perché la y_1 agisce con un ritardo quasi certamente superiore a quello relativo alla x, dovendo attraversare un maggior numero di NAND. Per avere, però, l'assoluta sicurezza di funzionamento, occorrerebbe introdurre un elemento di ritardo dopo il NAND 2.

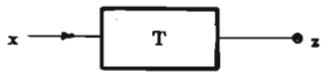
VI.10.5.1 - Contatori binari a più stadi.

Il circuito della fig.VI.87 è generalmente rappresentato col simbolo della fig.VI.88, ove la T sta ad indicare che il suo comportamento è analogo a quello del flip-flop T, presentando un'uscita $z = 1$ se e solo se, a partire dallo stato iniziale, si sono verificate un numero dispari di variazioni 0-1 degli ingressi.

Il contatore binario si utilizza per costruire contatori a base $n = 2^k$, cioè i circuiti con un ingresso e k uscite che contano gli impulsi

da 0 a n - 1, manifestando sulle uscite $z_1 z_2 \ldots z_k$, rispettivamente, i bit di peso 1, 2, 4, 2^{k-1}; il numero di impulsi sull'ingresso x è dato – a meno di un fattore di n – dalla somma dei pesi delle uscite al valore 1.

Fig.VI.88 - Simbolo di un contatore binario.



Ad esempio, nel contatore a 4 uscite della fig.VI.89, capace di contare da 0 a 15, il valore 1-0-0-1 su $z_1 \ldots z_4$ sta ad indicare che su x sono arrivati 9 (= 8 + 1) impulsi (+ in).

Fig.VI.89 - Simbolo di un contatore a base 16.

Per capire come dal flip-flop T si ottengono i contatori a base n, esaminiamo il comportamento delle funzioni y_1 e y_2 per effetto di un treno di impulsi x, comportamento mostrato dal diagramma temporale della fig.VI.90a, costruito dalla tabella degli stati di fig.VI.83c.

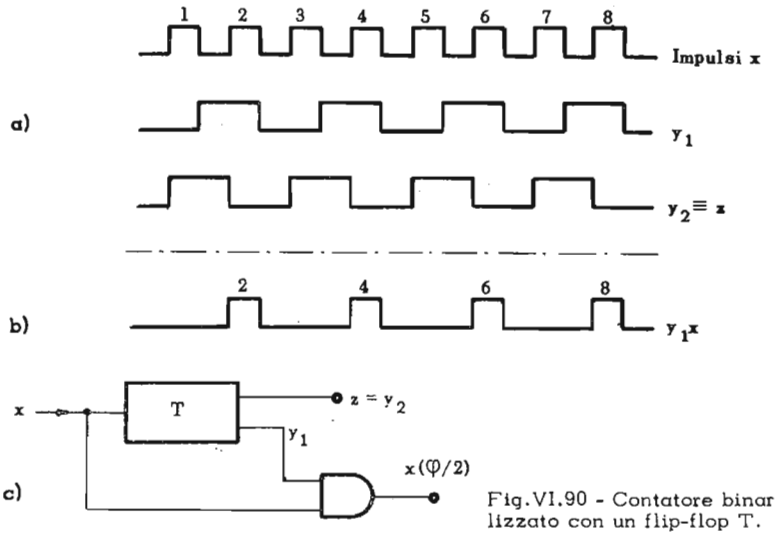


Fig.VI.90 - Contatore binario realizzato con un flip-flop T.

Il diagramma, che riporta tutte le relazioni temporali tra $x_1 y_1$ e y_2 mostra che il prodotto $y_1 \cdot x$ (fig.VI.90b) è 1 soltanto durante gli im-

pulsi di un numero pari. Di conseguenza, il circuito della fig.VI.90c, che blocca gli impulsi dispari e fa passare i pari, costituisce un *divisore di frequenza* perché, se la frequenza (φ) degli impulsi x è costante, alla sua uscita si ha un treno d'impulsi di eguale ampiezza e durata, ma di frequenza $\varphi/2$.

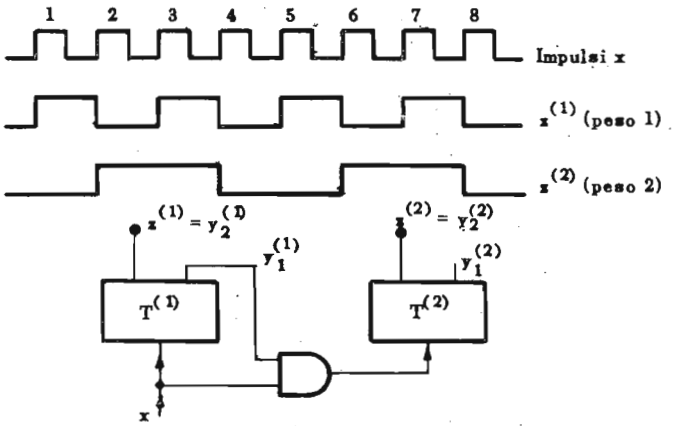


Fig.VI.91 - Contatore a base 4 con 2 flip-flop T.

Un secondo contatore binario, al cui ingresso si mandi l'uscita del divisore di frequenza, conterà per due gli impulsi che riceverà; cioè in definitiva gli impulsi x saranno contati per quattro ($z = 0$ per 0, 4, 8 ... impulsi; $z = 1$ per 2, 6, 10 ... impulsi). Considerate insieme, $z^{(1)}$ e $z^{(2)}$ rappresentano così le uscite di un contatore a base 4, che conta gli impulsi da 0 a 3 (fig.VI.91).

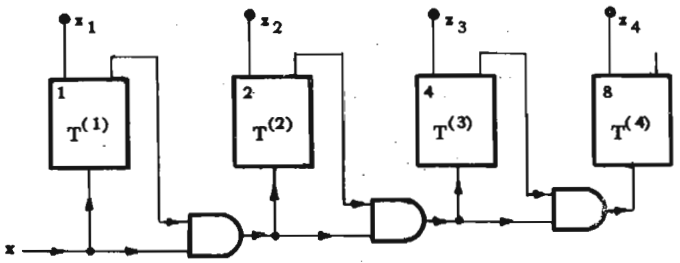


Fig.VI.92 - Contatore a base 16 con 4 flip-flop T.

Sullo stesso principio, si costruiscono contatori più complessi: nella fig.VI.92 è mostrato il contatore della fig.VI.89 (contatore asincrono in serie a base 16).

VI.10.5.2 - Contatore ad anello.

Si chiama contatore ad anello (Ring counter) un circuito ad un ingresso ed n uscite: l'uscita z_j è 1 quando e solo quando si sono avute $j + k_n$ ($k = 0, 1 \ldots$) variazioni dell'ingresso. Tutte le uscite z_k ($k \neq j$) sono 0.

Per fissare le idee, prendiamo $n = 4$.

Siano $z_1 z_2 z_3 z_4$ le uscite del circuito; x l'ingresso; la matrice primitiva, a 4 righe e 2 colonne (fig.VI.93) non può essere minimizzata.

La tavola di flusso (fig.VI.94) origina le equazioni:

$$\begin{cases} y_1' = \overline{x}y_2 + x\overline{y_1} + \overline{y}_1 y_2 \\ y_2' = \overline{x}y_2 + xy_1 + y_1 y_2 \\ z_1 = \overline{xy_2} \\ z_2 = x\overline{y}_1 \\ z_3 = \overline{x}y_2 \\ z_4 = xy_1 \quad . \end{cases}$$

Il circuito corrispondente, senza alee statiche e dinamiche, è piuttosto complesso, e richiede 8 NAND e 6 invertitori; uno più economico può essere ottenuto ripartendo diversamente gli elementi delle reti di memoria e d'uscita, precisamente aggiungendo due loop di reazione ri-

x		
0	1	$z_1z_2z_3z_4$
①	2	1000
3	②	0100
③	4	0010
1	④	0001

Fig.VI.93 - Matrice primitiva di un contatore ad anello a base 4.

y_1y_2	x = 0	x = 1
00	(00)/1000	01/0100
01	11/0010	(01)/0100
11	(11)/0010	10/0001
10	00/1000	(10)/0001

$y_1' y_2' z_1 z_2 z_3 z_4$

Fig.VI.94 - Tavola di flusso di un contatore ad anello a base 4.

dondanti sulla prima, per eliminare completamente la seconda. La soluzione migliore deriva dalla tavola di flusso della fig.VI.95 dove, per maggior chiarezza, sono state indicate anche le transizioni multiple. Scegliendo opportunamente le caselle opzionali, si ricavano le equazioni:

$$\begin{cases} y_1' = z_2 = xy_4 + y_1\overline{y}_2 \\ y_2' = z_3 = \overline{x}y_1 + y_2y_3 \\ y_3' = z_4 = xy_2 + y_3\overline{y}_4 \\ y_4' = z_1 = \overline{x}y_3 + \overline{y}_1y_4 \end{cases}$$

cui corrisponde un circuito a 12 NAND (fig.VI.96) che ha l'importante proprietà di essere iterativo, cioè estensibile a qualsiasi numero di uscite, molto semplice nei collegamenti e poco influenzabile da eventuali ritar-

$y_1y_2y_3y_4$	x = 0	x = 1
0000	–	–
0001	(0001)/0001	1001/0001
0011	0001/–	–
0010	0011/0010	(0010)/0010
0110	–	0010/–
0111	–	–
0101	–	–
0100	(0100)/0100	0110/0100
1100	0100/–	–
1101	–	–
1111	–	–
1110	–	–
1010	–	–
1011	–	–
1001	–	1000/–
1000	1100/1000	(1000)/1000

$y_1'y_2'y_3'y_4'z_1z_2z_3z_4$

Fig.VI.95 - Tavola di flusso per il contatore ad anello.

di, in quanto ogni stadio – ricevendo come ingressi le reazioni dagli stati seguenti – conosce quando un'informazione è stata raccolta e può essere cancellata.

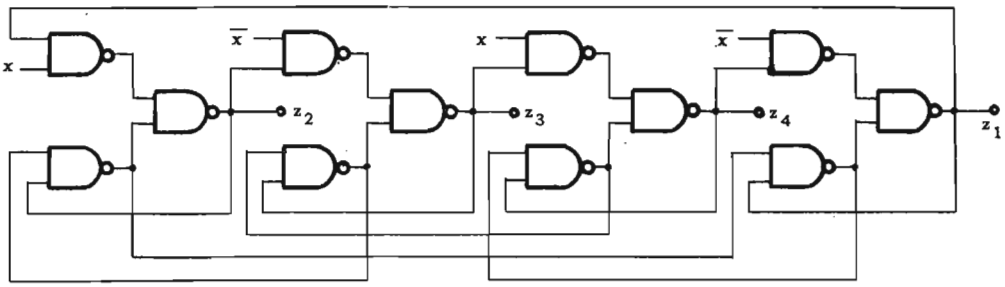


Fig.VI.96 - Circuito più economico per il contatore ad anello.

VI.10.5.3 - Registro a scorrimento.

Si chiama registro a scorrimento (shift register) un circuito iterativo la cui cella i^{ma} ($i = 1, 2, \ldots, n$) ha 2 ingressi: uno a livelli (x_i) coincidente con l'uscita z_{i-1} della cella precedente, l'altro (S) impulsivo. Dopo ogni impulso, il livello all'ingresso della cella compare sull'uscita, e vi rimane fino al prossimo impulso. Il registro a scorrimento, mostrato nello schema della fig.VI.97, serve a far avanzare un segnale di n bit in serie, alla velocità di un bit ad ogni impulso.

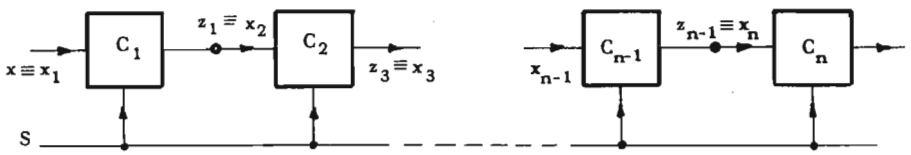


Fig.VI.97 - Schema a blocchi di un circuito a scorrimento.

La matrice primitiva del circuito è mostrata nella fig. VI.98a; la minimizzazione, che si tralascia per semplicità, porta alla matrice delle sequenze della fig.VI.98b, dove si è posto $1' = (1 \cdot 2 \cdot 3)$; $2' = (4)$; $3' = (5 \cdot 6 \cdot 7)$, $4' = (8)$. La tabella delle transizioni, codificata in modo da avere $z = y_1$ (fig.VI.98c), porta alla matrice di flusso della fig.VI.98d da cui, eliminando le alee statiche e dinamiche, si hanno le equazioni:

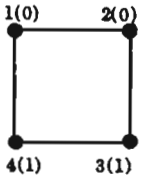
$$\begin{cases} y_1' = \overline{S}y_2 + Sy_1 + y_1y_2 = (S + y_2)\, y_1 + \overline{S}y_2 \\ y_2' = Sx + y_2x + \overline{S}y_2 = (S + y_2)\, x + \overline{S}y_2 \\ z = y_1 \ . \end{cases}$$

Sx 00	01	11	10	z
①	2	–	3	0
–	②	4	–	0
1	–	–	③	0
–	5	④	–	0
6	⑤	7	–	1
⑥	–	–	8	1
–	5	⑦	–	1
1	–	–	⑧	1

a)

	Sx 00	01	11	10	z
1	①	①	2	①	0
2	–	3	②	–	0
3	③	③	③	4	1
4	1	–	–	④	1

b)



c)

y_1y_2 \ Sx	00	01	11	10
00	(00)/0	(00)/0	01/0	(00)/0
01	–/0	11/0	(01)/0	–/0
11	(11)/1	(11)/1	(11)/1	10/1
10	00/1	–/1	–/1	(10)/1
				$y_1'y_2'z$

d)

Fig.VI.98 - Sintesi di un registro a scorrimento.

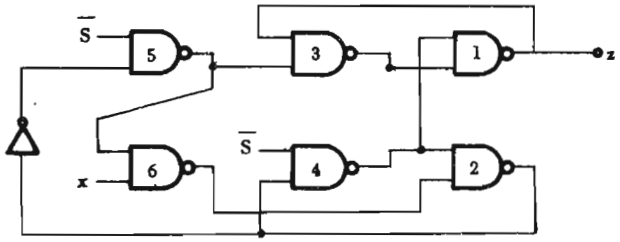


Fig.VI.99 - Registro a scorrimento.

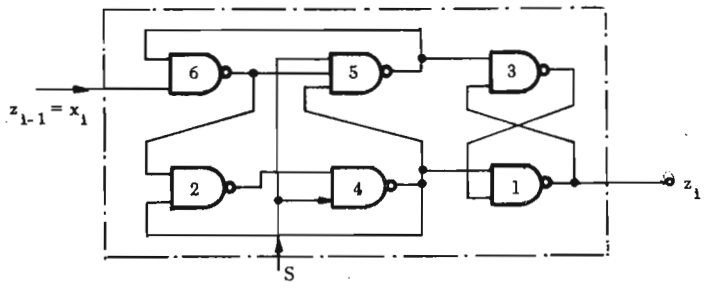


Fig.VI.100 - Schema semplificato del registro a scorrimento.

Il relativo circuito (fig.VI.99) richiede 6 NAND e un invertitore.

Si può eliminare l'invertitore prendendo y_2 dalle uscite dei NAND 4 e 6. Il circuito risultante, che è la generica cella del registro a scorrimento, è mostrato nella fig.VI.100.

*

BIBLIOGRAFIA

1. CALDWELL S.H.: *Switching Circuits and Logical Design* - John Wiley & Sons, 1958.

2. HUFFMAN D.A.: *The Synthesis of Sequential Switching Circuits* - Journal of the Franklin Institute, 1954.

3. HUFFMAN D.A.: *The Design and Use of Hazard-Free Switching Networks* - Journal of the ACM - Vol.4, Gennaio 1957.

4. MALEY G.A. - EARLE J.: *The Logic Design of Transistor Digital Computer* - Prentice Hall, 1963.

5. MARCUS M.P.: *Switching Circuits for Engineering* - Prentice Hall, 1962.

6. METZE G.A.: *Many - Valued Logic and the Design of Switching Circuits* - University of Illinois, 1955.

7. MULLER D.E.: *Asynchronous Logics and Application to Information Processing - Proceedings of a Symposium on the Application of Switching Theroy* - Stanford University, 1962.

8. UNGER S.H.: *Hazard and Deloys in Asynchronous Sequential Switching Circuits* - IRE Transaction on EC - Vol.6, n.1, Marzo 1959.

*

CAPITOLO VII

CIRCUITI SEQUENZIALI SINCRONI

VII.1 - Generalità.

I circuiti sequenziali sincroni, come già accennato, sono quelli in cui le variazioni degli ingressi esterni e interni avvengono su comando di un clock, mentre il ricordo delle sequenze passate, cioè lo stato del circuito, viene immagazzinato in appositi organi di memoria (flip-flop), non affidato ai ritardi concentrati nei transistor o distribuiti nei

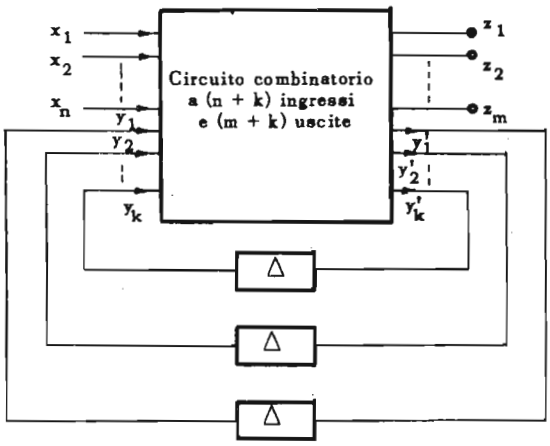


Fig.VII.1 - Schema a blocchi di un circuito sequenziale sincrono.

loop di reazione dei circuiti asincroni. Il modello ideale dei circuiti sequenziali sincroni è sempre quello del cap.V, riportato nella fig.VII.1; diverso è però il significato degli elementi di ritardo, realizzati con cir-

cuiti che dipendono anche dal tipo di flip-flop adoperato. A titolo d'esempio, nella fig. VII.2 è mostrato lo schema logico completo di un *elemento di ritardo* con un flip-flop SR a NOR.

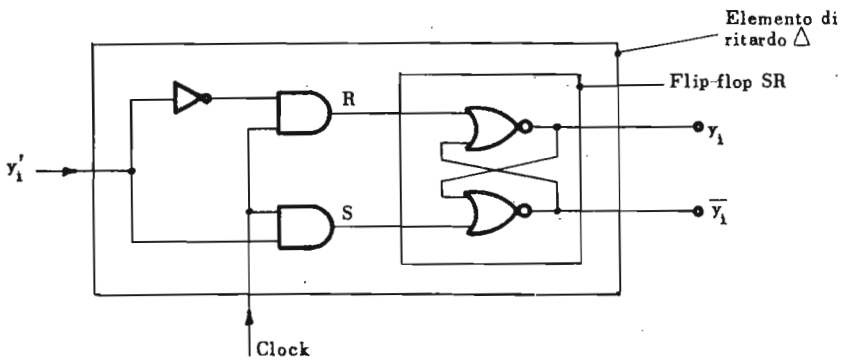


Fig. VII.2 - Elemento di ritardo.

L'elemento di ritardo è qui simulato dall'azione di un flip-flop e di un clock, che ha lo scopo di sincronizzare le operazioni del circuito, essendo la sua frequenza coincidente col ritardo Δ del modello fondamentale. In questo modo, la configurazione delle x relativa all'istante di clock t, insieme alla configurazione delle y corrispondenti alle y' generate dal circuito combinatorio all'istante $t - \Delta$, fornirà all'uscita del circuito combinatorio stesso un nuovo insieme di segnali z(t) e y'(t). Questi ultimi segnali saranno portati sugli ingressi solamente al nuovo impulso di clock, e reagiranno con la configurazione delle x relativa allo istante $t + \Delta$.

Ovviamente, l'impulso di clock deve essere terminato prima che ogni y abbia assunto il valore della relativa y'.

In questo capitolo tratteremo prima la sintesi dei circuiti sincroni, poi la loro analisi, ritenendo più logico quest'ordine rispetto a quello tradizionale. Descriveremo poi una particolare categoria di circuiti impulsivi, per la quale è possibile compiere importanti semplificazioni, e concluderemo l'argomento confrontando le caratteristiche e le prestazioni dei circuiti sincroni e asincroni.

VII.2 - Sintesi dei circuiti sequenziali sincroni.

La sintesi dei circuiti sequenziali sincroni si effettua con un procedimento analitico proposto da Moore e Mealy, procedimento che presenta molte analogie con quello di Huffman e si basa sempre sulla teoria

delle macchine a stati finiti. Per renderne più semplice l'esposizione, dividiamo questo metodo in cinque parti:

1) Costruzione del diagramma e della tabella degli stati per una macchina sequenziale sincrona $\mathfrak{M}$, a partire dalla descrizione verbale del comportamento del circuito.

2) Minimizzazione del numero degli stati di $\mathfrak{M}$ e costruzione di una tabella degli stati (matrice primitiva) per la macchina equivalente minima $\mathfrak{M}'$.

3) Codificazione degli stati di $\mathfrak{M}'$.

4) Costruzione della tavola di flusso per il modello ideale del circuito sequenziale.

5) Scelta del tipo di flip-flop da usare come elemento di memoria, e costruzione del circuito sequenziale reale.

Al contrario di quanto avviene nei circuiti asincroni, la codificazione degli stati è molto semplice, e non richiede precauzioni particolari; complessa è invece la trasformazione del circuito ideale nel circuito reale.

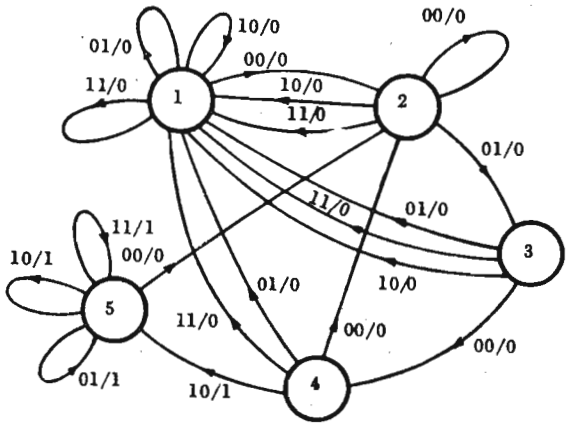
VII.2.1 - Diagramma degli stati.

Il diagramma degli stati descrive in maniera completa e precisa tutte le relazioni tra le sequenze di ingressi e di uscite del circuito, riferendosi al modello di Moore o Mealy di una macchina sequenziale sincrona $\mathfrak{M}$. Per essere il circuito sincrono, non è necessario raggiungere uno stato stabile per variare gli ingressi; anzi, perde completamente significato la distinzione tra stati stabili e instabili; l'evoluzione del circuito stesso, del resto, è controllata dalla successione degli impulsi di clock.

Esempio 1: *Costruire il diagramma degli stati per una macchina sincrona* $\mathfrak{M}$ *a 2 ingressi* x_1x_2 *e un'uscita z che va ad 1 al termine della sequenza* $x_1x_2 = 00 \rightarrow 01 \rightarrow 00 \rightarrow 10$ *e vi rimane finché gli ingressi non siano entrambi tornati a 0.*

Il diagramma di Mealy della $\mathfrak{M}$ è mostrato nella fig.VII.3a. Occorre prevedere 5 stati, perché 4 sono le condizioni della sequenza da ricordare e uno stato serve a prevedere le sequenze con uscita 0. Nella fig.VII.3b è ricavata la tabella degli stati corrispondente al diagramma. Si noti che gli stati 2 e 5 sono i soli stati *stabili*, e che la macchina $\mathfrak{M}$ è completamente specificata.

a)



b)

Stati	x_1x_2 00	01	11	10
1	2/0	1/0	1/0	1/0
2	2/0	3/0	1/0	1/0
3	4/0	1/0	1/0	1/0
4	2/0	1/0	1/0	5/1
5	2/0	5/1	5/1	5/1

$\mathcal{M}$

Fig.VII.3 - Diagramma degli stati e relativa tabella per una macchina $\mathcal{M}$.

I due esempi che seguono mostrano come, già da ora, una macchina sincrona si differenzi logicamente da quella asincrona che realizza la stessa relazione tra le sequenze d'ingresso e d'uscita.

Esempio 2: *Diagramma degli stati di una macchina sequenziale* $\mathcal{M}$ *a 2 ingressi* x_1x_2 *e un'uscita z che va a 1 al termine della sequenza* $x_1x_2 = 00 \rightarrow 01 \rightarrow 11$ *e torna a 0 per qualsiasi variazione degli ingressi.*

Se la $\mathcal{M}$ deve essere sincrona, possiamo realizzarla con 4 stati, che ricordano appunto i 3 valori della sequenza e distinguono gli ingressi $x_1x_2 = 11$ con uscita 0 e 1 (fig.VII.4).

Se la $\mathcal{M}$ deve essere asincrona, il diagramma deve mettere in evidenza che ogni variazione di ingresso deve avvenire su uno stato stabile (fig.VII.5).

Il numero degli stati è rimasto lo stesso, e i 2 diagrammi sono simili, perché la natura del problema era *quasi* asincrona: nella fig.VII.4 ci sono, infatti, ben 3 stati (1--2-4) stabili.

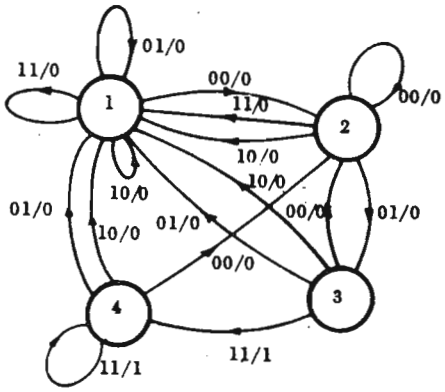


Fig.VII.4 - Diagramma di stato di una macchina sincrona.

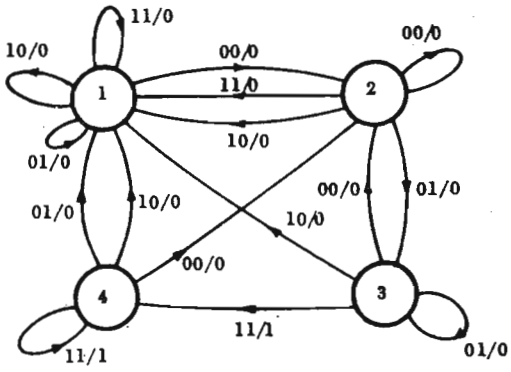


Fig.VII.5 - Diagramma di stato di una macchina asincrona.

Esempio 3: *Diagramma degli stati di una macchina sequenziale* $\mathfrak{M}$ *con 2 ingressi e una uscita che va ad 1 al termine della sequenza* $x_1x_2 = 00 \rightarrow 01 \rightarrow 11$ *e torna a 0 dopo un tempo* Δ *stabilito dalla frequenza di un clock che comanda anche le variazioni degli ingressi.*

Essendo, nei confronti dell'uscita, la sequenza $x_1x_2 = 00 \rightarrow 01 \rightarrow 11 \rightarrow 11$ del tutto eguale alla $x_1x_2 = 00 \rightarrow 01 \rightarrow 11 \rightarrow 10$ (00, 01), la macchina sincrona $\mathfrak{M}$ avrà 3 soli stati (figura VII.6).

Volendo realizzare la stessa sequenza ingressi-uscite con una macchina asincrona $\mathfrak{M}^*$ (in pratica, con un circuito senza flip-flop), bisogna considerare il clock co-

me un ulteriore ingresso c. Poiché tutti i cambiamenti degli ingressi dovranno avvenire in coincidenza con gli impulsi di clock, la sequenza $x_1x_2 = 00 \rightarrow 01 \rightarrow 11$ diventa: $cx_1x_2 = 100 \rightarrow 000 \rightarrow 101 \rightarrow 001 \rightarrow 111 \rightarrow 011$.

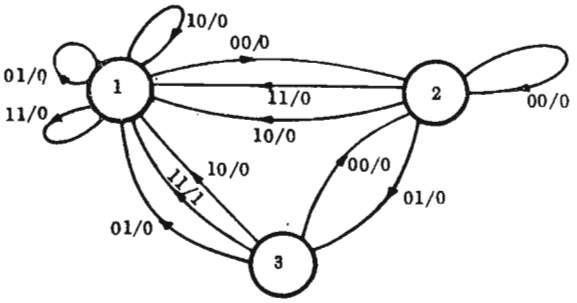


Fig.VII.6 - Diagramma di stato di una macchina sincrona.

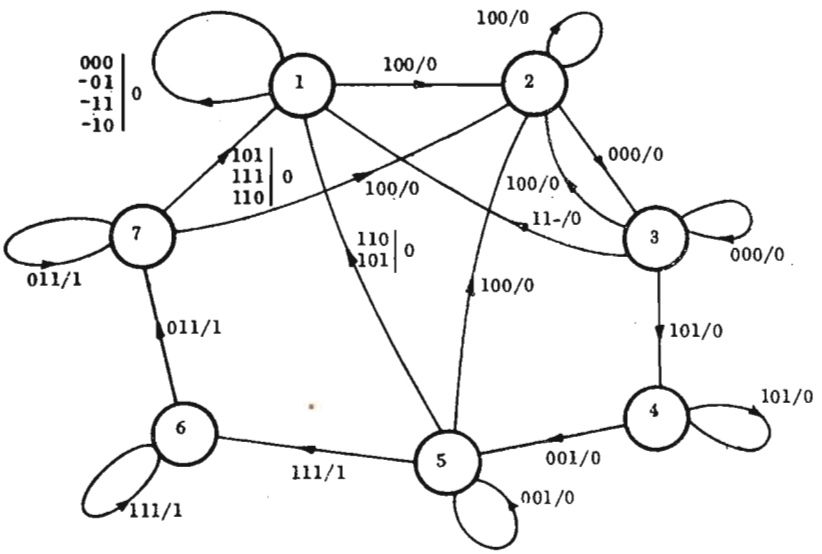


Fig.VII.7 - Diagramma di stato di una macchina asincrona.

Il diagramma degli stati della $\mathcal{M}^*$ è mostrato nella fig.VII.7.

Questa volta, per aver voluto trattare come asincrono un problema di natura sincrona, si è dovuto aumentare il numero degli stati e complicare notevolmente le transizioni.

Gli esempi 2 e 3 mostrano che per realizzare la stessa sequenza, una macchina sincrona impiega un numero di stati non maggiore della corrispondente macchina asincrona: si può dimostrare che questa proprietà è valida in generale, per qualsiasi tipo di problema. E' evidente, pertanto, la convenienza di risolvere problemi come quelli dell'esempio 3 con circuiti sincroni. Altri paragoni fra i 2 tipi di circuiti saranno fatti in seguito.

VII.2.2 - Minimizzazione e codificazione degli stati di $\mathfrak{M}$; equazioni del circuito.

La minimizzazione e la codificazione degli stati non vengono qui trattate perché non presentano aspetti nuovi. Basterà dire che, per essere tutte le variazioni delle y comandate dal clock, e per le ipotesi fatte sulla sincronizzazione tra le y', le y e le x, nei circuiti sincroni non possono verificarsi corse critiche. Non occorre, quindi, l'adiacenza degli stati legati da transizioni. Il numero delle variabili interne è sempre quello minimo, cioè il più piccolo intero k non minore del $\lg_2 = s$, essendo s il numero degli Stati. Ad esempio, per la macchina sequenziale sincrona minima $\mathfrak{M}$ della fig.VII.8, sono valide entrambe le codificazioni $q_1 = 0$, $q_2 = 1$ o $q_1 = 1$, $q_2 = 0$.

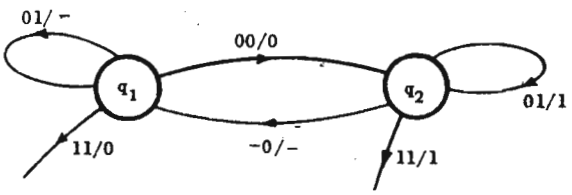


Fig.VII.8 - Diagramma degli stati di una macchina sincrona.

Anche per piccoli valori di s, sono possibili moltissime codifiche degli stati, che portano a diverse equazioni del circuito ideale e, in definitiva, a circuiti reali di differente complessità. Non esiste, purtroppo, un algoritmo che risolve il problema della codifica ottima in maniera semplice o, almeno, in modo sicuro.

Conviene, pertanto, provare un certo numero di assegnazioni, basandosi sull'intuito e sulla pratica, e paragonare i risultati ottenuti. Il problema è notevolmente complicato dal fatto che le equazioni delle y non sono quelle finali del circuito, come vedremo nei prossimi paragrafi.

Un metodo empirico per una buona assegnazione delle variabili interne consiste nell'assegnare gli stati secondari in modo che le risul-

tanti matrici di eccitazione abbiano il maggior numero di 0 e contengano gli 1 in caselle adiacenti, come mostra il seguente esempio.

Esempio 1: *Assegnazione degli stati secondari per la macchina* $\mathfrak{M}$ *della fig.VII.9 (per semplicità, non sono indicati i valori delle uscite).*

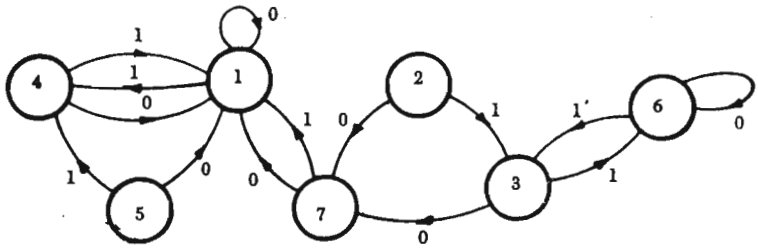


Fig.VII.9 - Diagramma degli stati di una macchina sincrona.

Essendo 7 gli stati, occorrono 3 loop di reazione.

Per avere il maggior numero di 0 nella matrice di eccitazione, il valore $y_1y_2y_3 = 000$ va assegnato allo stato che compare più volte nella matrice di flusso.

Conviene ricavare una *tabella inversa di flusso* che considera, per ogni ingresso e per ogni stato al tempo t, l'insieme degli stati in cui si trovava il circuito allo istante $t-\Delta$. Nella fig.VII.10 è riportata la tabella relativa alla macchina della figura VII.9. Lo stato che ha più termini nella colonna degli ingressi è lo stato 1, che pertanto viene codificato con $y_1y_2y_3 = 000$. Cerchiamo, ora, di rendere adiacenti il massi-

Stato presente	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

Fig.VII.10 - Tabella inversa di flusso.

mo numero di 1. Le adiacenze vanno ricercate per righe e per colonne: le prime provengono da transizioni verso uno stesso stato, e si riconoscono perché sono formate da stati che stanno nelle stesse caselle della tabella inversa (eccetto, naturalmente, quelli

della riga 1, che conterranno soltanto 0): nel nostro caso (2 • 6), (1 • 5), (2 • 3). Le seconde sono determinate dagli stati che si trovano nelle stesse colonne e appartengono al primo tipo. Così, gli stati della coppia (2 • 6) nella colonna x = 0, provocano l'adiacenza degli stati (6 • 7); gli stati (2 • 3) nella colonna x = 1 provocano l'adiacenza degli

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

a)

b)

$y_1y_2y_3$	x = 0	x = 1
000	000	100
001	110	000
011	011	101
010	000	100
110	000	000
111	–	–
101	110	011
100	000	000

$y'_1y'_2y'_3$

Fig.VII.11 - Assegnazione degli stati secondari alla macchina della fig.VII.9.

stati (3 • 6). In definitiva, la forma più economica delle y si otterrebbe rendendo adiacenti le coppie (2 • 3), (2 • 6), (1 • 5), (3 • 6), (6 • 7): questo non è tuttavia possibile con 3 valori di y. Una buona soluzione è quella indicata nella mappa della fig.VII.11a che, attraverso la tavola di flusso della fig.VII.11b, porta alle equazioni:

$$\begin{cases} y'_1 = x\overline{y_1} + \overline{xy_2}y_3 \\ y'_2 = y_1y_3 + \overline{x}y_3 \\ y'_3 = xy_3 + y_2y_3 \end{cases}.$$

Le coordinate della mappa sono state assegnate in un ordine qualsiasi; se sono note le uscite, invece, dato il valore 000 allo stato 1, le restanti coordinate vanno specificate in modo da minimizzare la rete d'uscita, coi criteri illustrati nel par.VI.6.

Esempio 2: *Assegnazione degli stati secondari alla macchina della fig.VII.12.*

Per i 4 stati di , occorrono 2 variabili interne. La tabella inversa (fig. VII.13) assegna il valore $y_1y_2 = 00$ allo stato 1. Risultano adiacenti le coppie (1 • 4) e (2 • 3). La relativa codificazione (fig.VII.14a), attraverso la tavola di flusso della fig.VII.14b, conduce alle equazioni:

$$\begin{cases} y'_1 = \overline{x} \\ y'_2 = \overline{y_1}\overline{y_2} + \overline{x}\,\overline{y_2} \\ z = \overline{x}\,\overline{y_1}\overline{y_2} + xy_1y_2 \end{cases}.$$

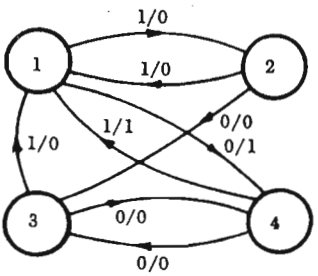


Fig.VII.12 - Diagramma degli stati di una macchina sequenziale sincrona.

Stato presente	Stato precedente x = 0	Stato precedente x = 1	n. termini
1	-	2, 3, 4	3
2	-	1	1
3	1, 4	-	2
4	2, 3	-	2

Fig.VII.13 - Tabella inversa per la macchina di fig.VII.12.

y_2 \ y_1	0	1
0	1	4
1	2	3

a)

b)

y_1y_2 \ x	0	1
00	11/1	01/0
01	10/0	00/0
11	10/0	00/1
10	11/0	00/0

$y_1' y_2'$ z

Fig.VII.14 - Assegnazione degli stati secondari alla macchina di fig.VII.12.

VII.2.3 - Costruzione del circuito reale.

Usando elementi di ritardo nei loop di reazione è possibile costruire il circuito sequenziale direttamente dalle equazioni delle y' e delle z, come mostrano gli esempi seguenti.

(Si noti che, nei circuiti sequenziali sincroni, essendo presenti elementi attivi nella rete di memoria, si possono usare elementi logici di qualsiasi tipo).

Esempio 1: *Costruire il circuito sequenziale sincrono di equazioni:*

$$(VII.1) \qquad \begin{cases} y_1' = x \\ y_2' = y_1 y_2 + \overline{x}\,\overline{y_2} \\ z = y_2 \end{cases}$$

usando, come elementi di memoria, le celle di un registro a scorrimento (par. VI.10.5.3).

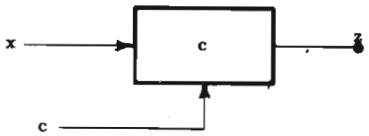


Fig.VII.15 - Cella di un registro a scorrimento.

Una cella di un registro a scorrimento ha un ingresso a livelli (x), un ingresso di clock (c) e un'uscita z (fig.VII.15); ad ogni impulso su c, l'uscita z prende lo stesso valore dell'ingresso x.

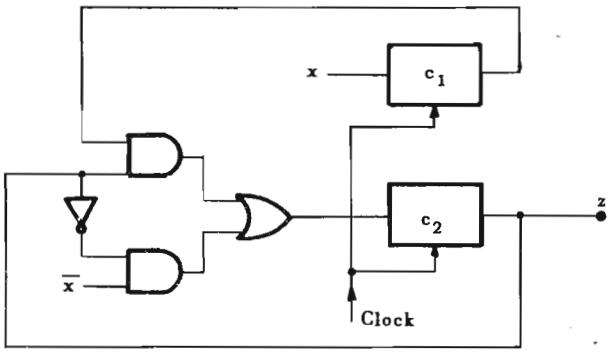


Fig.VII.16 - Circuito sequenziale sincrono con celle di un registro a scorrimento.

Il circuito per le (VII.1) è riportato nella fig.VII.16 (si suppone che l'ingresso x rispetti l'ipotesi del par.VII.1).

Esempio 2: *Costruire il circuito sequenziale sincrono descritto dalle equazioni (VII.1) usando l'elemento di ritardo con un flip-flop SR mostrato nella fig.VII.2.*

Il circuito, in cui l'elemento di ritardo di fig.VII.2 è rappresentato come un blocco di ingressi D e C, è mostrato nella fig.VII.17; la presenza del flip-flop in D permette di eliminare l'invertitore per ottenere la variabile $\overline{y}_1$. Il circuito è, per il resto, identico al precedente (l'ingresso x deve sempre rispettare le condizioni di cui al paragrafo VII.1).

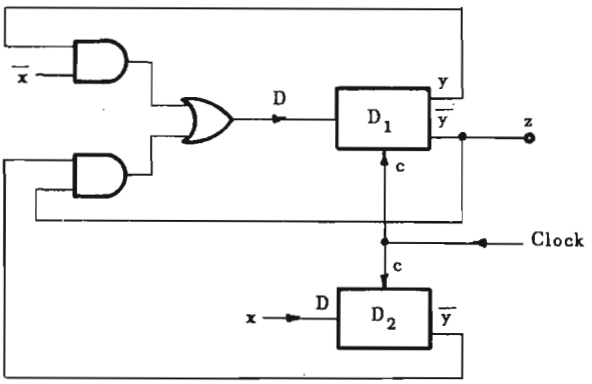


Fig.VII.17 - Circuito sequenziale sincrono con elementi di ritardo.

Se, invece degli elementi di ritardo, si usano dei normali flip-flop, conviene ricavare direttamente le equazioni degli ingressi dei flip-flop stessi in funzione delle x e delle y (*equazioni di eccitazione*).

Questo metodo conduce, generalmente, a circuiti più economici di quelli costruiti a partire dalle y'. Per ricavare le equazioni di eccitazione si costruisce, dalla tavola di flusso, una *matrice di variazione delle y*, e da questa una *matrice di eccitazione*, diversa a seconda del tipo di flip-flop scelto, che porta alle equazioni cercate.

VII.2.3.1 - Matrice di variazione delle y.

La matrice di variazione è una tabella che ha le stesse coordinate della tavola di flusso e che si ottiene scrivendo, in ogni casella e per ogni y, uno dei simboli 0, 1, $\vec{0}$, $\vec{1}$ col significato seguente:

0 : y_i' ed y_i hanno entrambi valore 0: non sono cioè previste variazioni dell'uscita del flip-flop i;

1 : y_i' ed y_i hanno entrambi valore 1: non sono previste variazioni dell'uscita del flip-flop i;

$\vec{0}$: y_i' ha valore 0 e y_i ha valore 1: è prevista una variazione da 1 a 0 dell'uscita del flip-flop i;

$\vec{1}$: y_i' ha valore 1 e y_i ha valore 0: è prevista una variazione da 0 a 1 dell'uscita del flip-flop i.

Esempio 1: *Matrice di variazione delle y per la tavola di flusso della fig.VII.18.*

y_1y_2 \ x	0	1
00	11/1	01/0
01	10/0	00/0
11	10/0	00/1
10	11/0	00/0

$y_1'y_2'z$

Fig.VII.18 - Tavola di flusso per una macchina sequenziale sincrona.

y_1y_2 \ x	0	1
00	$\vec{1}\vec{1}$	$0\vec{1}$
01	$\vec{1}\vec{0}$	$0\vec{0}$
11	$1\vec{0}$	$\vec{0}\vec{0}$
10	$1\vec{1}$	$\vec{0}0$

y_1y_2

Fig.VII.19 - Matrice di variazione per la macchina di figura VII.18.

Poiché, per $xy_1y_2 = 000$, nella tavola di flusso si ha $y_1'y_2' = 11$, la corrispondente casella della matrice di variazione conterrà $\vec{1}\vec{1}$. La casella $xy_1y_2 = 100$, per essere $y_1'y_2' = 01$, conterrà invece $0\vec{1}$. La matrice di variazione completa è riportata nella figura VII.19. In pratica è la stessa tavola di flusso con una freccia sui valori di y_1' diversi dai corrispondenti y_1'.

Vediamo ora come si comporta ogni tipo di flip-flop nelle diverse situazioni dalla matrice di variazione. I flip-flop considerati sono quelli definiti nella tab.1 del par.III.14.3. L'estensione del metodo esposto ai flip-flop realizzati con circuiti integrati non dovrebbe presentare difficoltà.

Di solito, però, il progetto dei circuiti sequenziali con i flip-flop a circuiti integrati si fa con una diversa tecnica (si veda, per questo, lo ultimo capitolo, dedicato ai sistemi numerici).

a) *Flip-flop SR*. Un impulso su S(R) lascia l'uscita a 1(0). Pertanto, se l'uscita deve restare a 0(1) non deve arrivare nessun segnale su S(R); se l'uscita deve andare a 1(0) sugli ingressi SR · debbono esserci i segnali 10(01).

b) *Flip-flop JK*. Poiché i 2 impulsi contemporanei sugli ingressi invertono l'uscita, per mandare l'uscita a 0(1) si può avere su JK l'una o l'altra delle 2 condizioni 10 o 11 (01 o 11): in definitiva la condizione 1-(-1). Se l'uscita resta a 0(1), nessun segnale deve arrivare su J(K).

c) *Flip-flop T*. Cambia di stato ad ogni impulso; quindi se y deve restare costante, T = 0; se deve cambiare, T = 1.

d) *Flip-flop PQ*. Se arrivano due impulsi contemporanei, l'uscita non cambia. Se la y deve passare a 0(1) si deve avere PQ = 01 (PQ = 10); se deve restare a 0(1) si può avere PQ = 01, 00, 11 (PQ = 10, 00, 11), cioè PQ = = 0- oppure -1 (PQ = 1- oppure -0).

e) *Flip-flop STR*. Se y deve restare a 0(1) si deve avere STR = 00- (STR = -00). Se y deve passare a 0(1) si può avere una delle 2 condizioni: STR = 01- o 0-1 (STR = 1-0 o -10).

f) *Flip-flop JTK*. Se y deve rimanere a 0(1) si avrà JTK = 00- (-00); se deve andare a 0(1) si potrà avere JTK = 01- o --1 (JTK = 1-- o -10).

Variazioni uscite

Flip-flop.	0	1	$\vec{0}$	$\vec{1}$
SR	0-	-0	01	10
JK	0-	-0	-1	1-
T	0	0	1	1
PQ	0- -1	-0 1-	01	10
STR	00-	-00	0-1 01-	-10 1-0
JTK	00-	-00	--1 01-	-10 1--

Fig.VII.20 - Ingressi dei vari tipi di flip-flop in funzione delle variazioni delle uscite.

I valori degli ingressi dei vari flip-flop, per tutte le condizioni delle uscite che compaiono nella tabella delle variazioni, sono riassunti nella tabella della fig.VII.20.

VII.2.3.2 - Equazioni di eccitazione dei flip-flop.

Costruita la matrice delle variazioni, per determinare i valori degli ingressi dei flip-flop si costruisce una ulteriore tabella (la *matrice di eccitazione dei flip-flop*), che ha le stesse coordinate della tavola di flusso e contiene – in ogni casella – i valori degli ingressi (SR, JK, T...) ricavati dalla fig.VII.20.

Dalla matrice di eccitazione si ottengono così le equazioni degli ingressi (SR, JK, T); le equazioni delle uscite Z si ricavano invece dalla matrice di flusso.

A titolo d'esempio, realizziamo – in tutti i modi possibili – il circuito la cui tavola di flusso porta alle equazioni (VII.1) dell'esempio 1 del par.VII.2.3.

Esempio 1: *Circuiti a flip-flop per la tavola di flusso della fig.VII.21.*

Nella fig.VII.22 è mostrata la matrice di variazione, ricavata dalla fig.VII.21.

y_1y_2	$x=0$	$x=1$
00	01/0	10/0
01	00/1	10/1
11	01/1	11/1
10	01/0	10/0
	$y'_1y'_2z$	

Fig.VII.21 - Tavola di flusso di una macchina sequenziale sincrona.

y_1y_2	$x=0$	$x=1$
00	$0\vec{1}$	$\vec{1}0$
01	$0\vec{0}$	$\vec{1}\vec{0}$
11	$\vec{0}1$	11
10	$\vec{0}\vec{1}$	10
	$\Delta_1\Delta_2$	

Fig.VII.22 - Matrice di variazione ricavata dalla tavola di flusso della figura VII.21.

Questa matrice va tradotta prima nelle matrici di eccitazione dei flip-flop, secondo le regole esposte nella tabella della fig.VII.20, poi nelle equazioni dei diversi circuiti; in ogni caso, dalla tavola di flusso si ha immediatamente l'equazione di uscita $z = y_2$.

a) Flip-flop SR.

La matrice di eccitazione dei 2 flip-flop SR è quella della fig.VII.23.

y_1y_2	$x=0$		$x=1$	
00	0–	10	10	0–
01	0–	01	10	01
11	01	–0	–0	–0
10	01	10	–0	0–
	S_1R_1	S_2R_2	S_1R_1	S_2R_2

Fig.VII.23 - Matrice di eccitazione dei flip-flop SR.

Scegliendo nel modo più conveniente le condizioni non specificate, si ricavano le equazioni:

$$\begin{cases} S_1 = x \\ R_1 = \overline{x} \\ S_2 = \overline{x y_2} \\ R_2 = \overline{y}_1 y_2 \end{cases} \qquad \text{(VII.2)}$$

Il relativo circuito è disegnato nella fig.VII.24: si noti la presenza del clock, necessaria anche se esso non compare esplicitamente nelle equazioni (VII.2). Confrontando il circuito con quello della fig.VII.17, appare evidente l'utilità di calcolare i valori di S ed R direttamente, piuttosto che attraverso le equazioni delle y'.

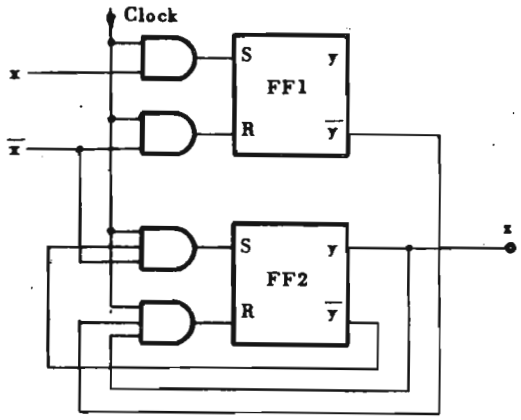


Fig.VII.24 - Circuito con flip-flop SR.

b) Flip-flop JK.

La matrice di eccitazione dei 2 flip-flop JK è quella della fig.VII.25.

y_1y_2	x=0		x=1	
00	0-	1-	1-	0-
01	0-	-1	1-	-1
11	-1	-0	-0	-0
10	-1	1-	-0	0-
	J_1K_1	J_2K_2	J_1K_1	J_2K_2

Fig.VII.25 - Matrice di eccitazione dei flip-flop JK.

Scegliendo nel modo più conveniente le condizioni non specificate, si ricavano le equazioni:

$$\begin{cases} J_1 = x \\ K_1 = \bar{x} \\ J_2 = \bar{x} \\ K_2 = \bar{y}_1 y_2 \end{cases}$$

e da queste, introducendo il clock, il circuito reale della fig.VII.26.

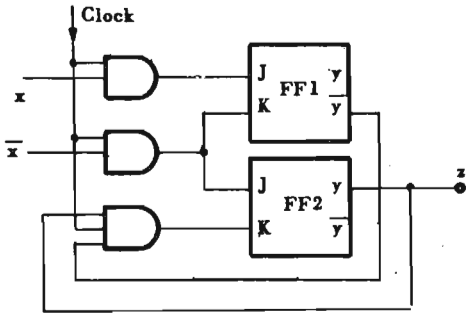


Fig.VII.26 - Circuito con flip-flop JK.

c) Flip-flop T.

La matrice di eccitazione dei 2 flip-flop T (fig.VII.27) conduce alle equazioni:

$$\begin{cases} T_1 = \overline{x}y_1 + x\overline{y_1} \\ T_2 = \overline{x}\overline{y_2} + \overline{y_1}y_2 \end{cases} .$$

Fig.VII.27 - Matrice di eccitazione dei flip-flop T.

y_1y_2	x = 0		x = 1	
00	0	1	1	0
01	0	1	1	1
11	1	0	0	0
10	1	1	0	0
	T_1	T_2	T_1	T_2

Dalle equazioni si costruisce il circuito reale della fig.VII.28.

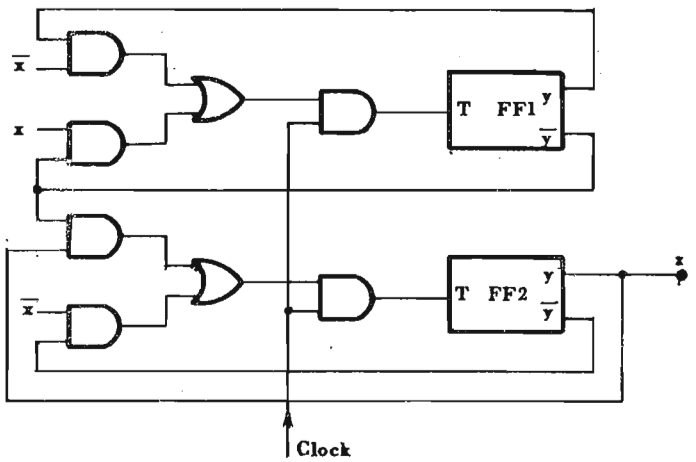


Fig.VII.28 - Circuito con flip-flop T.

d) Flip-flop PQ.

La matrice di eccitazione dei 2 flip-flop PQ è riportata nella fig.VII.29.

y_1y_2	x = 0		x = 1	
00	0 - - 1	1 0	1 0	0 - - 1
01	0 - - 1	0 1	1 0	0 1
11	0 1	- 0 1 -	- 0 1 -	- 0 1 -
10	0 1	1 0	- 0 1 -	0 - - 1
	P_1Q_1	P_2Q_2	P_1Q_1	P_2Q_2

Fig.VII.29 - Matrice di eccitazione dei flip-flop PQ.

La matrice consente una larghissima possibilità di scelte, essendo possibili, per parecchie caselle, 4 scelte diverse. Una buona soluzione è quella che porta alle equazioni:

$$\begin{cases} P_1 = x \ ; & P_2 = \overline{x}\,\overline{y_2} \\ Q_1 = \overline{x} \ ; & Q_2 = \overline{y_1}\,\overline{y_2} \end{cases}$$

quindi, al circuito della fig.VII.30.

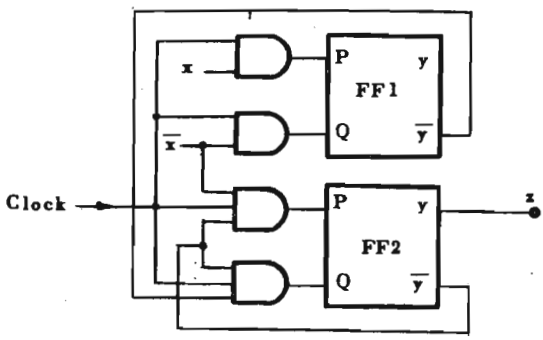


Fig.VII.30 - Circuito con flip-flop PQ.

e) Flip-flop STR.

La matrice di eccitazione dei due flip-flop è quella della fig.VII.31.

y_1y_2	$x=0$		$x=1$	
00	00-	-10 1-0	-10 1-0	00-
01	00-	0-1 01-	-10 1-0	0-1 01-
11	0-1 01-	-00	-00	-00
10	0-1 01-	-10 1-0	-00	00-
	$S_1T_1R_1$	$S_2T_2R_2$	$S_1T_1R_1$	$S_2T_2R_2$

Fig.VII.31 - Matrice di eccitazione dei flip-flop STR.

Sfruttando le condizioni non specificate, si hanno le equazioni:

$$\begin{cases} S_1 = x \\ T_1 = \overline{x}\,y_1 \\ R_1 = 0 \\ S_2 = \overline{x}\,y_1 \\ T_2 = \overline{x}\,\overline{y_1} \\ R_2 = \overline{y_1}\,y_2 \end{cases}$$

che conducono al circuito della fig.VII.32.

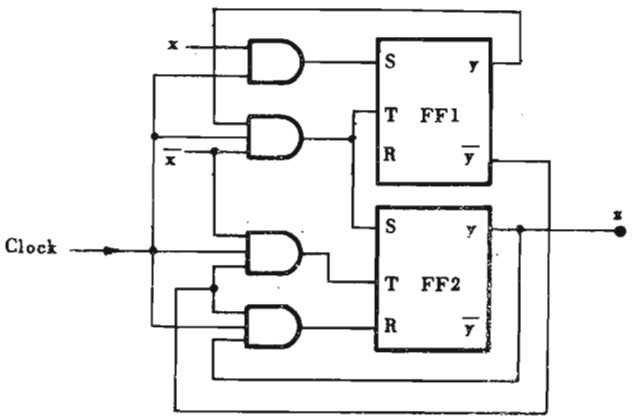


Fig.VII.32 - Circuito con flip-flop STR.

f) Flip-flop JTK.

La matrice di eccitazione dei 2 flip-flop è riportata nella fig.VII.33.

y_1y_2	$x=0$		$x=1$	
00	00-	-01 1--	-10 1--	00-
01	00-	--1 01-	-10 1--	--1 01-
11	--1 01-	-00	-00	-00
10	--1 01-	-10 1--	-00	00-
	$J_1T_1K_1$	$J_2T_2K_2$	$J_1T_1K_1$	$J_2T_2K_2$

Fig.VII.33 - Matrice di eccitazione dei flip-flop JTK.

La scelta più conveniente delle condizioni non specificate porta alle equazioni:

$$\begin{cases} J_1 = x \\ T_1 = 0 \\ K_1 = \overline{x} \\ J_2 = 0 \\ T_2 = \overline{x}\, y_2 \\ K_2 = \overline{y}_1 y_2 \end{cases},$$

e al circuito della fig.VII.34.

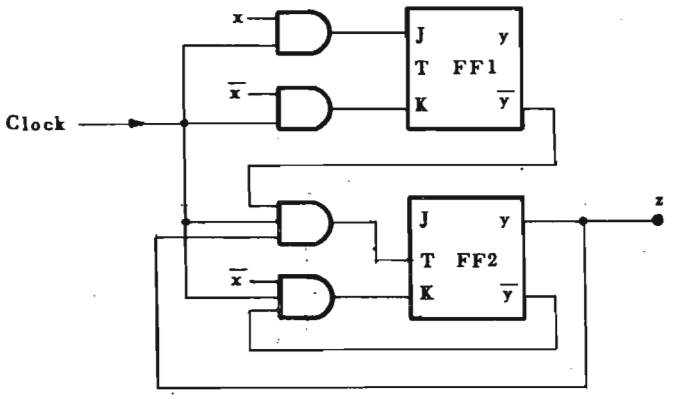


Fig.VII.34 - Circuito con flip-flop JTK.

Esaurita l'esposizione del metodo di sintesi, riteniamo utile – per riassumere quanto detto – eseguire la sintesi completa di 2 circuiti sequenziali.

Esempio 2: *Sintesi di un circuito sequenziale sincrono a 2 ingressi (x_1x_2) il cui stato iniziale è 00, e un'uscita (z). L'uscita va a 1 al termine della sequenza:*

$$x_1x_2 = 01 \rightarrow 11$$

e vi rimane finché il sistema non torna nello stato iniziale. Usare flip-flop T.

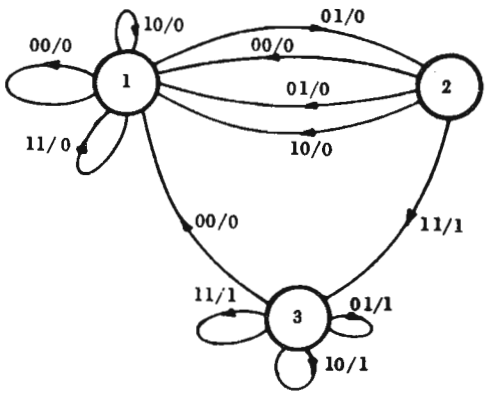


Fig.VII.35 - Diagramma degli stati di una macchina sequenziale sincrona.

Stati	x_1x_2 00	01	11	10
1	1/0	2/0	1/0	1/0
2	1/0	1/0	3/1	1/0
3	1/0	3/1	3/1	3/1

a)

Stati	x_1x_2 00	01	11	10	Numero
1	1·2·3	2	1	1·2	7
2	–	1	–	–	1
3	–	3	2·3	3	4

b)

y_1y_2	x_1x_2 00	01	11	10
00	00/0	10/0	00/0	00/0
01	00/0	01/1	01/1	01/1
11	–	–	–	–
10	00/0	00/0	01/1	00/0

c)

Fig.VII.36 - Tavola di flusso per la macchina di fig.VII.35.

Il diagramma degli stati è mostrato nella fig.VII.35.

Non è possibile (fig.VII.36a) nessuna riduzione del numero degli stati; dalla tabella inversa della fig.VII.36b, imponendo allo stato 1 il valore $y_1y_2 = 00$, nonché l'adiacenza degli stati 2 e 3, si ottiene la tavola di flusso della fig.VII.36c.

y_1y_2 \ x_1x_2	00	01	11	10
00	00	10	00	00
01	00	01	01	01
11	–	–	–	–
10	00	00	01	00

$\Delta_1\Delta_2$

a)

y_1y_2 \ x_1x_2	00	01	11	10
00	00	10	00	00
01	01	00	00	00
11	–	–	–	–
10	10	10	11	10

T_1T_2

b)

Fig.VII.37 - Matrici di variazione e di eccitazione dei flip-flop T.

L'equazione dell'uscita z è:

$$z = x_1x_2y_1 + x_1y_2 + x_2y_2 \quad .$$

La matrice di variazione della fig.VII.37a permette poi di ricavare la matrice di eccitazione dei 2 flip-flop T necessari (fig.VII.37b). Si hanno, in definitiva, le equazioni:

$$\begin{cases} T_1 = y_1\overline{y_2} + \overline{x}_1x_2\overline{y_2} \\ T_2 = \overline{x}_1\overline{x}_2y_2 + x_1x_2y_1 \end{cases} ,$$

e il circuito della fig.VII.38.

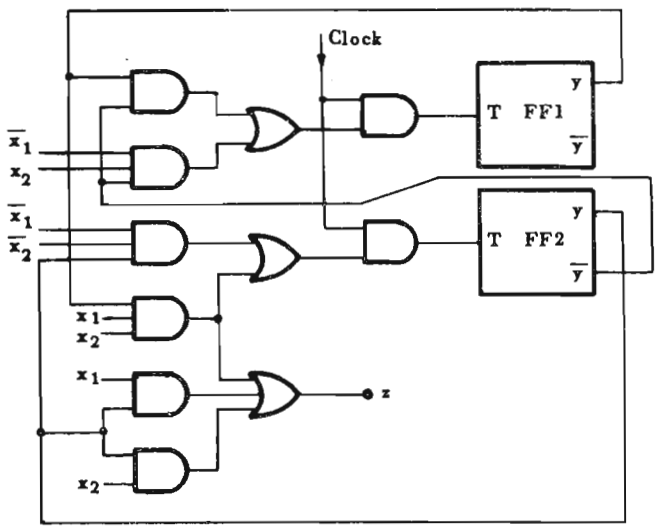
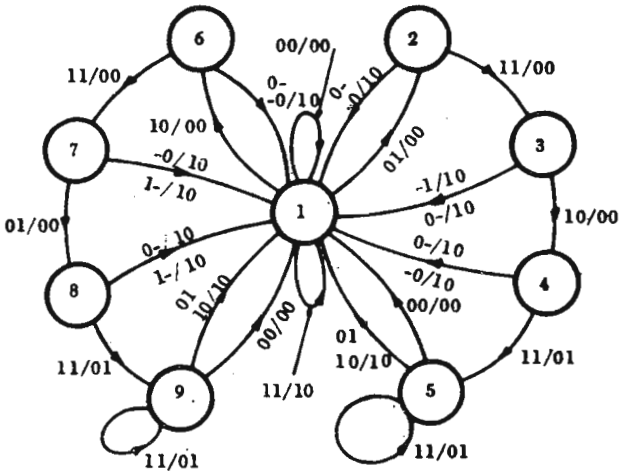


Fig.VII.38 - Circuito con flip-flop T.

Esempio 3: *Sintesi di un circuito sequenziale a 2 ingressi* (x_1x_2) *e 2 uscite* (z_1z_2) *per rivelare una delle 2 sequenze:*

$$\begin{cases} [S_1] \equiv x_1x_2 = 01 \rightarrow 11 \rightarrow 10 \rightarrow 11 \\ [S_2] \equiv x_1x_2 = 10 \rightarrow 11 \rightarrow 01 \rightarrow 11 \end{cases} .$$

In uscita si avrà $z_1z_2=00$ *per* $x_1x_2=00$ *e per tutti i valori di* $[S_1]$ *e* $[S_2]$ *che precedono l'ultimo;* $z_1z_2=01$ *quando* $[S_1]$ *e* $[S_2]$ *sono state completate;* $z_1z_2=10$ *per ogni altra sequenza diversa da* $[S_1]$ *ed* $[S_2]$. *Usare come memorie degli elementi di ritardo.*



Stati	x_1x_2 00	01	11	10
1	1/00	2/00	1/10	6/00
2	1/10	1/10	3/00	1/10
3	1/10	1/10	1/10	4/00
4	1/10	1/10	5/01	1/10
5	1/00	1/10	5/01	1/10
6	1/10	1/10	7/00	1/10
7	1/10	8/00	1/10	1/10
8	1/10	1/10	9/01	1/10
9	1/00	1/10	9/01	1/10

$y' z_1z_2$

Fig.VII.39 - Tabella degli stati e relativo diagramma per una macchina sequenziale sincrona.

A partire dallo stato 1 di riposo occorrono altri 8 stati per ricordare tutti i valori delle 2 sequenze da rivelare. Il diagramma e la tabella degli stati della macchina $\mathcal{M}$ che realizza il comportamento voluto sono riportati nella fig.VII.39.

Stato	x_1x_2 00	01	11	10
1	1/00	2/00	6/10	1/00
2	1/10	1/10	1/00	3/10
3	1/10	1/10	4/10	1/00
4	1/10	1/10	1/01	5/10
5	1/00	1/10	1/01	5/10
6	1/10	1/10	1/00	7/10
7	1/10	4/00	1/10	1/10

Fig.VII.40 - Tabella degli stati ridotta per la macchina di fig.VII.39.

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

a)

$y_1y_2y_3$	x_1x_2 00	01	11	10
000	010/10	010/10	010/01	001/10
001	010/00	010/10	010/01	001/10
011	010/10	000/00	010/10	010/10
010	010/00	100/00	101/10	010/00
110	-	-	-	-
111	010/10	010/10	000/10	010/00
101	010/10	010/10	010/00	011/10
100	010/10	010/10	010/00	111/10

$y_1'y_2'y_3'z_1z_2$

b)

Fig.VII.41 - Assegnazione degli stati secondari alla macchina della fig.VII.39.

La macchina $\mathcal{M}$ è completamente determinata; applicando il metodo di minimizzazione del par. V.6.1 si trova la seguente partizione degli stati:

$$\{1\}\ \{2\}\ \{3\}\ \{4\cdot8\}\ \{5\cdot9\}\ \{6\}\ \{7\}\ .$$

La tabella degli stati ridotta è mostrata nella fig.VII.40.

Occorrono tre variabili secondari per distinguere i 7 stati. Per la codificazione, non è molto utile il metodo della tabella inversa, avendo la rete d'uscita pressappoco la stessa importanza di quella di memoria.

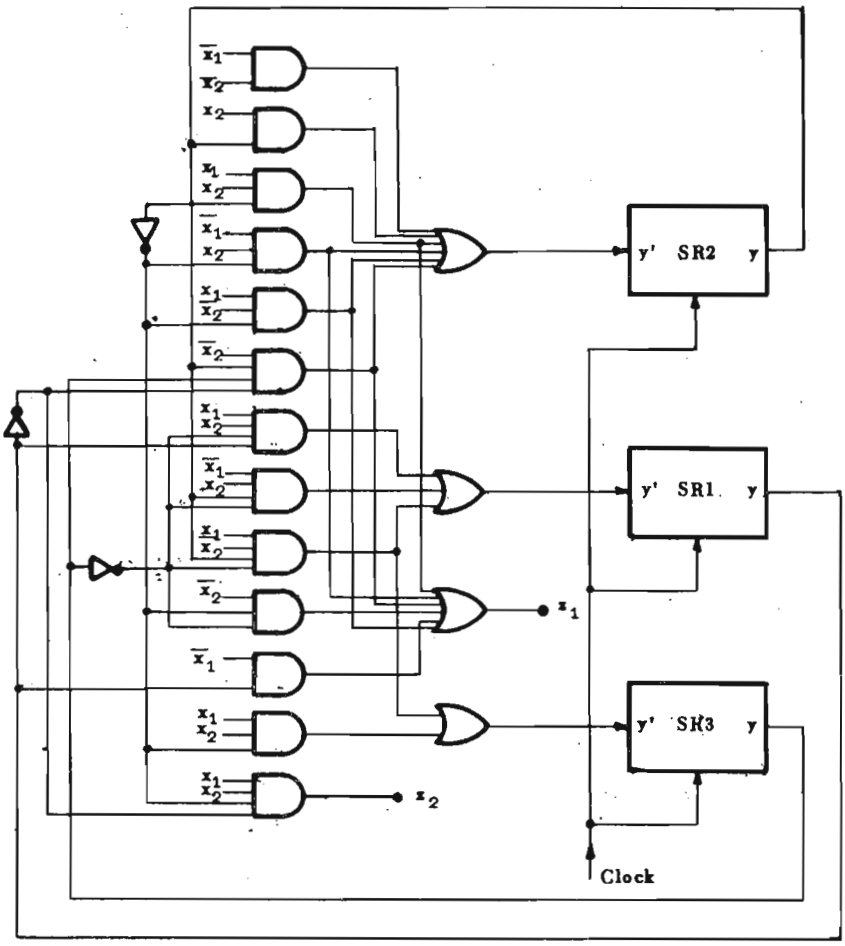


Fig.VII.42 - Circuito per la macchina di fig.VII.39.

Procedendo per tentativi, si trova che la codificazione migliore è quella della fig.VII.41a, da cui deriva la tavola di flusso della fig.VII.41b e le equazioni:

$$y_1' = y_1\overline{y}_3x_1x_2 + y_2\overline{y}_3\overline{x}_1x_2 + y_2\overline{y}_3x_1\overline{x}_2$$
$$y_2' = \overline{x}_1\overline{x}_2 + y_1x_2 + \overline{y}_1y_2y_3\overline{x}_2 + \overline{y}_2x_1\overline{x}_2 + \overline{y}_2\overline{x}_1x_2 + y_2x_1x_2$$
$$y_3' = \overline{y}_2x_1x_2 + y_2\overline{y}_3x_1\overline{x}_2$$
$$z_1 = \overline{y}_2\overline{y}_3\overline{x}_2 + y_1\overline{x}_1 + \overline{y}_2x_1\overline{x}_2 + \overline{y}_2\overline{x}_1x_2 + y_2x_1x_2 + \overline{y}_1y_2y_3\overline{x}_2$$
$$z_2 = \overline{y}_1\overline{y}_2x_1x_2$$

Impiegando tre celle di un registro a scorrimento per i tre elementi di ritardo, si ha finalmente il circuito della fig.VII.42.

VII.3 - Analisi dei circuiti sequenziali sincroni.

L'analisi dei circuiti sequenziali sincroni si effettua con un procedimento che non ha più bisogno di essere illustrato, essendo esattamente il contrario del procedimento di sintesi, come mostra il seguente esempio.

Esempio 1: *Analisi del circuito sequenziale sincrono ad un ingresso e un'uscita della fig.VII.43.*

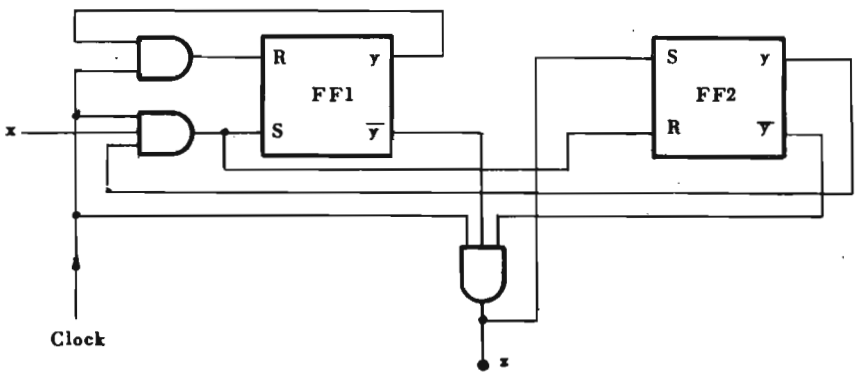


Fig.VII.43 - Circuito sequenziale sincrono.

Eliminando dagli ingressi il segnale di clock, non significativo dal punto di vista logico, il circuito può essere messo nella forma più semplice della fig.VII.44, a cui corrispondono le equazioni di eccitazione e d'uscita:

$$\begin{cases} S_1 = xy_2 \\ R_1 = y_1 \\ S_2 = \overline{y}_1\overline{y}_2 \\ R_2 = xy_2 \\ z = S_2 \end{cases}.$$

Da queste equazioni, attraverso le mappe di Karnaugh per le funzioni S_1R_1 e S_2R_2 (fig.VII.45) si ottengono le funzioni di eccitazione y' del circuito ideale il cui funzionamento corrisponde a quello del circuito reale. Formate tante tabelle SR quanti

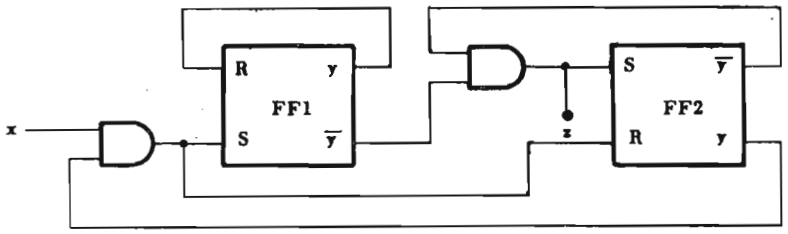


Fig.VII.44 - Forma semplificata per il circuito sequenziale sincrono della fig.VII.43.

sono i flip-flop del circuito (figg. VII.46a e VII.46b) è possibile trovare, con semplici ragionamenti, i valori che in corrispondenza ad ogni configurazione di $(xy_1y_2S_1R_1)$ assu-

y_1y_2 \ x	0	1
00		
01		1
11		1
10		

S_1

y_1y_2 \ x	0	1
00		
01		
11	1	1
10	1	1

R_1

y_1y_2 \ x	0	1
00	1	1
01		
11		
10		

S_2

y_1y_2 \ x	0	1
00		
01		1
11		1
10		

R_2

Fig.VII.45 - Mappe di Karnaugh per il circuito di fig.VII.44.

a)

y_1y_2 \ x	0	1
00	00	00
01	00	10
11	01	11
10	01	01

S_1R_1

b)

y_1y_2 \ x	0	1
00	10	10
01	00	01
11	00	01
10	00	00

S_2R_2

Fig.VII.46 - Matrici di eccitazione dei flip-flop del circuito di fig.VII.44.

mono le funzioni y_1'. Ad esempio, se $S_1R_1 = 00$, non ci sono segnali sugli ingressi del flip-flop 1, la cui uscita rimane perciò invariata; così, y_1' $(x = y_1 = y_2 = S_1 = R_1 = 0) = 0$.

Se $S_2R_2 = 10$, sul flip-flop 2 compare un impulso S, che lo manda nello stato 1: così y_2' $(xy_1y_2S_2R_2 = 00010) = 1$. Nella casella $xy_1y_2 = 111$ si ha $S_1R_1 = 11$; questa situazione

a)

y_1y_2 \ x	0	1
00	0	0
01	0	1
11	0	-
10	0	0

y_1'

b)

y_1y_2 \ x	0	1
00	1	1
01	1	0
11	1	-
10	0	0

y_2'

Fig.VII.47 - Funzioni y_1' e y_2' derivate dalle matrici di fig.VII.46.

y_1y_2 \ x	0	1
00	01	01
01	01	10
11	01	-
10	00	00

$y_1'y_2'$

y_1y_2 \ x	0	1	
00	2/1	2/1	(1)
01	2/0	4/0	(2)
11	2/0	-	(3)
10	1/0	1/0	(4)

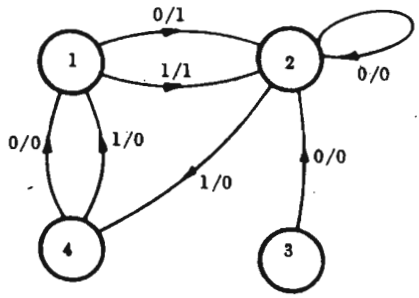


Fig.VII.48 - Diagramma degli stati del circuito di fig.VII.44.

non può mai verificarsi, pertanto la y_1' corrispondente deve provenire da una condizione non specificata. Con questi criteri sono state realizzate le mappe delle figg. VII.47a e VII.47b, derivate – rispettivamente – dalle figg. VII.46a e VII.46b.

Riunendo in un'unica tabella y_1' e y_2', ed aggiungendo l'uscita $z = S_2 = \overline{y_1}\,\overline{y_2}$, si ottengono la tavola di flusso e il diagramma degli stati del circuito (fig.VII.48).

Con gli algoritmi esposti in questo e nel precedente capitolo, è possibile risolvere ogni problema d'analisi e di sintesi relativo ai circuiti sequenziali nei quali siano verificate le ipotesi fondamentali del paragrafo VII.1.

Non riteniamo utile appesantire l'argomento discutendo gli effetti del mancato verificarsi di tali ipotesi, per ragioni che appariranno chiare nel capitolo successivo. Ci sembra opportuno, invece, esporre a parte i metodi di sintesi di quei circuiti, di uso assai comune, i cui ingressi sono costituiti esclusivamente da impulsi non contemporanei, metodi ba-

sati sull'osservazione che esistono soltanto n + 1, non 2^n valori possibili di n ingressi.

I metodi in questione sono applicabili anche ai circuiti con ingressi a livelli sempre tutti, o tutti meno uno, al valore 0. In questi casi, infatti, si possono formare degli impulsi derivando i livelli d'ingresso (v. par.V.7).

VII.4 - Circuiti a impulsi non contemporanei.

Quando all'ingresso di un circuito possono presentarsi soltanto impulsi di tensione non contemporanei e separati da un tempo minimo superiore al tempo di risposta del circuito, può essere applicato il procedimento semplificato riportato di seguito.

a) Si costruisce un diagramma considerando il circuito in tanti stati stabili diversi, tutti caratterizzati dall'assenza di segnali esterni; l'evoluzione del circuito è provocata dagli impulsi $x_1 x_2 \ldots x_n$, che si presentano uno alla volta a partire dai vari stati e secondo le sequenze previste.

b) La minimizzazione degli stati si effettua col solito metodo; poiché, poi, il circuito si realizza senza clock, con organi di memoria comandati direttamente dagli impulsi di ingresso, tutti i cambiamenti delle y avvengono nello stesso istante, non ci sono pericoli di corse critiche, e gli stati si possono codificare in un modo qualsiasi.

c) Si trasformano le equazioni delle y' nelle equazioni di eccitazione dei flip-flop scelti, e si costruisce il circuito collegando le uscite di questi ultimi con gli ingressi y della rete di memoria.

I circuiti cui si perviene con questo metodo, sia dal punto di vista teorico (per la forma del diagramma degli stati, quando gli impulsi si interpretano come 2 variazioni di livelli) sia dal punto di vista circuitale (per la mancanza di clock esterno) sono di tipo asincrono. Si è preferito tuttavia trattarli in questo capitolo sia perché usano dei flip-flop nei loop di reazione, quindi adoperano i metodi di sintesi dei circuiti sincroni, sia perché possono essere considerati dei particolarissimi circuiti sincroni in cui il clock, coincidendo con gli impulsi di ingresso, è a frequenza variabile.

VII.4.1 - Diagramma degli stati per i circuiti a impulsi non contemporanei.

I circuiti che stiamo considerando, impulsivi nei riguardi degli ingressi, possono avere contemporaneamente uscite a impulsi e a livelli:

per questa ragione, il loro diagramma degli stati è intermedio fra quello di Moore e di Mealy.

Nella costruzione del diagramma si considerano soltanto gli stati stabili del circuito compresi tra 2 impulsi, considerando ogni impulso come la causa del passaggio dallo stato j allo stato q, non necessariamente distinto da j.

Gli stati si rappresentano, al solito, con un cerchio numerato; le transizioni con un segmento orientato da j a q, su cui si scrive x_i, cioè il nome dell'impulso che manda il circuito da j a q. Le uscite impulsive avvengono in coincidenza con x_i, e si scrivono accanto ad esse. Le uscite a livelli permangono per tutta la durata dello stato stabile, cioè di riposo, e si indicano nell'interno del relativo cerchio.

Esempio 1: *Diagramma degli stati di un circuito sequenziale asincrono a due ingressi* (x_1x_2) *ed un'uscita* (z). *z va ad 1 solo quando si ha per la seconda volta il valore 1 di* x_2 *dopo almeno un valore 1 e un valore 0 di* x_1. *Gli ingressi* x_1 *e* x_2 *non possono essere 1 contemporaneamente.*

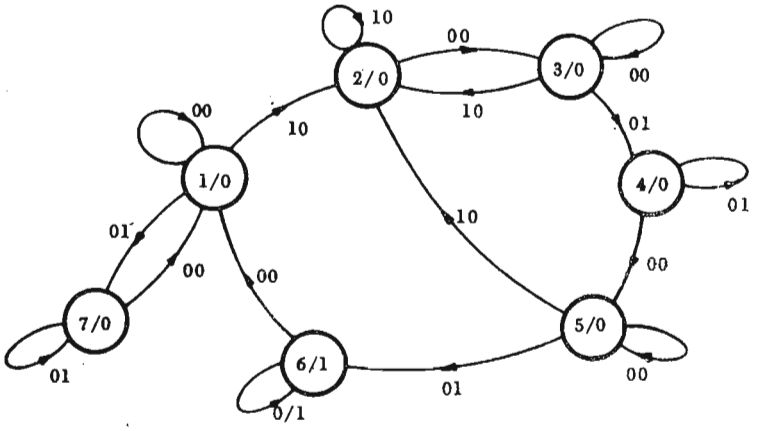


Fig.VII.49 - Diagramma di Moore di un circuito asincrono.

Il diagramma degli stati del circuito secondo Moore è riportato nella fig.VII.49.

Trasformando i livelli in impulsi, e associando i valori dell'uscita alle transizioni piuttosto che alle fasi, in modo da avere impulsi di uscita coincidenti con quelli d'ingresso, si ha il diagramma degli stati fortemente semplificato della fig.VII.50. In esso, i 3 stati sono tutti di riposo ($y_1y_2 = 00$), quindi stabili: la diversa forma del diagramma è dovuta alle nuove convenzioni introdotte.

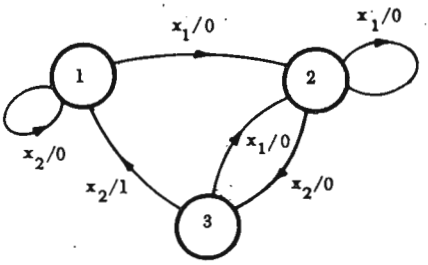


Fig.VII.50 - Diagramma equivalente a quello della fig.VII.49, con le transizioni associate agli impulsi.

Esempio 2: *Diagramma degli stati di un circuito che conta gli impulsi sul suo unico ingresso x, da 1 fino a 4. Le uscite avvengono su 4 terminali* ($z_0 z_1 z_2 z_3$) *che assumono il livello 1 rispettivamente dopo 0, 1, 2, 3 impulsi su x. Al quarto impulso, si ha di nuovo* $z_0 = 1$ *e, contemporaneamente, un impulso su una quinta uscita R.*

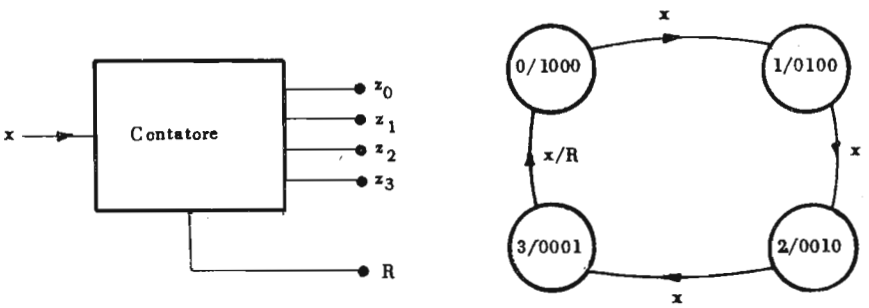


Fig.VII.51 - Schema a blocchi e diagramma di stato di un circuito contatore di impulsi.

Gli stati stabili debbono essere 4, per ricordare se all'ingresso non è arrivato nessun impulso, o ne sono arrivati 1, 2, 3. Le transizioni avvengono sempre in un unico senso, secondo il diagramma della fig.VII.51, dove è mostrato anche lo schema a blocchi del contatore.

Esempio 3: *Un circuito sequenziale ha 3 ingressi impulsivi* ($x_1 x_2 x_3$) *mai contemporanei e un'uscita (z) su cui compare un impulso quando all'ingresso si sono succeduti, ordinatamente, tre impulsi della sequenza* $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_1 \rightarrow x_2 \ldots$. *Ogni uscita è separata dalla successiva da almeno 3 impulsi d'ingresso.*

Occorre prevedere 3 stati per ricordare l'arrivo del primo impulso delle 3 sequenze diverse $x_1 \rightarrow x_2 \rightarrow x_3$, $x_2 \rightarrow x_3 \rightarrow x_1$ e $x_3 \rightarrow x_1 \rightarrow x_2$. Altri 3 stati ricorderanno l'arrivo dei primi due impulsi delle stesse sequenze. Un'ultimo stato, infine, terrà conto delle sequenze non previste. Il diagramma è mostrato nella fig.VII.52.

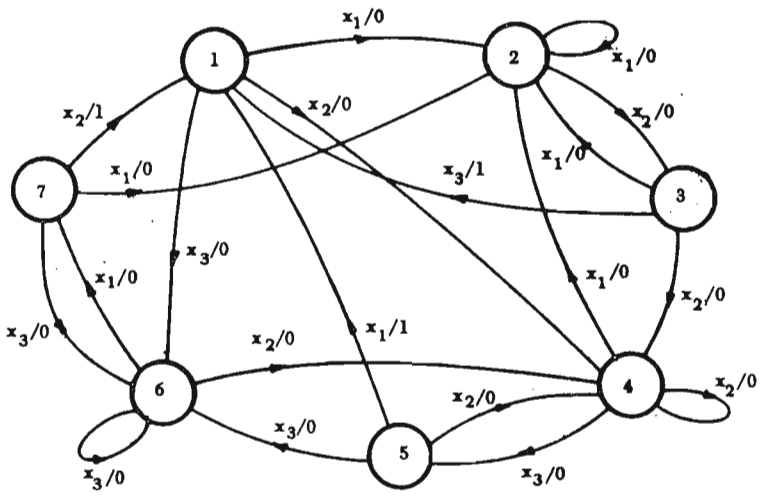


Fig.VII.52 - Diagramma degli stati di un circuito a impulsi non contemporanei.

VII.4.2 - Codificazione degli stati, equazioni di eccitazione dei flip-flop e costruzione del circuito.

La tabella degli stati si ottiene con i soliti criteri di minimizzazione. Quando esistono uscite a livelli e a impulsi, conviene scrivere le prime in corrispondenza degli stati stabili, cioè dei valori delle y che competono alle varie righe, le seconde in corrispondenza delle transizioni tra gli stati, cioè dei valori delle y'.
Ad esempio, nella fig.VII.53 è mostrata la tabella degli stati per il diagramma della fig.VII.51. Questa notazione va mantenuta anche per la minimizzazione e la costruzione della tavola di flusso.

Stati	x	$z_1z_2z_3z_4$
0	1	1000
1	2	0100
2	3	0010
3	1, R	0001

Fig.VII.53 - Tabella degli stati per il diagramma della fig.VII.51.

Con le ipotesi fatte, la tabella degli stati, la tavola di flusso e la matrice di eccitazione hanno tante righe quanti sono gli stati e tante colonne quanti sono gli ingressi. Gli stati stabili, in cui gli ingressi sono tutti a 0, sono le coordinate delle varie tabelle. Nella codificazione degli stati, e in tutto il procedimento di sintesi, per i motivi che preciseremo nei seguenti esempi, occorre considerare ogni colonna della tavola di flusso come a sé stante. Non si possono, pertanto, applicare le regole delle adiacenze per semplificare il circuito.

Nei riguardi della scelta del tipo di flip-flop e della scrittura delle equazioni di eccitazione, vale quanto detto nei parr. VII.2 e VII.3.

Il disegno del circuito avviene secondo le equazioni di eccitazione e comprende i collegamenti di reazione tra le uscite dei flip-flop e gli ingressi y della rete logica, collegamenti che vengono effettuati direttamente, senza alcun segnale di clock. Le uscite impulsive devono sempre contenere come fattore l'impulso d'ingresso della colonna in cui compaiono; le uscite a livelli saranno formate con reti combinatorie alimentate dai segnali d'uscita dei flip-flop. (I flip-flop usati sono ancora quelli definiti nella tab.1 del par.III.14.3).

Esempio 1: *Costruzione del circuito di equazioni:*

$$\begin{cases} S_1 = x_1 y_2 \\ R_1 = x_2 \\ S_2 = x_1 \overline{y_1} \\ R_2 = x_1 y_1 + x_2 \\ z_0 = x_2 y_1 y_2 \\ z_1 = \overline{y_1} y_2 \end{cases}$$

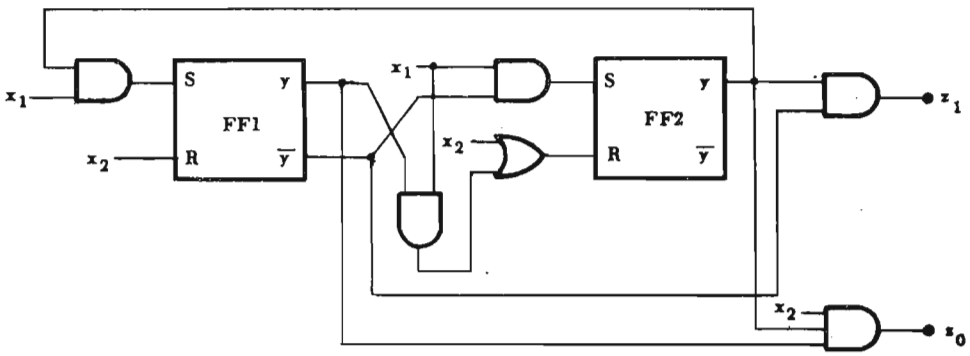


Fig.VII.54 - Circuito a flip-flop SR per impulsi non contemporanei.

L'uscita z_0, come mostrano le equazioni, è impulsiva; la z_1 a livelli. Il circuito è disegnato nella fig.VII.54, ed ha l'aspetto di un tipico circuito asincrono con flip-flop SR al posto degli elementi di ritardo.

Esposti i principi generali del metodo, vediamo ora due esempi di sintesi per i circuiti a impulsi non contemporanei.

VII.4.3 - Esempi di sintesi.

Esempio 1: *Sintesi di un circuito impulsivo a 3 ingressi ($x_1x_2x_3$) e un'uscita (z) su cui si ha un impulso in coincidenza col primo impulso x_3 che completa la sequenza $x_1 \rightarrow x_2 \rightarrow x_3$. Usare flip-flop T.*

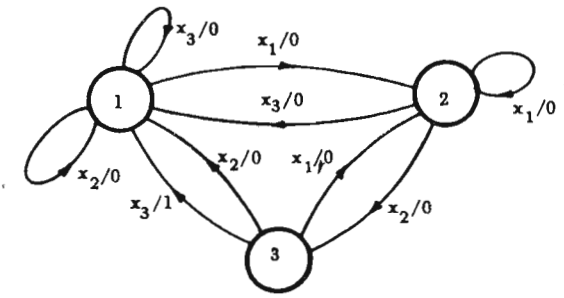


Fig.VII.55 - Diagramma degli stati di un circuito impulsivo.

Il primo passo della sintesi è la costruzione del diagramma degli stati (figura VII.55): a partire dallo stato iniziale 1, un impulso x_1 porta il circuito nello stato 2, un successivo impulso x_2 lo porta nello stato 3, infine un impulso x_3 lo riporta nello stato iniziale e provoca un impulso in uscita.

Non è possibile ridurre gli stati, per cui si dovranno usare 2 flip-flop.

	x_1	x_2	x_3
1	2	1	1
2	2	3	1
3	2	1	1,z

a)

y_2 \ y_1	0	1
0	2	1
1	3	-

b)

y_1y_2	x_1	x_2	x_3
00	00	01	10
01	00	10	10,z
11	-	-	-
10	00	10	10

c) $y_1'y_2'z$

Fig.VII.56 - Tabella (a) e codificazione (b) degli stati: tavola di flusso (c) del circuito di fig.VII.55.

Dal diagramma si ricava immediatamente la tabella degli stati (fig.VII.56a). Poiché le funzioni T d'ingresso ai flip-flop debbono essere tutte moltiplicate per uno degli impulsi, e questi non sono mai contemporanei, le colonne della tavola di flusso vanno considerate come tante mappe indipendenti.

Non esistendo perciò adiacenze tra le colonne x_i e x_{i+1}, non valgono le regole date precedentemente per la codificazione degli stati secondari, che va fatta per tentativi.

La codificazione che porta alla rete più economica è quella della fig.VII.56b, da cui deriva la tavola di flusso della fig.VII.56c e le matrici di variazione e di eccitazione dei 2 flip-flop della fig.VII.57.

y_1y_2	x_1	x_2	x_3
00	00	01	10
01	00	10	10
11	–	–	–
10	00	10	10

$\Delta_1\Delta_2$

a)

y_1y_2	x_1	x_2	x_3
00	0	0	1
01	0	1	1
11	–	–	–
10	1	0	0

T_1

y_1y_2	x_1	x_2	x_3
00	0	1	0
01	1	1	1
11	–	–	–
10	0	0	0

T_2

b)

Fig.VII.57 - Matrici di variazione (a) e di eccitazione (b) del circuito di fig.VII.55.

Scegliendo opportunamente i valori delle caselle non specificate, si ottengono le equazioni:

$$\text{(VII.3)} \qquad \begin{cases} T_1 = x_1y_1 + x_2y_2 + x_3\overline{y_1} \\ T_2 = x_1y_2 + x_2\overline{y_1} + x_3y \\ z = x_3y_2 \ . \end{cases}$$

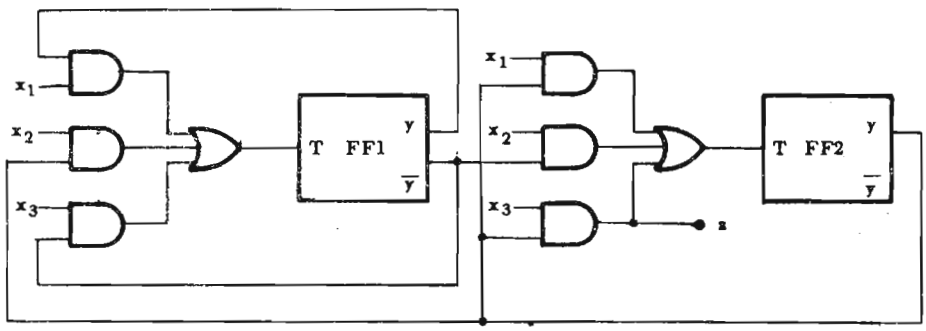


Fig.VII.58 - Circuito impulsivo derivato dalle matrici di eccitazione di fig.VII.57.

Come già detto, le (VII.3) sono state ricavate ognuna da una colonna delle matrici. Questa particolarità, pur non consentendo semplificazioni che nell'ambito di una stessa colonna, permette di trattare in modo molto semplice problemi ad elevato numero di variabili.

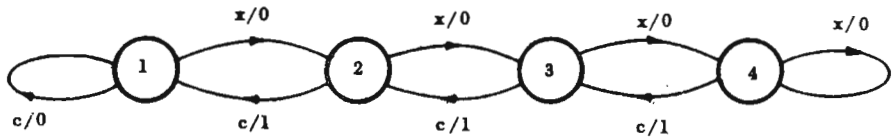
Nella fig.VII.58 è mostrato il circuito derivato dalle (VII.3).

È interessante vedere se la velocità dei componenti può causare un cattivo funzionamento del circuito progettato. Nella condizione $y_1y_2 = 01$, l'impulso x_3, per esse-

re $T_1 = T_2 = z = 1$, deve generare un impulso d'uscita e, contemporaneamente, variare lo stato dei 2 flip-flop; se y_1 va ad 1 prima che y_2 sia andato a 0, per un certo tempo si ha lo stato non previsto $y_1 y_2 = 11$. Questo transitorio non ha però conseguenze dannose, perché – per ipotesi – il successivo impulso di comando giungerà solo dopo che il circuito sarà passato nello stato stabile finale.

La relazione tra la durata degli impulsi di comando e la risposta dei flip-flop è importante perché se, e solo se, il cambiamento di stato avviene dopo che l'impulso è terminato, si possono codificare in un modo qualsiasi gli stati (si può ammettere, in realtà, che il flip-flop j^{mo} cambi stato prima della fine dell'impulso, purché nell'espressione delle sue funzioni di eccitazione non compaia l'uscita y_j). Con i circuiti a impulsi non contemporanei possono risolversi anche problemi di natura essenzialmente sincrona, come quello dell'esempio seguente.

Esempio 2: *Sintesi di un circuito ad un ingresso impulsivo x, un ingresso di clock c e un'uscita z. Il circuito ricorda gli impulsi x (che possono presentarsi in istanti qualsiasi, ma mai contemporaneamente agli impulsi c), fino a un massimo di 3. Gli impulsi x tra due segnali di clock possono essere 0, 1, 2. In coincidenza con un impulso di clock, se nel circuito sono registrati impulsi x, si ha un impulso su z e la contemporanea cancellazione di un impulso x. Usare flip-flop SR.*



	c	x
1	1	2
2	1,z	3
3	2,z	4
4	3,z	4

Fig.VII.59 - Diagramma e tabella degli stati di un circuito impulsivo.

Il diagramma e la tabella degli stati del circuito sono riportati nella fig.VII.59. Gli stati sono 4 e registrano, rispettivamente, l'arrivo di 0, 1, 2, 3 o più impulsi x tra 2 impulsi c. L'assegnazione più economica degli stati, ottenuta per tentativi, è quella della fig.VII.60a. Nelle figg.VII.60b e VII.60c sono riportate la tabella di flusso e le matrici di eccitazione che ne derivano. Le equazioni del circuito (fig.VII.61), con una opportuna scelta delle condizioni non specificate, sono:

$$\begin{cases} S_1 = xy_2 \\ R_1 = cy_2 \\ S_2 = x\overline{y_1} + cy_1 \\ R_2 = xy_1 + c\overline{y_1} \\ z = cy_1 + cy_2 \end{cases}$$

Ogni altra soluzione del problema, diversa per l'assegnazione degli stati secondari, risulta meno economica. Sembrerebbe, pertanto, che il circuito trovato, con 2 flip-flop e 18 diodi, fosse il più conveniente.

a)

y_2 \ y_1	0	1
0	1	4
1	2	3

b)

y_1y_2	c	x
00	00	01
01	00,z	11
11	01,z	10
10	11,z	10

$y_1'y_2'z$

c)

y_1y_2	c	x	c	x
00	0-	0-	0-	10
01	0-	10	01	-0
11	01	-0	-0	01
10	-0	-0	10	01

S_1R_1 S_2R_2

Fig.VII.60 - Assegnazione degli stati secondari (a), tabella di flusso (b) e matrice di eccitazione dei flip-flop SR (c) per il circuito di fig.VII.59.

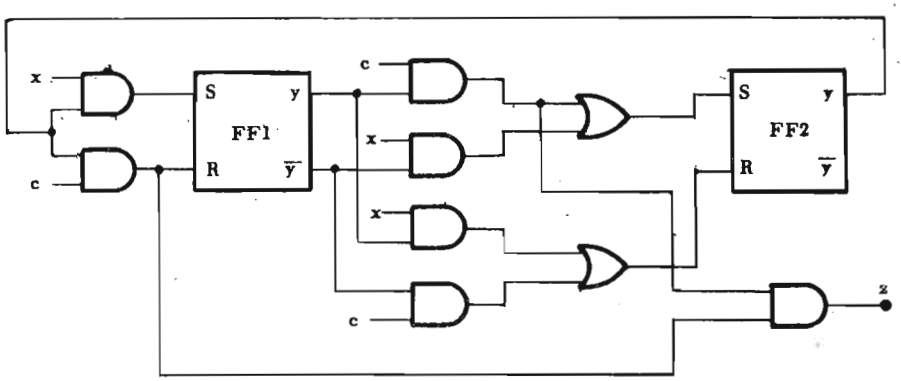


Fig.VII.61 - Circuito a flip-flop SR derivato dalle matrici di eccitazione di fig.VII.60.

Si può però, rinunciando a minimizzare gli elementi di memoria, assegnare alle righe della tavola di flusso 4 degli 8 possibili valori di 3 flip-flop SR. Operando questa scelta nel modo indicato nella fig.VII.62a, si ricava la matrice di eccitazione della figura VII.62b.

Scelti opportunamente i valori non specificati, si ottengono così le equazioni:

$$\begin{cases} S_1 = xy_2 \\ R_1 = c \\ S_2 = xy_3 \\ R_2 = c\bar{y}_1 \\ S_3 = x \\ R_3 = c\bar{y}_2 \\ z = cy_3 \end{cases},$$

cui corrisponde il circuito della fig.VII.62c, a 3 flip-flop e 10 diodi.

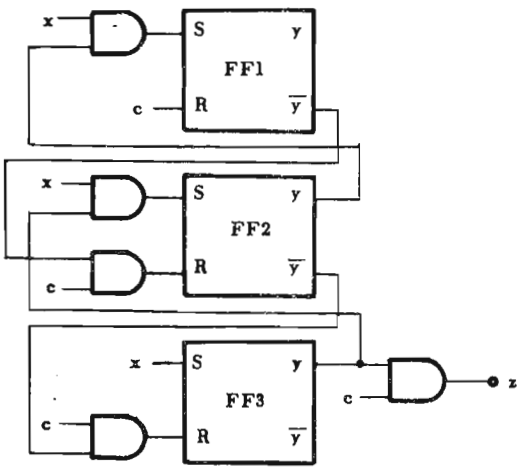
a)

	c	x	
1	1	2	000
2	1,z	3	001
3	2,z	4	011
4	3,z	4	111

b)

$y_1y_2y_3$	c	x	c	x	c	x
000	0-	0-	0-	0-	0-	10
001	0-	0-	0-	10	01	-0
011	0-	10	01	-0	-0	-0
010	-	-	-	-	-	-
110	-	-	-	-	-	-
111	01	-0	-0	-	-	-
101	-	-	-	-	-	-
100	-	-	-	-	-	-
	S_1R_1		S_2R_2		S_3R_3	

Fig.VII.62 - Altra soluzione per il circuito impulsivo di fig.VII.59.



c)

VII.4.4 - Le alee nei circuiti impulsivi.

Nei circuiti impulsivi, il mancato verificarsi delle ipotesi fondamentali porta a un cattivo funzionamento detto *alea di mezzo impulso*.

Abbiamo supposto che 2 impulsi successivi fossero separati da un tempo minimo superiore a quello (tempo di risposta) che il circuito impiega per raggiungere uno stato stabile; vediamo ora cosa accade se questa condizione non viene rispettata. Nei circuiti reali, il passaggio delle uscite di un flip-flop dall'uno all'altro livello non è immediato, ma avviene in un intervallo finito di tempo, dopo il presentarsi dell'impulso che provoca il cambiamento di stato. Nella fig.VII.63a è mostrato l'andamento temporale di una uscita y che cambia valore per effetto di un impulso T. Se, dopo un primo impulso T, arriva un impulso T' nel momento in cui il valore di y non ha ancora raggiunto il massimo (fig.VII.63b), ogni uscita interna o esterna del circuito, legata allo impulso T', può essere fortemente attenuata: l'eventuale segnale ridotto si chiama *mezzo impulso*. Se, infine, 2 impulsi T_1 e T_2 che, susseguendosi esattamente, fanno variare le uscite y_1 e y_2 di 2 flip-flop nel modo indicato dalla fig.VII.63c, sono troppo vicini, cosicché T_2 arriva ai 2 flip-flop come *mezzo impulso*, la risposta del circuito sarà quella voluta solo se il mezzo impulso supera un certo valore di soglia; sarà nulla se il mezzo impulso è inferiore a un secondo valore limite, sarà errata (uno solo dei 2 flip-flop cambierà stato) nei casi intermedi. Se la distanza minima tra gli impulsi di ingresso non può essere controllata, occorre allora assegnare gli stati secondari in modo che ogni impulso faccia variare lo stato di un solo flip-flop.

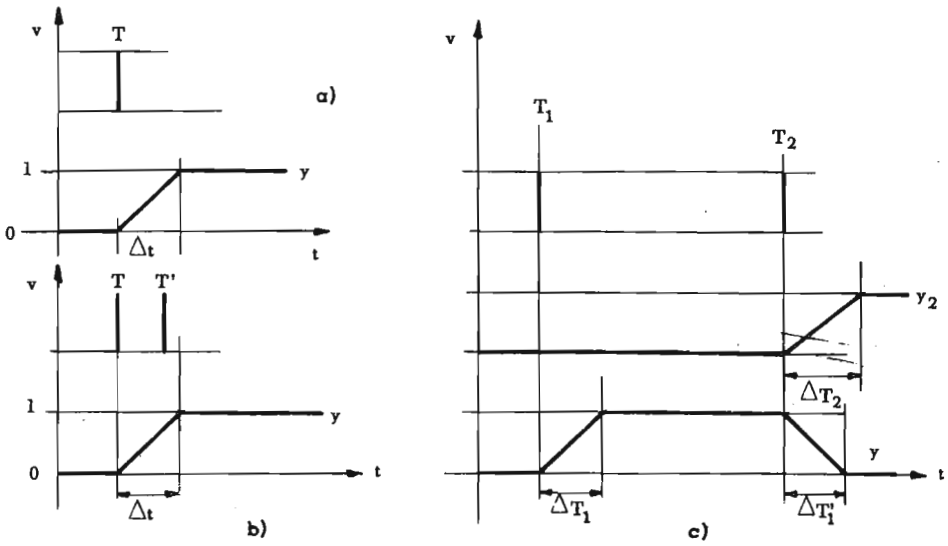
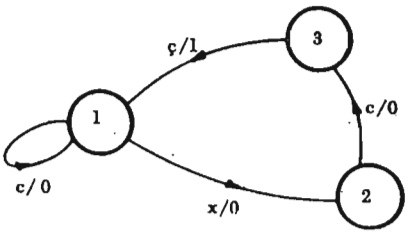


Fig.VII.63 - Alee nei circuiti impulsivi.

Quando, in più, è richiesto che non vadano perduti eventuali impulsi giunti all'ingresso dei flip-flop con valore inferiore alla soglia minima, occorre codificare gli stati in una maniera particolare, mostrata nell'esempio seguente.

Esempio 1: *Sintesi di un circuito impulsivo a 2 ingressi (x e c). C è un impulso di clock a frequenza fissa, x un impulso che arriva casualmente rispetto a c. Due impulsi x consecutivi sono separati da almeno 3 impulsi c. L'uscita z = 1 si ha contemporaneamente al secondo impulso di clock dopo l'impulso x. Il circuito deve essere realizzato in modo che nessun impulso x vada perduto e si chiama sincronizzatore d'impulsi.*



	x	c
1	2	1
2	–	3
3	–	1, z

Fig.VII.64 - Diagramma e matrice degli stati del circuito sincronizzatore d'impulsi.

Il diagramma e la matrice degli stati sono rappresentati nelle figg. VII.64a e VII.64b. Data la particolare natura del problema, occorre assegnare le variabili in modo da rendere impossibile il verificarsi di un'alea di mezzo impulso, che potrebbe presentarsi quando, in seguito a un impulso c dopo un impulso x, il circuito passa dallo stato 2 allo stato 3. Il passaggio 2-3 deve quindi avvenire con la variazione di una sola variabile secondaria. La transizione successiva, che riporta il circuito dallo stato 3 allo stato 1, è libera da alee, perché può arrivare soltanto un impulso c, non un altro impulso x. Un'alea può ancora presentarsi nel passaggio dallo stato 1 al 2, se il primo impulso c è troppo vicino a x: anche qui deve esserci dunque il cambiamento di una sola variabile secondaria.

y_2 \ y_1	0	1
0	1	–
1	2	3

a)

y_1y_2	x	c
00	01	00
01	–	11
11	–	00, z
10	–	–

b) $y_1'y_2'$

y_1y_2	x	c	x	c
00	0 –	0 –	10	0 –
01	–	10	–	– 0
11	–	01	–	01
10	–	–	–	–

c) S_1R_1 S_2R_2

Fig.VII.65 - Assegnazione degli stati secondari (a), matrice di flusso (b) e matrici di eccitazione (c) del sincronizzatore di impulsi.

Una possibile codifica è quella della fig.VII.65a, cui corrisponde la tavola di flusso della fig.VII.65b, la matrice di eccitazione della fig.VII.65c e le equazioni:

$$\begin{cases} S_1 = c\overline{y_1}y_2 \\ R_1 = R_2 = z = cy_1 \\ S_2 = x \ . \end{cases}$$

È interessante analizzare il circuito ottenuto dalle equazioni (fig.VII.66) dal punto di vista elettrico non logico, iniziando dall'esame della sua risposta nelle condizioni normali. Partendo dallo stato di riposo in cui le uscite y_1 e y_2 sono entrambi 0:

– un impulso x fa passare il flip-flop 2 nello stato 1: si ha così un altro ingresso 1 sull'AND A_1, dove già $\overline{y}_1 = 1$;

– il primo impulso c che segue x passa attraverso A_1 e commuta nello stato 1 il flip-flop FF1: cioè mette un ingresso 0 su A_1 e un ingresso 1 su A_2;

– il secondo impulso c passa attraverso A_2, mette a 0 i 2 flip-flop e – contemporaneamente – esce sul terminale z.

Se, ora, l'intervallo tra un impulso x e il primo impulso c è tanto piccolo che y_2 su A_1 non riesce a raggiungere il livello di regime, l'impulso c può o no mettere a 1 il FF1. Nel primo caso, tutto è a posto; nel secondo, il primo impulso c è senza effetto, ma il secondo, trovando y_2 al livello giusto, effettua l'operazione che il primo non ha compiuto. Il terzo impulso c, infine, rimette i 2 flip-flop nello stato di riposo, ed esce sul terminale z.

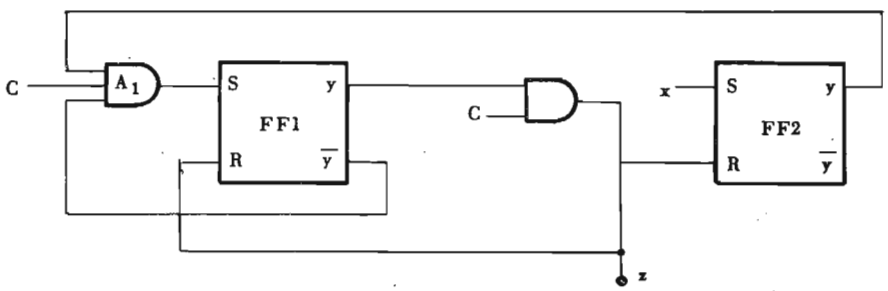


Fig.VII.66 - Circuito sincronizzatore d'impulsi.

Il circuito si è comportato come se x seguisse, anziché precedere, il primo impulso c: la sua uscita è stata ritardata di un impulso di clock, ma il suo effetto non è andato perduto. Anche l'eventuale vicinanza di x a c, quando il circuito si trova nello stato 1, è senza effetto, perché x agisce da solo sull'ingresso del FF2 che è nello stato 0.

VII.5 - Confronto tra circuiti sincroni e asincroni.

Si hanno in pratica notevoli differenze tra 2 circuiti che, l'uno in modo asincrono, l'altro sincrono, realizzano le medesime funzioni: per esempio tra un sommatore sincrono e uno asincrono che lavorano in serie su dati dello stesso tipo e della stessa lunghezza.

Nel primo caso, ogni operazione avviene in un intervallo di tempo prefissato, ed è possibile che la velocità della macchina non venga sfruttata interamente. Nel secondo, invece, ogni operazione inizia ad un segnale di fine dell'operazione precedente, ed il tempo è sempre quello strettamente necessario.

Ad esempio, l'operazione binaria (00000 + 00000) è più rapida della (11111 + 11111), ma un sommatore sincrono in serie dà il risultato delle 2 operazioni dopo lo stesso tempo che, ovviamente, è quello della operazione più lenta. Per aumentare la velocità non si può che diminuire il numero dei livelli: ma questa soluzione, anche quando è possibile, non è economica.

I circuiti asincroni, d'altra parte, presentano un grosso svantaggio nei confronti dei più lenti circuiti sincroni: in essi un'operazione inizia quando la precedente è finita, cioè quando compare un segnale alla uscita di un circuito a monte. Ma se il risultato dell'operazione a monte è 0, non è possibile distinguerlo dallo 0 che indica mancanza di segnale, cioè operazione non terminata.

TABELLA - *Significato dei simboli DR.*

Simbolo	Significato
00	Operazione a monte non terminata
01	Risultato 0
10	Risultato 1
11	Errore

Fig.VII.67 - Segnali usati nella tecnica del Double-Rail e relativi significati.

Per eliminare questa ambiguità, si potrebbe ricorrere a una logica a 3 valori (0 - 1 - niente); ma i circuiti relativi sono complessi e costosi. Si usa, invece, la tecnica della *doppia via* (double rail = DR), consistente nell'usare 2 linee separate per portare un solo bit di informazione; vengono utilizzati soltanto 3 dei 4 segnali possibili (00 - 01 - 11 - 10): il quarto non ha significato e, presentandosi, viene interpretato come un errore (fig. VII.67).

Per identificare un bit d'informazione, si fa riferimento alla coppia dei fili che portano lo stesso segnale, coppia che viene indicata con la stessa lettera, maiuscola e minuscola (es.: Xx, Yy ...).

La tecnica del DR modifica, più o meno profondamente, la forma dei circuiti logici. A titolo d'esempio, illustriamo i circuiti fondamentali AND, OR, NOT.

Esempio 1: *Circuito AND-DR.*

La mappa di Karnaugh per la funzione AND XY, quando ogni variabile è espressa con una coppia di valori, è riportata nella fig.VII.68a. Sfruttando opportunamente le condizioni non assegnate, si ottiene:

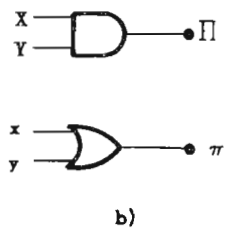
$$\begin{cases} \Pi = XY \\ \pi = x + y \end{cases}$$

cioè il circuito della fig.VII.68b.

a)

Yy \ Xx	00	01	11	10
00	01	01	-	00
01	01	01	-	01
11	-	-	-	-
10	00	01	-	10

$\Pi\pi$



b)

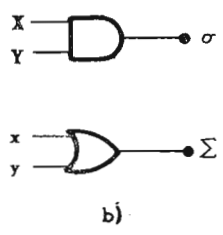
Fig.VII.68 - Funzione AND-DR.

Esempio 2: *Circuito OR-DR.*

a)

Yy \ Xx	00	01	11	10
00	00	00	-	10
01	00	01	-	10
11	-	-	-	-
10	10	10	-	10

$\Sigma\sigma$



b)

Fig.VII.69 - Funzione OR-DR.

La mappa di Karnaugh per la funzione OR X + Y, quando ogni variabile è espressa come una coppia di valori, è riportata nella fig.VII.69.

Le equazioni del circuito (fig.VII.69b) sono:

$$\begin{cases} \Sigma = X + Y \\ \sigma = xy \ . \end{cases}$$

Nella fig.VII.70a e VII.70b sono mostrati i circuiti AND e OR per più di 2 variabili, ottenuti come ovvia estensione di quelli delle figg.VII.68 e VII.69. Nella figura VII.70c è riportato un invertitore DR: come si vede, non c'è bisogno di elementi logici: è questa la ragione per cui nei circuiti a livelli si ammette di avere a disposizione le variabili negate senza ulteriore spesa.

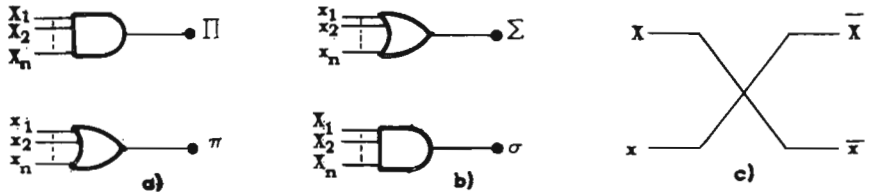


Fig.VII.70 - Circuiti AND-OR-NOT in DR.

Adoperando circuiti DR, per sapere se un'informazione è passata attraverso una serie di blocchi logici, si fa entrare in un OR di tipo normale l'ultima coppia di linee di uscita: l'uscita dell'OR sarà 1 solo se agli ingressi non si presentano i valori 00. Evidentemente, tutta la catena dei circuiti DR va messa a 00 prima che inizi l'elaborazione dei dati.

Esempio 3: *Sintesi di un circuito DR a diodi per la funzione «somma modulo 2»:*

$$F = X \oplus Y = X\overline{Y} + \overline{X}Y \quad .$$

Fig.VII.71 - Mappa di Karnaugh per la funzione ⊕ in DR.

Yy \ Xx	00	01	11	10
00	00	00	-	00
01	00	01	-	10
11	-	-	-	-
10	00	10	-	01

Ff

La mappa di Karnaugh per la F nella versione DR è quella della fig.VII.71. Pertanto:

$$\begin{cases} F = Xy + Yx \\ f = XY + xy \ . \end{cases}$$

Il circuito corrispondente alle equazioni trovate è disegnato nella fig.VII.72a. Lo stesso circuito è ridisegnato nella fig.VII.72b, mettendone in evidenza l'origine, a partire da un circuito normale, mediante le relazioni:

$$\begin{cases} x = \overline{X} \\ y = \overline{Y} \end{cases}.$$

Sul terminale φ, chiamato *linea di fine* (finish line) si ha il segnale 1 quando, e solo quando, l'operazione è terminata.

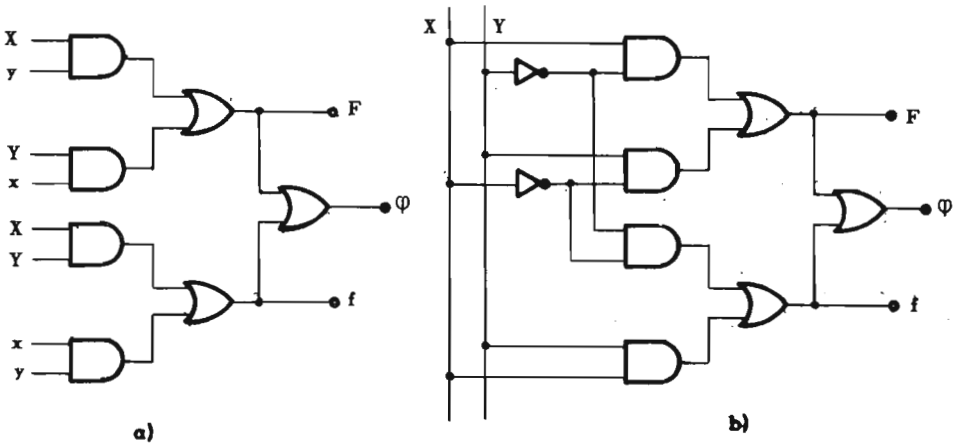


Fig.VII.72 - Circuito ⊕ in DR.

La tecnica del DR e l'aggiunta del terminale φ, complicano notevolmente la realizzazione dei circuiti sequenziali asincroni e dei circuiti combinatori ad essi associati. Ad esempio, se un circuito combinatorio ne alimenta uno sequenziale, conviene avere ad 1 la linea di fine del primo solo dopo che l'informazione è stata immagazzinata nel secondo: ne derivano interazioni reciproche tra i due circuiti, non sempre immediatamente intuibili. Per uno studio approfondito del DR, rimandiamo, comunque, ai testi citati nella bibliografia.

Da quanto esposto in questo paragrafo, si può concludere che un calcolatore interamente asincrono richiede, grosso modo, il doppio dei componenti di uno interamente sincrono: ciò, pur tenendo conto della maggiore velocità dei circuiti asincroni, è sempre antieconomico. Un buon compromesso è quello di costruire una parte del calcolatore funzionante in modo sincrono e l'altra (ovviamente quella che più risente del vantaggio di una maggiore velocità) in modo asincrono.

*

BIBLIOGRAFIA

1. MILLER R.E.: *Switching Theory* - vol.II - John Wiley & Sons, 1965.

2. PHISTER M.J.: *Logical Design of Digital Computers* - John Wiley & Sons, 1958.

3. DAVIDOW W.H.: *A State Assignement Technique for Synchronous Sequential Networks* - Stanford University, 1961.

4. KARP R.M.: *Some Techniques of State Assignment for Synchronous Sequential Machines* - IBM Research Report - Maggio 1963.

5. MILLER R.E.: *Switching Theory and Logical Design of Automatic Digital Computer Circuits* - IBM Research Report - Giugno 1961.

6. SCHNEIDER M.I.: *State Assignement Algoritm for Clocked Sequential Machines* - MIT-LLTR - N. 270 - Maggio 1962.

7. FLORINE I.: *La Synthese des Machines Logiques* - Dunod, 1965.

8. NASLIN P.: *Circuits logiques et automatismes à séquences* - Dunod, 1965.

*

CAPITOLO VIII

TECNICHE DIGITALI

VIII.1 - Generalità.

Nei capitoli precedenti abbiamo trattato i fondamenti e le principali applicazioni della teoria della commutazione. I metodi esposti consentono di risolvere qualsiasi problema di natura combinatoria e sequenziale, e sono di carattere assolutamente generale; sulla loro praticità ed utilità è però necessario avanzare qualche riserva.

Mentre i circuiti combinatori si progettano e si realizzano secondo la corrispondente teoria, i sequenziali hanno, in pratica, forme più o meno diverse da quelle cui si giungerebbe applicando rigorosamente i metodi di Moore, Mealy, Huffman. Ciò dipende da varie cause:

a) Le sequenze d'ingresso possibili sono - per macchine appena un poco complesse - estremamente elevate: ne derivano diagrammi degli stati e tavole di flusso di dimensioni tali da rendere i calcoli relativi alla minimizzazione e alla costruzione delle equazioni caratteristiche praticamente irrisolvibili. Nè sono stati trovati metodi generali per trattare i problemi su calcolatori.

b) I metodi di minimizzazione degli stati non portano necessariamente al circuito più economico che realizza un determinato comportamento, ma solo a quello col minor numero di elementi di memoria, e limitatamente al caso in cui non esistono ridondanze nella tabella degli stati. Nel testo sono stati dati, a questo proposito, numerosi esempi di circuiti minimi realizzati con un flip-flop, o con un loop di reazione, in più rispetto al numero teorico minimo.

c) I problemi della codifica degli stati interni e della scelta del tipo di flip-flop, essenziali per minimizzare le reti d'uscita e di memoria, si sanno risolvere soltantò per tentativi: ma il numero dei tentativi stessi è, per altro, elevatissimo.

d) Le ipotesi fondamentali su cui è stata impostata la sintesi dei circuiti sequenziali sincroni, particolarmente nei riguardi della forma e della durata degli impulsi di clock e delle relazioni temporali fra impulsi e tempi di reazione del circuito, non sempre sono verificate.

Per la prima delle ragioni esposte, è necessario che ogni sistema digitale, cioè ogni apparecchiatura che realizza funzioni logiche complesse, venga diviso e concepito come l'insieme di un numero più o meno grande di circuiti più semplici legati in modo che le uscite di alcuni costituiscano gli ingressi di altri. Allo stato attuale della tecnica è, ad esempio, del tutto impensabile realizzare un calcolatore, per quanto modesto, come un solo circuito sequenziale.

Per gli ultimi tre motivi, poi, si rinuncia a progettare ex-novo i predetti circuiti, anche perchè i relativi problemi di sintesi appartengono essenzialmente ad un ristretto insieme, la cui soluzione è ormai ben nota e collaudata: si preferisce, in ogni caso, adottare queste soluzioni, forse meno economiche, per ragioni di modularità e semplicità costruttiva. La teoria dei circuiti sequenziali non è tuttavia inutile: essa permette di risolvere problemi nuovi e, talvolta, offre soluzioni più brillanti di quelle che si ottengono con i metodi empirici.

In questo capitolo intendiamo esporre le soluzioni adottate in pratica per i problemi più comuni nel campo delle tecniche digitali, senza fare riferimento a nessuna particolare applicazione. I circuiti illustrati sono tutti di tipo sincrono, generalmente a flip-flop IK. Per la versione asincrona degli stessi circuiti non esistono soluzioni generali. I circuiti stessi vanno quindi progettati secondo la teoria di Huffman, con qualche accorgimento particolare, di natura più circuitale che logica (si vedano, a questo proposito, gli esempi finali del cap. VI).

VIII.2 - Circuiti fondamentali dei sistemi digitali.

Per *sistema digitale* intendiamo qui ogni apparecchiatura in cui i dati vengano immagazzinati ed elaborati in forma binaria, mediante dispositivi di natura elettronica.

I dati immagazzinati sono ordinati in insiemi di bit, chiamati *parole*, aventi un determinato significato secondo un certo codice (vedi ca-

pitolo I). Il termine *parola* si riferisce alle informazioni immagazzinate, si chiama invece *registro* il dispositivo fisico che immagazzina una parola o, più in generale, un vettore binario, in funzione del tempo. Un sistema digitale è formato essenzialmente da registri e da circuiti logici di tipo combinatorio.

VIII.2.1 - Registri.

Registro è un circuito che immagazzina una parola, formato da tanti elementi di memoria (celle) quanti sono i bit della parola stessa. I registri di seguito descritti sono tutti a flip-flop; i flip-flop usati, conformemente a quanto avviene in pratica, sono quelli a circuiti integrati descritti nel cap.III; per semplicità, quando occorre un solo ingresso J K, si è evitato di disegnare le porte AND. Gli schemi forniti, con un'apposita temporizzazione dei segnali d'ingresso, sono validi anche per i flip-flop della tab.1 del cap.III.

Un registro che immagazzina una parola di n bit è formato da n flip-flop che ne costituiscono le celle. Si chiama *capacità* del registro il numero 2^n di parole diverse che esso può registrare, numero coincidente con gli stati delle sue n celle. *Scrittura* è l'introduzione di una parola in un registro; *lettura* la rilevazione della parola immagazzinata; *cancellazione*, l'operazione di azzeramento delle celle del registro, che precede sempre la scrittura.

Ogni informazione può essere scritta in un registro solo se la sua capacità è sufficiente a contenerla: è la natura dell'informazione che determina, quindi, il numero delle celle. Ad esempio, per immagazzinare un numero compreso tra 0 e 99 secondo un codice binario non ridondante, occorre un registro con una capacità di 100 parole, cioè a 7 celle ($2^6 < 100 < 2^7$). Usando un codice BCD, cioè codificando separatamente ogni cifra, occorrono invece $2 \times 4 = 8$ celle ($2^3 < 10 < 2^4$).

I registri possono essere in parallelo, in serie (o a scorrimento) e misti, a seconda che i bit delle parole registrate siano scritti (e letti) in parallelo, in serie o in ambedue i modi.

Un insieme di n bit costituenti un'informazione si dice *in parallelo* se ogni bit è associato al valore 0 o 1 che compare, nello stesso istante, su n terminali; si dice *in serie* se il bit i^{mo} è associato al valore che compare, su uno stesso terminale, all'istante $(t + i\Delta)$, essendo t l'istante in cui compare il primo bit, Δ un intervallo di tempo costante (periodo di clock) e $i = 0, 1, \ldots n-1$. Nei sistemi sincroni con periodo di clock Δ, occorrono $m \cdot \Delta$ secondi ed n fili per trasmettere da un punto all'altro m parole di n bit in parallelo, nonchè $m \cdot n \cdot \Delta$ secondi ed 1 filo per trasmettere m parole di n bit in serie.

VIII.2.1.1 - Registri in parallelo.

Un registro in parallelo è formato da n flip-flop JK indipendenti su cui vengono scritte, immagazzinate e lette parole di n bit disponibili come livelli 0 e 1 in parallelo su n terminali.

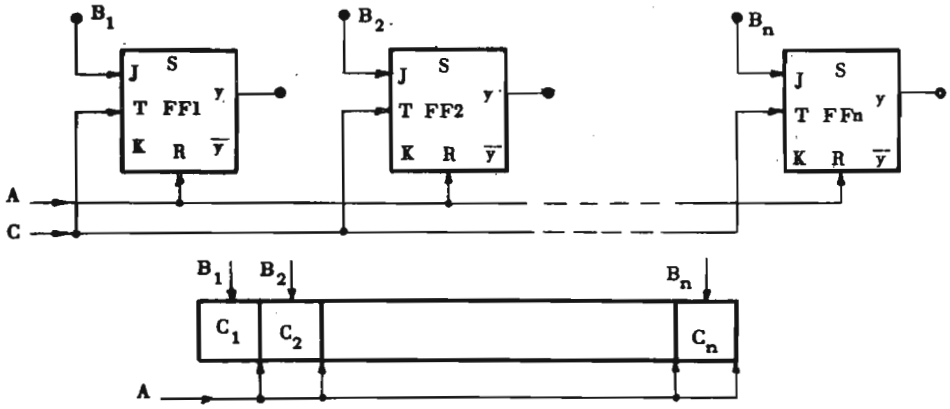


Fig.VIII.1 - Registro in parallelo.

Lo schema logico di un registro in parallelo è mostrato nella figura VIII.1. Prima della scrittura, un impulso A (impulso di azzeramento) collegato a tutti gli ingressi R dei flip-flop mette a 0 i flip-flop stessi. Gli n bit $B_1 B_2 \ldots B_n$ da immagazzinare si trovano, sotto forma di livelli, su altrettanti terminali collegati agli ingressi J dei flip-flop; all'istante in cui si vuole effettuare l'operazione di scrittura, si manda un impulso di clock C sugli ingressi T. Tutti e soli i flip-flop collegati ai bit 1 commutano, per cui - alla fine dell'impulso C - i valori B_i risultano trasferiti nelle celle C_i del registro, e sono rilevabili come valori delle uscite y_i.

Un registro in parallelo viene indicato, più semplicemente, anche col simbolo della fig.VIII.1 b.

VIII.2.1.2 - Registri a scorrimento (Shift Register).

I registri a scorrimento sono formati da n flip-flop in serie e, oltre che ad immagazzinare n bit, servono a ritardare di n periodi di clock un flusso d'informazioni. I flip-flop di questi registri sono collegati fra loro in modo che ad ogni istante di clock (fronte di discesa dell'impulso, in logica positiva) il contenuto di ciascun flip-flop venga trasferito al successivo.

Un registro a scorrimento può essere realizzato con flip-flop JK come mostrato nella fig.VIII.2a (nella fig.VIII.2b è mostrata la rappresentazione simbolica dello stesso registro). All'istante iniziale - dopo un segnale di azzeramento A - comincia il flusso delle informazioni: sui terminali B e $\overline{B}$ si presenta il primo bit B_1 (e il suo complemento $\overline{B}_1$) della parola da immagazzinare. Viene mandato un impulso di clock C che trasferisce B_1 su y_1. Successivamente, si manda il secondo bit B_2 sul terminale B; un secondo impulso di clock trasferisce B_2 su y_1 e y_1 (cioè B_1) su y_2.

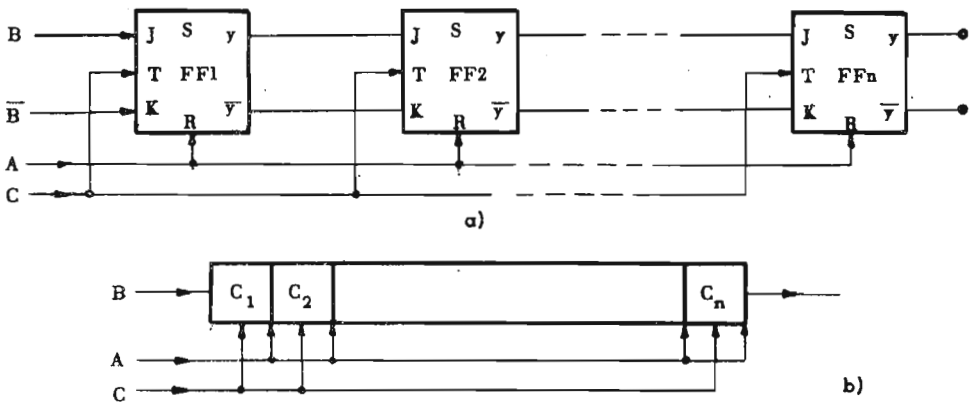


Fig.VIII.2 - Registro a scorrimento (shift avanti).

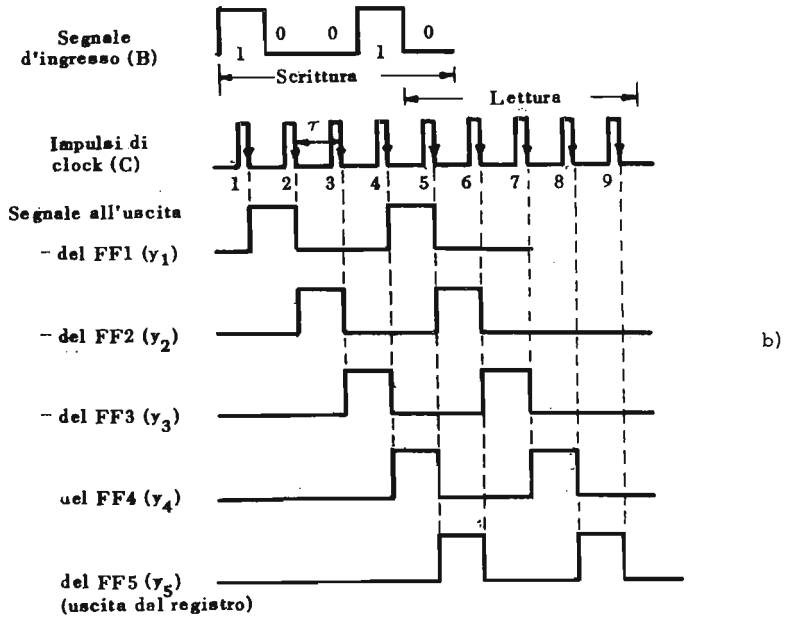
Dopo (p + K - 1) impulsi di clock, il K^{mo} bit (B_K) si trova, così, all'uscita del p^{mo} flip-flop del registro. Per avere il K^{mo} bit sull'uscita occorrono, quindi, (n + K - 1) impulsi di clock. Poichè una parola è formata da n bit, la sua trasmissione completa dall'ingresso B all'uscita B' avverrà in (2n - 1) impulsi di clock.

Nella fig.VIII.3a è mostrata la posizione di ognuno dei 5 bit di una parola in un registro a 5 celle, durante ciascuno dei 9 impulsi di clock. Nella fig.VI.3 b è mostrato il diagramma temporale dei segnali (che sono livelli di tensione di durata τ). Si è dato, a titolo d'esempio il valore 10010 ai bit $B_1 \ldots B_5$; l'impulso di clock è inviato a metà dell'intervallo τ, per ragioni di sicurezza.

Ovviamente, se un registro viene usato per ritardare un flusso di informazioni di N_p periodi di clock, deve essere formato da N_p celle e può essere adoperato per parole di lunghezza qualsiasi.

a)

Impulsi di clock (C)	Bit all'uscita del flip-flop K^{mo} dopo l'impulso C				
	K = 1	K = 2	K = 3	K = 4	K = 5
1	B_1				
2	B_2	B_1			
3	B_3	B_2	B_1		
4	B_4	B_3	B_2	B_1	
5	B_5	B_4	B_3	B_2	B_1
6	-	B_5	B_4	B_3	B_2
7	-	-	B_5	B_4	B_3
8	-	-	-	B_5	B_4
9	-	-	-	-	B_5



b)

Fig. VIII.3 - Configurazione delle 5 celle di un registro a scorrimento, in funzione del tempo, durante la lettura e la scrittura di una parola di 5 Bit ($B_1 \ldots B_5$ = 10010).

Nel registro della fig.VIII.2 i bit si muovono, ad ogni impulso di clock, da sinistra verso destra (shift in avanti). La direzione del movimento dei bit nel registro dipende dalle connessioni fra le uscite y e $\overline{y}$ dell'i^{mo} flip-flop e gli ingressi J e K del flip-flop $(i+1)^{mo}$. Con le connessioni della fig.VIII.4 il movimento avviene da destra verso sinistra (shift indietro).

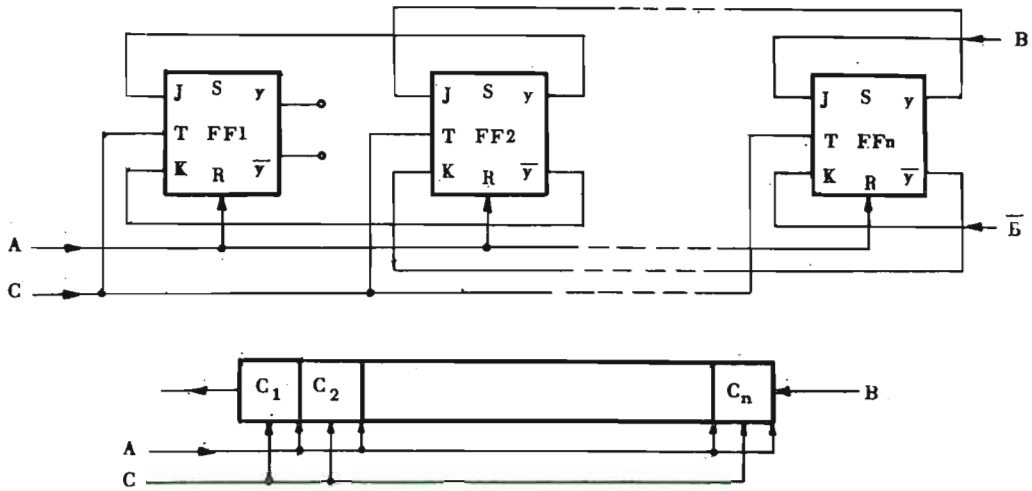


Fig.VIII.4 - Registro a scorrimento (shift indietro).

Se si vuole la possibilità di scegliere la direzione del moto, bisogna connettere, attraverso 2 AND, gli ingressi di ogni flip-flop alle uscite sia del flip-flop che lo precede sia di quello che lo segue, e realizzare l'una o l'altra connessione con un segnale di comando il cui valore dipende dalla direzione prescelta.

In molti casi, è necessario far *circolare* un'informazione immagazzinata in un registro, in modo da spostare l'ordine dei bit (con questo sistema, ad esempio, si moltiplicano o si dividono i numeri binari).

Un circuito per la circolazione dei bit è quello della fig.VIII.5a: le due reti combinatorie servono a permettere l'una o l'altra delle operazioni di scrittura e di circolazione. Se è presente il segnale di scorrimento (SC = 1), il primo flip-flop assume lo stato del terzo; il secondo quello del primo, e il terzo quello del secondo: in questo modo, ogni tre impulsi di clock, i bit tornano nella posizione di partenza. Quando si vogliono far entrare i bit di una nuova parola, si porta a 0 il segnale $\overline{SC}$, immettendo così nel registro i valori che compaiono sui terminali B e $\overline{B}$. dopo 3 impulsi di clock, la nuova informazione è completamente immagazzinata.

Nella fig.VIII.5b è rappresentato lo schema a blocchi di un registro a circolazione con scorrimento in avanti; esistono anche registri a circolazione con scorrimento all'indietro, misti, e con possibilità di scorrimento parziale (per esempio dall'ultima alla seconda cella).

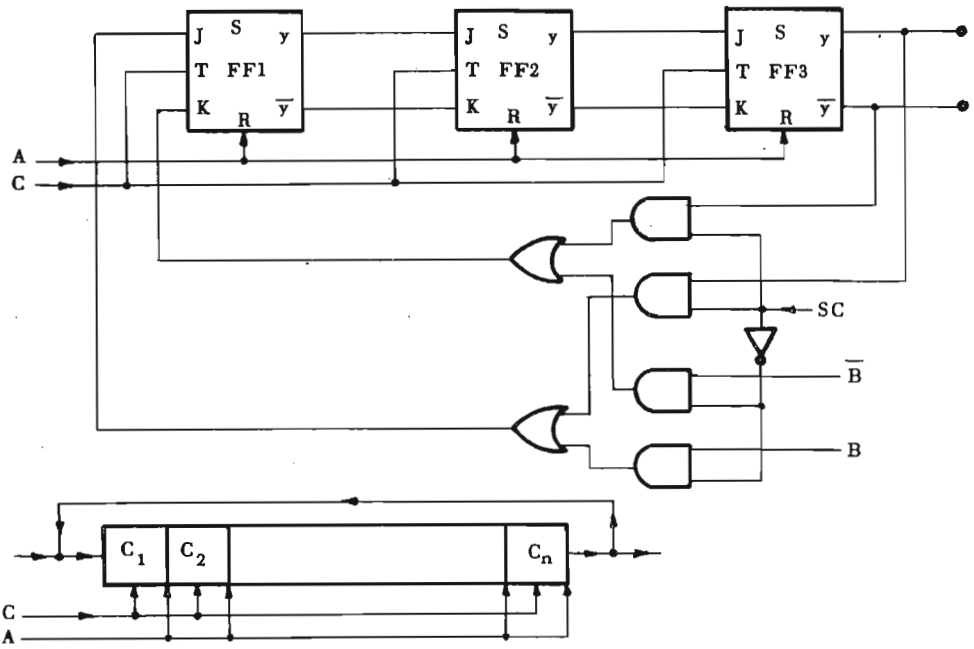


Fig.VIII.5 - Registro a circolazione con scorrimento in avanti.

VIII.2.1.3 - Registri misti (dual mode register).

Alcuni registri funzionano accettando i dati d'ingresso in serie, e presentandoli su n terminali in parallelo. Un esempio di questo tipo di registri è mostrato nella fig.VIII.6; dopo n impulsi di clock, tutti i bit della parola sono stati immagazzinati e sono disponibili ai terminali $B_1 B_2 \ldots B_n$, sia in forma diretta che in forma negata.

Il circuito illustrato in fig.VIII.6 serve alla conversione serie - -parallelo dei dati: altri circuiti realizzanti lo stesso obiettivo saranno mostrati nei prossimi paragrafi (il filo L serve per leggere le uscite nello stesso istante).

Con una disposizione leggermente diversa è possibile introdurre in un registro dei dati in parallelo, ed estrarli in serie. Nel circuito del-

la fig.VIII.7, all'arrivo dell'impulso di scrittura S, i bit $B_1 \dots B_n$ vengono immagazzinati negli n flip-flop del registro; successivamente, n impulsi di clock li fanno uscire, in serie, dai terminali B' e $\overline{B}'$.

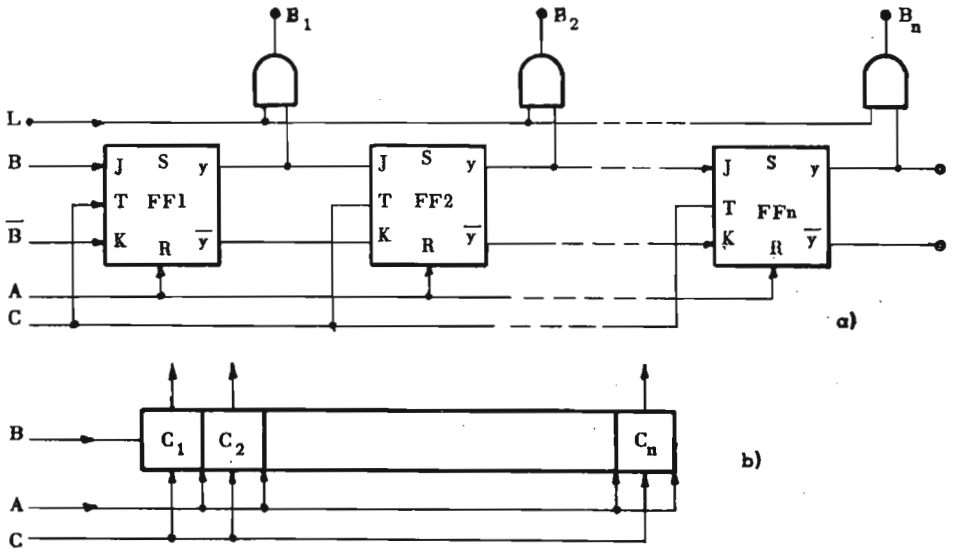


Fig.VIII.6 - Registro con ingresso in serie e uscita in parallelo.

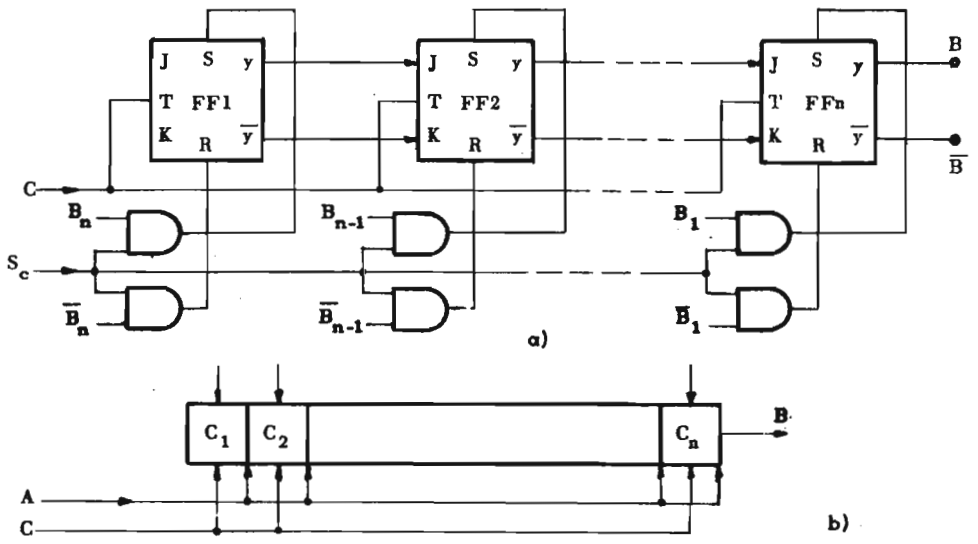


Fig.VIII.7 - Registro con ingresso in parallelo e uscita in serie.

VIII.3 - Contatori.

Una delle operazioni più comuni dei sistemi digitali è il conteggio del numero degli impulsi, effettuato da un contatore.

Per *contatore* si intende un circuito con n flip-flop che, pilotato da una successione di impulsi di *trigger*, presenta una sequenza periodica di M configurazioni delle sue n uscite ($2^{n-1} < M \leq 2^n$). Il numero M è il *modulo* del contatore; si chiamano *binari* quei contatori in cui M è una potenza di 2 e le uscite si susseguono secondo la serie naturale dei numeri binari da 0 a (M - 1).

I contatori si dividono, poi, in *sincroni* o *asincroni*, a seconda che l'impulso di trigger (T) sia inviato contemporaneamente a tutti i flip-flop o no. Nel primo caso, le uscite dei flip-flop debbono condizionare gli impulsi T, per realizzare la sequenza voluta; nel secondo, l'uscita di ogni flip-flop costituisce il trigger di un flip-flop seguente. Nei flip-flop sincroni tutte le uscite seguono - contemporaneamente - l'impulso T, col solo ritardo dovuto al tempo di commutazione del flip-flop; nel secondo, la configurazione finale è raggiunta dopo un ritardo che può essere n volte superiore.

I termini *sincrono* e *asincrono* sono, qui e in seguito, usati in senso completamente diverso da quello che gli omonimi termini avevano nella teoria dei circuiti e delle macchine sequenziali.

VIII.3.1 - Contatori Binari Asincroni.

Un contatore binario asincrono può essere formato da n flip-flop JK in serie, collegati nel modo mostrato nella fig. VIII.8 (per n = 4).

Descriviamo brevemente il funzionamento del circuito supponendo di usare la logica positiva, quindi flip-flop che commutano coi fronti di caduta degli impulsi di trigger (questa convenzione resterà valida per tutti i circuiti di questo capitolo).

Gli impulsi da contare vengono inviati sul terminale I del contatore, dopo averlo azzerato con un segnale sul terminale A. Il primo flip-flop cambia di stato ad ogni fronte di caduta degli impulsi; il secondo flip-flop ad ogni fronte di caduta dell'uscita y del primo flip-flop; il terzo e il quarto - rispettivamente - in coincidenza coi fronti di caduta del secondo e del terzo flip-flop. Nella fig. VIII.9 sono mostrate le tensioni d'uscita dei 4 flip-flop ad ogni impulso.

Il contatore della fig. VIII.8 può esistere in 16 stati, o posizioni, diverse: dopo il sedicesimo impulso, tutti i flip-flop tornano nello stato iniziale.

Un contatore binario a n flip-flop può, in generale, assumere 2^n configurazioni diverse e, pertanto, contare fino a 2^n impulsi, non distinguendo l'impulso il cui numero d'ordine è K da quelli i cui numeri d'ordine sono $(K + m \cdot 2^n)$.

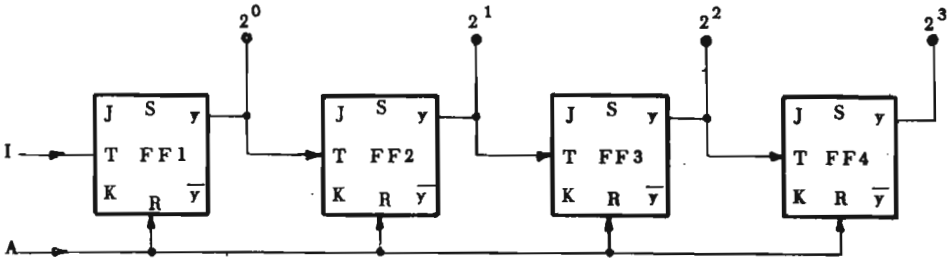


Fig.VIII.8 - Contatore binario (modulo 16).

Se si dà, come in fig.VIII.8, un peso alle uscite dei vari flip-flop, il numero degli impulsi arrivati sull'ingresso I si ottiene, a meno del modulo 2^n, sommando i pesi delle uscite che si trovano al valore 1.

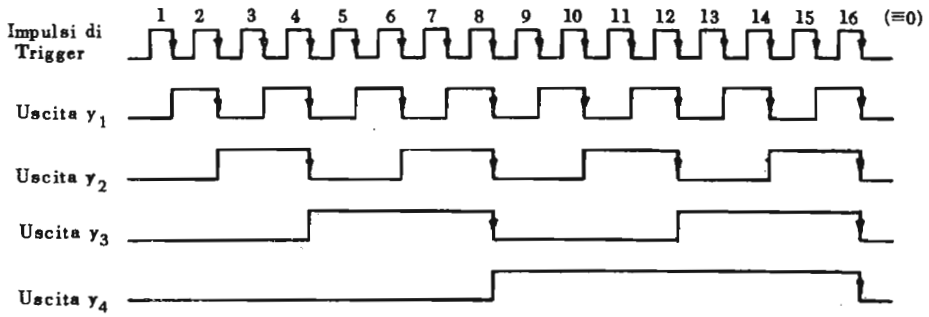


Fig.VIII.9 - Tensioni d'uscita dei 4 flip-flop del contatore di fig.VIII.8.

Si possono costruire contatori *scalanti* che, invece di contare secondo l'ordine crescente $(0, 1, \ldots 2^n - 1)$, contano dal valore più alto

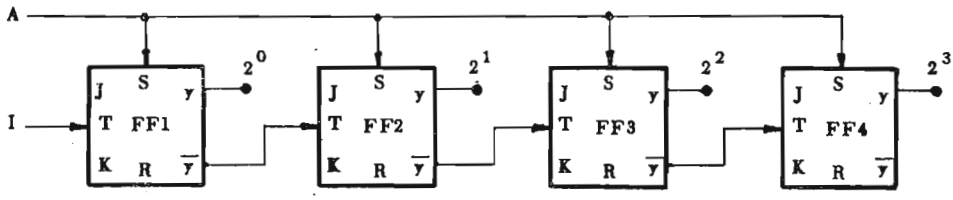


Fig.VIII.10 - Contatore asincrono scalante (modulo 16).

verso il più basso ($2^n - 1$, $2^n - 2$, $2^n - 3$, ... 2 1 0). Un contatore scalante deve essere messo al valore massimo all'inizio del conteggio ed avere le connessioni secondo lo schema della fig.VIII.10 (con ovvia disposizione dei circuiti di controllo, si possono realizzare contatori che siano - insiemi - normali e scalanti).

VIII.3.2 - Contatori binari sincroni.

Un contatore sincrono binario è formato da n flip-flop JK collegati come mostrato nella fig.VIII.11 (per n = 4): qui e in seguito, gli ingressi J e K non collegati si intendono a 1 (v. cap.III).

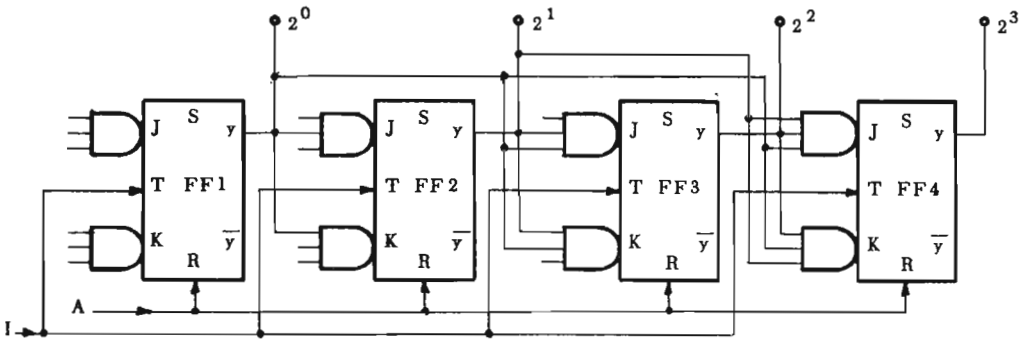


Fig.VIII.11 - Contatore sincrono modulo 16.

Il primo impulso I inviato, dopo quello di azzeramento, a tutti i flip-flop del contatore, mette a 1 soltanto il primo dei flip-flop stessi, perchè gli altri hanno almeno un ingresso J a zero. Poichè, dopo questo impulso, $y_1 = 1$, al secondo impulso commutano insieme i primi due flip-flop. Il terzo impulso, trovando $y_1 = 0$ e $y_2 = 1$ fa commutare soltanto il primo flip-flop e lascia quindi y_1 e y_2 a 1 .... Le tensioni d'uscita dei vari flip-flop ad ogni impulso sono, in definitiva, ancora quelle della fig.VIII.9.

VIII.3.3 - Contatori di modulo $M \neq 2^k$.

Modificando i contatori binari, si costruiscono contatori secondo un modulo M qualsiasi, partendo - se $2^{n-1} < M \leq 2^n$ - da un contatore binario a n flip-flop.

Il metodo più semplice per progettare questi contatori è quello esposto da M. Phister in [1], metodo che qui riassumiamo brevemente, limitatamente ai flip-flop JK.

Le proprietà del flip-flop JK, espresse nella tabella (già riportata nel cap.VII) che fornisce i valori degli ingressi J e K in funzione dello stato presente y e dello stato futuro y':

y	y'	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

possono essere espresse dall'equazione:

$$y' = y\overline{K} + \overline{y}J \qquad \text{(VIII.1)}$$

La (VIII.1) si chiama *equazione caratteristica del flip-flop JK* e fornisce la risposta del flip-flop stesso ai segnali d'ingresso (in modo analogo si possono ricavare le equazioni caratteristiche di ogni tipo di flip-flop).

Supponiamo, ora, di dover progettare un contatore che *conti* secondo una determinata sequenza, nel senso che gli stati dei suoi flip-flop assumano - ad ogni impulso di clock - i valori della sequenza stessa.

Se con $(y_1 y_2 \dots y_n)$ indichiamo la generica configurazione del contatore all'istante t, e con $(y'_1 y'_2 \dots y'_n)$ la configurazione all'istante $t+\Delta$, successiva all'arrivo di un impulso di clock, possiamo descrivere il funzionamento del contatore in una tabella di verità come quella della figura VIII.12, relativa ad un contatore a 3 flip-flop che conta secondo la sequenza:

$$0 \to 2 \to 3 \to 7 \to 6 \to 4 \to 0 \to \dots \quad .$$

Dalla tabella di verità si possono ricavare le equazioni:

$$y'_i = f_i(y_1, y_2, y_3) \qquad (i = 1, 2, 3) \quad . \qquad \text{(VIII.2)}$$

Le (VIII.2), la cui forma dipende dal particolare problema trattato, si chiamano *equazioni di applicazione*; esse definiscono il ruolo che ogni flip-flop gioca nel contatore, in funzione delle configurazioni di quest'ultimo e del tutto indipendentemente dalle (VIII.1), cioè dal tipo

di flip-flop. [Si noti che le (VIII.2) coincidono con le equazioni delle y' dei circuiti a impulsi non contemporanei, se moltiplicate per il clock].

Stato presente			Stato futuro		
$y_1(1)$	$y_2(2)$	$y_3(4)$	$y'_1(1)$	$y'_2(2)$	$y'_3(4)$
0	0	0	0	1	0
1	0	0	-	-	-
0	1	0	1	1	0
1	1	0	1	1	1
0	0	1	0	0	0
1	0	1	-	-	-
0	1	1	0	0	1
1	1	1	0	1	1

Fig.VIII.12 - Tabella di verità di un contatore modulo 6 (i pesi di ogni y sono indicati tra parentesi).

Scrivendo le (VIII.2) nella forma:

(VIII.2') $$y'_i = f_{1i}\, y_i + f_{2i}\, \overline{y}_i$$

è possibile, considerando che [v. la (VIII.1)]:

$$y'_i = \overline{K}_i\, y_i + J_i\, \overline{y}_i$$

ricavare immediatamente le espressioni di J_i e K_i in funzione delle y, ponendo:

$$\begin{cases} J_i = f_{2i} \\ K_i = \overline{f}_{1i} \end{cases} .$$

Nel nostro caso, sfruttando le condizioni non specificate nella tabella della fig.VIII.12, attraverso le mappe di Karnaugh di fig.VIII.13 si ricava:

$$\begin{cases} y'_1 = \overline{y}_1\, (y_2 \overline{y}_3) + y_1\, (\overline{y}_3) \\ y'_2 = \overline{y}_2\, (\overline{y}_3) + y_2\, \overline{(\overline{y}_1\, y_3)} \\ y'_3 = \overline{y}_3\, (y_1) + y_3\, (y_2) \end{cases}$$

e pertanto:

$$\begin{cases} J_1 = y_2 \overline{y}_3 \\ K_1 = y_3 \end{cases} \qquad \begin{cases} J_2 = \overline{y}_3 \\ K_2 = \overline{y}_1 y_3 \end{cases} \qquad \begin{cases} J_3 = y_1 \\ K_3 = \overline{y}_2 \end{cases}$$

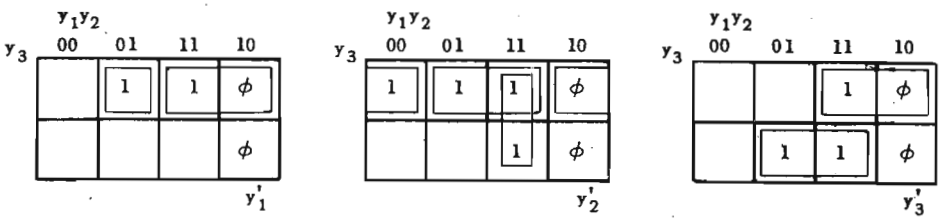
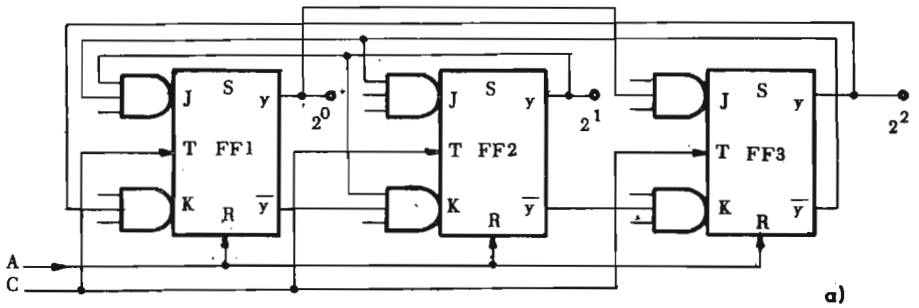


Fig.VIII.13 - Mappe di Karnaugh per le funzioni descritte nella tabella di fig.VIII.12 (si noti che le semplificazioni sono fatte in modo da separare y_1 da $\overline{y}_1$).



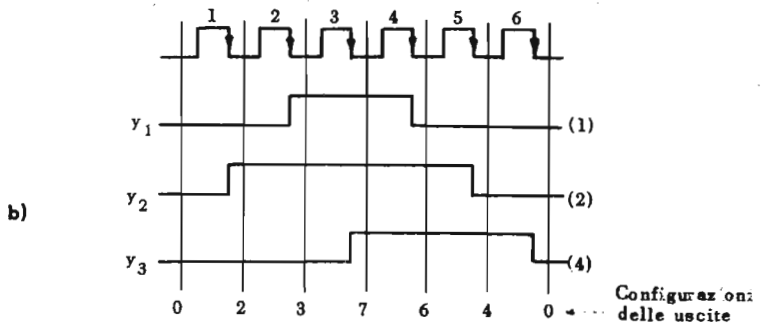


Fig.VIII.14 - Contatore modulo 6 secondo la sequenza: 0→2→3→7→6→4→0....

Il contatore, costruito secondo queste equazioni, è riportato nella fig.VIII.14a; il diagramma temporale delle sue tensioni d'uscita è riportato, per maggior chiarezza, nella fig.VIII.14b.

Fra tutti i contatori, una particolare importanza hanno i contatori decimali, in special modo quelli che contano secondo la serie naturale dei numeri binari (in codice 1 • 2 • 4 • 8).

VIII.3.4 - Contatori decimali 1 • 2 • 4 • 8.

Nella fig. VIII.15a è riportata la tabella di verità che descive il funzionamento di un contatore decimale; dalle mappe di Karnaugh per le

Stato presente				Stato futuro			
$y_1(1)$	$y_2(2)$	$y_3(4)$	$y_4(8)$	$y'_1(1)$	$y'_2(2)$	$y'_3(4)$	$y'_4(8)$
0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0
0	1	0	0	1	1	0	0
1	1	0	0	0	0	1	0
0	0	1	0	1	0	1	0
1	0	1	0	0	1	1	0
0	1	1	0	1	1	1	0
1	1	1	0	0	0	0	1
0	0	0	1	1	0	0	1
1	0	0	1	0	0	0	0

a)

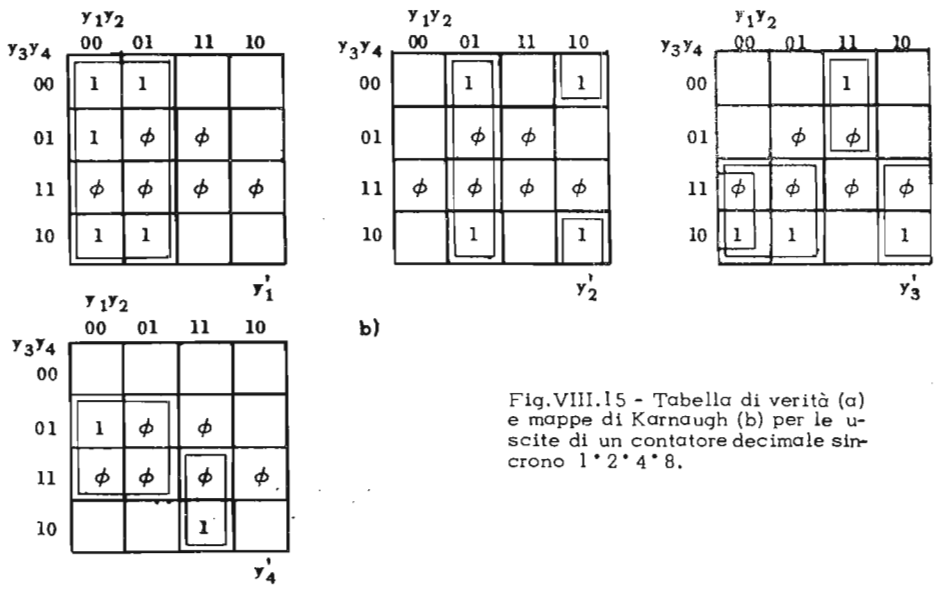


Fig. VIII.15 - Tabella di verità (a) e mappe di Karnaugh (b) per le uscite di un contatore decimale sincrono 1 • 2 • 4 • 8.

funzioni y_1' (fig.VIII.15b) si ricavano le equazioni:

$$\begin{cases} y_1' = \overline{y_1} + 0 \cdot y_1 \\ y_2' = (y_1 \overline{y_4})\,\overline{y_2} + (\overline{y_1})\,y_2 \\ y_3' = (y_1 y_2)\,\overline{y_3} + (\overline{y_1 y_2})\,y_3 \\ y_4' = (y_1 y_2 y_3)\,\overline{y_4} + (\overline{y_1})\,y_4 \end{cases}$$

pertanto:

$$\begin{cases} J_1 = 1 \\ K_1 = 1 \end{cases} \qquad \begin{cases} J_2 = y_1 \overline{y_4} \\ K_2 = y_1 \end{cases} \qquad \begin{cases} J_3 = y_1 y_2 \\ K_3 = y_1 y_2 \end{cases} \qquad \begin{cases} J_4 = y_1 y_2 y_3 \\ K_4 = y_1 \end{cases}.$$

Il circuito, costruito connettendo in questo modo 4 flip-flop, è mostrato nella fig.VIII.16; il contatore realizzato è di tipo sincrono.

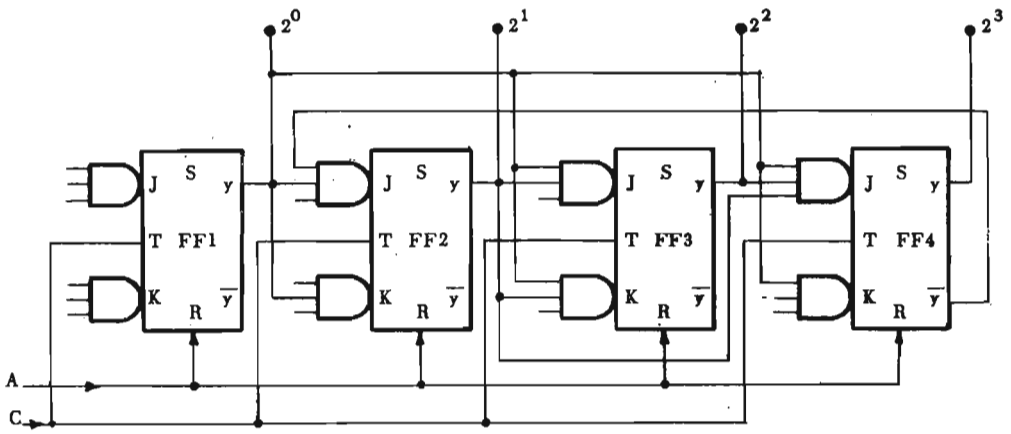


Fig.VIII.16 - Contatore decimale sincrono.

Se si vuole costruire un contatore asincrono, bisogna tener conto che alcuni flip-flop non sono comandati direttamente dal clock, ma dalle uscite di precedenti flip-flop. Pertanto, nei riguardi del flip-flop i^{mo} sono significative tutte e sole le configurazioni degli ingressi che si verificano contemporaneamente all'impulso di trigger su T_i.

Il progetto di un contatore asincrono comprende anche la scelta del trigger per il flip-flop i^{mo}: si può usare l'uscita di ogni flip-flop precedente, durante il suo passaggio da 1 a 0: se si sceglie quella (K) che presenta il minor numero di transizioni $1 \rightarrow 0$, è possibile semplificare al massimo la rete logica che comanda J_i e K_i. Per quanto detto sopra, in-

fatti, si debbono prendere in considerazione i soli valori di y_i' contemporanei alle transizioni da 1 a 0 di y_k (o di $\overline{y}_k$).

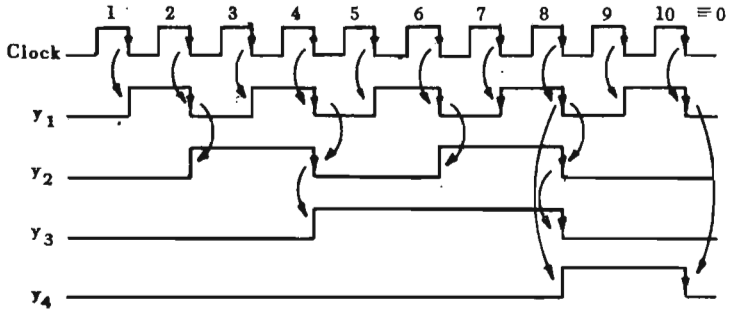


Fig.VIII.17 - Tensioni d'uscita di un contatore decimale asincrono.

Per realizzare un contatore decimale asincrono, cominciamo col tracciare il diagramma temporale delle sue uscite, per scegliere su di esso il trigger di ogni flip-flop (fig.VIII.17).

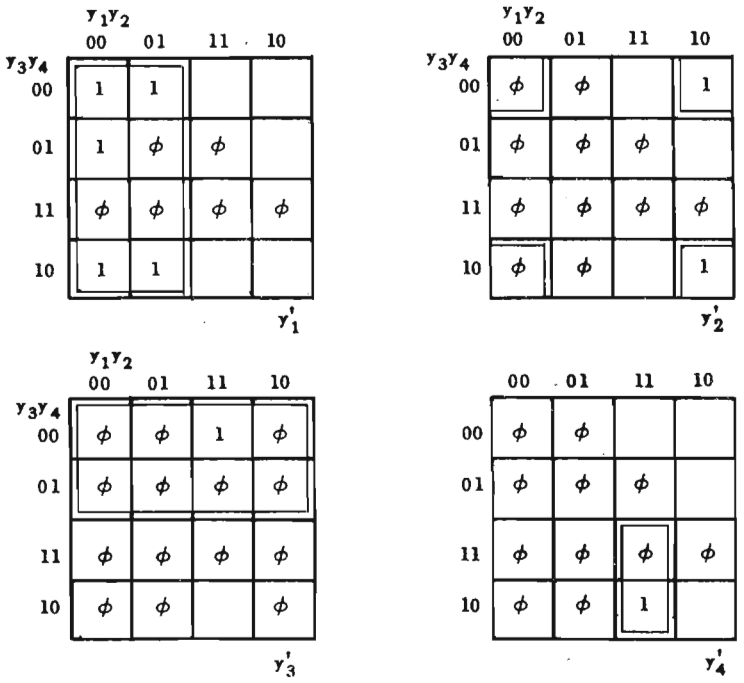


Fig.VIII.18 - Mappe di Karnaugh per le uscite di un contatore decimale asincrono.

Seguendo il criterio di far comandare ogni flip-flop da quello che lo precede e che presenta il minor numero di transizioni $1 \rightarrow 0$, conviene far pilotare y_2 da y_1 e y_3 da y_2; y_4 non può dipendere da y_3 né da y_2, perché questi segnali non presentano fronti di discesa al decimo impulso di clock, pertanto deve essere comandato da y_1 (quest'ultimo, a sua volta, è pilotato dal clock, come in tutti i contatori asincroni).

Nella fig.VIII.18 sono riportate le mappe di Karnaugh delle funzioni y_i'.

La mappa per y_i' è uguale a quella della fig.VIII.15, perché y_1' è sempre comandato dal clock. La mappa per y_2' contiene valori 0 e 1 solo durante i fronti negativi di y_1, quindi in corrispondenza al 4°, 8°, 10° impulso (valori 0) e al 2° e 6° impulso (valori 1); sono indifferenti i valori di y_2' in corrispondenza agli impulsi dispari (questa indifferenza va intesa nel senso che, non variando le condizioni d'ingresso per l'assenza del trigger, il flip-flop rimane nel suo stato precedente, qualunque siano i valori di J e K). In modo analogo sono costruite le mappe y_3' e y_4'.

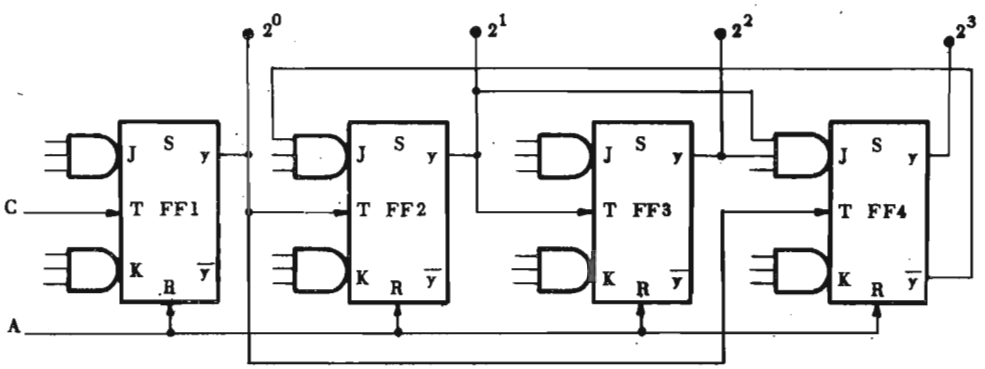


Fig.VIII.19 - Contatore decimale asincrono.

Sfruttando opportunamente le condizioni di indifferenza, si ricava:

$$\begin{cases} y_1' = \overline{y}_1 + 0 \cdot y_1 \\ y_2' = (\overline{y}_4)\,\overline{y}_2 + 0 \cdot y_2 \end{cases} \qquad \begin{cases} y_3' = \overline{y}_3 + 0 \cdot y_3 \\ y_4' = (y_2 y_3)\,\overline{y}_4 + 0 \cdot y_4 \end{cases}.$$

Quindi, essendo:

$$\begin{cases} J_1 = 1 \\ K_1 = 1 \end{cases} \quad \begin{cases} J_2 = \overline{y}_4 \\ K_2 = 1 \end{cases} \quad \begin{cases} J_3 = 1 \\ K_3 = 1 \end{cases} \quad \begin{cases} J_4 = y_2 y_3 \\ K_4 = 1 \end{cases}$$

si ottiene il circuito della fig.VIII.19.

VIII.3.5 - Contatori ad anello.

Si chiamano *contatori ad anello (Ring Counter)* quei contatori modulo n realizzati con n flip-flop. Un contatore ad anello realizzato con flip-flop JK è mostrato nella fig.VIII.20 a (per n = 4).

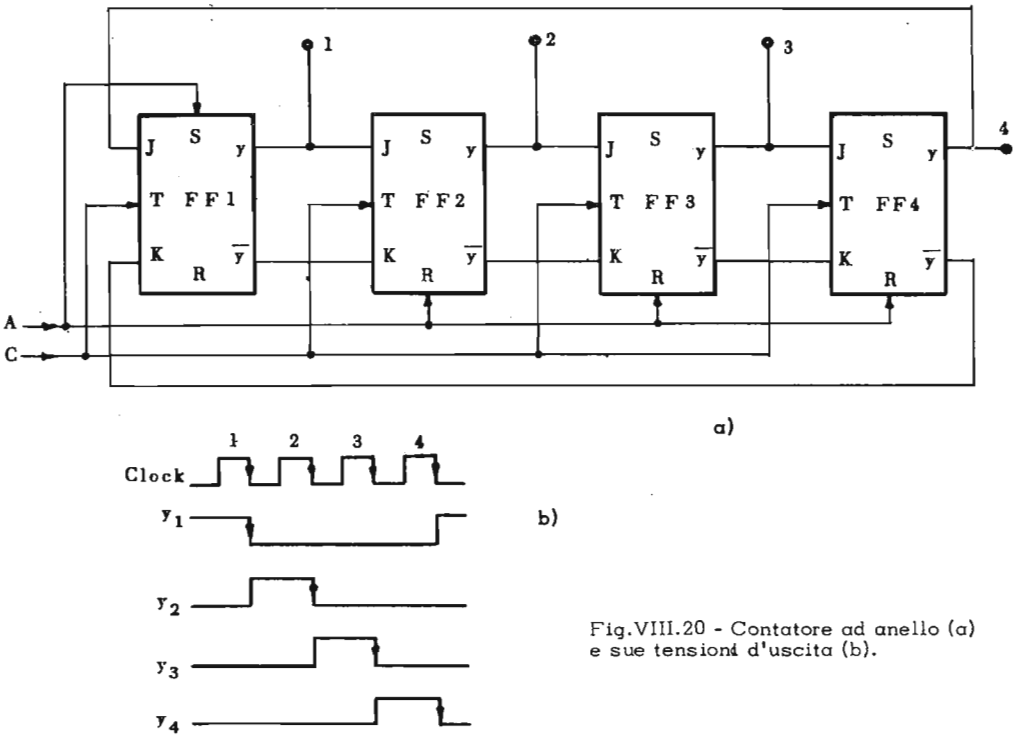


Fig.VIII.20 - Contatore ad anello (a) e sue tensioni d'uscita (b).

Le configurazioni del contatore sono quelle in cui uno e un solo flip-flop (FF_i) si trova nello stato 1. All'arrivo di ogni impulso il flip-flop FF_i assume lo stato di FF_{i-1} (e FF_1 quello di FF_n). Il flip-flop FF_{j+1} passa così nello stato 1 e FF_j torna a 0. Non esistendo nessuna configurazione in cui tutti i flip-flop sono a 0, il circuito di azzeramento deve mettere a 1 il flip-flop FF_1, secondo quanto indicato dal diagramma temporale delle tensioni d'uscita (fig.VIII.20 b).

Il contatore ad *anello intrecciato* (fig.VIII.21 a per n =4) è un contatore ad anello in cui il primo e l'ultimo flip-flop vengono connessi in modo che – ad ogni impulso – FF_1 assuma lo stato opposto di FF_n.

Esiste uno stato *di riposo* in cui tutti i flip-flop sono a 0; al primo impulso di clock, il generico FF_i prende lo stato di FF_{i-1}, cioè rimane a 0, ma FF_1 passa a 1. Al secondo impulso, FF_1 rimane a 1 ed FF_2 – assumendo lo stato di FF_1 – passa a 1; tutti gli altri flip-flop rimangono a zero. Dopo n impulsi, tutti i flip-flop si trovano ad 1; all'impulso seguente, FF_1 passa a 0.... Lo stato 0 si comunica, successivamente, a tutti i flip-flop, e il contatore ritorna nello stato iniziale dopo $2 \cdot n$ impulsi (v. fig.VIII.21 b). un contatore ad anello intrecciato con n flip-flop conta – in definitiva – 2 n impulsi.

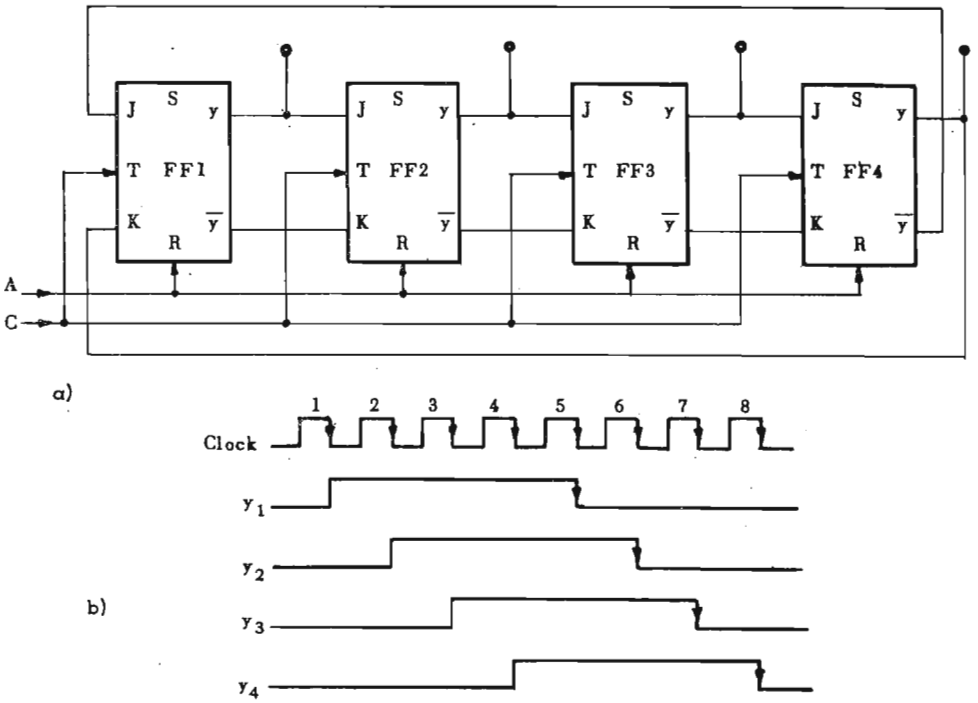


Fig.VIII.21 - Contatore ad anello intrecciato (a) e sue tensioni di uscita (b).

VIII.3.6 - Decodifica di un contatore.

Si chiamano *decodificatori* quei circuiti che danno un segnale su una di M uscite in corrispondenza ad ognuna delle M configurazioni in cui può trovarsi un contatore modulo M. I decodificatori si realizzano con elementi logici di tipo combinatorio.

Ad esempio, decodificare un contatore che conta da 0 a 3 (cioè con M = 4) significa collegarne le uscite y_1, y_2, $\overline{y_1}$, $\overline{y_2}$ agli ingressi di un circuito multiterminale come quello di fig.VIII.22: l'uscita 1 indica – a meno del modulo – l'arrivo di 0, 1, 2, 3 impulsi.

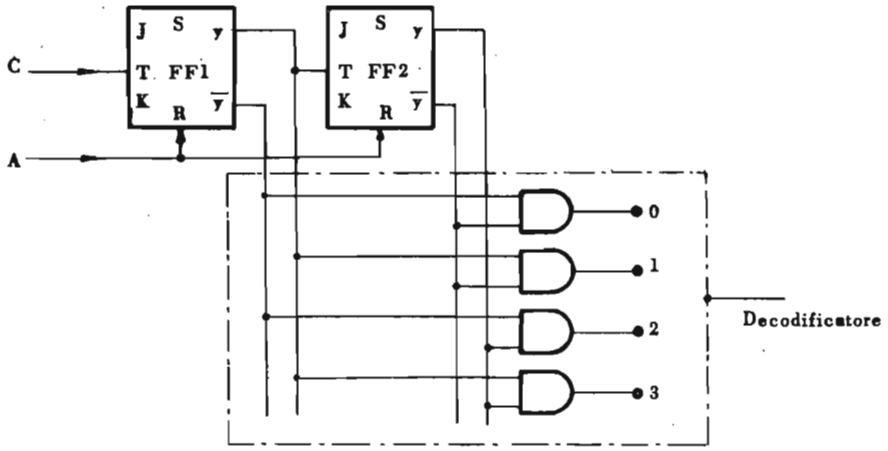


Fig.VIII.22 - Decodificatore di un contatore M = 4.

Come per decodificare 4 stati occorrono 8 diodi, per decodificarne $M = 2^n$ ne occorrono $n \cdot 2^n$: nessuna semplificazione è infatti possibile, tranne l'uso delle matrici per $n > 3$ (v. par.IV.13).

Se il contatore ha un modulo $M < 2^n$, si possono sfruttare gli stati indifferenti per semplificare il decodificatore. Ad esempio, nella figura VIII.23b è mostrato un contatore modulo 5 con relativo decodificatore: quest'ultimo è stato realizzato ricavando dalla mappa di Karnaugh di figura VIII.23a – che mostra le configurazioni delle uscite del contatore dopo l'arrivo di 1, 2, ..., 5 impulsi – le equazioni:

$$1 = y_1\overline{y_2} \qquad 3 = y_1y_2 \qquad 5 = 0 = \overline{y_1}\,\overline{y_2}\,\overline{y_3} \quad .$$

$$2 = \overline{y_1}y_2 \qquad 4 = y_3$$

Quando il contatore è realizzato con un numero di flip-flop maggiore di quello strettamente necessario, la sua decodifica diventa molto più semplice. Così, per decodificare un contatore modulo M ad anello intrecciato occorrono soltanto M porte AND a 2 ingressi, mentre un contatore ad anello ha già sulle uscite i segnali decodificati.

Considerando il complesso contatore-decodificatore come un circuito sequenziale ad un ingresso (il clock) ed M uscite, si verifica in definitiva quanto già visto per tutti i circuiti sequenziali: la complessità della rete di memoria (cioè del contatore) è compensata dalla semplicità della rete logica (cioè del decodificatore), e viceversa.

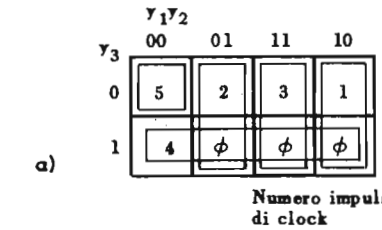
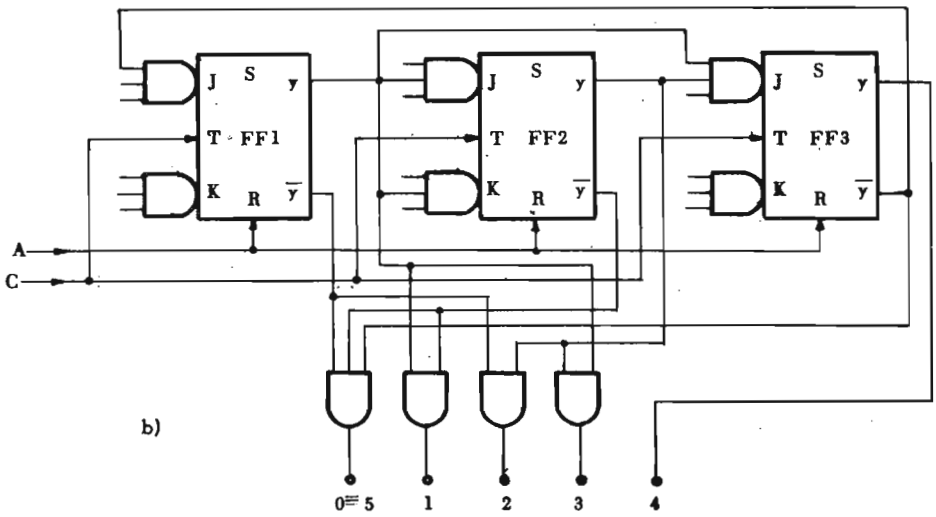


Fig.VIII.23 - Contatore modulo 5 e suo circuito decodificatore.



VIII.4 - Generazione di segnali periodici.

Nella progettazione dei circuiti digitali è spesso richiesta la generazione di segnali periodici a partire da un segnale di clock. Di seguito descriveremo i casi più comuni in pratica.

Un segnale rettangolare a frequenza sottomultipla, secondo una potenza di 2, della frequenza di clock, si ottiene prelevando il segnale stesso all'uscita di un opportuno contatore binario. Ad esempio, sulle u-

scite $y_1 y_2 y_3 y_4$ del contatore della fig.VIII.8 sono immediatamente disponibili segnali a frequenza 1/2, 1/4, 1/8, 1/16 della frequenza di clock (v. fig.VIII.9).

Combinando le uscite dei vari flip-flop di un contatore è possibile ricavare dei segnali a livelli (S) aventi una qualsiasi configurazione periodica, con periodo e durate dei segnali multipli del periodo di clock. Per questo si adopera un contatore con modulo uguale al numero degli impulsi di clock dopo il quale si ripete la configurazione degli S, nel modo illustrato dal seguente esempio.

Esempio: *Realizzare il segnale S della fig.VIII.24b, a partire dal clock di fig.VIII.24a.*

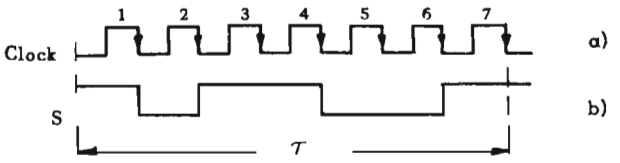


Fig.VIII.24 - Segnale S e relativo segnale di clock.

Il periodo T di S è uguale a 7 impulsi di clock: si può allora ottenere il livello S all'uscita di una rete combinatoria alimentata dalle uscite y_1, y_2, y_3 di un contatore *modulo* 7, imponendo che la funzione S sia uguale a 0 durante le configurazioni assunte dal contatore dopo 1, 4, 5 impulsi, e uguale a 1 per le restanti configurazioni. Per realizzare questa rete, costruiamo una mappa di Karnaugh (fig.VIII.25a) che lega le configurazioni delle y al numero degli impulsi di clock, e imponiamo quindi le condizioni volute (fig.VIII.25b). Si ricava così:

$$S = y_2 + \overline{y_1}\,\overline{y_3} \quad ,$$

quindi il circuito di fig.VIII.25c (il contatore modulo 7 è ottenuto coi metodi già esposti).

Da un segnale a livelli del tipo precedentemente descritto si possono ottenere configurazioni periodiche di impulsi, alimentando un AND a 2 ingressi con un segnale S di forma opportuna e con il clock.

A titolo d'esempio, nella fig.VIII.26 è mostrata la costruzione di un circuito per ottenere configurazioni periodiche di 5 impulsi in coincidenza con il 1°, 3°, 4°, 7°, 8° impulso di ogni gruppo di 8 impulsi di clock (fig.VIII.26a). Il segnale S, che ha un periodo di 8 impulsi di clock, sarà fornito da un contatore modulo 8, e sarà uguale a 1 dopo gli impulsi 2, 3, 6, 7, 8 e uguale a 0 dopo gli impulsi 1, 4, 5. In questo modo, in-

fatti, usando la logica positiva, il prodotto di S per C dà per risultato la configurazione I.

Dalle mappe di Karnaugh (fig.VIII.26b e c) si ottiene:

$$S = y_2 + \overline{y}_1\overline{y}_3 \quad .$$

Il circuito, realizzato con un contatore asincrono, è riportato nella fig.VIII.26d.

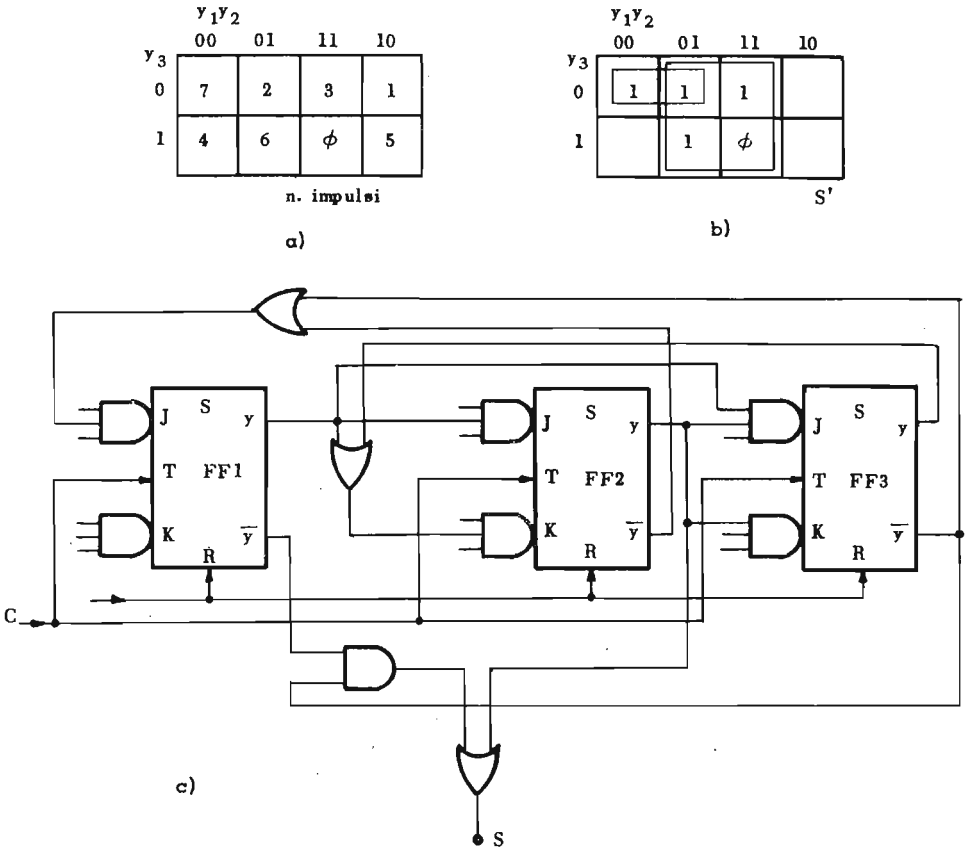


Fig.VIII.25 - Circuito per generare il segnale S della fig.VIII.24.

Quando sono richiesti degli impulsi a intervalli di tempo Δ che non sono in rapporto col periodo di clock, si usa una linea di ritardo e un amplificatore-riformatore di impulsi, secondo lo schema della figura VIII.27a: un impulso inviato sull'ingresso I continua a circolare indefinitamente, producendo una sequenza di segnali impulsivi sfasati di Δ secondi (fig.VIII.27b).

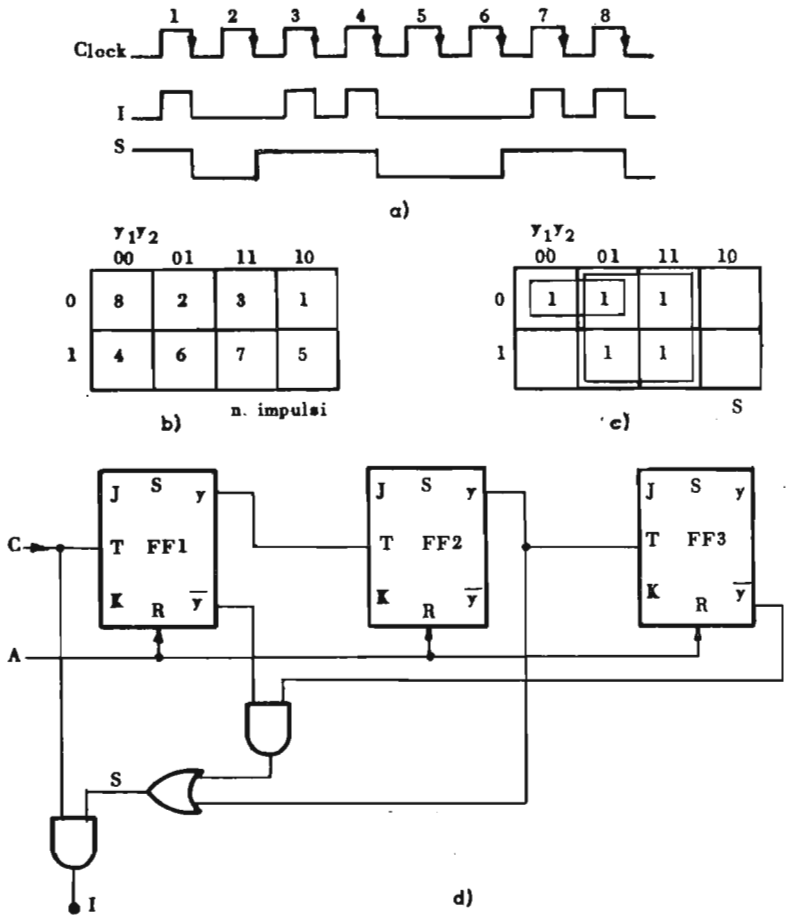


Fig.VIII.26 - Circuito per una configurazione periodica di impulsi.

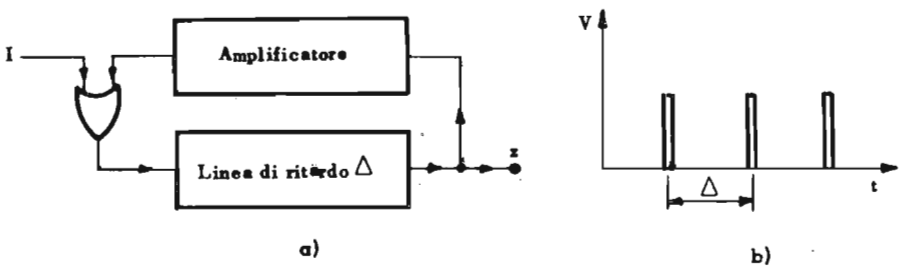


Fig.VIII.27 - Circuito per generare una sequenza di segnali impulsivi.

Con linee di ritardo diverse, o con un'unica linea a uscite multiple, si può infine generare una sequenza di impulsi sfasati di $\Delta_1\Delta_2...\Delta_k$ secondi. A titolo d'esempio, nella fig.VIII.28a è mostrato il circuito che genera la sequenza di 5 impulsi ritardati di $\Delta_1\Delta_2\Delta_3\Delta_4$ secondi della fig.VIII.28b.

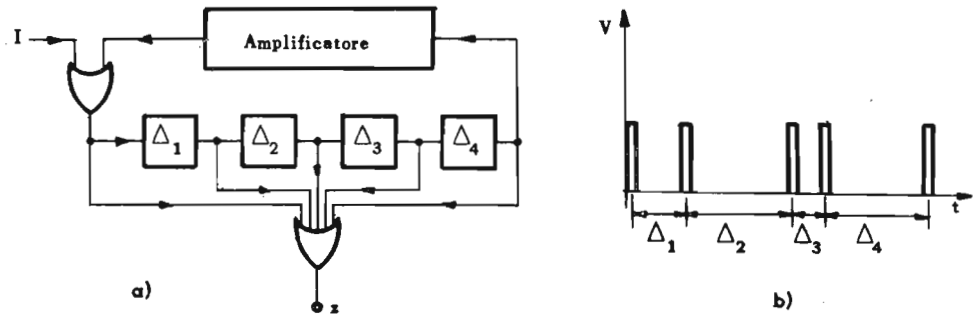


Fig.VIII.28 - Circuito per la generazione di una sequenza di segnali impulsivi non equidistanti.

VIII.5 - Selezione, istradamento e trasformazione delle informazioni.

Il flusso di informazioni tra i blocchi che costituiscono un'apparecchiatura digitale complessa deve essere regolato da opportuni organi che, comandati, da appositi *segnali di controllo*, inviano un'informazione su un certo canale, o collegano un canale a un determinato segnale o, infine, operano una trasformazione serie-parallelo o parallelo-serie delle informazioni immagazzinate in un organo di memoria.

Per *istradamento* di un segnale S di tipo qualsiasi si intende la possibilità di far avanzare il segnale S, che si presenta in un punto P da cui si diramano K linee, su una sola di tali linee, determinata dai valori di n variabili di controllo $C_1C_2...C_n$, essendo $2^{n-1} < K \leq 2^n$. A questo scopo, si invia il segnale S in parallelo sugli AND finali di una matrice per le variabili $C_1C_2...C_n$. Per ogni ennupla di valori degli C, S passerà attraverso l'unico AND i cui restanti ingressi sono tutti a 1. Naturalmente, vanno aggiunti alla matrice 2^n diodi, per creare i 2^n ingressi di S sugli AND. A titolo d'esempio, nella fig.VIII.29 è mostrato il circuito per istradare un segnale su una di 4 direzioni: ad ogni valore di C_1 e C_2, S passa su una delle uscite: la prima (terminale 0) per $C_1C_2 = 00$, la seconda (terminale 1) per $C_1C_2 = 01$, la terza (terminale 2) per $C_1C_2 =$

= 10 e la quarta (terminale 3) per $C_1C_2 = 11$. C_1 e C_2 sono, in genere, prelevati da un contatore.

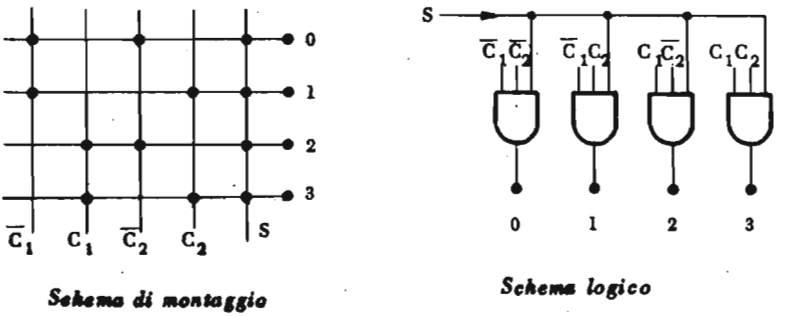


Fig.VIII.29 - Circuito per istradare un segnale su una di 4 direzioni.

Per *selezione* di un segnale S_j da K segnali $S_1S_2...S_k$ presenti contemporaneamente su altrettante linee convergenti in un punto, si intende l'operazione che blocca tutti i segnali $S_i \neq S_j$ e fa passare S_j su una linea l uscente da P. La selezione si realizza inserendo in P un circuito a K ingressi e un'uscita che, per ogni valore di n variabili di controllo $C_1C_2...C_n$ ($2^{n-1} < K \leq 2^n$), si collega a uno degli ingressi. Praticamente, si mandano i segnali S_i sugli AND finali di una matrice per le variabili $C_1C_2...C_n$, e si collegano le uscite della matrice stessa agli ingressi di un OR. Per ogni ennupla di valori degli C_i, entra sulla linea

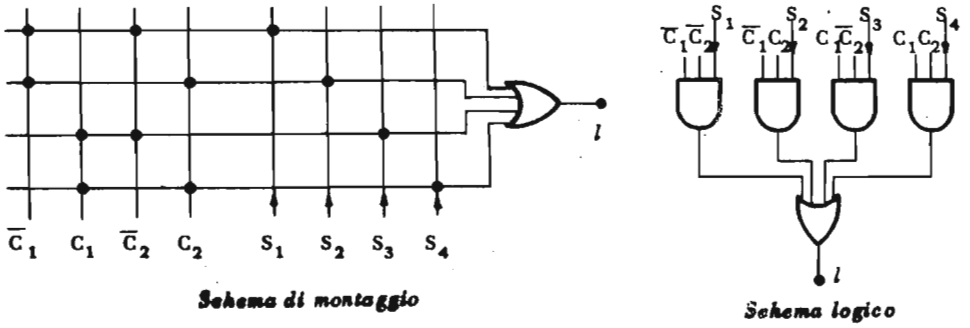


Fig.VIII.30 - Circuito per selezionare un segnale.

il solo segnale presente sull'AND i cui restanti ingressi sono ad 1. Naturalmente si dovranno aggiungere alla matrice 2^n diodi, per consentire lo ingresso degli S_i sui relativi AND, nonché 2^n diodi per l'OR d'uscita.

Nella fig.VIII.30 è mostrato un circuito per selezionare uno dei segnali $S_1S_2S_3S_4$; ad ogni valore di C_1 e C_2 un solo S_i passa su l: S_1 per $C_1C_2 = 00$, S_2 per $C_1C_2 = 01$, S_3 per $C_1C_2 = 10$ e S_n per $C_1C_2 = 11$.

I circuiti di selezione e istradamento si realizzano anche a transistor: in generale, con 2^n NOR per n variabili. Nella fig.VIII.31 è disegnato un circuito istradatore a 4 vie.

La trasformazione di un'informazione in parallelo in un flusso di bit in serie (*trasformazione parallelo-serie*) si può realizzare con diversi tipi di circuiti. I più comuni, oltre al registro con ingresso in parallelo e uscita in serie, sono realizzati con selettori e contatori, o con elementi di ritardo.

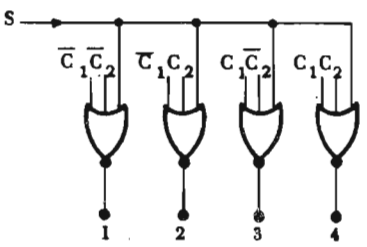


Fig.VIII.31 - Circuito istradatore realizzato a NOR.

Nella fig.VIII.32a è mostrato un circuito con un selettore e un contatore modulo 4, per inoltrare sul terminale B, in 4 istanti successivi determinati da altrettanti impulsi di clock, i bit $B_1 \ldots B_4$ presenti contemporaneamente sui terminali 1...4.

Dopo un impulso A di azzeramento, un impulso S di scrittura (figura VIII.32 b) registra il bit B_i nella cella C_i. In questo istante $y_1y_2 = 00$, quindi l'uscita della prima cella si trova sul terminale B. Dopo un tempo τ, si manda al contatore un impulso di clock: si ha allora $y_1y_2 = 10$ e il contenuto della seconda cella passa su B. Altri 2 impulsi di clock completano, poi, l'uscita in serie dei bit.

Nella fig.VIII.32 a il contatore e il registro sono stati disegnati come 2 blocchi. Questa rappresentazione, introdotta qui e in seguito per ragioni di semplicità, è in linea con le più avanzate tecniche elettroniche, che tendono a realizzare circuiti complessi di uso frequente con un solo circuito integrato. Nel campo dei circuiti digitali si trovano oggi – in un unico contenitore – shift-register, contatori binari e decimali, addizionatori e persino memorie a parecchi bit coi relativi circuiti di lettura e scrittura.

La conversione parallelo-serie con elementi di ritardo è realizzata con circuiti simili a quello di fig.VIII.33 a. Nei punti $A_1 \ldots A_5$ di una linea di ritardo vengono immessi – all'istante t_0 – i bit $B_1 \ldots B_5$; nello stesso istante, essendo A_1 collegato al terminale B, il bit B_1 è disponibile in uscita.

All'istante $t_0 + \Delta$, B_2 sostituisce B_1 in A_1, cioè compare sul terminale B... .

All'istante $t_0 + 4\Delta$ inizia l'uscita del bit B_5, che si mantiene sul terminale B fino all'istante $t_0 + 5\Delta$. Un impulso di scrittura S manda poi 5 nuovi bit nei punti $A_1 \ldots A_5$. Nella fig.VIII.33 b è riportato il diagramma temporale del circuito (si è posto $B_1 \ldots B_5 = 10010$).

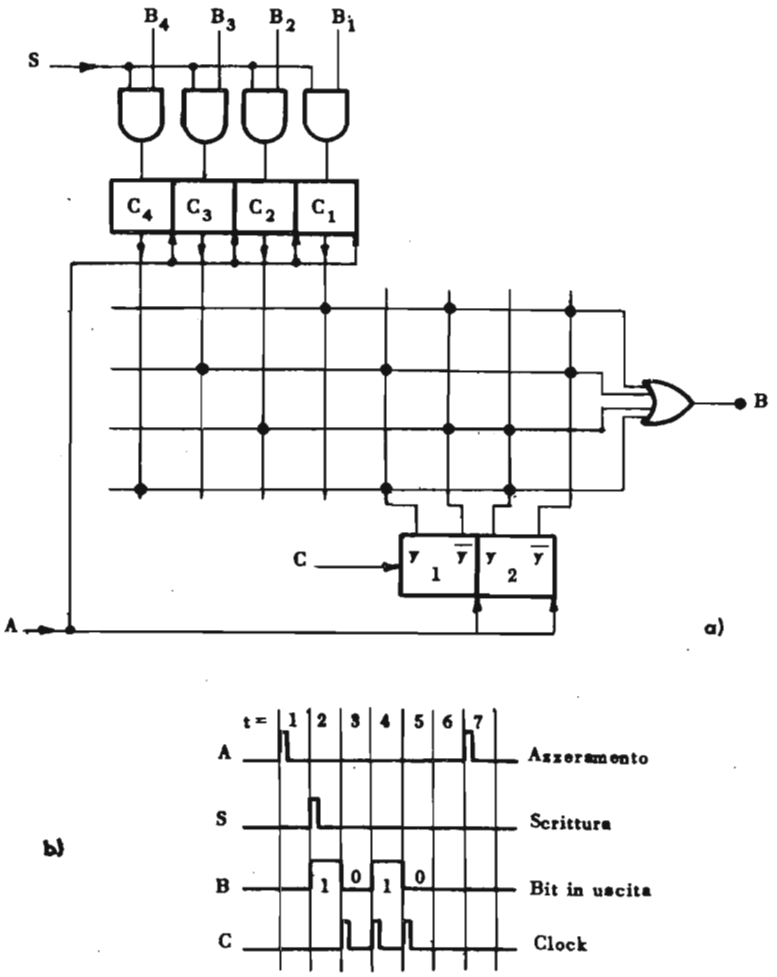


Fig.VIII.32 - Conversione parallelo-serie con selettore e contatore.

La trasformazione di un flusso di bit in serie in un'informazione in parallelo (*trasformazione serie-parallelo*) avviene in modo complementare alla trasformazione parallelo-serie. Vengono adoperati ancora registri *dual-mode*, circuiti con istradatori e contatori, o elementi di ritardo.

La conversione serie-parallelo con istradatori e contatori non presenta aspetti particolarmente nuovi. Nella fig.VIII.34 è mostrato – a ti-

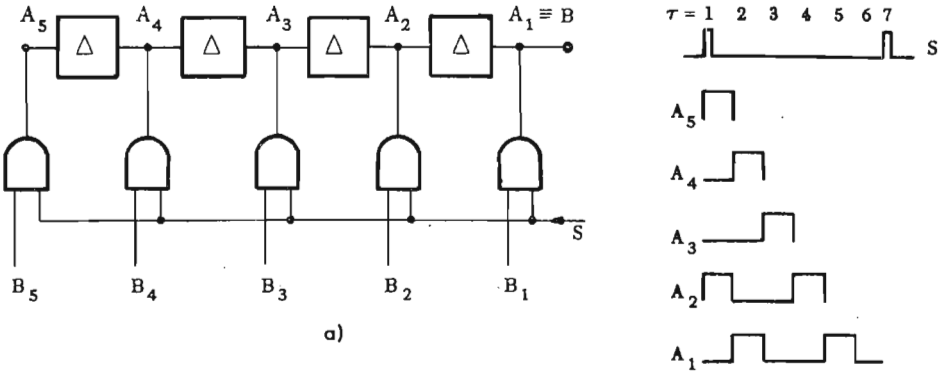


Fig.VIII.33 - Conversione parallelo-serie con elementi di ritardo.

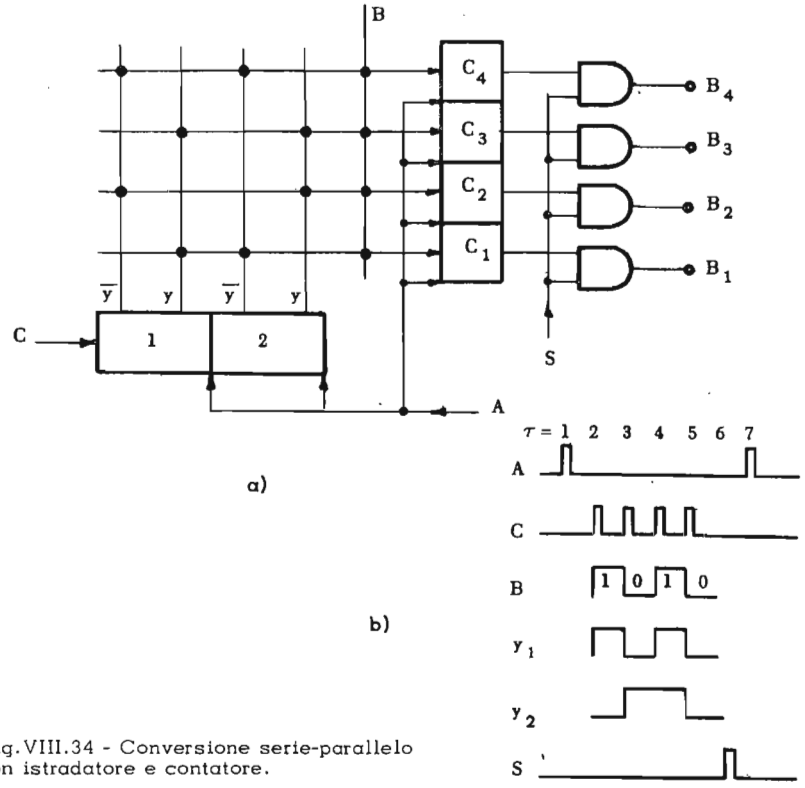
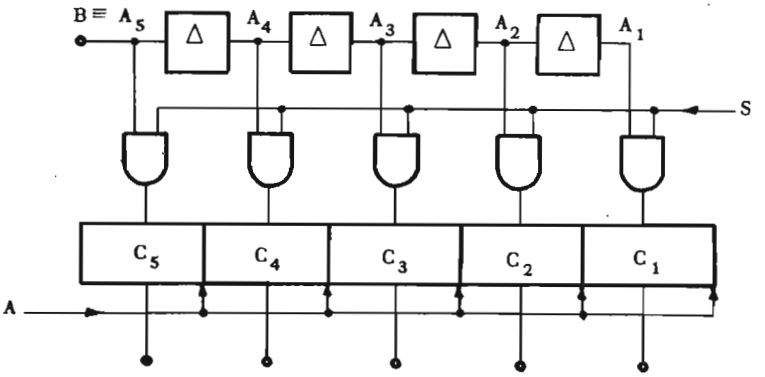


Fig.VIII.34 - Conversione serie-parallelo con istradatore e contatore.

tolo d'esempio – il circuito e il diagramma temporale per un convertitore serie-parallelo a 4 bit: il circuito di istradamento smista il segnale B – successivamente – su $C_1C_2C_3C_4$, a seconda degli stati in cui gli impulsi di clock lasciano il contatore modulo 4. I bit immagazzinati nelle celle $C_1 \dots C_4$ vengono poi presentati sui terminali $B_1 \dots B_4$ da un impulso S.



La conversione serie-parallelo con elementi di ritardo è realizzata inviando i bit che giungono sul terminale B all'ingresso di una linea di ritardo con tanti elementi quanti sono i bit del segnale meno 1. Ogni elemento provoca un ritardo Δ eguale a quello con cui si succedono i bit sull'ingresso (figura VIII.35 a). Il bit B_1 arriva all'istante t_0 nel punto A_5 e vi rimane per un tempo Δ. All'istante $t_0+\Delta$, B_1 passa in A_4 e B_2 arriva in A_5: questa situazione permane fino all'istante $t_0 + 2\Delta$, in cui B_1 passa in A_3, B_2 in A_4 e B_3 in A_5.... All'istante $t_0 + 4\Delta$, in cui B_1 si trova in A_1, B_2 in $A_2 \dots B_5$ in A_5, si invia un impulso di scrittura S su tutti gli AND, per registrare il bit B_i nella cella C_i; nella figura VIII.35 b è riportato il diagramma temporale dei segnali nei punti $A_1 \dots A_5$ (si è posto $B_1 \dots B_5 = 10010$).

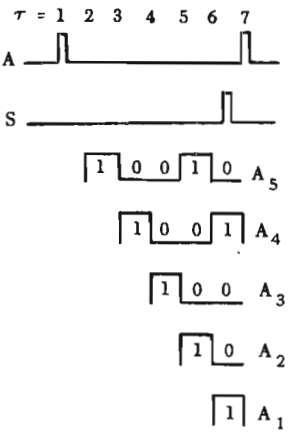


Fig. VIII.35 - Conversione serie-parallelo con elementi di ritardo.

VIII.6 - Applicazioni: circuiti per il controllo delle informazioni nei sistemi di trasmissione digitali.

Le tecniche descritte nei paragrafi precedenti, e la conoscenza di quelle relative ai circuiti combinatori, permettono di affrontare e risolvere i problemi che si presentano nella progettazione di apparecchiature digitali non eccessivamente complesse.

Per mostrare alcune applicazioni dei concetti introdotti in questo capitolo, tratteremo – a solo titolo d'esempio – i circuiti che verificano l'esattezza di un messaggio ricevuto in un generico sistema di trasmissione di informazioni digitali nonché l'insieme dei dispositivi che compongono l'unità aritmetica di un piccolo calcolatore; descriveremo i circuiti del primo tipo in questo paragrafo, gli altri nel seguente.

Abbiamo visto, nel primo capitolo, come i messaggi alfanumerici vengano codificati secondo un codice, e come – per salvaguardare i messaggi stessi da eventuali distorsioni che possono presentarsi durante la trasmissione – vengono introdotti, tra i loro bit, alcuni bit di ridondanza. Si chiamano *circuiti di controllo* quei circuiti che verificano – in ricezione – la presenza delle ridondanze introdotte, e danno un allarme quando non trovano le ridondanze stesse. I controlli che vengono eseguiti sono sostanzialmente di due tipi: di parità e di peso, e dipendono, ovviamente, dal codice usato. Un terzo tipo di controllo, quello di *identità*, viene infine compiuto quando la ridondanza del messaggio consiste semplicemente nella sua completa ripetizione. Esamineremo i tre casi, indicando qualcuna delle numerose soluzioni adottate in pratica. Non faremo alcuna ipotesi sulle modalità di trasmissione, ma supporremo che dal ricevitore i messaggi escano come livelli di durata costante ed uguale al periodo degli impulsi di clock presenti nel sistema.

VIII.6.1 - Controllo di parità.

Il più semplice circuito di questo tipo è quello che effettua un solo controllo di parità su tutti gli n bit del messaggio. I segnali arrivano a un registro a scorrimento collegato al ricevitore e vengono inviati in parallelo (fig. VIII.36) anche a un flip-flop JTK. Il flip-flop riceve un impulso, e cambia di stato, quando l'uscita dell'AND A è a 1, cioè quando è 1 il segnale ricevuto. Se, quindi, il flip-flop è azzerato prima che inizi la ricezione del messaggio, il flip-flop stesso si trova nello stato 1 negli istanti in cui si sono presentati 1, 3, 5 ... bit. Quando l'intero messaggio è stato ricevuto, un impulso F sull'AND B collegato con l'uscita y del contatore provoca un segnale d'allarme se $y = 1$, cioè se è arrivato un numero dispari di bit 1.

Quando le verifiche da effettuare sono in numero di K ($K \geq 2$), si realizzano K circuiti separati, ognuno alimentato dai soli bit da controllare, e si raccolgono i loro

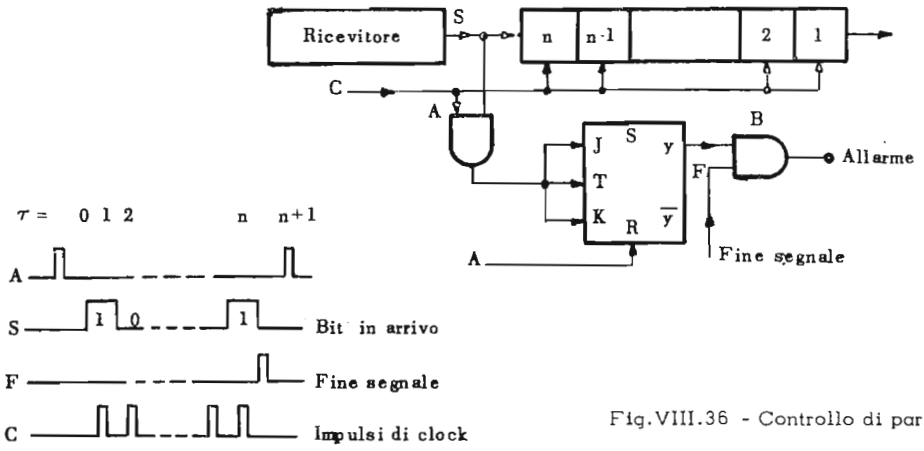


Fig.VIII.36 - Controllo di parità.

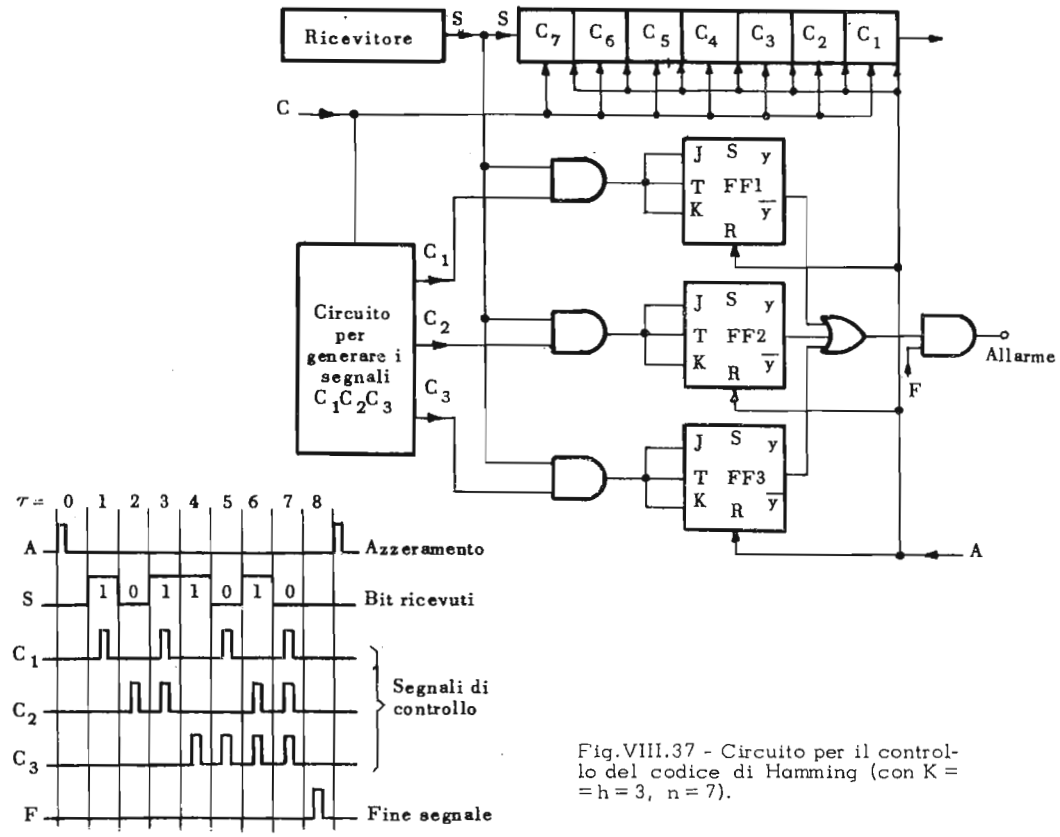


Fig.VIII.37 - Circuito per il controllo del codice di Hamming (con K = = h = 3, n = 7).

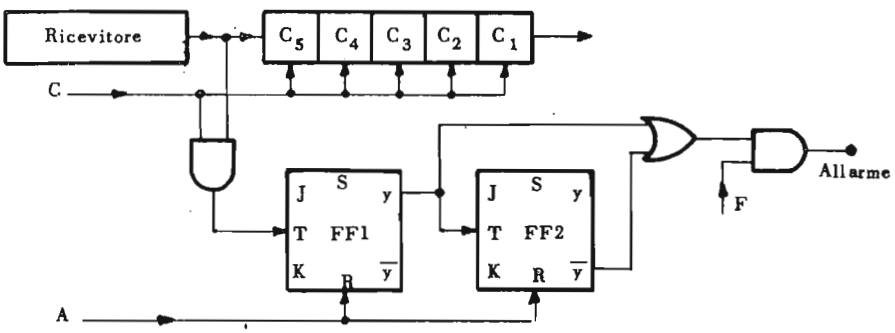
segnali d'uscita su un OR a K ingressi. Il segnale d'allarme viene prodotto da un AND, nel modo già visto. La fig.VIII.37 mostra un circuito per controllare un codice di Hamming con K = 3, h = 3, n = 7 (vedi cap.I). I controlli sono effettuati sui tre seguenti gruppi di bit:

1) bit 1, 3, 5, 7;
2) bit 2, 3, 6, 7;
3) bit 4, 5, 6, 7.

I segnali $C_1C_2C_3$ si ottengono dal segnale di clock, secondo quanto indicato nel par.VIII.4.

VIII.6.2 - Controllo del peso.

Nei codici a peso costanti (vedi cap.I) del tipo $\binom{m}{p}$ si deve verificare che p degli m bit ricevuti abbiano valore 1. Si può costruire, a questo scopo, un circuito derivato da quello della fig.VIII.36, sostituendo al contatore binario uno di modulo M. La scelta di M dipende dalla necessità di determinare con esattezza il peso p, scartando quei valori multipli di p, che un contatore con M = p non può distinguere; precisamente, deve essere M = p + 1, se $p \geq m/2$, e M = (m − p + 1) se $p \leq m/2$.



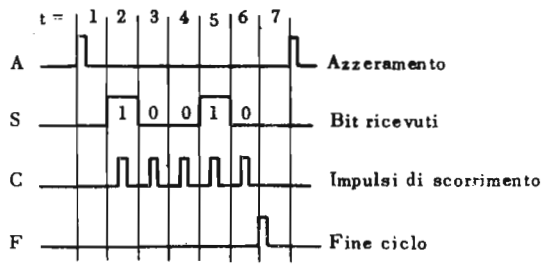


Fig.VIII.38 - Circuito per il controllo del codice $\binom{5}{2}$.

Supponiamo, ad esempio, di voler realizzare un circuito di controllo per un codice $\binom{5}{2}$. Se scegliamo un contatore con M = 2, non abbiamo la possibilità di distingue-

re l'uscita 2 dalle uscite 0 e 4, quindi i segnali di peso 2 da quelli di peso 0 e 4. Un contatore con M = 3 non distingue l'uscita 2 dalla 5; un contatore con M = 4 non determina invece equivoci nella durata di un ciclo. L'allarme deve essere dato se, alla fine dei 5 impulsi, il contatore non si trova nello stato 2, in cui $y_1 y_2 = 01$, ma in uno qualsiasi degli altri stati, cioè nella condizione:

$$\overline{\overline{y}_1 y_2} = y_1 + \overline{y}_2 \ .$$

Il circuito di controllo completo è disegnato nella fig.VIII.38.

VIII.6.3 - Controllo di identità.

Il confronto fra 2 bit B_1 e B_2 provenienti dalla doppia trasmissione di uno stesso bit B può avvenire in un circuito per la funzione logica *somma modulo 2* ($f = B_1 \oplus B_2 = B_1\overline{B}_2 + \overline{B}_1 B_2$).

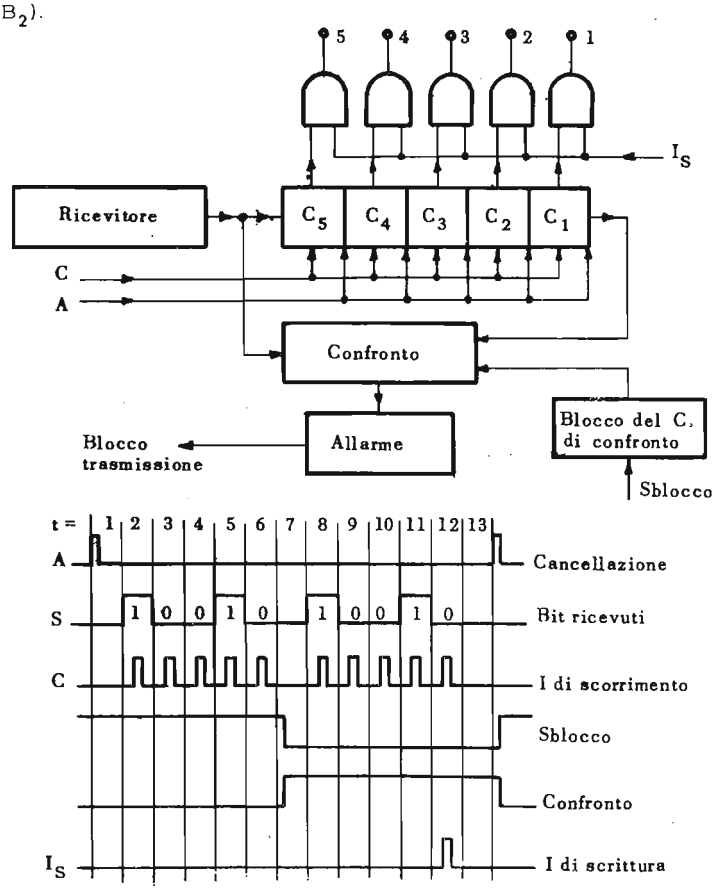


Fig.VIII.39 - Circuito per controllare l'identità di parole a 5 bit.

Quando uno stesso segnale viene trasmesso due volte, il circuito convertitore serie-parallelo in ricezione comprende anche un circuito di confronto, un circuito di blocco e uno di allarme, secondo lo schema circuitale e temporale della fig.VIII.39.

La prima ricezione del messaggio avviene in modo normale: per tutta la sua durata il circuito di confronto è bloccato e alla fine, benché tutte le informazioni si trovino registrate nelle celle C_i nel giusto ordine, non viene inviato l'impulso di scrittura.

Dopo una pausa, inizia la seconda trasmissione e ricezione del messaggio, e – – alla cella C_5 – arriva il bit $B_1^{(2)}$ (distinguiamo con gli apici la prima e la seconda trasmissione del bit B_i). L'impulso di scorrimento mette $B_1^{(2)}$ in C_5, $B_5^{(1)}$ in C_4... $B_2^{(1)}$ in C_1. $B_1^{(1)}$ esce dal registro ed entra nel circuito di confronto, che intanto è stato sbloccato dal circuito di blocco. Nel circuito di confronto passa anche $B_1^{(2)}$, e i 2 bit vengono paragonati. Se $B_1^{(1)} = B_1^{(2)}$, l'uscita del circuito di confronto è 0, se $B_1^{(1)} \neq B_1^{(2)}$, si ha un'uscita 1 che aziona il circuito di allarme e blocca la trasmissione. Al termine del secondo messaggio, i bit $B_i^{(2)}$, se sono risultati eguali ai $B_i^{(1)}$, si trovano immagazzinati nel registro, nella giusta posizione. Viene allora inviato l'impulso di scrittura: i bit passano nel circuito a valle, il registro viene azzerato e il ciclo ricomincia.

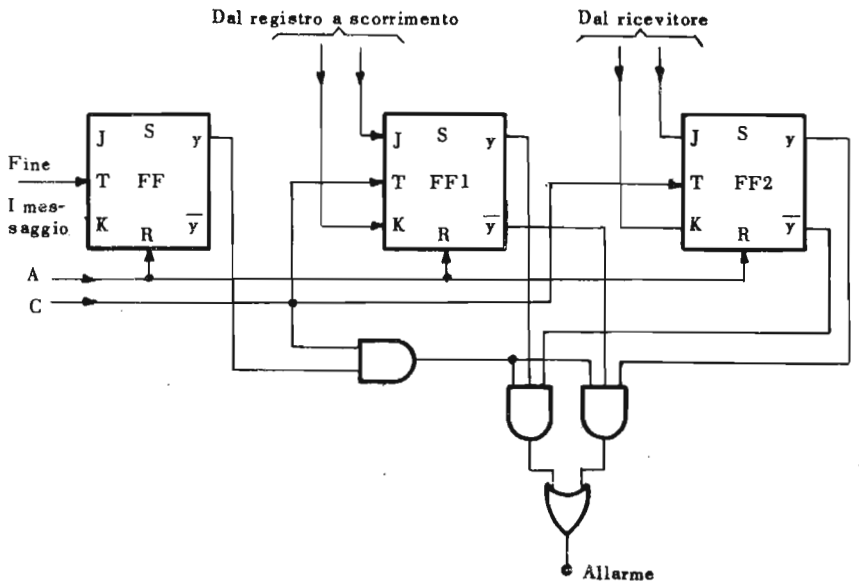
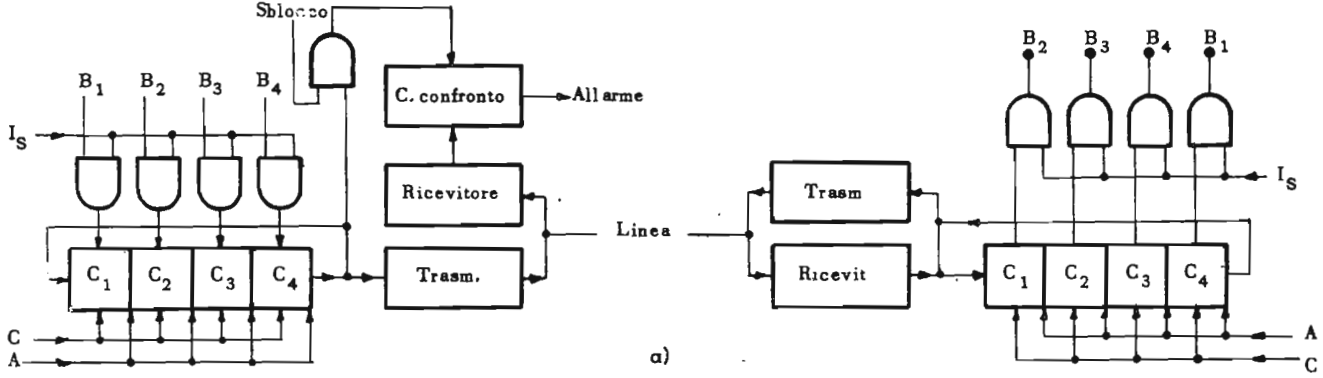


Fig.VIII.40 - Circuito di confronto, allarme e blocco.

I circuiti di confronto, allarme e blocco costituiscono, in realtà, un unico complesso, rappresentato nella fig.VIII.40. Il segnale di fine messaggio, non riportato nella fig.VIII.39 per semplicità, può essere fornito dal trasmettitore, o rilevato da un contatore dopo n impulsi in arrivo.



Tempi τ	Lato trasmittente								C. di confronto	Linea	Lato ricevente							Note
	Contenuto celle				A	C	I_s	Sblocco			Contenuto celle				A	C	I_s	
	C_1	C_2	C_3	C_4							C_1	C_2	C_3	C_4				
1	-	-	-	-	1	0	0	0	-	-	-	-	-	-	1	0	0	→ Trasmissione in avanti
2	B_1	B_2	B_3	B_4	0	0	1	0	-	$\overrightarrow{B_4}$	B_4	-	-	-	0	1	0	
3	B_4	B_1	B_2	B_3	0	1	0	0	-	$\overrightarrow{B_3}$	B_3	B_4			0	1	0	
4	B_3	B_4	B_1	B_2	0	1	0	0	-	$\overrightarrow{B_2}$	B_2	B_3	B_4	-	0	1	0	
5	B_2	B_3	B_4	B_1	0	1	0	0	-	$\overrightarrow{B_1}$	B_1	B_2	B_3	B_4	0	1	0	
6	B_1	B_2	B_3	B_4	0	1	0	1	$\overrightarrow{B_4} : \overleftarrow{B_4}$	$\overleftarrow{B_1}$	B_1	B_2	B_3	B_4	0	0	0	← Trasmissione indietro
7	B_4	B_1	B_2	B_3	0	1	0	1	$\overrightarrow{B_3} : \overleftarrow{B_3}$	$\overleftarrow{B_2}$	B_4	B_1	B_2	B_3	0	1	0	
8	B_3	B_4	B_1	B_2	0	1	0	1	$\overrightarrow{B_2} : \overleftarrow{B_2}$	$\overleftarrow{B_3}$	B_3	B_4	B_1	B_2	0	1	0	
9	B_2	B_3	B_4	B_1	0	1	0	1	$\overrightarrow{B_1} : \overleftarrow{B_1}$	$\overleftarrow{B_4}$	B_2	B_3	B_4	B_1	0	1	0	
10	-	-	-	-	1	0	0	1	-	-	-	-	-	-	0	0	1	

b)

Fig.VIII.41 - Circuito per il controllo di un codice asservito (a) e sua tabella degli stati (b).

VIII.6.3.1 - Doppia trasmissione di un messaggio nei due sensi.

La doppia trasmissione del messaggio per la protezione dagli errori, intesa in maniera diversa, può essere effettuata con una prima trasmissione dall'estremità trasmittente verso quella ricevente (trasmissione in avanti) e una seconda dall'estremità ricevente verso la trasmittente (trasmissione indietro). Durante quest'ultima trasmissione, avviene, bit per bit, la comparazione dei 2 messaggi: ed è l'estremità trasmittente che blocca i due circuiti, manda l'allarme e ritrasmette il segnale in caso di mancata identità di due bit qualsiasi. La doppia trasmissione di uno stesso messaggio nei due sensi, usata nei sistemi cosiddetti *asserviti*, può essere ottenuta col circuito di figura VIII.41 a (per messaggi di 4 bit). Il funzionamento temporale del circuito è dettagliatamente illustrato nella tabella della fig.VIII.41 b.

VIII.6.4 - Circuiti sequenziali sincroni con flip-flop JK.

Il metodo esposto nel par.VIII.3.3 può essere applicato anche ai circuiti sequenziali sincroni, e costituisce un'alternativa a quelli esposti nel capitolo VII.

Illustriamo con un esempio il metodo di sintesi.

Esempio: *Sintesi di un circuito per il controllo del codice BCD. Sull'ingresso di un circuito (fig.VIII.42) arriva una serie di cifre decimali codificate in binario, secondo il codice BCD (par.I.10.1); i 4 bit di ogni cifra arrivano in serie, a cominciare dal bit meno significativo. All'arrivo del bit di peso 8, si vuole un'uscita 1 se i bit ricevuti rappresentano uno dei numeri compresi tra 10 e 15 (cioè se non rappresentano una cifra decimale BCD).*

Fig.VIII.42 - Schema a blocchi di un circuito per la rivelazione di cifre BCD errate.

Iniziamo la sintesi col diagramma di Mealy (fig.VIII.43). A partire da uno stato iniziale 1, si possono avere 2 possibilità: l'arrivo del bit 1 e quello del bit 0. Per ognuno dei 2 stati raggiunti si hanno ancora 2 possibilità, e così via.

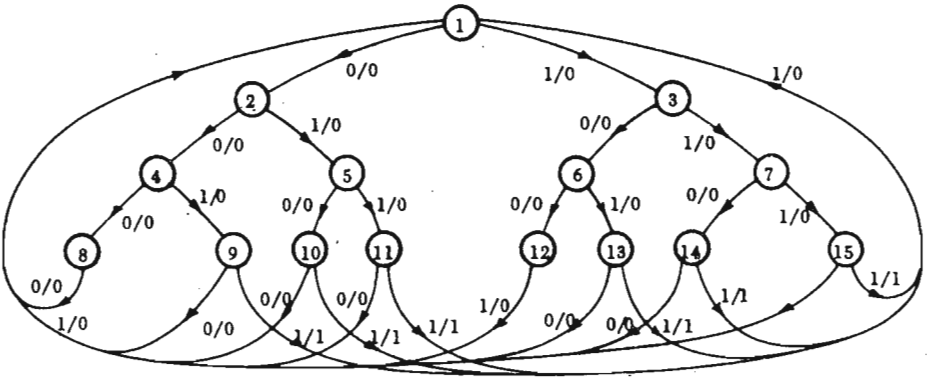


Fig.VIII.43 - Diagramma di Mealy dal circuito di fig.VIII.42.

Dal diagramma si ricava la seguente tavola di flusso:

Stato	x = 0	x = 1
1	2/0	3/0
2	4/0	5/0
3	6/0	7/0
4	8/0	9/0
5	10/0	11/0
6	12/0	13/0
7	14/0	15/0
8	1/0	1/0
9	1/0	1/1
10	1/0	1/1
11	1/0	1/1
12	1/0	1/0
13	1/0	1/1
14	1/0	1/1
15	1/0	1/1

i cui stati si riducono, applicando il metodo di minimizzazione, a 6. La tavola di flusso corrispondente alla macchina minima è:

Stato	x = 0	x = 1
1	2/0	2/0
2	3/0	4/0
3	5/0	6/0
4	6/0	6/0
5	1/0	1/0
6	1/0	1/1

Assegniamo ora una codifica arbitraria agli stati. Poiché gli stati sono 6, occorrono 3 variabili interne y: stabiliamo tra stati e variabili la corrispondenza seguente:

Stato (n)	Codifica $y_1\,y_2\,y_3$
1	000
2	100
3	110
4	101
5	010
6	001

La tavola di flusso può, a questo punto, essere trasformata in modo da mettere in evidenza le relazioni tra le variabili y_i, le y_i' e la z. (Da questa tabella, che va interpretata come le tabelle di verità delle funzioni y_1' e z in funzione delle y_i, possono essere ricavate le equazioni interne e l'equazione di uscita del circuito).

$y_1y_2y_3$	x = 0	x = 1
000	100/0	100/0
100	110/0	101/0
110	010/0	001/0
101	001/0	001/0
010	000/0	000/0
001	000/0	000/1

$y_1'y_2'y_3'/z$

Riportiamo, ora, su 4 mappe di Karnaugh (fig.VIII.44) le funzioni $y_1'y_2'y_3'z$:

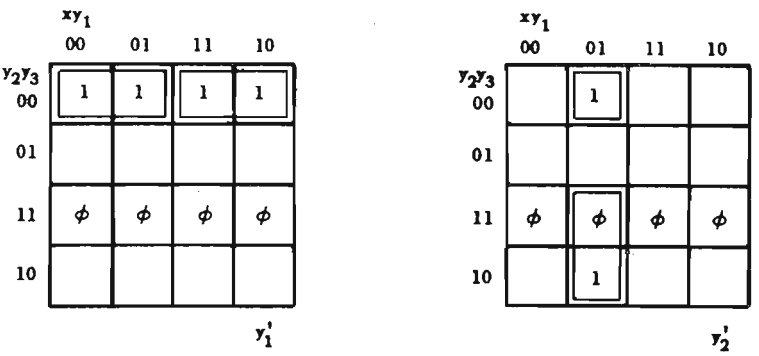


Fig.VIII.44

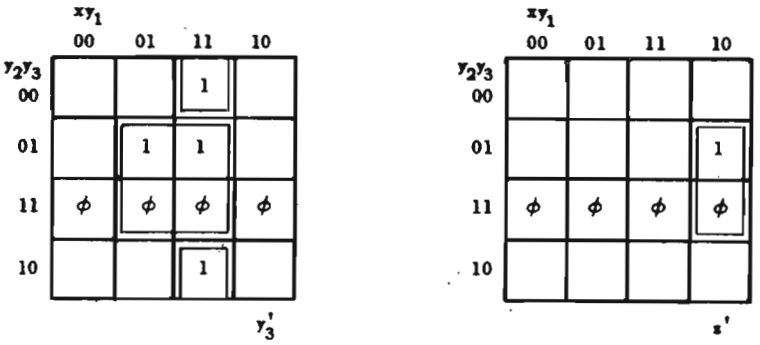


Fig.VIII.44 - Mappe di Karnaugh per le funzioni y'_1, y'_2, y'_3, z.

Dalle mappe, ricaviamo le equazioni delle y'_i e l'equazione della z:

$$y'_1 = y_1 (\overline{y_2}\,\overline{y_3}) + \overline{y_1} (\overline{y_2}\,\overline{y_3})$$

$$y'_2 = y_2 (\overline{x}\,y_1) + \overline{y_2} (\overline{x}\,y_1\overline{y_3})$$

$$y'_3 = y_3 (y_1) + \overline{y_3} (x\,y_1)$$

$$z = x\,\overline{y_1}\,y_3$$

dalle prime 3 equazioni, possiamo finalmente scrivere le espressioni di J_i e K_i:

$$\begin{cases} J_1 = \overline{y_2}\,\overline{y_3} \\ K_1 = y_2 + y_3 \end{cases} \qquad \begin{cases} J_2 = \overline{x}\,y_1\,\overline{y_3} \\ K_2 = x + y_1 \end{cases} \qquad \begin{cases} J_3 = x\,y_1 \\ K_3 = \overline{y_1} \end{cases}$$

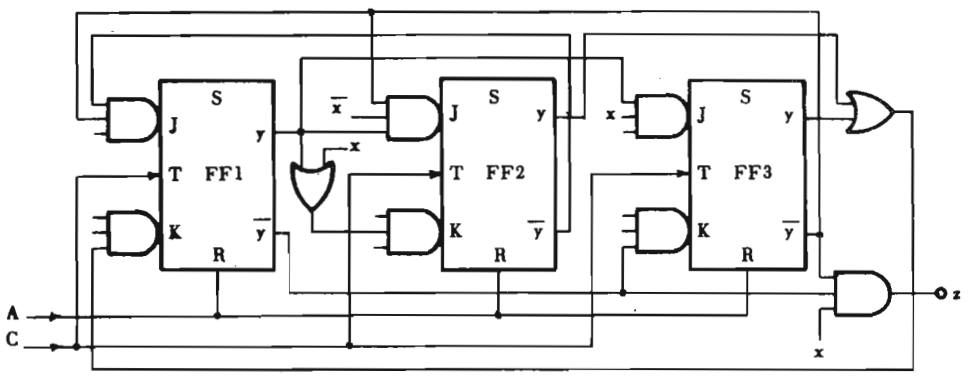


Fig.VIII.45 - Circuito sequenziale sincrono.

Nella fig.VIII.45 è disegnato il circuito (il terminale A serve per metterlo nello stato iniziale).

Sia chiaro che l'assegnazione degli stati è il punto più critico del progetto: non esiste nessun criterio di validità generale e, poiché ad assegnazioni diverse corrispondono circuiti diversi, occorre procedere per tentativi.

Dopo questo breve cenno sul progetto di un generico circuito sequenziale, passiamo ora ai più comuni circuiti dei sistemi digitali.

VIII.7 - Uso dei circuiti integrati nelle tecniche digitali.

In questo capitolo abbiamo usato flip-flop a circuiti integrati per la realizzazione di ogni tipo di circuiti.

E' bene avvertire, a questo punto, che tutti i costruttori realizzano oggi circuiti con funzioni molto più complesse di quelle viste nel cap.III. A titolo d'esempio, riportiamo nella tab.VIII.1 alcune caratteristiche di un contatore decimale asincrono realizzato secondo la logica DTL.

Tra i molti altri circuiti realizzati, esistono Full Adder, Divisori, registri a scorrimento a 8 bit, ecc.

I circuiti integrati si possono combinare facilmente per formare circuiti logici combinatori e/o sequenziali anche notevolmente complessi. Gli stessi costruttori, anzi, sono soliti accompagnare i loro cataloghi con pubblicazioni contenenti i più svariati esempi di realizzazioni pratiche (circuiti decodificatori, registri, contatori di ogni tipo, sommatori in parallelo). Come esempi di questi circuiti, riportiamo nella tabella VIII.2 un convertitore binario-decimale e uno shift-register reversibile.

I criteri di progetto sono molto semplici: si parte dalle equazioni logiche, si disegna il circuito, e si sostituiscono i suoi elementi coi più convenienti circuiti integrati di cui si dispone. Quando è possibile, si usa poi una sola famiglia logica.

Occorre, ovviamente, attenersi strettamente a quanto riportato nei cataloghi per quanto riguarda:

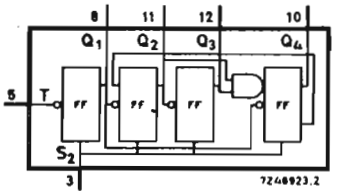
1) le tensioni di alimentazione e le relative tolleranze;
2) il *fan-in* e il *fan-out*;
3) i ritardi introdotti dagli elementi adottati;
4) i requisiti dei fronti di caduta e di salita dei segnali impulsivi;
5) l'immunità dal rumore;
6) le temperature di funzionamento;
7) la riformazione degli impulsi;
8) le dislocazioni dei microcircuiti in modo da realizzare collegamenti il più possibile diretti e brevi.

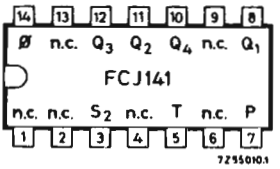
TABELLA VIII.1 - Contatore Decimale Asincrono da «Dati Tecnici Philips»

The FC family of DTL silicon monolithic integrated circuits has been designed for medium speed digital applications in computing, office electronics, telecommunication, instrumentation and industrial control.

DEVELOPMENT SAMPLE DATA

SINGLE ASYNCHRONOUS 10-COUNTER



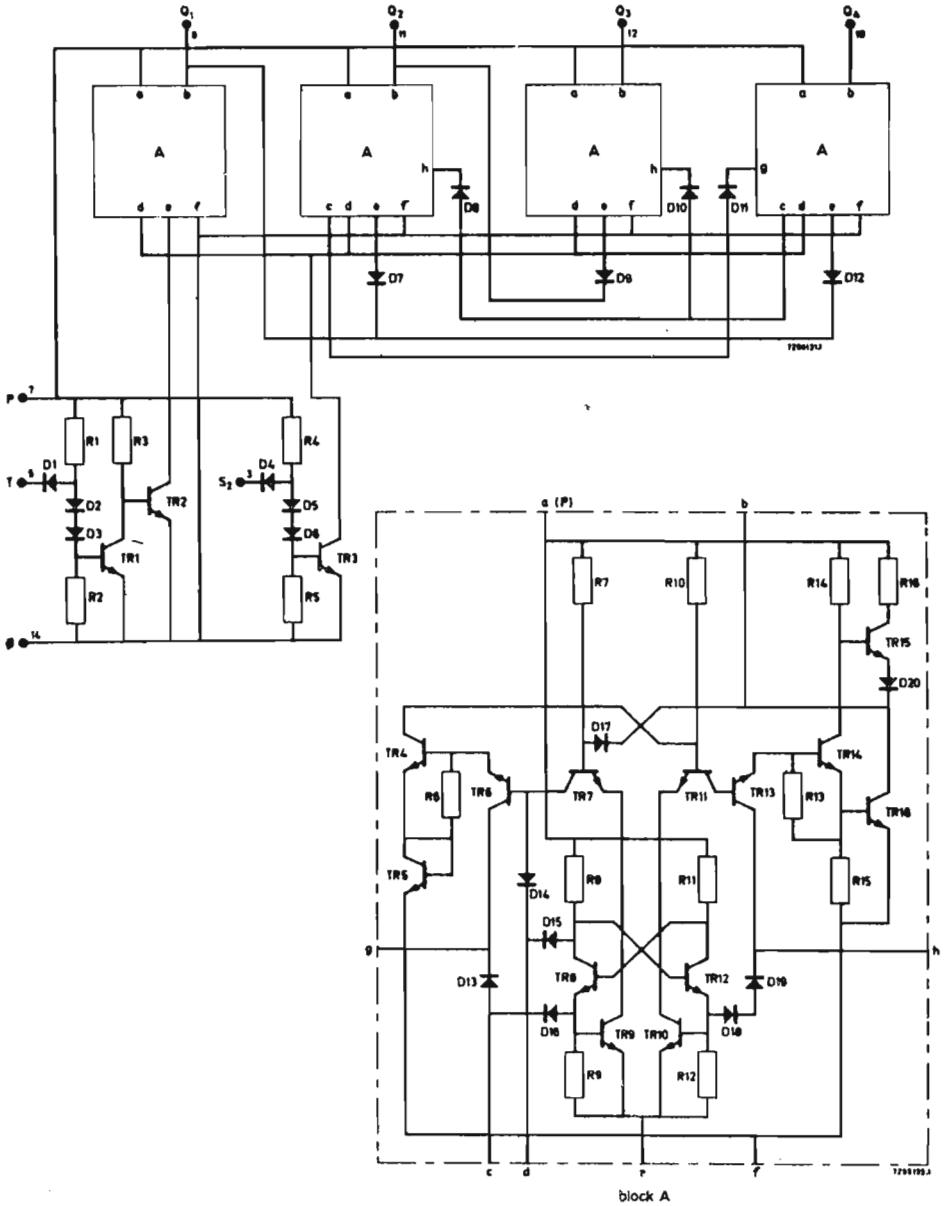


QUICK REFERENCE DATA			
Supply voltage	V_P	6.0 ± 5%	V
Operating ambient temperature range	T_{amb}	0 to +75	°C
Clock rate	f_c	typ. 7	MHz
Available d.c. fan out T_{amb} = 25 °C	N_a	≥ 7	
D.C. noise margin T_{amb} = 25 °C	M_L	typ. 1.2	V
Power consumption 50% duty cycle, T_{amb} = 25 °C	P_{av}	typ. 180	mW

The FCJ141 is four master-slave flip-flops interconnected to form an a-synchronous decade counter in the 8-4-2-1 code. The information is transferred to the master when the trigger signal is HIGH (the first flip-flop is triggered by the count input at T). When the trigger signal is LOW the information is transferred to the slaves and appears at the outputs. A common reset input S_2 directly resets the outputs and overrides the T input.

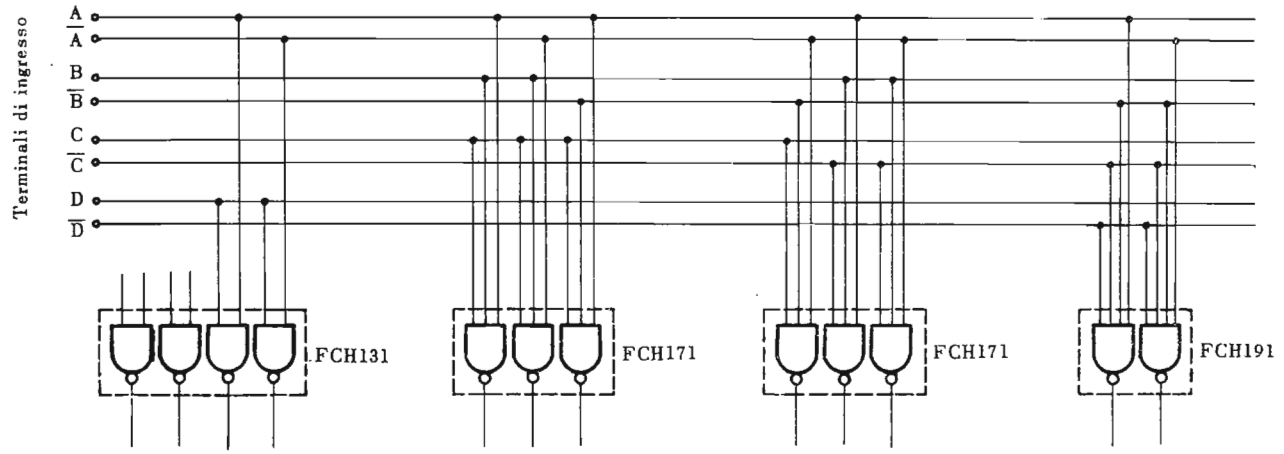
These data, based on the specifications and measured performance of development samples, afford a preliminary indication of the characteristics to be expected of the described product. Distribution of development samples implies no guarantee as to the subsequent availability of the product

CIRCUIT DIAGRAMS

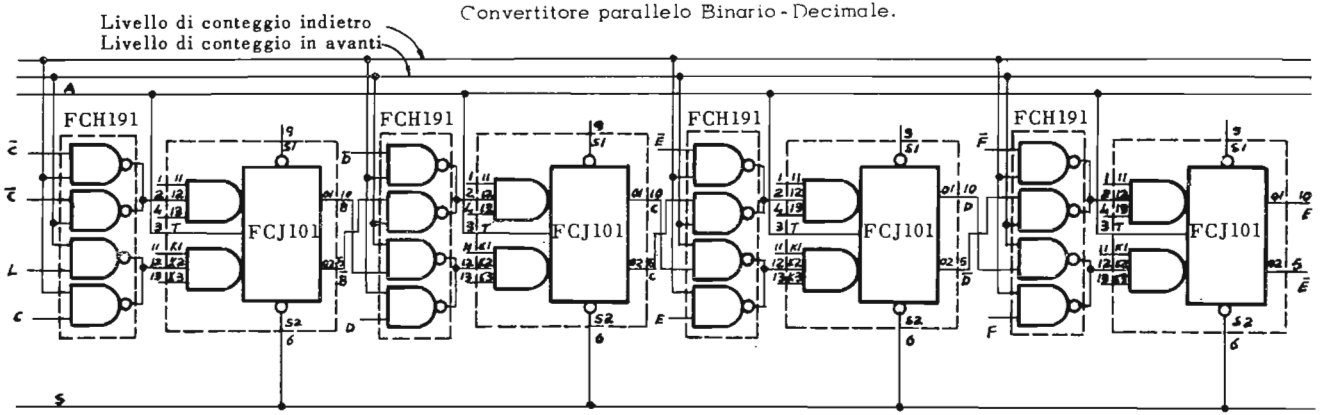


block A

TABELLA VIII.2 - da Circuiti Integrati digitali serie FC.



Convertitore parallelo Binario - Decimale.



Shift register Ingresso serie - Uscita serie, reversibile.

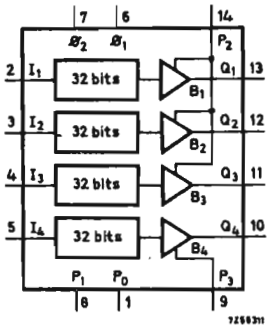
TABELLA VIII.3 - Circuiti MOS
da «Dati Tecnici Philips»

SHIFT REGISTERS

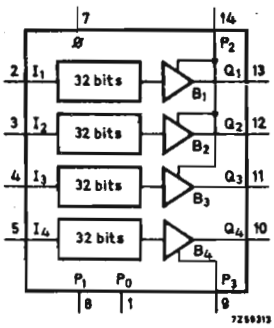
Supply voltage	V_P	=	- 24 V to - 28 V
D.C. noise margin	M_L	>	1 V
Clock rate (two-phase)	f_{ϕ}	=	0.01 MHz to 3 MHz
Clock rate (single-phase)	f_{ϕ}	=	0.01 MHz to 1 MHz
Power dissipation	P_{tot}	=	300 mW
Operating ambient temperature range	T_{amb}	=	- 55 °C to + 85 °C

Metal/ceramic hermetic dual in-line package (14 lead)

Quadruple 32 bit dynamic shift registers, FDN106 and FDN116

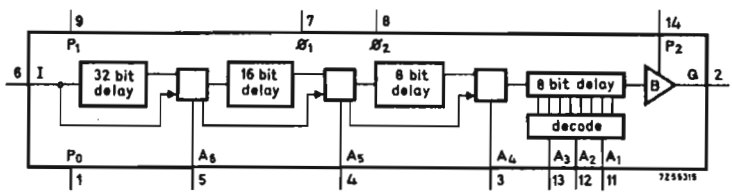


FDN106
Two-phase

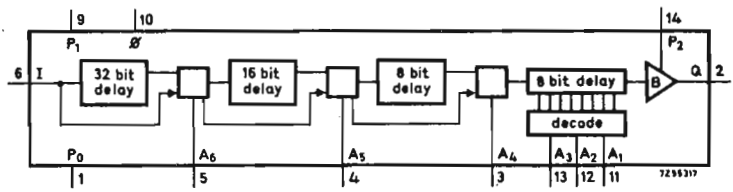


FDN116
Single-phase

Variable length 1 to 64 bit dynamic shift registers, FDN126 and FDN136

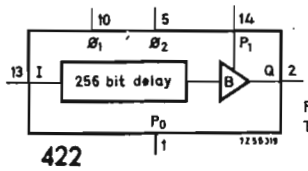


FDN126
Two-phase

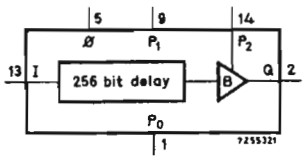


FDN136
Single-phase

256 bit dynamic shift registers, FDN146 and FDN156



FDN146
Two-phase



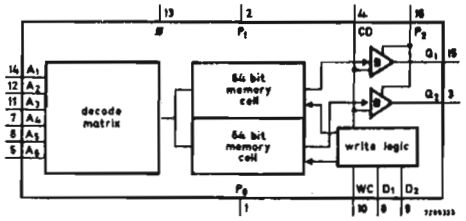
FDN156
Single-phase

RANDOM ACCESS MEMORY

Supply voltage	V_P	=	−24 V to −28 V
Stand-by power per bit		=	3 μW
Power consumption per bit	P_{av}	≈	135 mW
Read access time	t_{AR}	<	1 μs
D.C. noise margin	M_L	>	1 V
Data read rate	f_{DR}	>	1 MHz
Data write rate	f_{DW}	>	1 MHz
Operating ambient temperature range	T_{amb}	=	−55 °C to +85 °C

Metal/ceramic dual in-line package (16 lead)

FDQ106 – Random access memory 128 bit



READ ONLY MEMORIES

Read access time FDR 106 Z . .	t_{AR}	<	1 μs
FDR 116 Z . .		<	850 ns
Clock rate	f_ϕ	>	1 MHz
Power dissipation at f_ϕ = 1 MHz	P_ϕ	=	36 mW

Metal/ceramic dual in-line package (24 lead)

D.C. noise margin	M_L	>	1 V
Operating ambient temperature range	T_{amb}	=	−55 °C to +85 °C

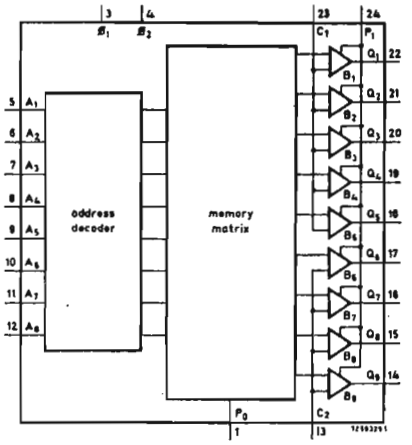
256 words; 9 bits/word

FDR106Z... bit pattern to customer's specification

FDR106Z1 fixed bit pattern, specified below

Bit pattern. With bit patterns stored in the FDR106Z1 the following functions can be performed:

1. **Starburst character generation.** Generation of activating signals for 64 17-segment starburst pattern characters, using an ASCII input code (128 words).
2. **Starburst pattern generation.** Generation of X and Y signals for a cathode ray tube producing the starburst character segment pattern (22 words).
3. **Seven segment decode.** Generation of activating signals for a 7-segment character generator with BCD input codes (16 words).
4. **Selectric to ASCII code conversion.** Conversion of I.B.M. selectric input/output writer key-board code to ASCII code with even or odd parity (90 words).

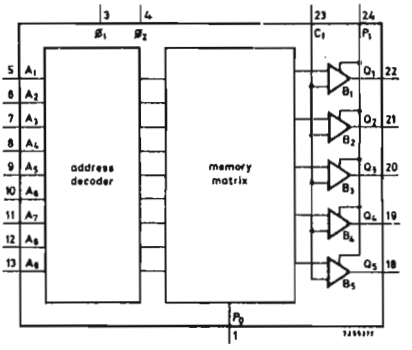


512 words; 5 bits/word

FDR116Z... bit pattern to customer's specification

FDR116Z1 with fixed bit pattern for dot code matrix character generator

The FDR 116 Z 1 is intended as a character generator for symbols for a display system in which each character is presented as a dot code matrix of seven rows of five bits each, 35 bits in all. The FDR 116 Z 1 holds 64 characters that are selected by a 6 bit address on inputs A4 to A9 (character select inputs). The seven rows are selected by a 3 bit address code on inputs A1 to A3 (word select inputs). When generating a complete line of characters, each word address is maintained for the full line, the character select address being changed as appropriate. The line of characters is completed by addressing each row in turn and selecting the characters in the same sequence. All zeros on the word select inputs produces all zeros at the output, thus giving line spacing.



VIII.7.1 - Circuiti MOS ed LSI.

La tecnica dei circuiti integrati consente attualmente di produrre dei circuiti digitali molto più complessi di quelli finora esaminati, con dei particolari procedimenti costruttivi. Tali procedimenti, la cui descrizione è del tutto al di fuori dei nostri scopi, sono generalmente indicati con le sigle MOS ed LSI.

La logica MOS (*Metal Oxide Silicon*) riduce le dimensioni di un circuito nel rapporto 1/10 rispetto ai comuni circuiti integrati, e presenta – rispetto ad essi – una serie di vantaggi e di svantaggi per i quali rimandiamo ai manuali dei costruttori. Vogliamo solo accennare che con questa tecnica si costruiscono agevolmente, oltre ai normali circuiti logici, anche i cosiddetti LSI, in cui si hanno centinaia di circuiti elementari nello stesso contenitore. Sono, così, stati realizzati, ad esempio, shift register a 256 bit e memorie di sola lettura (*Read Only Memories* - ROM), con relative matrici di decodifica, a parecchie migliaia di bit (per l'uso di queste memorie rimandiamo ai testi citati in bibliografia [13, 14, 15].

Nella tab.VIII.3 sono riportati alcuni tipici circuiti MOS.

*

BIBLIOGRAFIA

1. PHISTER M.J.: *Logical Design of Digital Computers* - John Wiley & Sons, 1958.
2. HUMPREY W.S.: *Switching Circuits with computer Applications* - Mc Graw Hill, 1958.
3. MARCUS M.P.: *Switching circuits for engineers* - Prentice Hall, 1967.
4. IRWIN W.C.: *Digital Computer Principles* - Van Nostrand, 1960.
5. LEDLY R.S.: *Digital Computer and Control Engineering* - Mc Graw Hill, 1960.
6. RICHARDS R.R.: *Arithmetic Operations in Digital Computer* - D.Van Nostrand, 1955.
7. GORDON B.D.: *An Introduction to Electronic Computers* - Mc Graw Hill, 1967.
8. BARON R.C. & PICCIRILLI A.T.: *Digital Logic and Computer Operations* - Mc Graw Hill, 1968.
9. MILMAN J. - TAUB H.: *Pulse, Digital and Switching Waveforms* - Mc Graw Hill, 1968.
10. NASHLESKY L.: *Digital Computer Theory* - John Wiley & Sons, 1966.
11. PHILIPS: *Elementi logici in apparecchiature digitali* - Manuale tecnico Philips, 1967.
12. *The Application of diode-transistor Micrologic* - Manuale SGS, 1967.
13. TEXAS INSTRUMENTS: *Integrated circuits catalog* - 1968.
14. PHILIPS - Dati Tecnici: *Circuiti integrati digitali Bipolari e MOS* - Philips - Elcoma 1971.
15. SACCHI P.F.: *Circuiti Integrati Digitali serie FC - Generalità e Applicazioni* - Biblioteca Tecnica Philips, 1971.

*

CAPITOLO IX

CALCOLATORI ELETTRONICI

IX.1 - Generalità.

I calcolatori digitali risolvono una vasta gamma di problemi, sia scientifici che tecnici: sono in grado di eseguire lunghe serie di calcoli matematici a velocità estremamente elevate, di preparare liste stampate per estratti di conti bancari, di immagazzinare un numero elevatissimo di dati, di ricercare e presentare un dato immagazzinato, di controllare processi industriali complessi nell'istante in cui avvengono, ecc.

Scopo di questo capitolo, che ha carattere generale e introduttivo, è la descrizione dei principali componenti circuitali dei calcolatori (hardware)[1]; useremo in esso alcuni termini la cui definizione esatta non rientra nei nostri scopi, e che sarà pertanto sufficiente introdurre intuitivamente.

Se vogliamo effettuare un calcolo, consistente ad esempio nella somma di 1000 numeri, dovremo introdurre su un idoneo supporto fisico (es. schede perforate) e secondo un opportuno codice i 1000 numeri, che costituiranno i *dati* da elaborare. Dovremo introdurre, insieme coi dati, le *istruzioni* relative al lavoro da effettuare, sullo stesso o su un diverso supporto fisico. Le istruzioni sono delle parole di codice, scritte in un particolare *linguaggio*, che agiscono sui circuiti del Calcolatore predisponendoli per una determinata sequenza di operazioni (es. lettura dei 1000 numeri su scheda, loro somma, stampa del risultato); esse vengo-

(1) - Il termine *hardware* è usato per indicare l'insieme dei componenti elettrici, magnetici e meccanici di un calcolatore, in contrapposizione al termine *software* che indica l'insieme dei programmi di ogni tipo, associato con un particolare calcolatore.

no registrate nella *memoria* del calcolatore e costituiscono, nel loro insieme, il *programma*.

I codici secondo cui i dati vengono immessi nell'elaboratore, immagazzinati ed elaborati sono tutti di tipo binario.

IX.2 - Schema a blocchi di un calcolatore.

Un calcolatore elettronico può considerarsi composto essenzialmente da 4 parti distinte (fig.IX.1):

1) la *memoria centrale* che contiene le istruzioni del programma, i dati da elaborare, i risultati da stampare, gli eventuali risultati intermedi;

2) l'*unità di controllo* che è l'organo attivo del calcolatore. Preleva le istruzioni dalla memoria, le interpreta, invia informazioni di ogni tipo alle altre unità, controllo l'esecuzione del programma;

3) l'*unità aritmetica* che effettua, su comandi ricevuti dall'unità di controllo, operazioni aritmetiche e logiche sui dati contenuti in memoria;

4) l'*unità di ingresso e uscita* che trasferisce le istruzioni e i dati da elaborare alla memoria, e i risultati su un mezzo esterno.

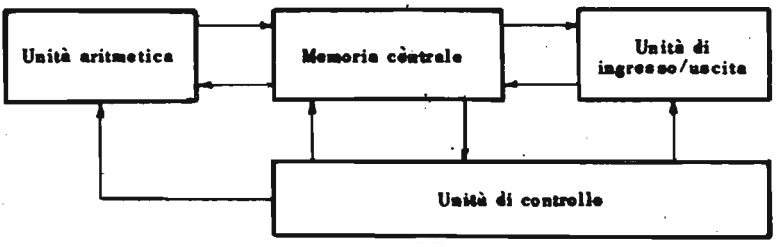


Fig.IX.1 - Schema a blocchi di un calcolatore elettronico.

Vedremo ora, con qualche dettaglio, le varie parti del calcolatore, a cominciare dall'unità aritmetica, la cui costituzione è più semplice da capire, alla luce delle conoscenze finora acquisite.

Per ragioni di semplicità, supporremo poi che il calcolatore lavori in modo sincrono.

IX.3 - Unità aritmetica.

L'unità aritmetica realizza le operazioni aritmetiche (somma, prodotto, sottrazione, divisione) e alcune operazioni logiche, tra cui il confronto, su dati binari contenuti in una parola.

L'unità aritmetica comprende sempre un certo numero di registri e dei circuiti aritmetici, tutti più o meno derivati dall'addizionatore.

Descriveremo soprattutto i circuiti che realizzano operazioni aritmetiche, anche perchè quelli per le funzioni logiche sono stati già trattati nei precedenti capitoli.

IX.3.1 - Addizionatore binario.

Nel cap.IV abbiamo mostrato la realizzazione circuitale di un addizionatore (ADDER) che fornisce, su 2 terminali distinti, il risultato della somma S_i e del riporto R_i relativi all'addizione dei bit A_i e B_i, di ordine i, e del riporto R_{i-1} relativo ai bit di ordine i - 1 dei numeri A e B.

Per addizionare n bit, occorre disporre di n ADDER in parallelo oppure mandare le cifre A_i e B_i n volte di seguito sullo stesso circuito, ad intervalli successivi di tempo (addizione in parallelo o in serie). Supponiamo, per intanto, di sommare numeri senza segno.

IX.3.1.1 - Addizionatore in serie.

La fig.IX.2 mostra uno schema a blocchi di un addizionatore in serie.

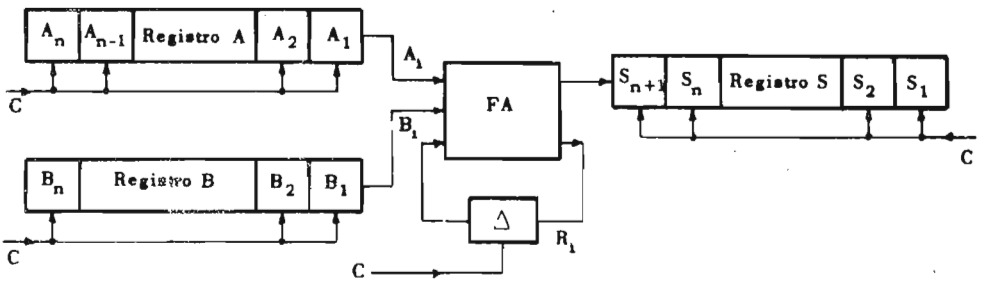


Fig.IX.2 - Schema a blocco di un addizionatore in serie (I schema).

I bit A_1 e B_1 si presentano per primi al Full Adder (FA); il risultato S_1 è trasferito nella 1ª cella a sinistra (S_{n+1}) del registro S; il riporto R_1 si presenta all'ingresso del flip-flop Δ. Successivamente, vengono addizionati in FA: A_2, B_2, e il riporto R_1. Il risultato S_2 viene immagazzinato in S_{n+1}, dopo che S_1 è shiftato in S_n; il riporto R_1 si presenta all'ingresso di Δ...

Dopo n + 1 impulsi di clock, l'addizione è completa. Il bit più significativo S_{n+1} è il riporto R_n, che viene sommato con $A_{n+1}=B_{n+1}=0$.

Si noti che:

1) il registro S ha (n + 1) bit, per contenere in ogni caso la somma dei 2 numeri di n bit (questa è una soluzione non sempre adottata, evitabile con la segnalazione di un overflow se $R_n = 1$);

2) un solo FA basta per addizionare i 2 numeri a n bit;

3) il tempo necessario per effettuare la somma A + B è il prodotto del tempo necessario ed effettuare la somma nel FA per n+1; a questo tempo vanno aggiunti i tempi di trasferimento di A e B dalla memoria nei registri, e del registro S in memoria (tempi che variano notevolmente a seconda che la scrittura avvenga in parallelo o in serie);

4) i segnali C sono segnali di controllo derivati direttamente dal clock, e generati dall'unità di controllo nei modi ampiamente descritti nel capitolo precedente.

Dallo schema della fig.IX.2 deriva direttamente l'addizionatore della fig.IX.3, che usa due soli registri invece che tre: il risultato infatti viene scritto nelle celle dal registro B che man mano vanno vuotandosi. Essendo i registri A e B ad n bit, un eventuale overflow (soluzione qui adottata) è rivelato da un 1 su R_i all'$(n+1)^{mo}$ clock. La velocità di questo circuito è la stessa del precedente.

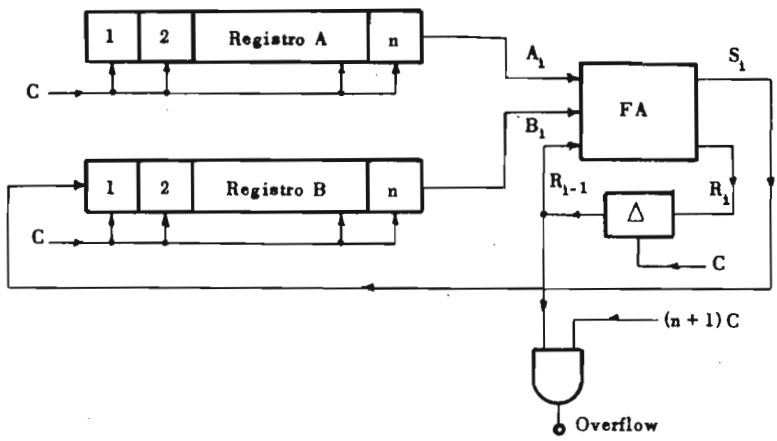


Fig.IX.3 - Addizionatore in serie (II schema).

Nei circuiti delle figg.IX.2 e IX.3, la lettura e la scrittura dei bit avvengono generalmente in parallelo (i registri adoperati sono registri a scorrimento, con uscita e ingresso in parallelo. Se la velocità non

è importante, si possono usare registri più semplici, con uscita e ingresso in serie. Uno degli schemi adoperati, che impiega tra l'altro un solo registro, è mostrato nella fig.IX.4. I bit A_i, vengono mandati uno alla volta al registro A (accumulatore); dopo n impulsi di clock si mandano al FA i bit B_i, insieme agli A_i provenienti - per scorrimento - dal registro A. I risultati S_i vengono scritti - durante n impulsi di clock - di nuovo nell'accumulatore, da sinistra verso destra. Un segnale M rivela un eventuale overflow, e abilita la trasmissione in altri n impulsi di clock.

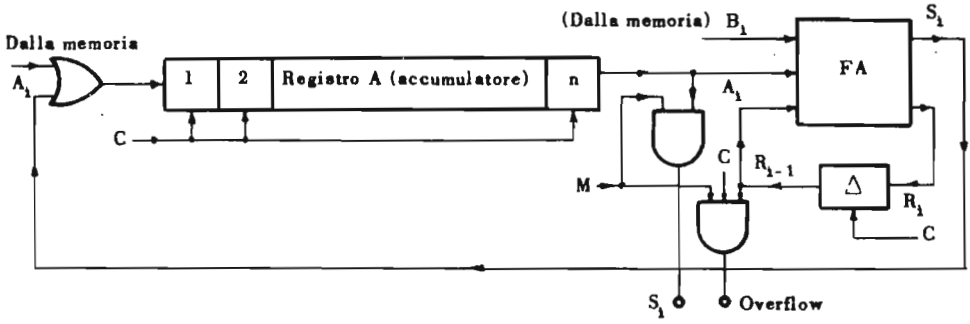


Fig.IX.4 - Addizionatore seriale (III schema).

IX.3.1.2 - Addizionatore in parallelo.

L'addizione in parallelo è più rapida di quella in serie, ma richiede circuiti più complessi, quindi più costosi. Precisamente, occorrono: 1 HA e n-1 FA, se n sono i bit delle parole da sommare (vedi figura IX.5).

Si noti che ogni coppia di bit A_i e B_i fornisce 1 bit di riporto al FA_i. I riporti vengono trasmessi dal primo all'ultimo FA, per cui la somma finale appare solo quando il primo riporto si è propagato fino all'ultimo FA. Solo allora viene inviato dall'unità di governo un segnale di fine operazione (FO) che scrive i risultati della somma nel registro S. Il circuito è molto rapido: impiega il tempo necessario per effettuare una somma più quello occorrente per la propagazione dei riporti.

Per l'addizionatore in parallelo sono possibili, evidentemente, altri schemi (in pratica, ad esempio, il registro S coincide con uno dei registri A e B) che riteniamo inutili esporre.

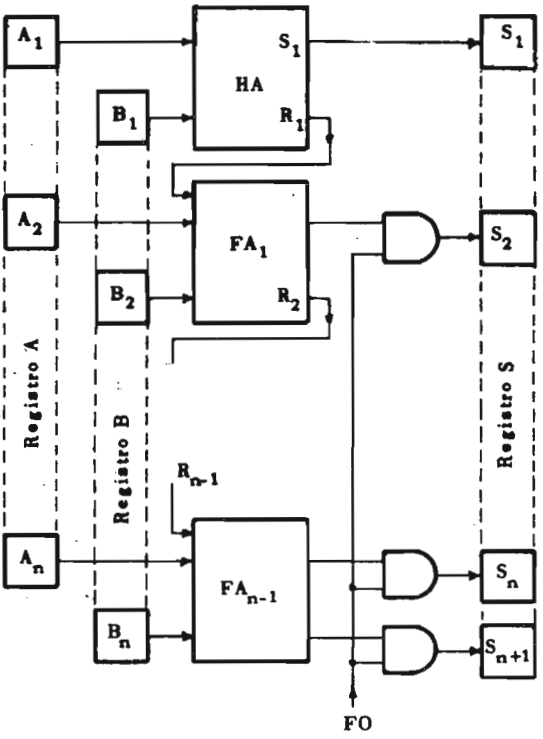


Fig.IX.5 - Addizionatore in parallelo.

IX.3.2 - Sottrazione binaria.

Prima di esporre i metodi con cui si può effettuare la sottrazione di 2 numeri A e B ad n bit, ricaviamo rapidamente, sulla scorta di quanto fatto al cap.IV per l'addizionatore, gli schemi di un Half-Subtractor e di un Full-Subtractor.

IX.3.2.1 - Half-Subtractor.

Nella fig.IX.6 è mostrata la tabella di verità delle funzioni D (differenza) e P (prestito) per l'operazione A - B.

Fig.IX.6 - Tabella di verità dell'HF (Half-Subtractor).

A	B	D	P
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Dalla tabella si ricavano le equazioni:

$$D = \overline{A}B + A\overline{B} = A \oplus B$$
$$P = \overline{A}B \ .$$

Poichè la D coincide con la S dell'HA, e la P coincide con la C, se si cambia il segno di A, il FS è molto simile all'HA: in fig.IX.7 sono riportati alcuni schemi [si noti il circuito (c), realizzato con elementi AND-OR-NAND].

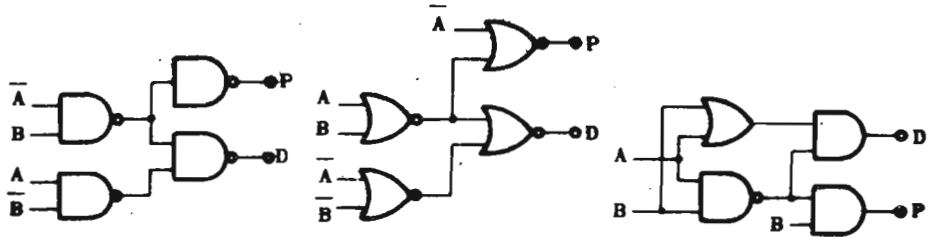


Fig.IX.7 - Circuiti HS.

IX.3.2.2 - Full-Subtractor.

Nella fig.IX.8 è mostrata la tabella di verità delle funzioni D_i e P_i per l'operazione $A_i - B_i - P_{i-1}$, effettuata per tutti i bit A_i e B_i di 2 numeri A e B da sottrarre, ad eccezione dei primi 2.

A_i	B_i	P_{i-1}	D_i	P_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fig.IX.8 - Tabella di verità del FS (Full-Subtractor).

Dalla tabella si ricavano le equazioni:

$$D_i = \overline{A}_i\overline{B}_iP_{i-1} + \overline{A}_iB_i\overline{P}_{i-1} + A_i\overline{B}_i\overline{P}_{i-1} + A_iB_iP_{i-1}$$
$$P_i = A_iB_iP_{i-1} + \overline{A}\,(B_i + P_{i-1}) = \overline{A}_i\overline{B}_iP_{i-1} + \overline{A}_iB_i\overline{P}_{i-1} + B_iP_{i-1} \ .$$

Alcuni circuiti FS sono mostrati nella fig.IX.9; altri circuiti possono ricavarsi facilmente modificando i FA del cap.IV, dato che la D_i coincide con la S_i del FA stesso.

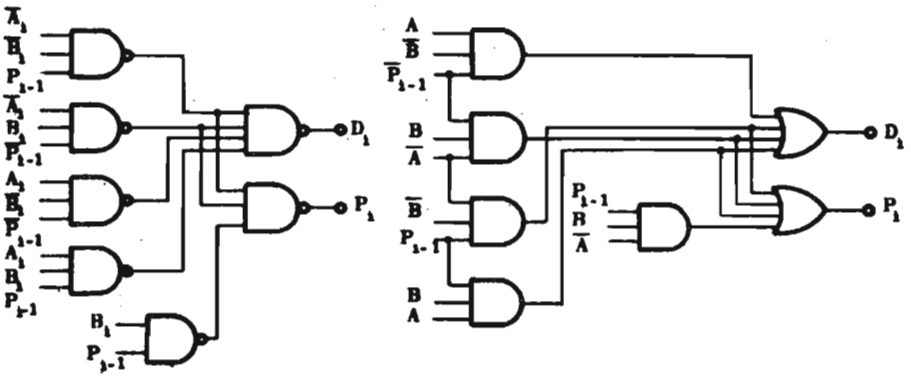


Fig.IX.9 - Circuiti FS.

Il FS e l'HS possono essere utilizzati per costruire sottrattori in serie e in parallelo, con procedimenti simili a quelli usati per gli addizionatori, mantenendo valida l'ipotesi che A e B siano numeri senza segno, e che sia sempre $A \geq B$.

Nel cap.I, esponendo i principali metodi di rappresentazione dei numeri con segno, abbiamo detto che il modo più semplice per realizzare la loro somma è quello di usare una rappresentazione complementata. Possiamo ora facilmente spiegare perchè. Un circuito costruito per sommare numeri rappresentati con modulo e segno dovrebbe:

1) poter trattare a parte i bit di segno;

2) funzionare come addizionatore se i bit di segno sono uguali, assegnando il valore comune al risultato della somma;

3) stabilire quale numero è maggiore in valore assoluto se i bit di segno sono diversi; sottrarre da questo il minore, e attribuire al risultato della sottrazione il segno del minuendo.

Un circuito così fatto può essere realizzato, ma è certamente più complesso di quelli che utilizzano la rappresentazione complementata dei numeri negativi.

IX.3.2.3 - Addizionatore-Sottrattore per numeri rappresentati in complemento a 1.

L'addizionatore-sottrattore è realizzato tenendo presente le regole esposte al par.I.8.1.2 per la somma di 2 numeri rappresentati in complemento a 1: *l'addizione di 2 numeri a n bit si fa sommando i numeri stessi. La sottrazione va preceduta dal complemento del sottraendo. Eventuali bit 1 nella posizione (n + 1) vanno aggiunti al risultato.*

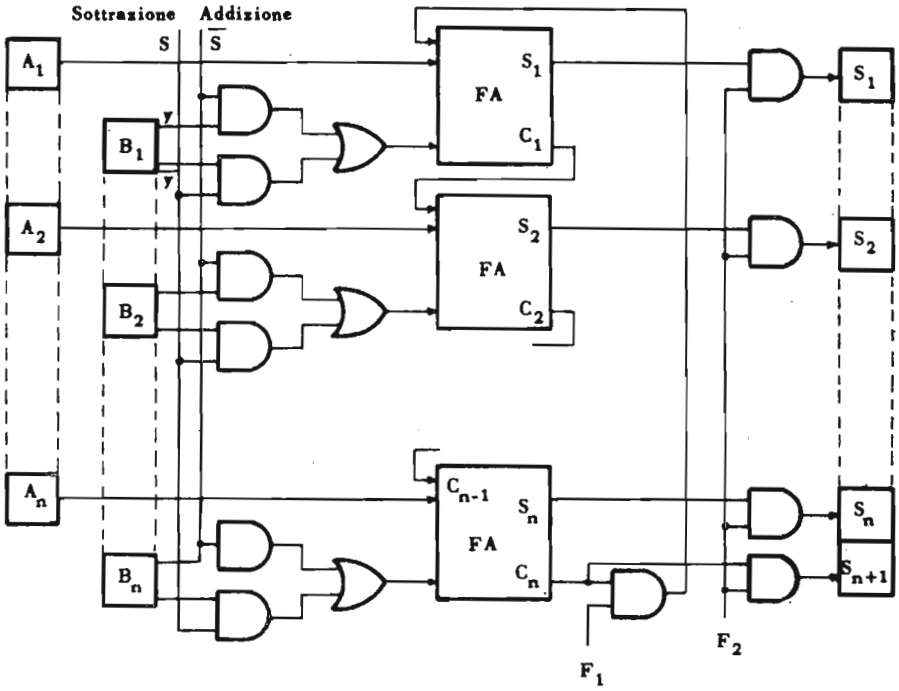


Fig.IX.10 - Circuito addizionatore-sottrattore in parallelo per numeri rappresentati in complemento a 1.

Nella fig.IX.10 è mostrato un circuito in parallelo realizzato secondo questo principio, e derivato da quello della fig.IX.5. Se si deve effettuare la sottrazione A - B, l'unità di governo invia un segnale S = 1, per cui tutti i bit contenuti nel registro B vengono invertiti. Il riporto dell'ultimo FA, letto dal segnale F_1 inviato dall'unità di governo quando l'addizione è terminata (vedi FO nel par.IX.3.1.2), viene quindi sommato al bit meno significativo. Il risultato viene letto e immagazzinato da un successivo segnale F_2.

Per realizzare la somma A + B. l'unità di Governo invia invece il segnale S = 0, per cui i bit di B non vengono complementati.

IX.3.2.4 - Addizionatore e sottrattore per numeri rappresentati in complemento a 2.

L'addizionatore-sottrattore per numeri rappresentati in complemento a 2, è molto simile a quello della fig.IX.10; non occorre aggiungere l'ultimo riporto al risultato, per cui l'operazione è più rapida. Il circuito (fig.IX.11) è tuttavia leggermente più complicato, perchè nel caso in cui occorra fare la sottrazione A - B il numero B deve essere complementato e questo richiede, oltre all'inversione dei bit B_i, l'addizione di 1 al risultato.

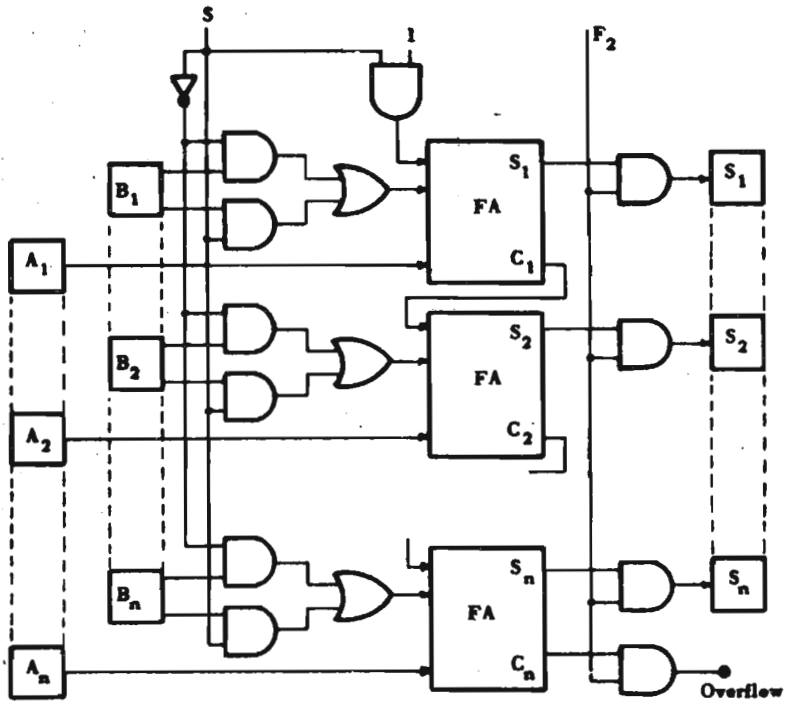


Fig.IX.11 - Circuito addizionatore-sottrattore in parallelo per numeri rappresentati in complemento a 2.

IX.3.2.5 - Altri tipi di addizionatori: addizionatori decimali.

Oltre a quelli visti, si costruiscono addizionatori per numeri rappresentati in codici particolari, Tra questi, una certa importanza ha quello per la somma dei numeri espressi nel codice «eccesso a 3», che ha la

proprietà di essere autocomplementante, e quello *decimale*, che effettua la somma di numeri decimali codificati in BCD, dando il risultato, sempre in decimale, nello stesso codice.

Nella fig.IX.12 è mostrato lo schema a blocchi di un FA decimale, che somma i bit $A_{i1}A_{i2}A_{i3}A_{i4}$ della cifra decimale A_i con i bit $B_{i1}B_{i2}B_{i3}B_{i4}$ della cifra decimale B_i e con il riporto C_{i-1} proveniente dalla somma delle cifre decimali A_{i-1} e B_{i-1}; le uscite del circuito sono i 4 bit $S_{i1}S_{i2}S_{i3}S_{i4}$ della i^{ma} cifra decimale del risultato S, e il riporto C_i, il cui peso è 10^{i+1}.

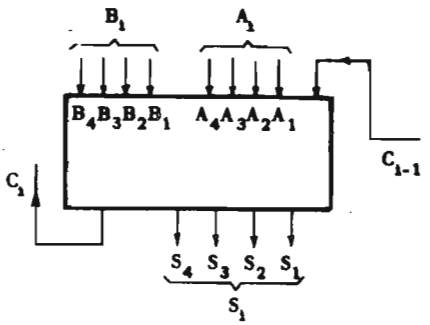
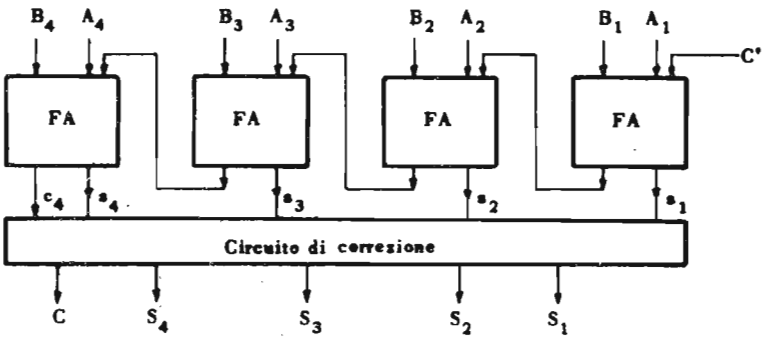


Fig.IX.12 - Schema a blocchi di un FA decimale.

Il FA decimale può essere progettato come un normale circuito multiterminale combinatorio a 9 ingressi e 5 uscite o, più semplicemente, come una serie di FA binari (fig.IX.13) le cui uscite vengono opportunamente corrette. Tali correzioni derivano dal fatto che quando le cifre A e B danno come somma un numero compreso tra 10 e 15, tale numero viene espresso in binario secondo configurazioni (1010, 1011, ..., 1111) non significative in BCD; mentre quando la somma è compresa tra 16 e 19, il risultato è espresso col riporto c = 16, non C = 10 come dovrebbe essere in BCD.



N.B. - C'è il riporto del FA decimale precedente.

Fig.IX.13 - Schema a blocchi di un FA decimale realizzato con FA binari e circuito di correzione.

Indicando con c_4 ed s_i il riporto e le somme non corrette, e con C ed S_i i riporti e le somme corrette, il circuito di correzione dovrà realizzare la seguente tadella di verità:

Valore in decimale della somma A+B+C'	Uscite dai FA, non corrette					Uscite corrette				
	c_4	s_4	s_3	s_2	s_1	C	S_4	S_3	S_2	S_1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

Esaminando la tabella si può notare che:

1) le correzioni debbono intervenire su tutte e sole le configurazioni per le quali si ha $c_4 = 1$ oppure $s_4 (s_2 + s_3) = 1$;

2) le configurazioni corrette - lette come numeri binari a 5 cifre - si ottengono semplicemente sommando il numero 6 alle configurazioni non corrette;

3) $s_1 = S_1$.

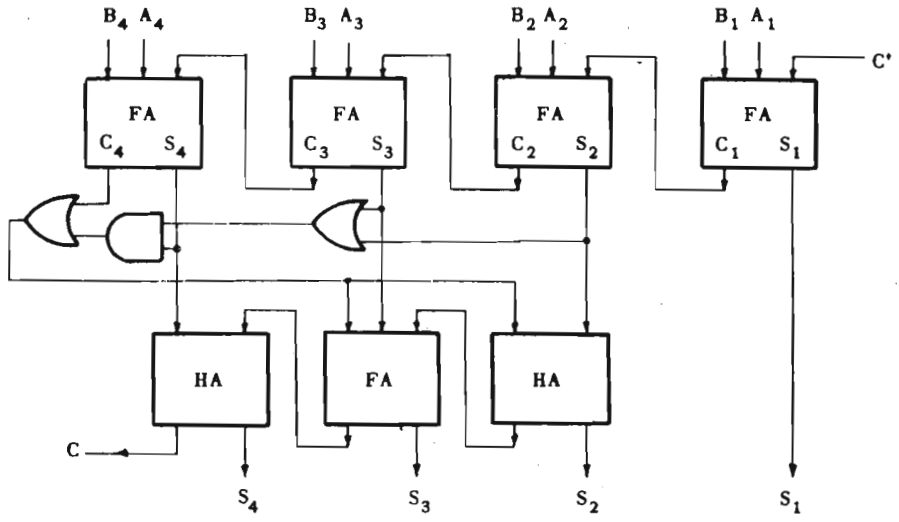


Fig.IX.14 - Addizionatore decimale.

Ciò permette di realizzare il circuito di correzione con 1 FA e 2 HA binari in cui si sommano le uscite $s_2 s_3 s_4$ con il numero 110, per tutte le configurazioni per le quali si ha $c_4 + s_4 (s_2 + s_3) = 1$. Il FA decimale completo è mostrato nella fig.IX.14.

Ovviamente, il circuito di correzione si può ottenere, dalla tabella di verità, come un circuito combinatorio che ha come ingresso un numero binario a 5 bit e come uscita una cifra EBCD e un riporto di valore 10.

Con i FA decimali, si possono realizzare addizionatori decimali in serie o in parallelo per numeri a n cifre, in modo simile a quanto fatto per gli addizionatori binari.

IX.3.3 - Moltiplicatori.

Il prodotto $M = A \cdot B$ si può effettuare - in serie o in parallelo - per addizione ripetuta o per scorrimento. Se A e B hanno n bit ciascuno, M può avere fino a 2 n bit.

IX.3.3.1 - Moltiplicatori per addizione ripetuta.

Per l'*addizione ripetuta* in parallelo si può adoperare il circuito di fig.IX.15. All'inizio si scrive il moltiplicando A nel registro R_A e il moltiplicatore B nel contatore scalante: si azzera l'accumulatore, si somma il moltiplicando allo 0 e si scala il contatore di 1. Si somma poi - di nuovo - il contenuto (A) dell'accumulatore al contenuto (A) di R_A, e si scala il contatore di 1: l'operazione termina quando il contatore è andato a 0. Si noti che l'accumulatore è a n bit, non a 2n, e conserva le cifre meno significative del risultato. Le cifre più significative si ottengono come uscite di un contatore che conta gli overflow verificatisi nell'addizionatore durante le varie somme.

Per chiarire il meccanismo di questa moltiplicazione, eseguiamo - passo per passo - il prodotto $M = 7 \times 4$ (7 = moltiplicatore; 4 = moltiplicando). All'inizio il contatore scalante riceve il 4, il registro R_A il 7; l'accumulatore è azzerato.

Al primo impulso di clock, avviene la somma A + 0 nell'Adder. Il contatore contiene 3; R_A contiene 7; l'accumulatore contiene 7.

Al secondo impulso di clock, si effettua la somma A + A = 2A. Il contatore contiene 2; R_A contiene 7; l'accumulatore contiene i bit meno significativi del risultato:

$$\begin{array}{r} 111 \\ 111 \; + \\ \hline 1\,110 \end{array}$$

cioè 6. Si è verificato un overflow, che viene registrato nel contatore.

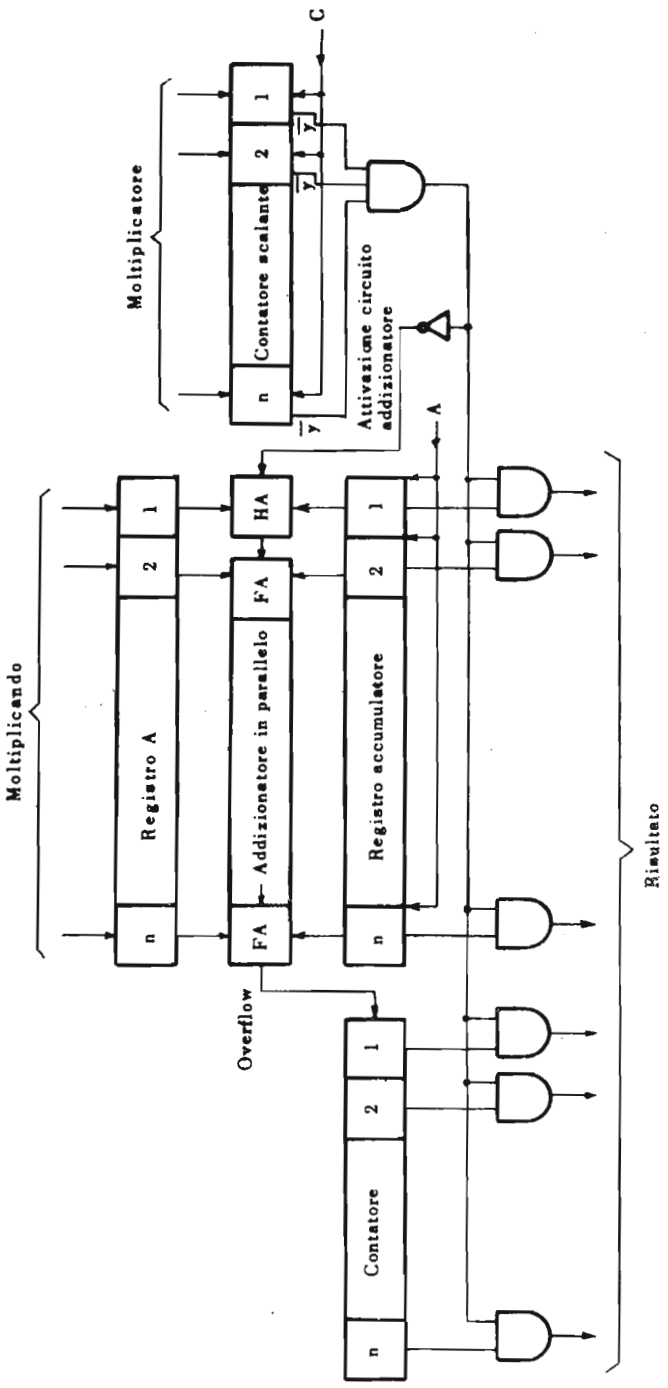


Fig.IX.15 - Moltiplicatore per addizioni ripetute, con addizionatore in parallelo.

Al terzo impulso di clock, si effettua la somma $R_A + A_{CC}$, cioè 7 + 6, che dà per risultato:

$$\begin{array}{r} 110 \\ 111\ + \\ \hline 1\,|\,101 \end{array}$$

le cifre meno significative (5) vengono conservate nell'accumulatore. L'overflow viene mandato al contatore (che così contiene 2); il contatore scalante passa a 1.

Al quarto impulso di clock, la somma $R_A + A_{CC}$, cioè 7 + 5, dà per risultato:

$$\begin{array}{r} 101 \\ 111\ + \\ \hline 1\,|\,100 \end{array}$$

Le cifre meno significative (4) vengono mantenute dall'accumulatore; l'overflow si manda al contatore (che così contiene 3); il contatore scalante passa a 0. La operazione è terminata. Il 3 (011) sul contatore e il (100) sull'accumulatore (insieme interpretati come il numero binario (011 100) corrispondono al numero decimale 28, risultato della moltiplicazione.

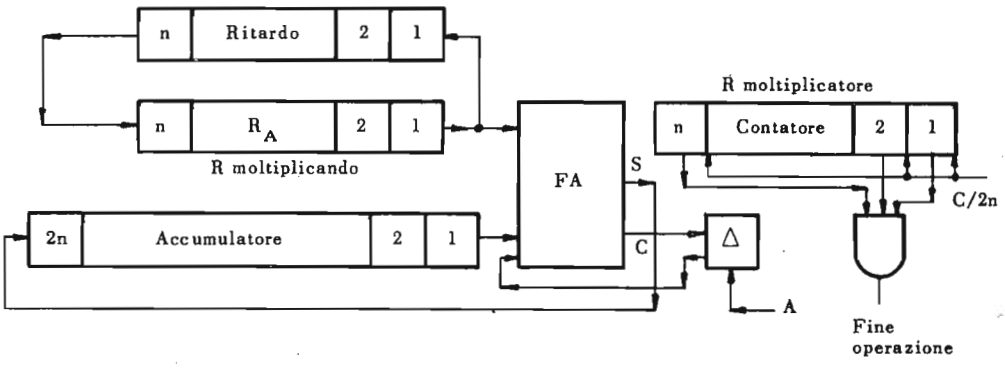


Fig.IX.16 - Moltiplicatore per addizioni ripetute in serie.

Il circuito per l'addizione ripetuta con addizionatore in serie è simile a quello della fig.IX.15, ma somma due bit alla volta. All'inizio, l'accumulatore è a 0 e gli operandi nei propri registri. L'Adder riceve le 2 cifre meno significative e mette il risultato della somma nell'accumulatore: il registro che contiene il moltiplicando scala di una posizione a destra. Il ciclo ricomincia: il moltiplicando, essendo sommato allo accumulatore, va riscritto nel registro, da cui esce per scorrimento; la riscrittura avviene da sinistra verso destra ma, comportando la diversa

lunghezza dei due registri uno sfasamento di n cicli fra accumulatore e registro R_A, occorre introdurre nel loop di scrittura un ritardo della stessa entità, per esempio con un altro registro a scorrimento di n celle. Dallo schema (fig.IX.16) si sono eliminati, per semplicità, i comandi di azzeramento e scorrimento.

IX.3.3.2 - Moltiplicatori a scorrimento.

I cicli richiesti per il prodotto possono essere ridotti realizzando la moltiplicazione, come mostrato nel cap.I, mediante *scorrimento* del moltiplicando. Nella fig.IX.17a è mostrata, a titolo d'esempio, la moltiplicazione in binario di 11×9. Si analizzano, da destra verso sinistra, i bit del moltiplicatore; ogni volta che si trova un 1, il moltiplicando 1011 viene aggiunto a se stesso, spostato di tanti posti a sinistra quanti sono i bit 0 del moltiplicatore che precedono l'1 considerato. Lo stesso risultato si ottiene se si effettuano le somme parziali e si sposta, dello stesso numero di bit, il moltiplicando verso destra (fig.IX.17b): un circuito con Adder in parallelo che funziona in quest'ultimo modo è mostrato nella fig.IX.18a.

a)

```
  1011
  1001  x
------
  1011
1011---
------
1100011
```

b)

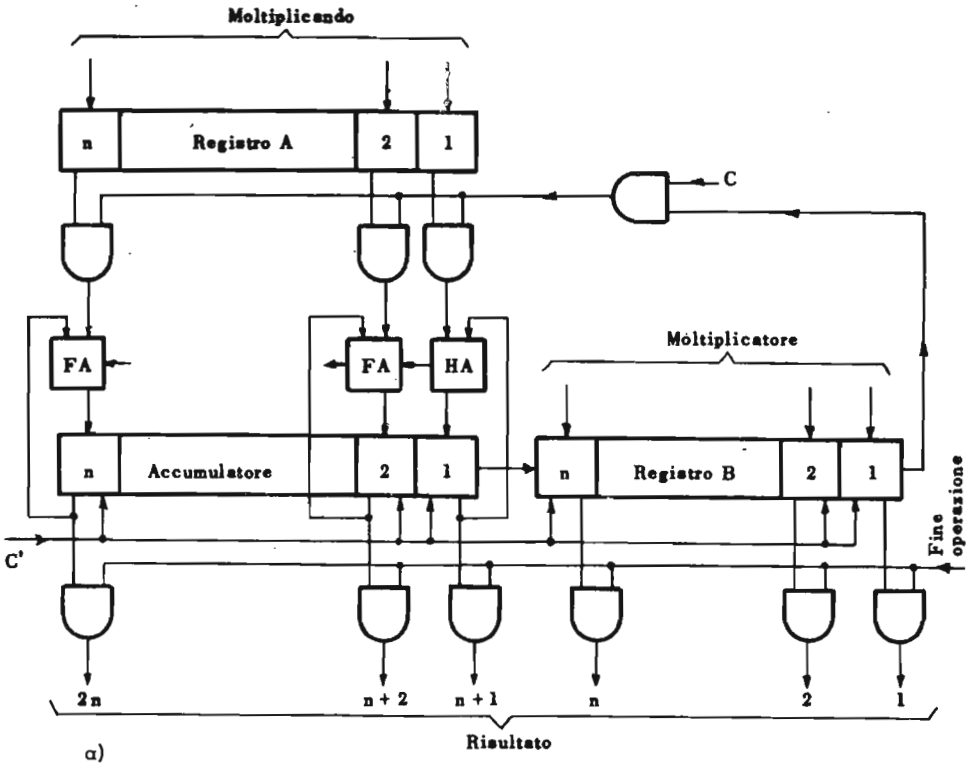
```
  1011
  1001  x
------
  ---1011
  1011
------
1100011
```

Fig.IX.17 - Moltiplicazione mediante scorrimento del moltiplicando.

Il circuito ispeziona i bit del moltiplicatore, a partire dal meno significativo. Trovato un 1, aggiunge il moltiplicando al contenuto dell'accumulatore (originariamente a 0) e fa scorrere di un posto a destra lo accumulatore stesso. Il bit che esce a destra dall'accumulatore entra da sinistra nel registro B, da cui, con un contemporaneo scorrimento a destra, esce la cifra meno significativa del moltiplicatore, ormai superflua. Se la cifra ispezionata è 0, si ha scorrimento senza addizione. Alla fine, le 2n cifre del prodotto si trovano nell'accumulatore e nel registro B, in ordine di importanza. I segnali C, C', ecc., sono tutti generati dall'unità di governo.

Nella fig.IX.18b è mostrato il contenuto dell'accumulatore e del registro B ad ogni ciclo dell'operazione di cui alla fig.IX.17; il registro A contiene sempre 1011.

La moltiplicazione per scorrimento con addizione in serie si basa sullo stesso principio: uno dei possibili schemi è mostrato nella figura IX.19.



Accumulatore	0 0 0 0	1 0 1 1	0 1 0 1	0 1 0 1	0 0 1 0
Registro B	1 0 0 1	molt. (1) 1 0 0 1 (1) moltiplicatore	molt. 1 1 0 0	molt. 1 1 0 0	molt. 1 1 1 0
Descrizione operazione	Inizio operazione	I cifra $R_B = 1$ $R_A + A_{CC} \rightarrow A_{CC}$	Scorrimento a destra di A_{CC} (in R_B) e di R_B	I cifra $R_B = 0$	Scorrimento a destra di A_{CC} e R_B
Accumulatore	0 0 1 0	0 0 0 1	1 1 0 0	Risultato 0 1 1 0 0 0 1 1	
Registro B	m 1 1 1 0	m 0 1 1 1	m 0 1 1 1	A_{CC} R_B	
Descrizione operazione	I cifra $R_B = 0$	Scorrimento a destra di A_{CC} e R_B	I cifra $R_B = 1$ $R_A + A_{CC} \rightarrow A_{CC}$	Scorrimento a destra di A_{CC} e R_B. Risultato.	

Fig.IX.18 - Contenuto dell'accumulatore e del registro B ad ogni ciclo dell'operazione di fig.IX.17b.

Il moltiplicatore e il moltiplicando entrano in parallelo nella memoria; l'accumulatore viene azzerato: il circuito di confronto CC_2 esamina il bit meno significativo del moltiplicatore: se è 0 fa scorrere a destra di un posto l'accumulatore e il moltiplicatore che, attraverso il circuito di controllo CC1, riceve nella cella n l'ultimo bit dell'accumulatore. Se la cifra esaminata è 1, attraverso CC1 e CC2, il moltiplicando viene aggiunto al contenuto dell'accumulatore, cifra per cifra.

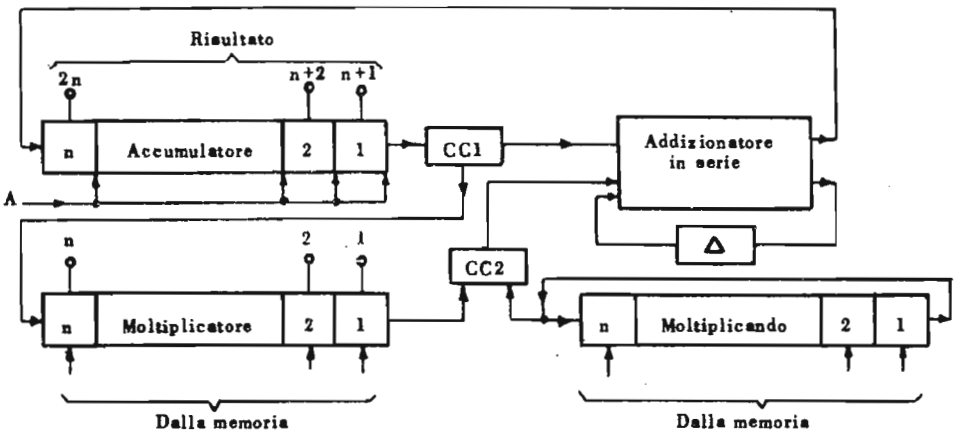


Fig.IX.19 - Moltiplicatore per scorrimento con addizionatore in serie.

Ad ogni impulso di clock, il risultato della somma è messo nello accumulatore e il moltiplicando viene riscritto nel proprio registro. Terminate le somme, un nuovo impulso di scorrimento fa avanzare di un posto l'accumulatore, che attraverso CC1 trabocca sul moltiplicatore; è il moltiplicatore stesso, così, che presenta all'esame di CC2 la cifra successiva. Al termine del procedimento, il risultato è disponibile in parallelo sui 2n terminali dell'accumulatore e del moltiplicatore; i bit più significativi sono nell'accumulatore.

Esistono anche altri metodi di moltiplicazione, ma quelli descritti ci sembrano sufficienti per una trattazione introduttiva dell'argomento.

IX.3.4 - Divisori.

Come la moltiplicazione, anche la divisione può essere realizzata mediante sottrazioni ripetute o per scorrimento, in serie o in parallelo. I circuiti sono simili a quelli del paragrafo precedente, pertanto ci limiteremo a una loro rapida descrizione.

Nel divisore per sottrazioni ripetute, il quoziente è il numero di sottrazioni complete dividendo - divisore. Le sottrazioni vengono eseguite in un Adder col metodo del complemento a 2. Il dividendo è immagazzinato nel registro accumulatore: ad ogni sottrazione è diminuito del valore del divisore, riciclato e riscritto nell'accumulatore. Il procedimento termina quando il dividendo è diventato minore del divisore, se si vuole un quoziente intero, o quando si sono ottenute tutte le cifre richieste, nel caso più generale. Il risultato si legge in un contatore che conta il numero di sottrazioni eseguite.

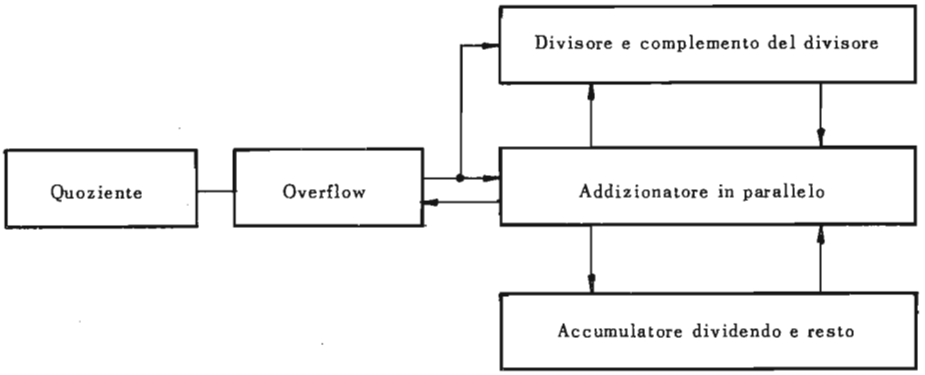


Fig.IX.20 - Divisore in parallelo per sottrazioni ripetute.

Nella fig.IX.20 è riportato uno schema a blocchi per la divisione in parallelo con sottrazioni ripetute. Il circuito effettua le sottrazioni finchè non si arriva a un risultato negativo: dopo il quale torna di un ciclo indietro, e riforma un resto positivo aggiungendo il divisore all'accumulatore. Essenziale è la funzione dell'*overflow*: questo elemento è azzerato quando il dividendo è mandato per la prima volta nell'accumulatore, va a 1 dopo ogni sottrazione la cui differenza è positiva, ed è rimesso a 0 prima della successiva sottrazione. Alla prima differenza negativa, l'overflow rimane a 0: si aggiunge allora il divisore non complementato al risultato negativo e si arresta l'operazione. Il contatore, d'altra parte, avanza di 1 soltanto quando l'uscita dell'overflow è 1.

Nella fig.IX.21 è riportato un circuito per la divisione con scorrimento e sottrazioni in serie. Il registro A contiene il divisore, il registro B il dividendo. La sottrazione A - B fornisce un risultato, che viene immagazzinato di nuovo nel registro A, e un overflow dopo l'ultima sottrazione. Se l'overflow è 0, il divisore è più piccolo della parte del

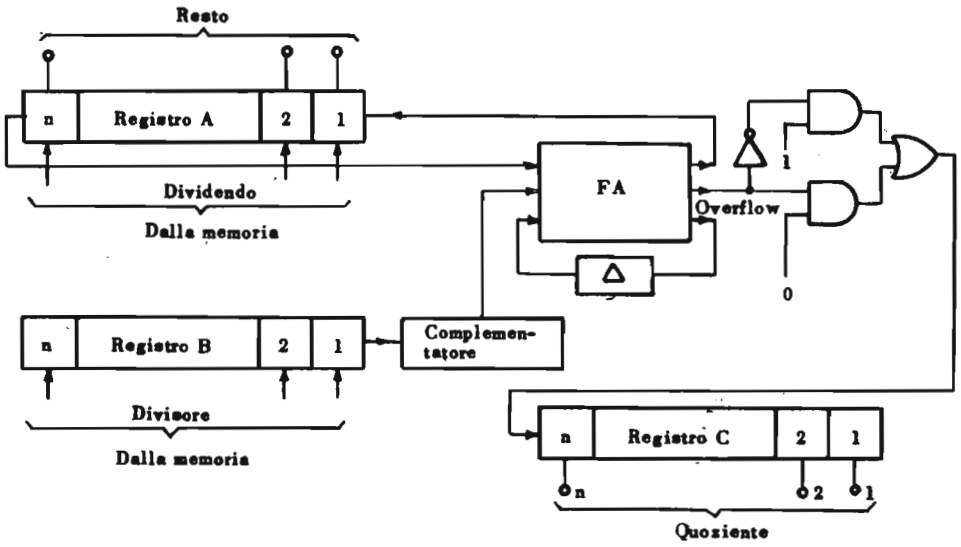


Fig.IX.21 - Divisore con scorrimento e sottrazione in serie.

Registro A	101001000	001001000	0 ← 010010000
Registro C	a) 000000	b) 000000	c) 100000
Registro A	110010000	1 ← 100100000	000100000
Registro C	d) 100000	e) 010000	f) 010000
Registro A	0 ← 001000000	101000000	1 ← 010000000
Registro C	g) 101000	h) 101000	i) 010100
Registro A	110000000	1 ← 100000000	000000000
Registro C	*l*) 010100	m) 001010	n) 001010
Registro A	000000000		
Registro C	o) 100101		

a) Inizio R_C è azzerato; b) $(R_A - R_B) \rightarrow R_A$; c) scorrimento a sinistra di R_A - overflow =0; $1 \rightarrow R_C$; d) $(R_A - R_B) \rightarrow R_A$; e) scorrimento a sinistra di R_A - overflow = $1 - 0 \rightarrow R_C$; f) $(R_A - R_B) \rightarrow R_A$; g) scorrimento a sinistra di R_A - overflow = 0, $1 \rightarrow R_C$; h) $(R_A - R_B) \rightarrow R_A$; i) scorrimento a sinistra di R_A - overflow = 1; $0 \rightarrow R_C$; *l*) $(R_A - R_B) \rightarrow R_A$; m) scorrimento a sinistra di R_A - overflow = $1 - 0 \rightarrow R_C$; n) $(R_A - R_B) \rightarrow R_A$; o) scorrimento a sinistra di R_A - overflow = $0 - 1 \rightarrow R_C$. La divisione è terminata. Il quoziente, scritto da sinistra verso destra, è 101001.

Fig.IX.22 - Contenuti dei registri A e C in ogni fase della divisione 328/8.

dividendo da cui è stato sottratto, e la corrispondente cifra del quoziente è 1. Se l'overflow è 1, il risultato della sottrazione è negativo e la corrispondente cifra del quoziente è 0. Il registro C riceve i bit del risultato e li sposta verso destra ad ogni uscita dell'overflow. Alla fine dell'operazione, il registro A contiene il resto della divisione.

Per maggior chiarezza, nella fig.IX.22 sono mostrati i contenuti dei registri A e C ad ogni passo della divisione 328/8. Il registro B contiene sempre il divisore (1000); avviene quindi una sottrazione tra i primi 4 bit di R_A e 1000, sottrazione che si traduce nell'addizione tra i primi bit di R_A e 1000 (complemento a 2 di 1000).

IX.3.5 - Aritmetica in virgola mobile.

Nella maggior parte dei calcolatori; i numeri vengono rappresentati con tanti bit quanti può contenerne una parola. Si chiama rappresentazione in *virgola fissa* quella in cui, per poter rappresentare anche i numeri decimali, si suppone che tra il bit in posizione K e quello in posizione K + 1 esista una virgola virtuale. Ad esempio, per una parola di 8 bit con virgola tra il 1° e il 2° bit, la stringa:

00110010

rappresenta il binario 0,0110010, cioè il numero decimale 0,390625 (vedi par.I.5.6).

Con la rappresentazione in virgola fissa, per effettuare calcoli su numeri molto grandi e/o molto piccoli sarebbero necessarie parole a molti bit. In genere, tuttavia, non si adotta la soluzione di fissare a priori la posizione della virgola nell'interno della parola, ma si lascia al programmatore la responsabilità di determinarla ad ogni passo del calcolo, utilizzando opportuni *fattori di scala.* In pratica, per ogni numero che entra in macchina, il programmatore userà opportune istruzioni per stabilire la posizione della virgola; shifterà i numeri nei registri, prima di ogni calcolo, in modo da ottenere la maggior precisione possibile e, infine, determinerà la posizione reale della virgola nel risultato ottenuto. Questo lavoro può essere evitato se il calcolatore è organizzato in modo da poter effettuare operazioni su numeri rappresentati con *virgola mobile*. Il progetto di questi calcolatori, d'altra parte, è notevolmente più complicato[1].

(1) - Una soluzione diversa, ma ugualmente comoda per il programmatore, consiste nell'avere a disposizione delle *istruzioni in virgola mobile*, che permettano di usare un calcolatore con unità aritmetica a virgola fissa come se avesse un'unità a virgola mobile. Tale soluzione non verrà qui trattata perchè riguarda il *software* non l'*hardware* del calcolatore.

Un numero viene rappresentato in virgola mobile come l'insieme di una *mantissa* e di un *esponente*. Nel sistema decimale, ad esempio, il numero 30000 si rappresenterà come:

$$3 \times 10^4 \qquad \text{oppure} \qquad 0{,}3 \times 10^5 \quad \text{ecc.}$$

Nel sistema binario, un numero in virgola mobile ha ugualmente una mantissa e un esponente, che occupano posizioni fisse nella parola. In entrambe le due zone, il primo bit è riservato al segno (0 per i numeri positivi, 1 per i negativi). La mantissa è sempre scritta in modo che subito dopo il bit di segno compaia il bit più significativo (cioè con la virgola a sinistra del bit più significativo).

Ecco, ad esempio, l'organizzazione di una parola a 12 bit con 8 bit di mantissa e 4 di esponente (fig.IX.23).

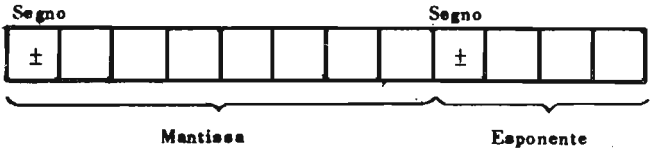


Fig.IX.23 - Organizzazione di una parola contenente un numero rappresentato in virgola mobile.

Se la parola contiene i bit:

010100000001 ,

supponendo di usare la rappresentazione in modulo e segno, la mantissa è (+0,1010000) e l'esponente +001; il numero rappresentato è quindi $+0{,}101 \times 2^{1\,(1)} = 1{,}01\ (= 1{,}25)_{10}$.

Se la parola contiene i bit:

010100001101 ,

la mantissa è +0,1010000 e l'esponente è -010. Il numero rappresentato è $+0{,}101 \times 2^{-2} = 0{,}00101\ (= -1{,}15625)_{10}$.

I vantaggi della rappresentazione in virgola mobile sono:

1) la possibilità di rappresentare con pochi bit numeri molto grandi e/o molto piccoli;

---

(1) - Questa notazione è usata per chiarezza, in realtà si dovrebbe scrivere, servendosi del sistema binario:

$$+0{,}101 \times 10^1 \ .$$

2) l'eliminazione automatica dei bit non significativi: si ottiene così la massima precisione possibile nei risultati, senza intervento del programmatore.

Il modo di operare di una unità aritmetica in virgola mobile varia a seconda del metodo usato per la rappresentazione dei numeri con segno e dell'organizzazione del calcolatore (in serie o in parallelo). In generale, si può dire che la divisione e la moltiplicazione si eseguono direttamente, mentre addizione e sottrazione richiedono operazioni di scalamento; anche nei risultati sono necessari scalamenti, se il valore assoluto della mantissa non è compreso tra 0 e 1.

L'*addizione* viene eseguita con scalatura della mantissa del numero il cui esponente è minore, finchè i due esponenti diventano uguali.

Ad esempio, l'addizione di $0{,}101010 \times 2^2$ con $0{,}111101 \times 2^3$ viene eseguita così:

$$\begin{array}{r} 0{,}10101 \times 2^2 = 0{,}010101 \times 2^3 \; + \\ \underline{\qquad\qquad 0{,}111101 \times 2^3} \\ 1{,}010010 \times 2^3 \end{array}$$

Il risultato deve essere poi messo nella forma $0{,}101001 \times 2^4$.

Le stesse operazioni vengono compiute per la *sottrazione* complementata.

La *moltiplicazione* viene eseguita moltiplicando le mantisse e sommando gli esponenti.

Esempio: $(0{,}1010 \times 2^{-2}) \times (0{,}1000 \times 2^4) = 0{,}0101 \times 2^2 = 0{,}1010 \times 2^1$.

La *divisione*, infine, viene eseguita dividendo le mantisse, sottraendo l'esponente del divisore da quello del dividendo ed, eventualmente, scalando il risultato.

IX.3.6 - Comparazione.

L'unità aritmetica effettua anche le operazioni logiche e quelle di confronto.

I circuiti per effettuare le operazioni logiche non hanno, a questo punto, bisogno di essere illustrati; è invece interessante esaminare come avviene l'operazione di confronto, cioè l'operazione che determina quale dei 2 numeri ad n bit A ($= A_1 A_2 A_3 \ldots A_n$) e B ($= B_1 B_2 B_3 \ldots B_n$) ha valore maggiore. Il circuito per il confronto di A con B ha lo schema a blocchi rappresentato nella fig.IX.24; A_1 e B_1 sono i bit più si-

gnificativi; non si tiene conto del bit di segno, anzi si suppone di confrontare solo valori positivi (l'estensione al caso generale dipende dalla rappresentazione adottata e non presenta difficoltà rilevanti).

La comparazione può essere fatta in serie o in parallelo, coi circuiti appresso descritti.

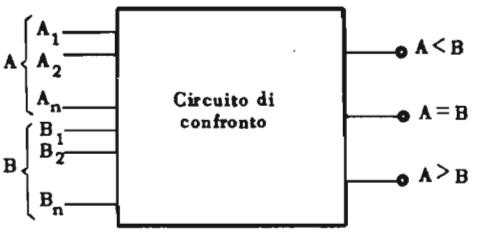
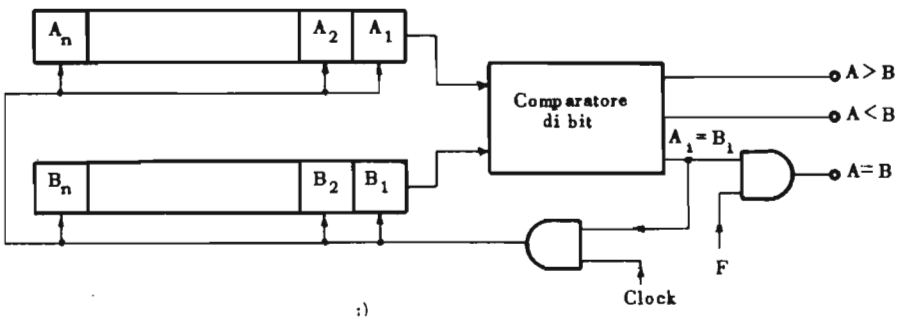


Fig.IX.24 - Schema a blocchi di un circuito per il confronto di 2 numeri positivi.

IX.3.6.1 - Comparatore in serie.

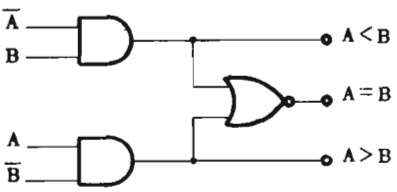
I numeri A e B, entrambi a n bit, vengono scritti in 2 shift-register, col bit più significativo nella posizione più a destra. L'uscita del registro è collegata col circuito comparatore; se il risultato del confron-



a)

A	B	A=B	A>B	A<B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

b)



c)

Fig.IX.25 - Comparatore in serie (a); tabella di verità (b) e realizzazione circuitale (c) di un comparatore di bit.

to di A_1 con B_1 dà $A_1 = B_1$, si ha uno scorrimento a destra, e vengono paragonati i bit A_2 e B_2. Se $A_2 = B_2$, si ha uno scorrimento a destra e vengono paragonati A_3 e B_3. Quando $A_i \neq B_i$, si arrestano i confronti. L'operazione, realizzata nel circuito di fig.IX.25 a, termina quando una delle 3 uscite del circuito è al valore 1. Il segnale F (fine confronto) viene fornito, dopo (n - 1) impulsi di clock, dall'unità di governo. Nelle figg.IX.25 b e c) sono mostrate la tabella di verità e una realizzazione circuitale del comparatore di bit (notare la forma AND-NOR del circuito).

IX.3.6.2 - Comparatore in parallelo.

Il comparatore in serie è un circuito che può impiegare un tempo piuttosto lungo, inaccettabile nei calcolatori in cui la velocità è un requisito essenziale. Si usano, per aumentare la velocità, i *comparatori in parallelo* costituiti da n circuiti comparatori, in cui gli n bit A_i e B_i vengono confrontati separatamente e contemporaneamente. La prima coppia di bit A_j e B_j (j = 1,2,...) che differisce determina l'uscita finale del comparatore, indipendentemente dal valore di A_{j+1} e B_{j+1}, A_{j+2} e B_{j+2}...

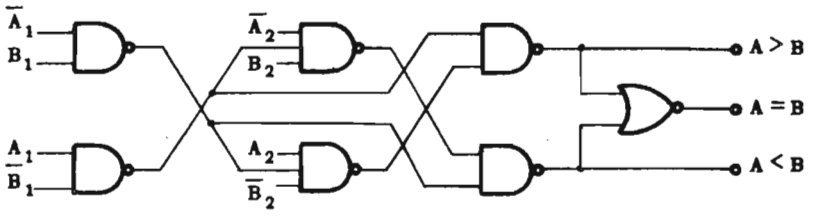


Fig.IX.26 - Circuito per la comparazione in parallelo di numeri a 2 bit.

Nella fig.IX.26 è mostrato, a titolo d'esempio, un circuito per il confronto simultaneo di 2 numeri a 2 bit. Sullo stesso principio si realizzano circuiti per paragonare numeri ad n bit (con n > 2); vengono paragonati i bit di ordine j solo se tutti quelli di ordine k = 1,2,...j - 1 sono uguali: il relativo circuito presenta lo stadio d'uscita uguale a quello di fig.IX.26, e 2 NAND in più per ogni coppia di bit addizionale.

IX.4 - Memoria centrale.

La memoria serve, come si è detto, a immagazzinare istruzioni, dati e risultati parziali.

La sua costituzione, sia per quanto riguarda la capacità, sia per quanto riguarda la velocità con cui le informazioni possono essere lette o scritte, deve essere proporzionata alle altre caratteristiche del calcolatore (velocità di esecuzione delle operazioni aritmetiche, importanza dell'istallazione, ecc.).

La memoria di un calcolatore non è unica: c'è una *memoria centrale* dove vengono immagazzinate le istruzioni e i dati relativi a un programma in esecuzione e ci sono delle *memorie ausiliarie*, in cui vengono conservate grandi quantità di informazioni a disposizione delle memoria centrale. La memoria centrale è caratterizzata da una limitata capacità e da una alta velocità operativa, la memoria ausiliaria ha caratteristiche opposte.

IX.4.1 - Caratteristiche della memoria.

Le principali caratteristiche della memoria, alcune delle quali sono state brevemente descritte nei capitoli precedenti, sono:

– *capacità*: numero di bit immagazzinabili; viene generalmente misurata in caratteri o in parole (vedi par.IX.5);

– *tempo di accesso*: tempo medio necessario per leggere o scrivere un dato in memoria;

– *tipo di accesso*: sequenziale (se il tempo d'accesso dipende dalla posizione del dato in memoria); casuale (o *random*, se il tempo d'accesso è lo stesso per qualsiasi dato). Le memorie sequenziali sono sempre più lente di quelle casuali.

Una memoria è tanto migliore quanto maggiore è la sua capacità, e minori il tempo di accesso, le dimensioni e il costo.

Descriveremo ora la memoria a nuclei, con cui vengono oggi realizzate la maggior parte delle memorie centrali dei calcolatori commerciali.

IX.4.2 - Memoria a nuclei di ferrite.

Un nucleo di ferrite è un anellino dal diametro esterno di pochi millimetri, costruito con materiale magnetico avente un ciclo di isteresi molto stretto. Il nucleo può essere magnetizzato facendo passare una corrente di intensità i opportuna in un filo che la attraversa (fig.IX.27). La direzione della corrente determina la polarità del campo magnetico; il nucleo rimane polarizzato anche quando la corrente è cessata. Per con-

venzione, ad un nucleo magnetizzato positivamente si dà il valore 1; ad uno magnetizzato negativamente, il valore 0: ogni nucleo, quindi, può immagazzinare un bit d'informazione.

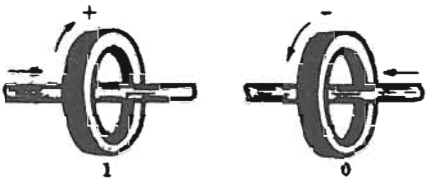


Fig.IX.27 - Relazione tra polarità del campo magnetico e direzione della corrente in un nucleo di ferrite.

La memoria a nuclei è costituita da un certo numero di reticoli (*matrici*) di nuclei di ferrite; ogni nucleo è sostenuto e attraversato da 4 fili: due (X,I) orizzontali, uno (Y) verticale e uno (L) a 45° col filo verticale (fig.IX.28). Le matrici vengono poi disposte secondo tanti piani

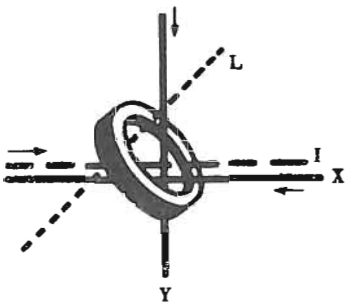


Fig.IX.28 - Nucleo di ferrite in una matrice.

paralleli quanto sono i bit necessari a rappresentare un carattere nel particolare codice usato; sulla perpendicolare ai piani della matrice, in corrispondenza a ogni nucleo, si trovano così tutti i nuclei di un carattere.

A titolo d'esempio, nella fig.IX.28 è mostrato un particolare di una memoria a nuclei per caratteri BCD (di 4 bit); per chiarezza, si sono rappresentati soltanto 3 ca-

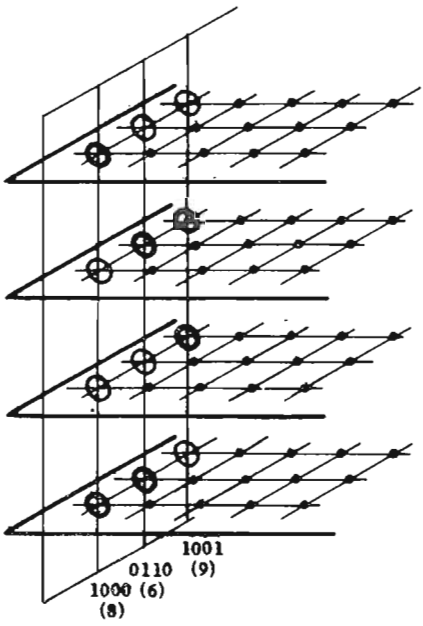


Fig.IX.29 - Particolare di una memoria a nuclei a 4 piani (matrici) per la rappresentazione di caratteri BCD. I caratteri immagazzinabili in memoria sono m × m, se ogni matrice contiene m righe ed m colonne.

ratteri, indicando con un cerchio marcato i nuclei magnetizzati positivamente (bit 1), e con un cerchio a tratto sottile quelli negativi (bit 0); si sono anche tralasciati in fili X, Y, I, L.

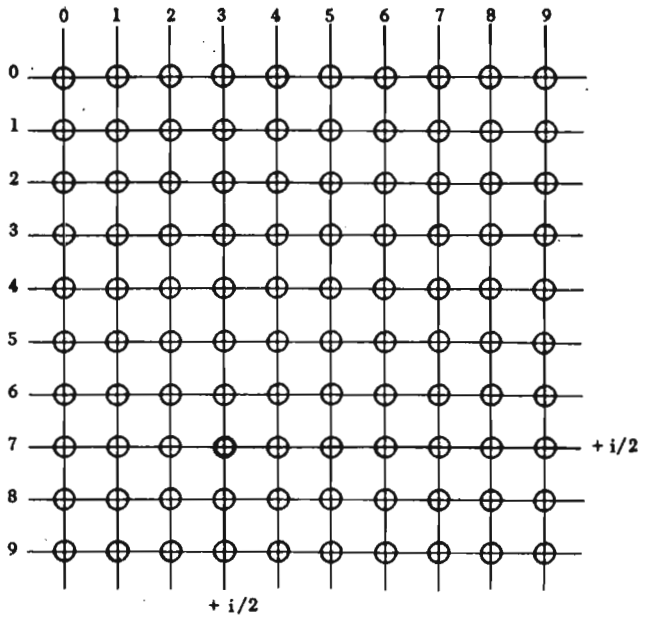


Fig.IX.30 - Scrittura del bit 1 nel nucleo di ascissa 3 e ordinata 7 (nucleo 37) di una memoria a 100 bit.

I conduttori X e Y servono per la scrittura: se + i è la corrente necessaria per portare un nucleo nello stato 1, vengono inviate 2 correnti + i/2 nei conduttori X e Y che attraversano il nucleo stesso.

Ad esempio, per scrivere un bit 1 nel nucleo 37 di un certo piano, si invia una corrente + i/2 sul conduttore X = 3 e sul conduttore Y = = 7 (fig.IX.30). Tutti i nuclei di ascissa 3 (30, 31, ..., 39) e tutti quelli di ordinata 7 (07, 17, ..., 97) sono attraversati da una corrente che non è sufficiente a far variare il loro stato di magnetizzazione; solo il nucleo 37, all'intersezione dei 2 conduttori, viene attraversato dalla corrente + i che lo lascia nello stato 1.

Nella memoria, però, viene scritto non un solo bit alla volta, ma tutti i bit di un carattere insieme. Torniamo, per fissare le idee, alla memoria a 4 piani della fig.IX.29: tutti i conduttori X (Y) che hanno la stessa ascissa (ordinata) sono collegati tra loro, per cui la scrittura del bit 1 nel nucleo 37 della 1ª matrice comporta la scrittura contemporanea

del bit 1 nel nucleo 37 delle altre 3 matrici; la scrittura di un bit 1 implica, pertanto, la scrittura del carattere 1111. Per scrivere qualsiasi carattere (ad esempio 1001) si manda allora, contemporaneamente alla corrente +i/2 su X e su Y, una corrente -i/2 sul filo I (*inibizione*) dei piani (nel nostro caso 3 e 4) dove si deve scrivere 0.

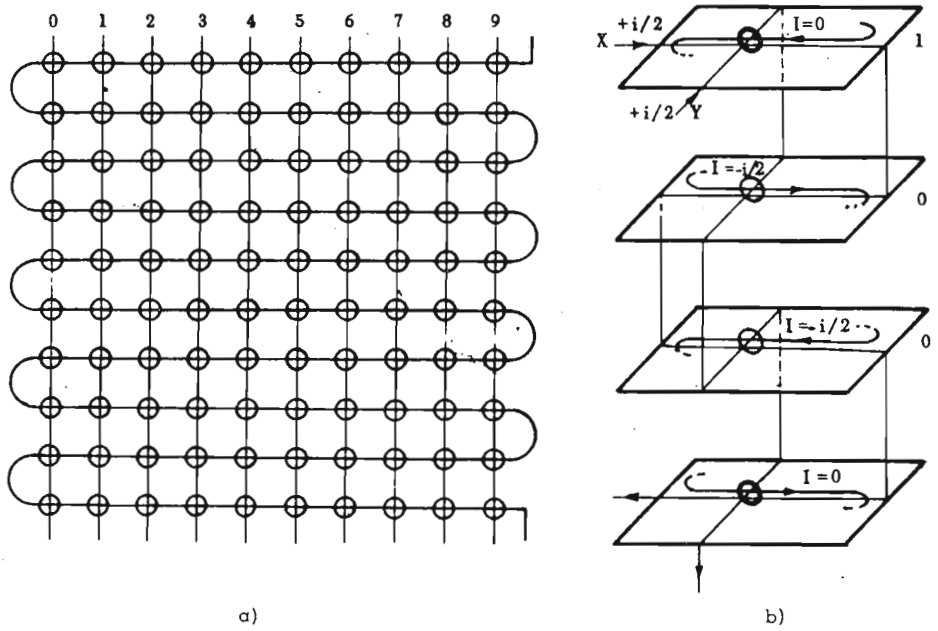


Fig.IX.31 - a) Filo di inibizione del piano i^{mo}; b) scrittura del carattere BCD 1001 nei 4 nuclei di ascissa X e ordinata Y di una memoria a 4 piani.

Il filo di inibizione attraversa tutti i nuclei di un piano, come mostrato nella fig.IX.31 a; esistono tanti fili di inibizione indipendenti quanti sono i piani. Nella fig.IX.31 b è mostrata la scrittura del carattere 1001, mettendo in evidenza soltanto i nuclei e i conduttori interessati alla scrittura.

Per la lettura di un carattere, viene utilizzato il filo L (*lettura*), che attraversa tutti e soli i nuclei di un piano, nel modo indicato nella fig.IX.32 a. Quando si invia una corrente -i/2 sui conduttori X e Y che attraversano i nuclei del carattere che si vuol leggere, i nuclei che si trovano nello stato 1 commutano nello stato 0, e pertanto nei fili di lettura dei corrispondenti piani viene indotto un impulso di corrente. I nuclei che si trovavano a 0 rimangono nel loro stato, e sul corrispondente filo di lettura non si ha alcun segnale. Nella fig.IX.32 b è mostrata la

lettura del carattere 1001, mettando in evidenza soltanto i nuclei e i conduttori interessati.

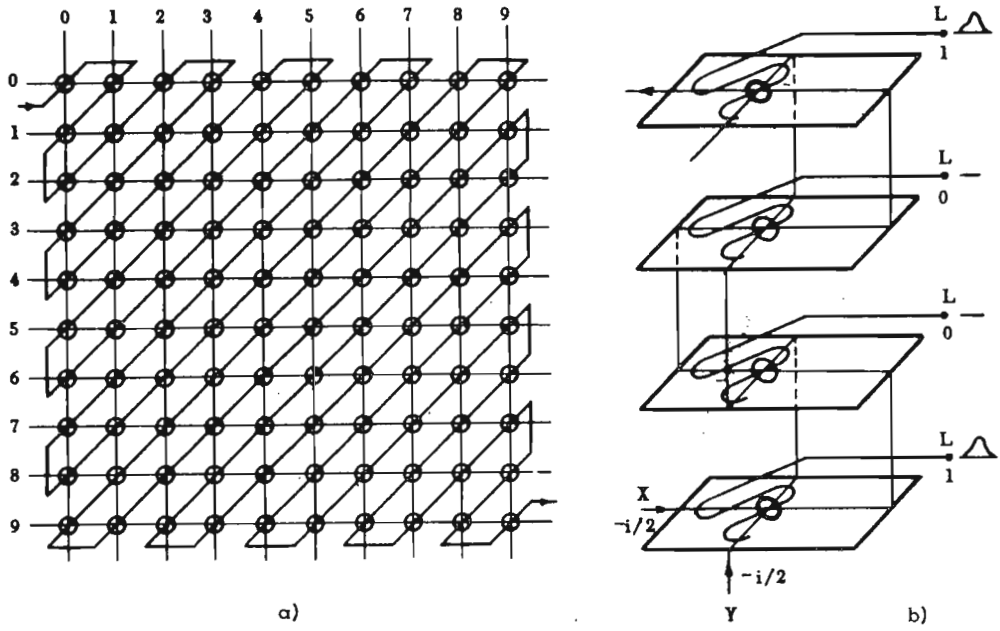


Fig.IX.32 - a) Filo di lettura del piano i^{mo}; b) lettura del carattere 1001 nei 4 nuclei di ascissa X e ordinata Y di una memoria a 4 piani.

Come si vede, il processo di lettura è distruttivo; i bit 1 vengono infatti cancellati, e vanno riscritti. Per questo, si immagazzina il carattere letto in un registro apposito e da qui, con un opportuno comando, il carattere stesso viene inviato da una parte all'unità dove sarà utilizzato (unità di governo, unità aritmetica, ...) dall'altra riscritto in memoria.

Nella fig.IX.33 è mostrato lo schema effettivo di un piano di memoria, con i fili X, Y, L, I. La disposizione alternata dei nuclei facilita la costruzione, in quanto permette che il filo I sia filato orizzontalmente attraverso le righe, e il filo L diagonalmente, in modo da cancellare i piccoli impulsi di interferenza generati dai nuclei attraversati da correnti i/2. Ne consegue che gli impulsi di scrittura in fili X e Y adiacenti devono avere direzioni opposte, in modo che in ogni nucleo l'impulso di inibizione su I si opponga agli impulsi su X e Y.

Molto usati, in pratica, sono i piani di memoria a 64 righe e 64 colonne: in un calcolatore con caratteri a 8 bit, 8 piani di memoria con-

tengono così 4096 caratteri. I tipi di nuclei usati sono molto diversi, e variano a seconda del tempo di lettura e di scrittura.

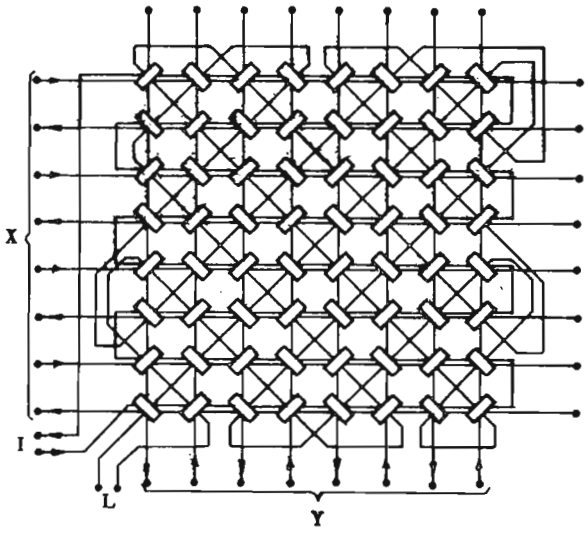


Fig.IX.33 - Schema completo di una matrice a 64 nuclei.

IX.4.2.1 - Organizzazione circuitale della memoria.

Nella memoria che abbiamo descritto sono contenuti $m \times (n \times n)$ anellini di ferrite, capaci di immagazzinare $n \times n$ caratteri di m bit ciascuno. Ogni carattere ha un *indirizzo*, che viene specificato dalle coordinate dei suoi bit (eguali per ogni piano). Gli indirizzi variano da $X = 0$, $Y = 0$ ad $X = n-1$, $Y = n-1$. Per leggere - o scrivere - una determinata configurazione di bit (*dato*) in un carattere, occorre specificare l'indirizzo di memoria del carattere stesso. Associati con l'organo di memoria - cioè con l'insieme delle m matrici - ci sono, per questo, due diversi registri (fig.IX.34) il *registro dell'indirizzo di memoria* (RIM) e il *registro dei dati* (RD).

Il RIM è un registro diviso in due parti, ognuna delle quali ha tante celle quante bastano a rappresentare in binario il numero $n-1$. Esso contiene, dunque, l'indirizzo del carattere in cui occorrerà scrivere, o da cui occorrerà leggere, un dato.

Il RD è un registro ad m celle, e serve a immagazzinare temporaneamente il dato da leggere - o da scrivere - nell'indirizzo XY.

Quando si vuole scrivere un dato in memoria, questo viene inviato nel registro RD dall'unità di governo che, contemporaneamente, invia l'indirizzo XY al RIM. Viene successivamente decodificato, in un'apposita matrice, l'indirizzo contenuto nel RIM, e selezionato il carattere XY; all'arrivo di un segnale di scrittura, il dato immagazzinato nel RD viene scritto nei nuclei del carattere XY.

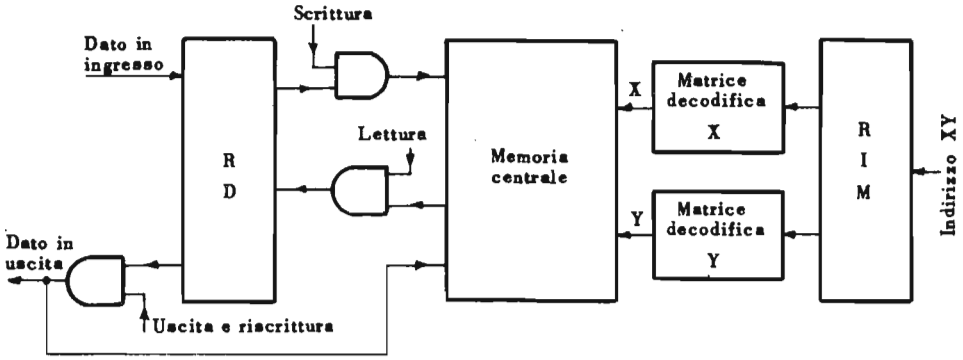


Fig.IX.34 - Organizzazione circuitale della memoria.

Per leggere un dato contenuto nel carattere XY, si invia l'indirizzo XY stesso al RIM, che provvede a decodificarlo e ad individuare il carattere in memoria. L'unità di governo invia quindi il segnale di lettura e il contenuto del carattere XY viene trasferito nel RD. Ad un segnale successivo il dato viene - contemporaneamente - trasferito all'unità che deve utilizzarlo e riscritto in memoria.

A maggio chiarimento di quanto detto, intendiamo richiamare la attenzione sui seguenti punti:

1) quella descritta in figura è una delle possibili organizzazioni della memoria, ed è stata descritta molto in generale;

2) l'indirizzo di un carattere XY, se n è il numero dei fili X e dei fili Y, è composto da $2 \times s$ ($s = \log_2 n$) bit. Di questi $2s$ bit, s entrano nella matrice di decodifica X ed s in quella Y. Ogni matrice ha n uscite;

3) si è supposto di leggere soltanto un carattere; è possibile leggere, contemporaneamente, più caratteri insiemi, ad esempio tutti quelli di una parola (v. par.IX.5.1) in questo caso i registri devono poter contenere tutti i bit della parola.

Occorre tener presente che sono state realizzate, ed usate, memorie a nuclei aventi organizzazione più o meno diversa da quella illustrata (che è chiamata 3 D). Tra queste memorie citeremo:

1) la memoria 2 D, che è organizzata - almeno concettualmente - su 2 dimensioni, invece che su 3, e richiede 2 fili per ogni nucleo;

2) la *memoria* $2^{1/2}$ D, con 3 fili per nucleo, e caratteristiche intermedie tra la 3 D e la 2 D.

Esistono poi altri tipi di memorie centrali, con caratteristiche nuove e importanti. Ad esempio: le memorie NDRO che usano dei cubetti di ferrite e sono a lettura non distruttiva; le memorie *pellicolari a filo*, il cui elemento principale è un filo dal diametro di ~ 1/10 di mm ricoperto da un sottilissimo strato di materiale magnetico: anche questa memoria è a lettura non distruttiva e presenta caratteristiche di velocità e di economicità assai interessanti. Ricordiamo, infine, le memorie a circuiti monolitici, in particolare le memorie permanenti LSIC, che sembrano avviate a diventare le memorie più usate in futuro. La descrizione dettagliata di queste memorie esula dagli scopi introduttivi del presente lavoro; per essa, rimandiamo ai testi citati in bibliografia.

IX.4.3 - Memorie ausiliarie.

Non esiste, da un punto di vista tecnico, alcuna limitazione alle dimensioni di una memoria centrale; esiste però una limitazione di natura economica. Attualmente, al di sopra di 10^7 bit, le memorie a nuclei sono troppo costose. Si ricorre, per capacità superiori, a memorie il cui costo per bit sia molto basso e la cui capacità sia altissima. Tali memorie sono i nastri, i dischi e i tamburi magnetici.

IX.4.3.1 - Memorie a nastri magnetici.

Il nastro magnetico è realizzato con un materiale plastico ricoperto di ossido di ferro; è largo 12 ÷ 15 mm, ed è lungo fino a qualche migliaio di metri. Il nastro è diviso longitudinalmente in *piste* (da 7 a 9) sulle quali vengono memorizzati i caratteri, secondo la larghezza del nastro (fig.IX.35).

Per ogni carattere, la scrittura dei bit viene effettuata magnetizzando o no le zone di nastro all'intersezione con le piste, con una densità variabile da 125 a 3.000 caratteri per pollice. Su una bobina, pertanto, possono essere contenuti più di 10 milioni di caratteri.

La scrittura e la lettura dei bit vengono effettuate da una testina sotto la quale scorre il nastro, svolgendosi da una bobina e riavvolgendosi su un'altra. Le bobine e le testine di lettura sono montate su un'*unità a nastro magnetico* che controlla il movimento del nastro e trasmette, o riceve, i dati dalla memoria centrale.

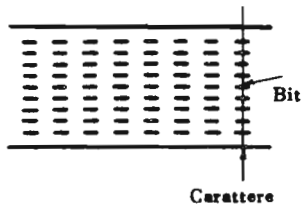


Fig.IX.35 - Nastro magnetico.

Il nastro permette un'elevata velocità di trasferimento dei dati; può immagazzinare un rilevante numero di caratteri, può essere usato più volte, ed ha un basso costo per bit. Presenta, d'altra parte, lo svantaggio di un elevato tempo d'accesso, dell'ordine di parecchie decine di secondi (è infatti un'unità ad accesso sequenziale) e richiede cure particolari per l'uso (ambienti privi di polvere, e a temperatura controllata).

IX.4.3.2 - Memorie a dischi.

La memoria a dischi è costituita da un gruppo di dischi uguali, disposti parallelamente su uno stesso asse, a distanza di circa un centimetro. Le facce dei dischi sono ricoperti di materiale magnetico e suddivise in piste concentriche o *tracce* (qualche migliaia per faccia); lo insieme delle piste su tutti i dischi si chiama *cilindro* (fig.IX.36).

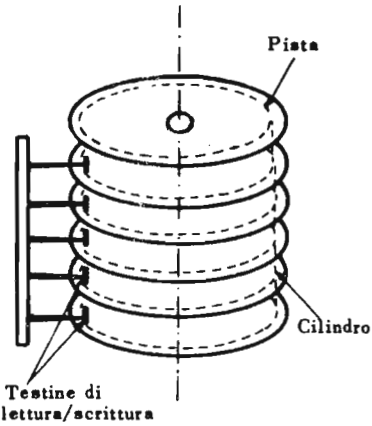


Fig.IX.36 - Dischi magnetici.

I bit che costituiscono un carattere vengono registrati serialmente su una sola pista; ogni pista può contenere migliaia di caratteri, per cui la capacità di una unità a dischi può raggiungere centinaia di milioni di caratteri.

La lettura e la scrittura avvengono per mezzo di apposite *testine*, una per ogni faccia del disco, che si posizionano su una pista con un solo movimento. In ogni posizione, sono accessibili tutte le piste di un cilindro, ma una sola testina è attiva. La scrittura avviene perciò riem-

piendo tutta una pista, poi attivando la testina successiva per scrivere sulla corrispondente pista dello stesso cilindro; quando tutte le piste di un cilindro sono esaurite, si passa al cilindro successivo. La lettura avviene posizionando le testine sul cilindro dove si trova il dato da leggere, attivando la testina corrispondente a una certa pista, ed esaminando i caratteri sulla pista stessa, mentre l'insieme dei dischi ruota.

Il pacco di dischi è montato su una *unità a dischi* e può essere facilmente sostituito; l'unità è considerata ad accesso diretto, in quanto un dato può essere localizzato senza esaminare tutti i precedenti, come avviene per il nastro. La capacità è molto alta, il tempo d'accesso relativamente basso (da 15 a 700 msec).

IX.4.3.3 - Memorie a tamburo.

Il tamburo è un cilindro ricoperto di materiale ferromagnetico e ruotante attorno a un asse (fig.IX.37). La superficie è divisa in parecchie centinaia di zone chiamate *piste*, di fronte alle quali sono collocate le testine di lettura/scrittura (una per pista).

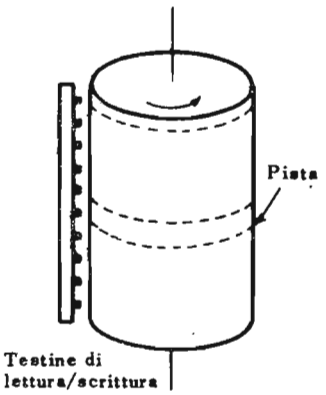


Fig.IX.37 - Tamburo magnetico.

Il tamburo può essere organizzato in serie o in parallelo: i bit di ogni carattere, cioè, possono essere scritti in serie su una pista, o in parallelo su più piste. Il tamburo ruota costantemente, ed è montato rigidamente sull'*unità a tamburo magnetico*, che effettua e controlla il trasferimento dei dati da e verso la memoria centrale.

Il tamburo è considerato un'unità ad accesso diretto; a parità di volume ha una capacità minore dei dischi, ma un più breve tempo d'accesso (da 9 a 100 millisecondi).

Dopo aver descritto le caratteristiche fisiche della memoria centrale e delle memorie ausiliarie, è a questo punto necessario parlare dell'organizzazione logica dei dati in memoria. La conoscenza di questo argomento è infatti indispensabile per introdurre le istruzioni e descrivere l'unità di governo.

IX.5 - Organizzazione della memoria.

Il termine *parola* è stato usato nel cap.I, parlando di codici, e nel capitolo precedente parlando di registri. In questo capitolo, dedica-

to ai calcolatori, abbiamo usato più volte il termine *carattere*. Vogliamo ora illustrare le differenze e le analogie tra questi 2 termini, che talvolta sono semplicemente sinonimi.

Riprendiamo, dunque, la descrizione della memoria centrale, che avevamo visto essere organizzata in caratteri, cioè in insiemi di nuclei individuati in modo univoco dalle coordinate X e Y. La memoria a nuclei è formata da un insieme di m caratteri, misurati in multipli di K($K = 2^{10} = 1024$). Ad esempio, una memoria di 16 K contiene $16 \times 1024 = 16384$ caratteri; ognuno dei 16384 caratteri è individuato da un indirizzo che va da 0 a 16383, ed è esprimibile con 14 bit ($2^{14} = 16384$).

I caratteri sono, generalmente, di 6 o 8 bit. Per fissare le idee, supponiamo che i caratteri abbiano 8 bit. Ogni *carattere* della memoria può così immagazzinare $2^8 = 256$ configurazioni di bit diversi. Ogni configurazione, secondo un determinato codice, ha un preciso significato, cioè rappresenta un carattere dell'alfabeto, o una cifra decimale, o un segno speciale (= , ! ≥ ecc.). Per evitare confusione tra il «carattere» inteso come insieme di nuclei e il «carattere» inteso come lettera dello alfabeto o cifra o segno speciale, indicheremo l'insieme di nuclei con il termine inglese *byte*. Diremo quindi, d'ora in poi, che un *byte* rappresenta un *carattere* secondo un determinato codice.

Generalmente, le informazioni che si trovano in memoria occupano una zona contigua di byte: questa zona si chiama *campo di memoria*: è individuata dall'indirizzo del suo primo byte e dalla sua lunghezza (in byte). Ad esempio, nella fig.IX.38 è mostrato un campo di 5 byte, di indirizzo 382: esso può contenere un'informazione di 5 caratteri.

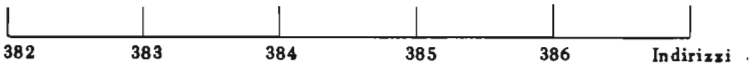


Fig.IX.38 - Campo di memoria di 5 byte (40 bit).

Quando i campi possono avere lunghezza qualsiasi, si dice che la memoria è organizzata in caratteri; quando i campi hanno tutti la stessa lunghezza, la memoria è organizzata in parole. Molti calcolatori possono essere organizzati sia a caratteri che a parole.

Abbiamo detto all'inizio del capitolo che un calcolatore esegue un *programma* composto da *istruzioni* su dei *dati* in memoria. Dati e istruzioni si trovano, al momento dell'esecuzione del programma, insieme in memoria; le zone di memoria occupate dalle istruzioni e quelle occupate

dai dati si presentano sempre come stringe di bit. Dati e istruzioni sono però rappresentati in modo logicamente diverso. Cominciamo a vedere come sono rappresentati i dati.

IX.5.1 - Rappresentazione dei dati.

I dati vengono rappresentati in modo diverso a seconda che siano di tipo numerico o non numerico. Il primo tipo comprende numeri su cui bisogna effettuare dei calcoli; il secondo, ogni tipo di carattere, comprese cifre decimali formanti stringhe su cui non si devono effettuare calcoli aritmetici.

IX.5.1.1 - Rappresentazione dei dati non numerici.

I codici a otto bit attualmente più usati sono: il codice ASCII e il codice EBCDIC.

Il codice ASCII (American Standards Code for Information Interchange) è un codice capace di codificare 256 simboli, cioè: le 26 lettere dell'alfabeto inglese minuscole e maiuscole, un elevato numero di caratteri speciali (es.: ? , * ,) e tutta una serie di comandi operativi (esempio: *fine messaggio, ritorno del carrello, salto di una riga*).

Nella fig.IX.39 sono rappresentati i più importanti simboli di questo codice. Per semplicità si è usata la notazione esadecimale;pertanto, le scritte «51» e «B4», rappresentanti il numero 1 e la lettera T vanno interpretate, rispettivamente, come «01010001» e «10110100».

I bit di ogni carattere vengono divisi in due gruppi di 4, ognuno rappresentato da una cifra esadecimale; i primi 4 bit si chiamano bit di zona (Z); i secondi, bit numerici (N). Per le cifre decimali, Z=5; per le lettere maiuscole, Z = A o B. Si noti che le cifre hanno la parte numerica coincidente con la rappresentazione BCD.

Anche il codice EBCDIC (Extended Binary Coded Decimal Interchange Code) è a otto bit, e codifica pressapoco gli stessi simboli dell'ASCII.

Nella fig.IX.40 sono rappresentati alcuni di questi simboli; si è usata ancora la notazione esadecimale. Vale, inoltre, quanto detto per il codice ASCII circa la rappresentazione delle lettere e dei numeri; si ha però: Z = F per le cifre decimali, e Z = C,D,E per le lettere maiuscole.

TABELLA IX.1 - Caratteri del codice ASCII							
Carattere	ASCII	Carattere	ASCII	Carattere	ASCII	Carattere	ASCII
0	50	>	5E	A	A1	O	AF
1	51	?	5F	B	A2	P	B0
2	52	!	41	C	A3	Q	B1
3	53	#	43	D	A4	R	B2
4	54	$	44	E	A5	S	B3
5	55	%	45	F	A6	T	B4
6	56	&	46	G	A7	U	B5
7	57	(	48	H	A8	V	B6
8	58	)	49	I	A9	W	B7
9	59	*	4A	J	AA	X	B8
:	5A	+	4B	K	AB	Y	B9
;	5B	-	4D	L	AC	Z	BA
<	5C	/	4F	M	AD	,	4C
=	5D	Spazio	40	N	AE	.	4E

Fig.IX.39

TABELLA IX.2 - Caratteri del codice EBCDIC							
Carattere	EBCDIC	Carattere	EBCDIC	Carattere	EBCDIC	Carattere	EBCDIC
0	F0	>	6E	A	C1	O	D6
1	F1	?	6F	B	C2	P	D7
2	F2	!	5A	C	C3	Q	D8
3	F3	#	7B	D	C4	R	D9
4	F4	$	5B	E	C5	S	E2
5	F5	%	6A	F	C6	T	E3
6	F6	&	50	G	C7	U	E4
7	F7	(	4D	H	C8	V	E5
8	F8	)	5D	I	C9	W	E6
9	F9	*	5C	J	D1	X	E7
:	7A	+	4E	K	D2	Y	E8
;	5E	-	6D	L	D3	Z	E9
<	4C	/	61	M	D4	,	6B
=	7E		40	N	D5	.	4B

Fig.IX.40

IX.5.1.2 - Rappresentazione dei dati numerici.

I numeri hanno 4 rappresentazioni diverse: due decimali e due binarie. Le rappresentazioni decimali rappresentano i numeri convertendo separatamente, nel codice ASCII o EBCDIC o BCD, ogni cifra.

Si chiama *zoned* (o *unpacked*) la rappresentazione di un numero decimale in uno dei codici ASCII o EBCDIC. Questa rappresentazione tratta un numero come una qualsiasi stringa di caratteri, e occupa tanti byte quante sono le cifre del numero stesso.

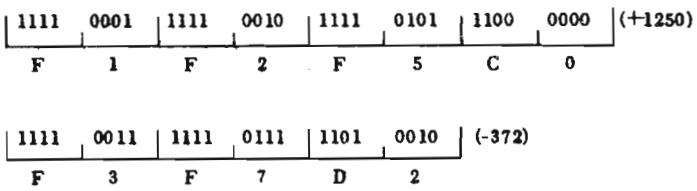


Fig.IX.41 - Rappresentazione *zoned* dei numeri +1250 e -372 (sotto ogni bytes è riportato il contenuto scritto in esadecimale).

Nella fig.IX.41 sono mostrate le rappresentazioni EBCDIC dei numeri +1250 e -372; il segno + o - viene inserito nell'ultima zona ed è, per convenzione, C o D.

La rappresentazione *packed* si ottiene dalla *zoned* eliminando i bit di zona (che sono sempre uguali per le cifre decimali) e spostando i bit di segno all'estrema destra del campo. Nella fig.IX.42 è mostrata la rappresentazione *packed* dei numeri +1250 e -372. Apposite istruzioni permettono il passaggio dall'una all'altra forma di rappresentatione dedecimale.

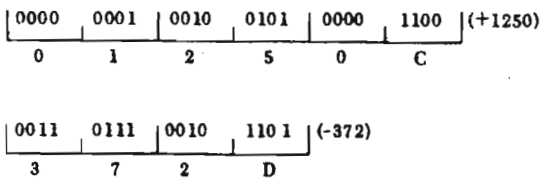


Fig.IX.42 - Rappresentazione *packed* dei numeri +1250 e -372.

Le rappresentazioni binarie sono quelle che abbiamo già viste, cioè quelle in virgola fissa o in virgola mobile. La lunghezza dei dati può

variare da 2 a 8 bytes, a seconda del tipo di calcolatore, della lunghezza del numero e delle istruzioni usate per eseguire i calcoli.

IX.5.2 - Rappresentazione delle istruzioni.

In memoria, insieme ai dati, sono immagazzinate le istruzioni di un programma. Prima di descrivere, sia pure sommariamente, i tipi e le caratteristiche delle istruzioni stesse, è necessario dire qualcosa sui programmi.

Il *programma* è una sequenza di istruzioni caricate in memoria prima che il calcolatore inizi a funzionare. Il calcolatore preleva ed esegue ogni istruzione del programma. Le istruzioni, come del resto i dati, sono codificate secondo regole precise. Quando le istruzioni sono codificate in una forma *direttamente* interpretabile del calcolatore si chiamano *istruzioni di macchina*. Tutte le istruzioni di cui ci occuperemo saranno in questa forma.

Esistono parecchie classificazioni delle istruzioni, basate sulla loro forma o sulla loro funzione. Poichè stiamo ora trattando la rappresentazione delle istruzioni in memoria, ci riferiremo alla prima classifica.

Le istruzioni possono comprendere un numero qualsiasi di caratteri ma, in pratica, occupano un campo di lunghezza fissa. Debbono sempre contenere un *codice operativo*, che dica al calcolatore cosa deve fare, e almeno un *indirizzo* di memoria. Le istruzioni vengono, anzi, classificate in base al numero di indirizzi che seguono il codice operativo: si hanno così istruzioni a uno, due, tre e quattro indirizzi.

La fig.IX.43 mostra un'istruzione a 4 indirizzi; il primo byte contiene il codice operativo; seguono 4 coppie di byte che individuano 4 indirizzi di memoria.(A,B,C,D) ciascuno di 16 bits. Dalle estensioni delle cinque zone di memoria si ricava che:

1) sono possibili $2^8 = 256$ configurazioni diverse del codice operativo, per cui il calcolatore può avere fino a 256 istruzioni diverse;

2) la memoria del calcolatore comprende $2^{16} = 65 \cdot 536$ bytes.

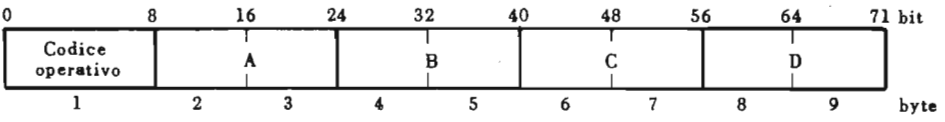


Fig.IX.43 - Istruzione a 4 indirizzi.

Un'esempio di istruzione a 4 indirizzi potrebbe essere una interpretata dal calcolatore come: «Somma il numero contenuto nella parola di memoria di indirizzo A al numero contenuto nella parola di indirizzo B; metti il risultato nella parola di indiririzzo C; preleva la prossima istruzione da eseguire dalla parola di indirizzo D».

La fig.IX.44 mostra un'istruzione a 3 indirizzi. Un'istruzione a 3 indirizzi potrebbe essere: «Somma il contenuto della parola di indirizzo A al contenuto della parola di indirizzo B, e scrivi il risultato nella parola di indirizzo C». Il calcolatore dovrebbe automaticamente prelevare la prossima istruzione da eseguire nel campo di 7 byte che, in memoria, segue quello contenente l'istruzione eseguita. Questa esecuzione sequenziale delle istruzioni deve però poter essere interrotta in qualche modo, con istruzioni particolari a disposizione del programmatore (vedi il prossimo paragrafo).

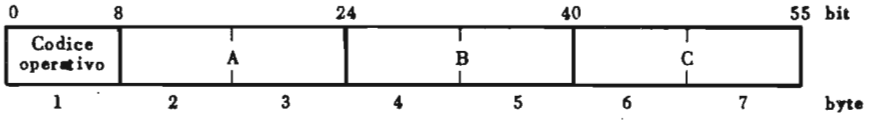


Fig.IX.44 - Istruzione a 3 indirizzi.

Talvolta, le istruzioni a 3 indirizzi sono del tipo: «Somma il contenuto di A al contenuto di B, e preleva la prossima istruzione da C». In questo caso, il risultato dell'addizione viene immagazzinato in un particolare registro.

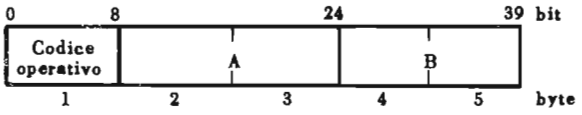


Fig.IX.45 - Istruzione a 2 indirizzi.

La fig.IX.45 mostra il tracciato di un'istruzione a 2 indirizzi, ad esempio: «Somma il contenuto di A al contenuto di B». Il risultato è immagazzinato in un registro, e può essere riportato in memoria con un'altra istruzione. L'esecuzione delle istruzioni è sequenziale, e deve poter essere interrotta con istruzioni particolari.

La fig.IX.46 mostra infine un'istruzione a 1 indirizzo, ad esempio: «Somma il contenuto dell'accumulatore al contenuto di A». Il risul-

tato rimane nell'accumulatore. Per compiere le stesse funzioni dell'istruzione di somma a 4 indirizzi, occorre in questo caso:

1) trasferire B nell'accumulatore;
2) sommare A all'accumulatore;
3) trasferire il contenuto dell'accumulatore in C.

L'esecuzione delle istruzioni è sequenziale e, al solito, deve poter essere interrotta.

Fig.IX.46 - Istruzione a 1 indirizzo.

E' chiaro, da quanto esposto, che all'aumentare degli indirizzi di memoria specificati, aumenta la lunghezza dell'istruzione, ma diminuisce il numero delle istruzioni necessarie per eseguire un certo lavoro. Le istruzioni illustrate, d'altra parte, sono puramente ipotetiche: in realtà assumono forme più complesse e organizzazioni concettualmente diverse. I calcolatori, inoltre, usano in generale un *repertorio di istruzioni* comprendente istruzioni di diverse lunghezze.

IX.5.3 - Tipi di istruzioni di macchina.

Ai fini di una migliore comprensione delle funzioni dell'unità di controllo, è utile descrivere le istruzioni a seconda delle loro funzioni, prescindendo dalla loro lunghezza. Si distinguono, in questo caso:

– *istruzioni di trasferimento*, che effettuano trasferimenti di dati tra 2 zone di memoria o tra la memoria e un'altra unità;

–*istruzione airtmetiche*, per eseguire operazioni tra registri e/o zone di memoria;

– *istruzioni di shift*, per manipolare e shiftare parole contenute in registri; contengono, invece dell'indirizzo, l'ampiezza dello shift da effettuare;

– *istruzioni logiche*, per eseguire operazioni booleane (AND,NOT,OR...) tra due stringhe di bit in memoria e/o in particolari registri;

– *istruzioni di controllo*, che modificano l'esecuzione di un programma; comprendono istruzioni di salto (condizionato o no), istruzioni per modi-

ficare altre istruzioni, istruzioni per il controllo delle unità di ingresso/ /uscita.

IX.6 - L'unità di controllo.

L'unità di controllo è la parte più complessa del calcolatore. Essa provvede a prelevare le istruzioni dalla memoria nella esatta sequenza logica, le decodifica, preleva i dati dalle relative zone di memoria, li invia all'unità in cui devono essere utilizzati (es.: unità aritmetica, unità di uscita, ...), genera tutti i segnali da inviare ai vari circuiti coinvolti nell'esecuzione di un'istruzione, fornisce indicazioni di allarme per errori di programma o di macchina.

Un programma, prima di essere eseguito da un calcolatore, deve entrare in memoria, completo di istruzioni e di dati. Ogni istruzione risiede in un particolare campo di memoria di cui è noto l'indirizzo. Per eseguire un programma, il calcolatore deve prelevare la prima istruzione ed eseguirla, poi prelevare ed eseguire le successive istruzioni in memoria. L'individuazione della locazione di memoria dove è immagazzinata la prossima istruzione da eseguire, quando il calcolatore non usa istruzioni a 4 indirizzi (nel qual caso è immediata), viene eseguita con un particolare contatore chiamato il *Registro dell'indirizzo delle istruzioni* (RII). Il RII contiene, all'inizio del lavoro, l'indirizzo della prima istruzione del programma. Questo indirizzo, appena inizia l'esecuzione del programma stesso, viene inviato nel RIM (vedi par.IX.4.2.1 e figura IX.34), dove viene decodificato. L'istruzione contenuta nell'indirizzo stesso - cioè la prima istruzione del programma - passa poi in una seconda sezione dell'unità di governo, il *Registro delle Istruzioni* (RI). Il RI trasferisce (fig.IX.47) il codice operativo dell'istruzione ad un *circuito decodificatore delle istruzioni* (CDI) e l'indirizzo contenuto nell'istruzione (si è considerato, per semplicità, il caso di istruzioni ad un solo indirizzo) al RIM che decodifica l'indirizzo da cui prelevare il dato nella memoria centrale. Il CDI è una matrice (completa o no) che ha tanti ingressi quanti sono i bit del codice operativo, e tante uscite quante sono le possibili operazioni del calcolatore: ogni uscita provvede ad attivare i circuiti necessari ad eseguire la corrispondente operazione.

Mentre vengono effettuati questi trasferimenti, il RII viene automaticamente incrementato di tante unità quanti sono i byte di un'istruzione (2, in questo caso).

L'istruzione viene quindi eseguita; quando l'esecuzione è terminata, il contenuto del RII viene trasferito al RIM; la seconda istruzione

del programma viene scritta nel RI; il RII viene incrementato di 2 (supponendo che tutte le istruzioni siano a 1 indirizzo), e il ciclo ricomincia. Ovviamente, perchè il programma venga eseguito correttamente, è necessario che le istruzioni occupino campi di memoria consecutivi.

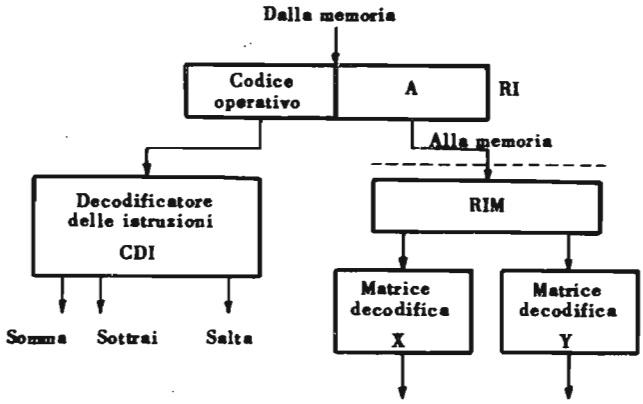


Fig.IX.47 - Decodificatore di una istruzione a 1 indirizzo nell'unità di governo.

In ogni programma, come è noto, è essenziale la possibilità di interrompere la esecuzione sequenziale, *saltando* ad un'istruzione contenuta in un campo di memoria diverso da quello indicato nel RII. Questi salti possono essere condizionati, dipendere cioè dal verificarsi di certe condizioni, o incondizionati. Con le istruzioni ad un indirizzo, un metodo seguito per realizzare salti incondizionati è quello di usare un'istruzione il cui codice operativo agisca proprio sul RII, che significhi cioè: « Sostituisci il contenuto del RII con X ». La prossima istruzione viene pertanto prelevata dall'indirizzo X.

Un'istruzione di salto incondizionato può avere un codice operativo che agisce sul RII solo se si verifica una determinata condizione. Ad esempio: « Sostituisci il contenuto del RII con X, se il contenuto dell'accumulatore è negativo ».

Altri modi di scrivere e realizzare le istruzioni di salto coinvolgono l'uso di *registri indice* o di metodi di *indirizzamento indiretto*: per essi rimandiamo ai testi specializzati. Basterà dire, soltanto, che il registro indice è un particolare registro il cui contenuto (nel nostro caso 2) si somma al RII: scrivendo un numero Y in questo registro, la prossima istruzione eseguita dopo quella di indirizzo N avrà indirizzo N + Y invece del N + 2. L'indirizzamento indiretto si ha invece quando l'indirizzo specificato in una istruzione è quello in cui è scritto l'indirizzo del dato da usare, non il dato stesso.

IX.6.1 - Segnali di temporizzazione.

Nel paragrafo precedente abbiamo descritto come vengono prelevate e decodificate le istruzioni di un programma. All'operazione di decodifica segue l'esecuzione dell'istruzione, esecuzione che occupa tempi - e impegna circuiti - diversi, se diverse sono le istruzioni.

Abbiamo visto (fig.IX.47) come dal CDI esca una sola linea al valore 1, a seconda del codice operativo dell'istruzione. Questa linea deve comandare opportunamente la distribuzione dei segnali di controllo ai circuiti interessati all'esecuzione dell'istruzione stessa.

I segnali si ottengono da un *distributore di impulsi*, con le tecniche descritte nel capitolo precedente. A titolo d'esempio, nella figura IX.48 a sono mostrati i circuiti che intervengono per eseguire un'operazione che implichi la sequenza temporale illustrata nella fig.IX.48 b.

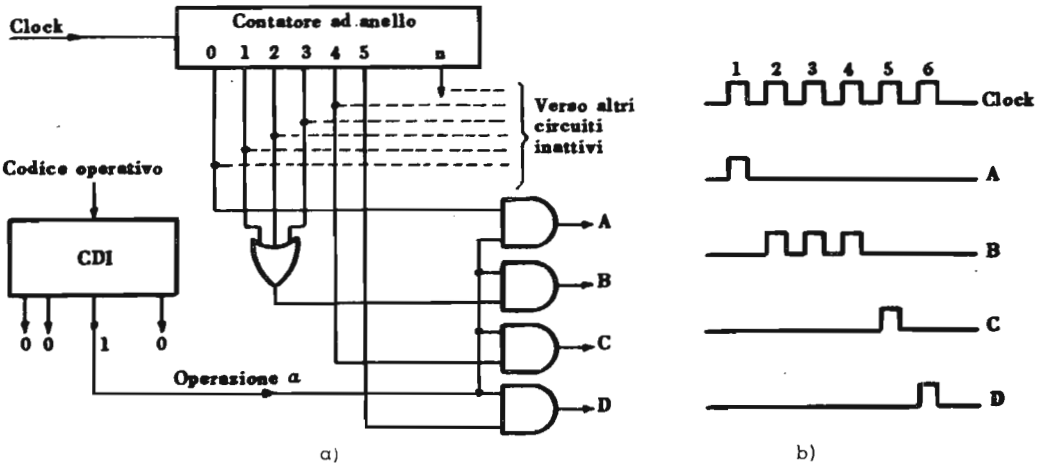


Fig.IX.48 - Sequenza temporale (b) e sua realizzazione (a) per l'operazione α.

IX.6.2 - Circuiti per il trasferimento dei dati.

L'unità di governo deve contenere i circuiti necessari per il trasferimento di dati tra registri, nei 2 sensi, e per shiftare i dati tra un registro e l'altro. Un modo per risolvere il problema è, chiaramente, quello di collegare direttamente i registri stessi; nei calcolatori però, dato il numero elevato dei registri, questa soluzione è praticamente impossibile. Si usano invece dei particolari circuiti chiamati *transfer-bus*. Questi non sono altro che normali circuiti selezionatori-istradatori, comandati

da appositi segnali di temporizzazione, e collegati nei 2 sensi con tutti i registri. Nella fig.IX.49 è mostrato uno di questi circuiti mentre effettua un trasferimento dei dati tra il registro B e il registro E (questi circuiti sono, ovviamente, tanti quanti sono i bit da trasferire in parallelo tra i registri).

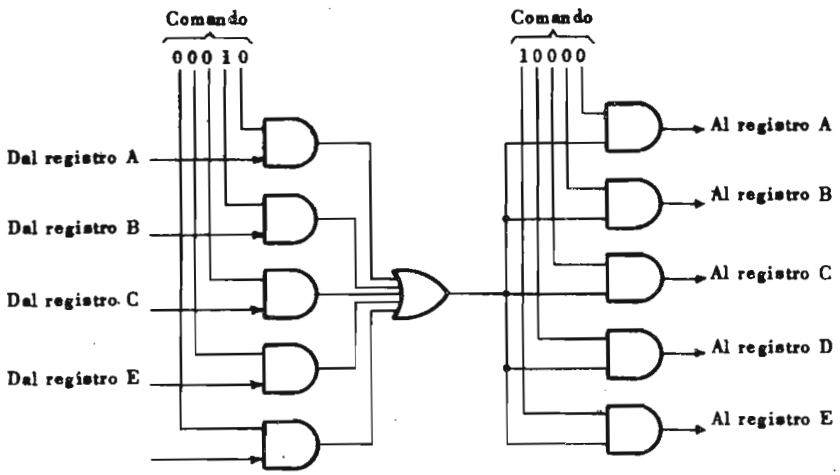


Fig.IX.49 - Circuito per il trasferimento di un bit dal registro B al registro E (se il registro ha 32 bit, e il trasferimento avviene in parallelo, esistono 32 di questi circuiti; tutti i relativi fili di comando sono collegati).

IX.6.3 - Governo a microprogramma.

Un metodo totalmente diverso da quello descritto per organizzare l'unità di governo è quello del *governo a microprogramma*; con questo sistema, ogni istruzione del repertorio delle istruzioni del calcolatore viene divisa in una serie di operazioni elementari fondamentali, ed eseguita secondo la sequenza delle operazioni stesse.

Alla fine degli anni '50, infatti, la somiglianza tra la sequenza dei segnali di controllo generati dall'unità di governo in seguito alla decodifica del codice operativo di un'istruzione in linguaggio macchina, e la successione delle istruzioni stesse in un programma, suggerì di realizzare una *unità di controllo a microprogramma*. Tali unità sono attualmente usate per alcuni calcolatori a media velocità, nei quali i segnali

di controllo alle varie sezioni, e ai relativi circuiti, sono generati da un vero e proprio secondo calcolatore (il *microcalcolatore*) il cui repertorio di istruzioni comprende tutte e sole istruzioni elementari, eseguibili cioè in un intervallo tra 2 impulsi di clok (*microistruzioni*). Il microcalcolatore, inoltre, non è in grado di eseguire qualsiasi programma, ma soltanto quei programmi che coincidono con le istruzioni del repertorio del calcolatore. Ovviamente, le microistruzioni comprendono anche istruzioni di salto, che possono essere eseguite o meno a seconda del risultato di test compiuti da precedenti microistruzioni sulle istruzioni provenienti dalla memoria del calcolatore.

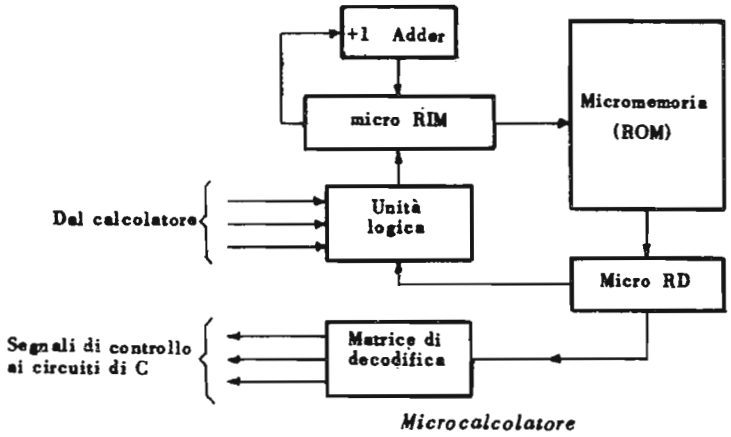


Fig.IX.50 - Unità di controllo a microprogramma.

Nella fig.IX.50 è mostrata l'organizzazione di un microcalcolatore, cioè dell'unità di governo di un calcolatore C. Il microcalcolatore ha una propria unità di memoria (micromemoria) in cui è immagazzinato il microprogramma, in successive parole di memoria. Poichè il microprogramma è univocamente determinato, una volta fissato il repertorio delle istruzioni di C, la micromemoria è una memoria a sola lettura (ROM = = *Read Only Memory)* il cui contenuto è stato fissato, una volta per tutte, dal progettista del calcolatore. Si noti, per inciso, che questa tecnica presenta alcuni notevoli vantaggi tra i quali: il minor prezzo e la maggior rapidità delle ROM, nonchè la possibilità di cambiare tutto il repertorio delle istruzioni del calcolatore cambiando *soltanto* la micromemoria.

Nel microcalcolatore esiste un micro-registro degli indirizzi di memoria (*micro*-RIM) con un addizionatore per incrementare di 1 l'indirizzo stesso (la ROM è sempre organizzata in parole di 1 carattere). Poichè il microcalcolatore esegue soltanto istruzioni di controllo e legge dati soltanto nella ROM, il micro-RIM serve soltanto per accedere alle microistruzioni, quindi ha una costituzione molto semplice ed è in pratica, il RII del microcalcolatore. Esso contiene infatti, in ogni istante, l'indirizzo della prossima microistruzione da prelevare dalla ROM. Ogni microistruzione prelevata dalla ROM, viene trasferita al micro-RD che provvede anche a decodificarla per determinare se vanno inviati dei segnali di controllo ai circuiti di C, attraverso un'opportuna matrice di decodifica, o se va eseguito un test su un dato proveniente dal C. In quest'ultimo caso, se il risultato del test, eseguito nell'apposita unità logica, è positivo, viene trasferito un nuovo indirizzo dal RD al RIM del microcalcolatore.

Si noti che si è descritto uno schema molto semplificato del microcalcolatore, senza dire come vengono agganciate le prime microistruzioni da eseguire in seguito alla lettura di ogni istruzione, nè quali e quanti bit di un'istruzione o di una parola di memoria vengono trasferiti all'unità logica.

Tutti questi problemi, del resto, sono risolti in maniera molto diversa dai vari costruttori, e la loro discussione non potrebbe prescindere dalla considerazione di una macchina particolare, o almeno da una approfondita conoscenza del software.

IX.7 - Unità di ingresso-uscita.

La memoria centrale riceve le informazioni costituenti il programma e i dati su cui esso deve operare attraverso particolari *unità periferiche*; altre unità periferiche ricevono dalla memoria i risultati del programma.

Le unità periferiche di ingresso sono delle apparecchiature *esterne* al calcolatore in grado di accettare informazioni contenute su appositi supporti delle informazioni e di inviarle alla memoria centrale; le unità periferiche di uscita sono apparecchiature, anch'esse esterne al calcolatore, che trasferiscono - sugli stessi o su altri supporti - i risultati contenuti in memoria.

Le unità periferiche *non* sono direttamente collegate alla memoria centrale, per ragioni che vedremo in seguito; l'unità di ingresso/uscita (unità I/U) è appunto quella parte del calcolatore che provvede al collegamento tra le unità periferiche e la memoria. Prima di esaminare la costituzione e le funzioni dell'unità di I/U, è quindi necessario esaminare i principali tipi di unità periferiche e i relativi supporti delle informazioni.

IX.7.1 - Apparecchiature periferiche.

Le apparecchiature periferiche più usate sono: la telescrivente, il lettore/perforatore di scheda, il lettore/perforatore di nastro, l'unità a nastri magnetici, la stampatrice.

IX.7.1.1 - Telescrivente.

Una telescrivente fa parte normalmente della *console*[1] di comando del calcolatore, e serve per scambiare brevi messaggi con l'operatore (ad esempio, per richiedere un intervento manuale o segnalare una condizione anomala di funzionamento).

Le telescriventi vengono anche usate come unità di ingresso lontane in sistemi time-sharing, e in questo caso possono essere collocate a distanze di centinaia di chilometri dal calcolatore ed avere un proprio buffer; i dati vengono trasmessi usando linee telefoniche, telegrafiche, o collegamenti radio.

La telescrivente è l'unità assai semplice da usare, ma è molto lenta, specialmente se usata come dispositivo d'ingresso diretto.

IX.7.1.2 - Lettore/perforatore di schede.

Il supporto più comune per l'ingresso dei dati nel calcolatore è la scheda perforata.

La scheda è un rettangolo di cartoncino di cm 19 × 8, divisa in 12 righe e 80 colonne. Su ogni colonna viene perforato un carattere secondo un codice (*Hollerith*) a 12 bit, uno per riga. Una perforazione corrisponde al bit 1, l'assenza della perforazione al bit 0. Le cifre decimali sono rappresentate con una sola perforazione sulle colonne 0 ÷ 9 della scheda; gli altri caratteri (lettere maiuscole e segni speciali) sono rappresentate da perforazioni su 2 o 3 colonne (fig.IX.51).

Le schede vengono perforate con una apposita macchina *perforatrice*, che ha una tastiera simile a quella di una comune macchina da scrivere. Premendo un tasto, si perfora un carattere sulla scheda che,

(1) - La *console* è un sistema di interruttori e indicatori che permette all'operatore di controllare l'andamento di un programma e immettere direttamente informazioni nei registri dell'unità di governo. La console assume aspetti e funzioni molto sofisticate nei grandi calcolatori, ed è sempre provvista di una telescrivente.

contemporaneamente, avanza di una colonna. La traslazione dell'informazione dalla scheda alla memoria centrale viene effettuata dal *lettore di schede*; un'altra unità, il *perforatore*, trasferisce l'informazione dalla memoria alla scheda (se questa viene usata come supporto dei risultati). La velocità del lettore varia da 200 a 1000 schede al minuto; quella del perforatore da 100 a 500.

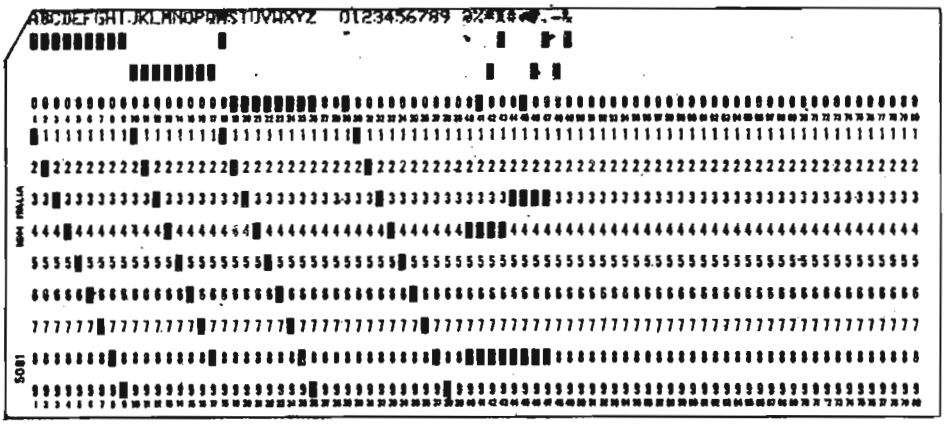


Fig.IX.51 - Scheda perforata.

I vantaggi della scheda perforata sono:

1) equipaggiamenti di lettura sicuri ed economici;
2) facilità di correzioni e aggiunte;
3) inalterabilità nel tempo.

Gli svantaggi sono:

1) lentezza dell'unità di lettura/scrittura;
2) eccessivo volume occupato.

IX.7.1.3 - Lettore/perforatore di nastro.

Il nastro di carta è una striscia di carta avvolta su una bobina, larga 20,5 mm e lunga fino a 150 m. È divisa idealmente in strisce longitudinali, chiamate *piste* o *canali*. Ogni carattere viene rappresentato perforando il nastro trasversalmente in corrispondenza delle piste, secondo un codice dipendente dal numero delle piste stesse (fig.IX.52).

I nastri vengono perforati con un *perforatore*, simile a quello che perfora le schede; un *lettore di banda* trasmette poi i dati alla memoria, a una velocità fino a 1000 caratteri al secondo. Quando il nastro perforato viene usato come supporto dei risultati, il perforatore di banda perfora fino a 300 caratteri/secondo.

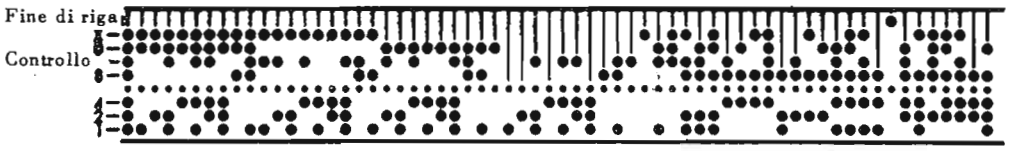


Fig.IX.52 - Nastro perforato a 8 piste. (La pista contenente i fori piccoli serve per il trascinamento del nastro).

I vantaggi del nastro perforato sono:

1) semplicità ed economicità del supporto e delle unità periferiche;
2) resistenza e inalterabilità nel tempo;
3) risparmio di spazio nei confronti delle schede.

Svantaggi:

1) bassa velocità di trasferimento;
2) difficoltà di correzione.

IX.7.1.4 - Unità a nastro magnetico, dischi, tamburi.

Il nastro magnetico e la relativa unità di lettura/scrittura sono stati già descritti come memorie ausiliarie. Ovviamente, l'unità a nastri è anche un'apparecchiatura di ingresso/uscita.

I nastri sono generalmente preparati da altri supporti di informazione. Ad esempio, un certo numero di programmi e/o di dati, perforati su schede, vengono letti e trasferiti su un nastro magnetico e poi inviati al calcolatore, ad alta velocità, per essere elaborati. Questo procedimento, tra l'altro, elimina i tempi d'atteso per la lentezza del lettore di schede e/o per interventi manuali, in quanto tutti i programmi entrano sequenzialmente dal nastro. Lo stesso avviene in uscita, dove un nastro riceve i risultati del programma sequenzialmente e ad alta velocità, per trasmetterli poi alle unità periferiche di uscita.

Anche le altre unità descritte come memorie ausiliarie (*dischi* e *tamburi*) vanno considerate tra le unità di I/U.

IX.7.1.5 - Stampatrice.

L'uscita su carta è di uso comune per visualizzare i risultati di un programma.

Tramite un'unità di stampa, o *stampatrice*, le informazioni in uscita vengono stampate su un foglio di dimensioni e caratteristiche variabili. La stampa viene effettuata riga per riga ad una velocità variabile tra 400 e 1400 righe/minuto (una riga comprende da 120 a 132 caratteri). L'avanzamento del foglio è automatico; la spaziatura tra le righe può essere comandata a programma.

IX.7.1.6 - Altre unità periferiche.

Per scopi particolari sono state introdotte - e vanno largamente diffondendosi - unità di ingresso/uscita particolari. Tra queste ricorderemo: l'*unità-video* che è sostanzialmente una telescrivente che ha - invece della carta - uno schermo su cui possono comparire, in uscita, parecchie centinaia di caratteri. In ingresso, l'intero messaggio da inviare al calcolatore viene memorizzato e visualizzato sullo schermo; può essere corretto, e viene trasferito, con apposito comando, ad alta velocità. Unità video più sofisticate permettono di mostrare, ed accettano in ingresso, figure, segni particolari e grafici.

Sullo schermo - in questo caso sotto controllo di un apposito programma del calcolatore - si può anche disegnare con una speciale *penna luminosa*, interagendo a vari livelli col calcolatore.

Un'altra apparecchiatura di ingresso è il *lettore di caratteri*, che è in grado di riconoscere particolari configurazioni, tracciate secondo regole stabilite e/o con speciali inchiostri magnetici. Questi dispositivi vengono usati soprattutto nelle applicazioni commerciali e bancarie. Non è stato tuttavia ancora risolto il problema di riconoscere caratteri alfanumerici tracciati senza precise regole nei riguardi della posizione e della forma.

Tra le apparecchiature d'uscita, molto utile, soprattutto in campo scientifico, si è dimostrato il tracciatore di curve (*plotter*), capace di produrre su un nastro di carta dei grafici delle funzioni di 1 variabile e - mediante famiglie di curve - di 2 variabili. In via sperimentale, sono state anche usate delle unità d'uscita che forniscono risposte acustiche pre-registrate.

IX.7.2 - Organizzazione dell'unità di ingresso/uscita.

Le funzioni essenziali dell'unità di I/U sono:

1) realizzare le operazioni di trasferimento dei dati tra il calcolatore e le unità periferiche lente, senza rallentare le operazioni del calcolatore stesso;

2) convertire delle informazioni dal codice adottato per rappresentare le informazioni sul supporto dell'unità periferica a quello usato in memoria centrale, con la generazione e/o il controllo dei bit di parità, e le eventuali trasformazioni serie-parallelo lungo le vie di trasmissione.

Tutte le operazioni indicate al punto 2) vengono compiute con i circuiti descritti più volte nei paragrafi precedenti, e non presentano pertanto caratteristiche particolari.

Il problema di scambiare informazioni con le unità periferiche senza rallentare le operazioni del calcolatore è stato risolto in modi assai diversi. Fino al 1964 era comune l'uso di un piccolo calcolatore che leggeva i dati da scheda e/o da nastro perforato e li registrava su nastro magnetico; l'unità a nastro era poi usata come unità di ingresso al calcolatore. Anche le velocità delle unità a nastri erano però basse rispetto a quelle dell'unità aritmetica. Si sono allora sviluppate tecniche per sovrapporre le operazioni di I/U con quelle relative all'elaborazione dei dati.

Tali tecniche prevedono l'uso di una memoria di *buffer* in cui si trasferiscono gruppi di dati dai nastri; dal buffer i dati vengono trasferiti in memoria ed elaborati; quando il buffer è vuoto, altri dati entrano dai nastri. Tutti i trasferimenti di caratteri vengono comandati dall'unità di governo.

Un ulteriore passo avanti fu quello di limitare ancora di più lo intervento dell'unità di governo, fornendo alla sezione di I/U l'indirizzo di memoria in cui trasferire i dati, la lunghezza del buffer, l'operazione da compiere (I o U), il dispositivo periferico da usare. Dopo queste informazioni, la sezione di I/U controlla da sola il trasferimento dei dati, mentre il calcolatore continua l'elaborazione. Un'unità di I/U che funziona così si chiama in genere *canale*. Il canale è un'apparecchiatura complessa, che possiede molte caratteristiche di un vero e proprio calcolatore (registri, contatori, ecc.). In genere, esistono più canali per ogni calcolatore; ogni canale può controllare più dispositivi di I/U (vedi figura IX.53).

Anche i canali possono avere realizzazioni diverse. Alcuni tipi ricevono i comandi direttamente dall'unità di governo (es.: «Leggi 15 pa-

role dall'unità periferica 4, e mettile nel campo di memoria di indirizzo 8320»), che subito dopo si svincola. L'unità di governo continua poi le sue operazioni sulle istruzioni del programma, finchè non gli arriva una *interruzione* di «operazione di lettura eseguita», mentre il canale riceve i dati dall'unità periferica e li manda in memoria, aggiornandone continuamente l'indirizzo.

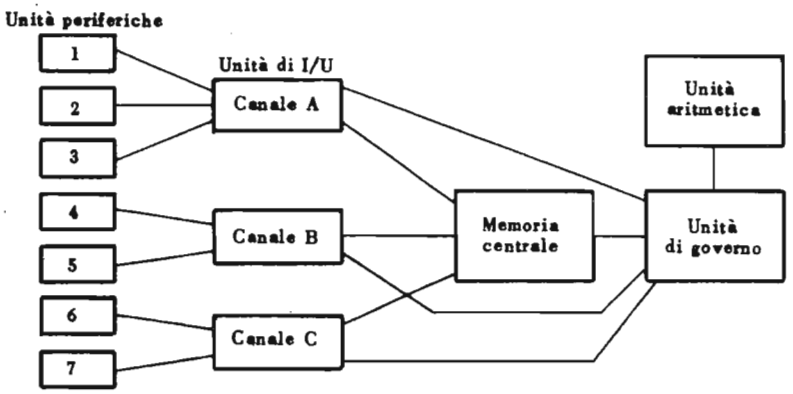


Fig.IX.53 - Sezione di ingresso/uscita realizzata con canali.

Altri tipi, più complessi, di canali eseguono le proprie istruzioni prelevandole direttamente dalla memoria, cioè senza intervento della unità di governo, che si limita ad attivare il canale (senza mandargli *anche* l'istruzione decodificata). In questa organizzazione, il canale ha un proprio RII e un proprio circuito di decodifica. L'attivazione del canale avviene da parte dell'unità di governo posizionando il RII del canale, cioè fornendogli l'indirizzo del *programma di canale* da eseguire. L'unità di governo coordina, inoltre, l'azione dei vari canali.

IX.8 - Cenni sui criteri di progetto di un calcolatore digitale.

La realizzazione di un calcolatore non comporta soltanto la soluzione di problemi tecnici, ma nasce ed è determinata principalmente da considerazioni economiche. Ogni calcolatore di cui si inizia il progetto deve cioè presentare qualche caratteristica nuova, e/o deve essere più

semplice da usare e costare meno rispetto a tutti gli altri già in commercio (non si considera, evidentemente, il caso di un'industria che costruisca calcolatori in regime di monopolio).

Il primo problema da affrontare riguarda il tipo di calcolatore e il repertorio delle istruzioni. Una prima lista di istruzioni va compilata analizzando il tipo di lavori che il calcolatore sarà chiamato a risolvere più frequentemente, e il metodo migliore per risolvere i problemi stessi. Dalla lista di istruzioni si può effettuare una prima grossolana valutazione delle dimensioni dei programmi in memoria e del relativo tempo di esecuzione.

Si passerà poi alla determinazione della *precisione* dei dati, sempre in base alle caratteristiche dei lavori che il calcolatore dovrà eseguire. Da questo studio si avranno delle indicazioni circa la lunghezza della parola di memoria. Una valutazione del volume di dati da trattare fisserà poi un primo dimensionamento della memoria stessa. Si tornerà quindi alla determinazione del repertorio delle istruzioni, del loro tipo e del formato di ogni tipo. Da queste considerazioni, e da quelle relative alla precisione da adottare, si arriva quindi a stabilire definitivamente le dimensioni della parola, prevedendo generalmente la possibilità di immagazzinare ed elaborare dati con una precisione doppia, trattando campi di 2 parole affiancate.

Lo studio preliminare verrà quindi portato a termine decidendo:

- il numero e il tipo di unità periferiche da impiegare;
- le dimensioni e il tipo di eventuali buffer;
- le dimensioni della memoria;
- i requisiti della memoria di massa;
- le velocità di calcolo e trasferimento dati;
- il sistema di protezione e localizzazione dei guasti.

IX.8.1 - Progetto di massima.

Stabiliti i requisiti generali del sistema, i tecnici cui è affidata la realizzazione del calcolatore dovrebbero, ciascuno per la propria sfera di competenza:

1) preparare un diagramma a blocchi dettagliato con l'organizzazione dei vari componenti il calcolatore;

2) stabilire se il trasferimento dei dati tra le varie unità sarà in serie, in parallelo o misto;

3) determinare i metodi per modificare le istruzioni;

4) fissare il numero e il tipo dei registri e dei contatori necessari;

5) scegliere i sistemi di controllo e clock;

6) stabilire le mobilità per il trasferimento delle informazioni;

7) definire l'organizzazione della memoria.

L'intero gruppo, quindi, sceglierà i circuiti logici da usare, determinerà la configurazione meccanica delle unità, farà previsioni sul costo, ricercherà eventuali compatibilità con altri calcolatori esistenti.

IX.8.2 - Progetto dettagliato.

Comincia a questo punto il progetto definitivo e dettagliato, che include un lavoro di Hardware e uno di Software.

La parte Hardware comprende, nell'ordine, le seguenti fasi:

1) elenco dettagliato delle istruzioni; decisione sul tipo di governo da adottare; descrizione del modo e dei tempi di esecuzione di ogni istruzione;

2) progetto logico dei circuiti combinatori e sequenziali delle varie unità: equazioni logiche e schemi AND-OR-NOT, NAND, NOR, ..., di tutti i circuiti;

3) progetto elettronico: scelta dei componenti, dei livelli operativi, del grado di affidabilità, dei livelli di alimentazione, della disposizione spaziale dei componenti, delle alimentazioni fornite dal clock.

Quando suesposto è ovviamente solo una traccia riportata a titolo d'esempio; solo una lunga esperienza di lavoro può in realtà suggerire gli effettivi metodi da usarsi in una realizzazione pratica tanto complessa.

*

BIBLIOGRAFIA

1. BARON R.C. - PICCIRILLI A.T.: *Digital Logic and Computer Operations* - Mc Graw Hill, 1967.
2. EADIE D.: *Introduction to the basic Computer* - Prentice-Hall, 1968.
3. GEAR W.C.: *Computer Organization and Programming* - Mc Graw Hill, 1969.
4. FALZONE V. - COSTANTINI E.: *Elaboratori Elettronici e tecnica della programmazione* - Hoepli, 1971.
5. FOSTER J.A.: *Computer Architecture* - Van Nostrand, 1971.
6. RALSTON A.: *Introduction to Programming and Computer Science* - Mc Graw Hill, 1971.

*

FINITO DI STAMPARE
GIUGNO 1972

PREZZO L. 8.000