CAPITOLO I

LE BASI DELL'ELETTRONICA DIGITALE

1.1 - INTRODUZIONE

L'informazione su un fenomeno fisico è in generale trasportata da "segnali" di varia natura (elettrica, ottica,...) generati da opportuni trasduttori. Si parla di segnale analogico se qualunque valore assunto da detto segnale è per noi significativo. La rappresentazione analitica di tale segnale è una funzione continua del tempo, x(t).

Un esempio è la tensione v(t) di uscita da un amplificatore audio, applicata ad un altoparlante: essa rappresenta istante per istante l'intensità del suono da riprodurre. Il numero dei valori significativi di v(t) è chiaramente infinito.

In molti casi, si ha a che fare con una informazione che ha oggettivamente un numero discreto, e finito, di valori significativi. Ci limiteremo a considerare il caso di due soli valori (informazione binaria), perché la tecnologia digitale è sviluppata essenzialmente per questo caso.

Se per esempio si vuole mantenere costante la temperatura di un ambiente, non ha importanza il valore attuale della temperatura (fornita per esempio sotto forma di segnale elettrico da una termocoppia): ciò che importa è se tale valore è maggiore o minore di un valore di riferimento, in modo da attivare opportunamente il sistema di condizionamento.

In generale, anche l'informazione contenuta in un segnale analogico può essere rappresentata in forma binaria. Infatti, il teorema di Shannon assicura che un segnale analogico, avente uno spettro di frequenza limitato a f_{max}, è completamente conosciuto (in ogni istante) qualora se ne conoscano i *campioni* (cioè i valori istantanei) presi con periodo $T_s <= 1/(2f_{max})$: con ciò si intende dire che dalla sequenza discreta x[n] dei campioni è possibile ricostruire il segnale continuo x(t). Ora, il valore di ciascun campione è un numero, e se tali numeri vengono espressi nel sistema di numerazione "binario", illustrato in un successivo paragrafo, ciascun campione (e quindi il segnale) sarà rappresentato da una stringa numerica in cui si fa uso di due soli simboli, quindi in forma binaria.

L'elaborazione dell'informazione rappresentata in forma numerica binaria è più affidabile di quella rappresentata in forma analogica. Si pensi, per esempio, alla trasmissione a distanza di un segnale elettrico analogico: una tensione di rumore che si sommi algebricamente alla tensione del segnale lungo il canale di trasmissione altererà in modo irrecuperabile il segnale vero. Se invece il segnale è rappresentato in codice binario, lungo il canale di trasmissione viaggeranno solo due diversi livelli di tensione, scelti a rappresentare i due simboli del codice binario. Queste due tensioni possono essere scelte in modo da non essere confondibili, una volta conosciuta la massima tensione di rumore che ci si aspetta.

La tecnologia elettronica permette di costruire macchine capaci di elaborare informazioni sempre più complesse in tempi sempre più brevi, se tali informazioni sono espresse in forma binaria. I calcolatori elettronici digitali sono l'aspetto più appariscente di questa innovazione, ma la tecnologia digitale è entrata in tutti i campi ove ci si serve di strumentazione per misure e controllo, con vantaggi inimmaginabili qualche decennio fa. Si pensi, per esempio, alla misura di una tensione elettrica: un voltmetro tradizionale, in cui la deviazione dell'ago di un galvanometro su una scala è proporzionale al valore della tensione di entrata, ben difficilmente permette una precisione di lettura migliore di qualche percento, date le ridotte dimensioni della scala, ed in generale tale precisione è inferiore a quella intrinseca del trasduttore (il galvanometro). Con strumenti di tipo digitale, la precisione della lettura è indipendente dal sistema di visualizzazione (un display alfanumerico) e coincide con quella del trasduttore (che in questo caso è un convertitore analogico-digitale, ADC), che può essere molto elevata.

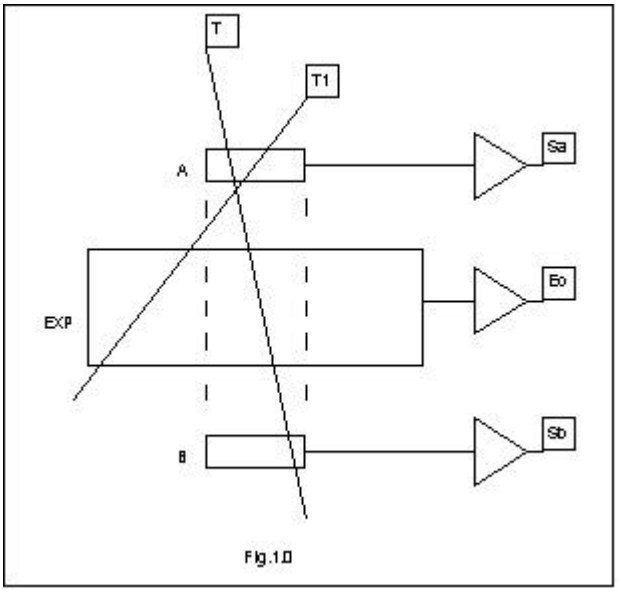


Fig.1.0

L'elaborazione digitale dell'informazione ha i suoi fondamenti teorici nell'algebra booleana, che verrà introdotta in un successivo paragrafo. Per ora, soffermiamoci su un semplice esempio il quale, seppure banale, permette di introdurre intuitivamente la problematica che l'algebra booleana affronta in modo formale. Consideriamo un semplice sistema di controllo di un esperimento di fisica delle particelle elementari, **Fig. 1.0**. Sia T una sorgente di particelle ionizzanti; EXP un apparato (per es. una camera proporzionale a fili) che

produce un segnale (elettrico) E_o legato ad una proprietà della particella ionizzante (per es. l'energia persa dalla particella nell'attraversare l'apparato). Supponiamo di essere interessati ai segnali E_o generati dalle particelle, provenienti da T, che attraversano la regione del rivelatore delimitata dal tratteggio. Per selezionare queste particelle, si possono usare due sensori (che sono in realtà anch'essi rivelatori di particelle) A e B posti lungo la traiettoria di interesse e utilizzare i segnali da essi prodotti, S_A e S_B, per selezionare le particelle di nostro interesse secondo una *logica* di funzionamento che possiamo così formulare:

"accettare l'evento E_o se è presente il segnale S_A e se è presente il segnale S_B".

Quello descritto è un semplice sistema di trigger a coincidenza.

Nella formulazione precedente possiamo distinguere la *funzione logica* che la macchina deve implementare:

"accettazione dell'evento E_o"

e le *variabili logiche* da cui l'implementazione dipende:

"presenza del segnale S_A"

"presenza del segnale S_B"

Dovrebbe apparire chiaro che siamo di fronte ad una funzione e a delle variabili che possono assumere solo due stati, che per comodità chiameremo stato *vero* e stato *falso*:

- l'evento E_o può essere accettato o rigettato;
- gli eventi S_A e S_B possono essere presenti o assenti.

Il problema è ora come costruire un apparato che *meccanizzi* la logica di controllo sopra descritta, cioè una rete che riceva in entrata i segnali S_A, S_B, ed E_o, e risponda con un segnale Y che ci dica se accettare o meno E_o. Per un approccio razionale al problema, occorre innanzi tutto costruire il modello matematico del problema stesso, cioè la relazione funzionale fra variabili logiche di entrata e funzione logica di uscita. Questo porta a definire le operazioni che si devono effettuare sulle variabili per ottenere la funzione desiderata. La funzione logica verrà indicata generalmente con la lettera Y e le variabili logiche con le lettere A, B, C,...; pertanto, una relazione funzionale sarà scritta genericamente come segue:

(1.1) $Y = f(A,B,C,...)$

È anche necessario fissare dei simboli per distinguere i due valori che Y, A, B,… possono assumere. A tale scopo, conveniamo di indicare con Y, A, B, … e con $\overline{Y}$, $\overline{A}$, $\overline{B}$ i due stati *vero* e *falso*, rispettivamente. È comunque arbitrario associare per es. il simbolo A al valore vero e $\overline{A}$ al valore falso, o viceversa.

OR(+)

A	B	Y
F	F	F
F	V	V
V	F	V
V	V	V

Fig.1.1

Per esplicitare la generica funzione Y = f(·), occorre definire delle *operazioni* sulle variabili, che esprimano come, nella formulazione di un problema, le variabili sono legate fra loro per determinare lo stato della funzione Y. Riferiamoci per semplicità ad una funzione di due variabili; la generalizzazione apparirà ovvia.

Poiché la funzione e le variabili logiche possono assumere due soli stati, la relazione funzionale più semplice è intuitivamente una delle due seguenti:

- *una funzione logica Y = f(A,B) ha valore vero se almeno una (cioè una o più) delle variabili ha valore vero.*

OR(-)

A	B	Y
F	F	F
F	V	F
V	F	F
V	V	V

Fig.1.2

Si dice che la relazione funzionale è realizzata dall'operatore **OR**, e l'espressione esplicita di Y può essere scritta

Y = A OR B

L'azione dell'operatore OR è descritta dalla *tavola della verità*, **Fig. 1.1**.

È anche evidente che l'operatore OR può essere introdotto con riferimento alla "falsità" della funzione e delle variabili, cioè si potrebbe anche dire:

una funzione Y = f(A,B) è falsa se almeno una delle variabili è falsa.

AND(-)

A	B	Y
F	F	F
F	V	V
V	F	V
V	V	V

Fig.1.3

AND(+)

A	B	Y
F	F	F
F	V	F
V	F	F
V	V	V

Fig.1.4

Nel caso di due variabili, la tavola della verità è quella di **Fig. 1.2**.

Appare quindi evidente che si può ragionare secondo due logiche, "duali" una dell'altra; esse vengono distinte chiamando per convenzione la prima *logica positiva* e la seconda *logica negativa*.

- *Una funzione logica Y = f(A,B) è vera (falsa) se*

entrambe le variabili sono vere (false).

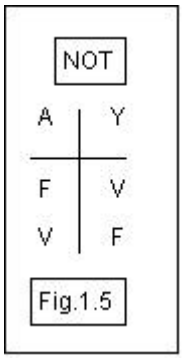
Si dice che la relazione funzionale è realizzata dall'operatore **AND**, e l'espressione esplicita di Y è

Y = A AND B

Le tavole della verità sono in **Fig. 1.3**, per la logica negativa, e in **Fig. 1.4** per la logica positiva.

In un problema concreto, una variabile è il segnale fornito da un generatore, e questo segnale ad un certo istante è in uno stato definito (o vero, o falso). Può essere necessario disporre contemporaneamente di entrambi gli stati logici della variabile: pertanto, occorre introdurre un altro operatore che fornisca lo stato logico complementare, che viene più comunemente chiamato stato negato. Questo operatore si chiama **NOT**. La tavola della verità di questo operatore è in **Fig. 1.5**, ed è unica per entrambe le logiche.

Una generica funzione logica avrà quindi una espressione esplicita in cui compaiono tre soli operatori matematici.



La scrittura della espressione algebrica della funzione risulta più snella se i tre operatori matematici vengono rappresentati con simboli algebrici. Tali simboli sono i seguenti:

OR = "+" (simbolo della somma aritmetica)

AND = "" (simb. del prodotto aritm., generalmente sottinteso)

NOT = "–" (sovrasegnatura)

Questi simboli hanno suggerito l'ovvia nomenclatura di "somma logica", "prodotto logico", "complementazione", rispettivamente, per le tre operazioni. È opportuno sottolineare che questi simboli vanno qui intesi in senso *logico*, non *aritmetico*.

Vediamo ora come il modello matematico può essere tradotto in una rete. Per questo, ad ogni operatore occorrerà associare un "elemento di circuito" che meccanizza la tavola della verità dell'operatore stesso. Questi elementi di circuito, che vengono comunemente chiamati *porte logiche*, possono essere costruiti con tecnologie elettroniche oppure elettromeccaniche, oppure ancora ottiche, ecc. Noi ci interesseremo del primo caso, poiché le tecnologie elettroniche sono a tutt'oggi quelle massimamente sviluppate, e le porte saranno reti elettriche, oggi disponibili sotto forma di circuito integrato. In questo caso, i due stati logici sono due diversi livelli di tensione,

variabili a seconda della tecnologia costruttiva usata, per cui è usuale parlare genericamente di *livello alto*, H, e di *livello basso*, L, anziché di vero e falso, rispettivamente. In una tecnologia di uso molto comune, nota come TTL, i valori nominali delle due tensioni sono L = 0V, H = 5V.

Dall'esame delle tavole della verità di **Fig. 1.1-1.4** appare che

un AND per logica positiva è anche un OR per logica negativa,

e che

un OR per logica positiva è anche un AND per logica negativa.

Ne segue che gli elementi di circuito sufficienti sono tre. Si farà vedere più avanti che non tutti sono necessari.

AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Fig.1.6a

OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Fig.1.6b

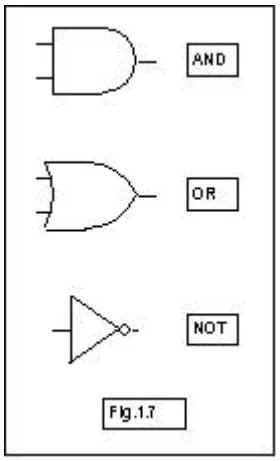
NOT

A	Y
0	1
1	0

Fig.1.6c

Nella formulazione di un problema logico è comodo potersi svincolare dal tipo di logica (positiva o negativa), e ciò viene realizzato se si conviene che

- *in logica positiva lo stato "vero" (o il livello H) di una variabile è rappresentato col simbolo 1, e lo stato "falso" (o il livello L) col simbolo 0.*



- *Viceversa in logica negativa.*

Le nuove tavole sono in **Fig. 1.6.** Quindi, è possibile scrivere la tavola della verità dell'operatore senza dover specificare la logica. Solo quando si passerà alla costruzione della rete occorrerà specificare *quale significato fisico (cioè quale tensione) associare a 1 e a 0*, per poter scegliere gli elementi di circuito adatti. Così, se nel nostro progetto il simbolo 1 è associato a un livello alto, sceglieremo, per realizzare il progetto, porte per logica positiva; e viceversa.

La realizzazione di una rete è di norma preceduta dal disegno dello schema della rete stessa. Per far questo, occorre associare un simbolo circuitale a ciascun operatore. I simboli a tutt'oggi più diffusi sono indicati in **Fig. 1.7**. Nello schema, i simboli sono connessi con linee che rappresentano il percorso dei segnali.

1.2 - SISTEMI DI NUMERAZIONE

Poiché le variabili logiche assumono due soli valori, si è introdotto un sistema di numerazione, chiamato binario, nel quale ogni simbolo (comunemente chiamato "bit") può assumere uno dei due valori numerici 0 oppure 1, a differenza del sistema di numerazione decimale in cui ogni simbolo può assumere uno dei dieci valori 0,1,...9.

Il significato aritmetico attribuito ad una stringa di simboli decimali è espresso dalla ben nota relazione illustrata nell'esempio che segue (l'indice è la "base di numerazione")

$$925{,}73_{(10)} = 9\cdot10^{2} + 2\cdot10^{1} + 5\cdot10^{0} + 7\cdot10^{-1} + 3\cdot10^{-2}$$

Questa relazione mette in evidenza come il sistema di numerazione decimale sia di tipo posizionale, cioè il *peso* attribuito ad un simbolo dipende dalla sua posizione nella stringa: tale peso è una potenza della base di numerazione, essendo l'esponente eguale, in valore assoluto, al numero d'ordine del posto occupato dal simbolo, contando a partire dalla virgola. Il sistema di numerazione romano è un esempio di codice numerico non posizionale.

L'esempio precedente si presta ad una immediata generalizzazione, considerando una stringa numerica in un sistema di numerazione, posizionale, in base $b \neq 10$. Se

$$N_{(b)} = a_n a_{n-1} \ldots a_1 a_0 , a_{-1} a_{-2} \ldots a_{-k}$$

è la stringa, il suo significato aritmetico è

$$N_{(b)} = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b^1 + a_0 b^0 + a_{-1} b^{-1} + \ldots + a_{-k} b^{-k}$$

essendo: b la base di numerazione e a_i un simbolo del sistema di numerazione, che è un intero compreso fra 0 e b-1. La potenza b^j è il peso attribuito al simbolo a_j.

Così, nel sistema di numerazione binario è b=2 ed ogni simbolo nella stringa può assumere i valori 0 e 1.

Se le operazioni a II membro (somma, prodotto, …) vengono eseguite secondo le regole dell'aritmetica decimale, si ottiene l'equivalente decimale del numero in base b.

Ci sono altri due sistemi di numerazione che conviene considerare, data l'importanza che hanno nei sistemi di calcolo: il sistema di numerazione in base 8 (octal), che fa uso degli 8 valori numerici

0,1,...7, ed il sistema di numerazione in base 16 (esadecimale) che fa uso dei 16 valori numerici 0,1,..9,10,..15 i quali, per ovvi motivi, sono scritti più comodamente 0,1,..9,A,B...F.

Esempi:

$$11010_{(2)} = 1\cdot2^4+1\cdot2^3+0\cdot2^2+1\cdot2^1+0\cdot2^0 = 26_{(10)}$$

$$371_{(8)} = 3\cdot8^2+7\cdot8^1+1\cdot8^0 = 249_{(10)}$$

$$3F5_{(16)} = 3\cdot16^2+15\cdot16^1+5\cdot16^0 = 1013_{(10)}$$

Per esprimere un numero $N_{(10)}$ in un sistema in base $b\neq10$, conviene considerare separatamente la parte intera e quella decimale di N. La parte intera viene divisa per b, annotando quoziente e resto; il quoziente viene ancora diviso per b annotando il nuovo quoziente ed il nuovo resto; e così via, fino ad ottenere quoziente 0. La stringa dei resti è il risultato cercato.

Nell'esempio che segue, si trasforma in binario il numero decimale 13:

quoz.	0	1	3	6	13
resti	1	1	0	1	

Quindi, il decimale 13 si scrive in binario 1101. Si noti che il bit più significativo (*Most-Significant-Bit*, MSB) è quello più a sinistra (viene moltiplicato per la massima potenza di 2), mentre quello meno significativo (*Least-Significant-Bit*, LSB) è quello più a destra.

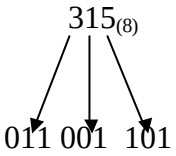
La conversione della parte decimale nella nuova base di numerazione b avviene moltiplicandola per b, annotando separatamente la parte intera e la parte frazionaria del risultato; la parte frazionaria viene ancora moltiplicata per b, annotando la nuova parte intera e la nuova parte frazionaria; e così via. Il processo si arresta spontaneamente se la parte frazionaria diventa 0; altrimenti, il processo viene arrestato quando si è ottenuto il numero desiderato di cifre. È chiaro che in questo secondo caso la conversione è approssimata.

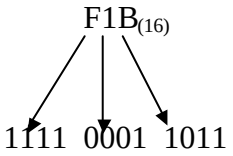
Nell'esempio che segue, si converte in octal il numero decimale 0.624:

0.624	0.992	0.936	0.448	0.904	....
	4	7	7	3	....

Quindi, $0.624_{(10)} = 0.4773..._{(8)}$.

Per convertire un numero da una base b≠10 ad una base b'≠10, conviene generalmente fare una conversione intermedia in decimale. Tuttavia, il codice octal e quello esadecimale permettono una conversione immediata in codice binario, e viceversa. Infatti, si può facilmente verificare che per tradurre un numero octal (esadecimale) in binario basta tradurre singolarmente ciascuna cifra in un numero binario a 3 bit (4 bit), conservando l'eventuale virgola; ciò è mostrato negli esempi che seguono:





Viceversa, per passare dal binario all'octal (esadecimale), basta dividere la stringa binaria, partendo dalla virgola, in gruppi di tre (quattro) bit e tradurre ciascun gruppo in octal (esadecimale). Esempio:

```
                1 1 0 1 1, 0 0 1 0 1
                ___| |_____| |_____| |____
octal           3     3   ,   1     2
              ___||_______||_________||___
esadec.         1     B   ,   2       8
```

Quanto ora detto fa comprendere che il sistema octal e quello esadecimale sono utili per rappresentare in maniera più compatta un numero binario.

1.3 - ALGEBRA BOOLEANA

L'algebra booleana permette di esprimere in forma algebrica le relazioni funzionali fra le variabili di entrata e l'uscita di una rete logica.

Per introdurla in modo generale, consideriamo un insieme di elementi $I=(i_1,i_2,i_3,...)$ e indichiamo con A,B,C,... delle variabili definite su I; associamo ad I una struttura algebrica definendo le due operazioni di somma (+) e prodotto ($\cdot$), per le quali si postulano le seguenti proprietà:

(1.2a) $A+B = B+A$ proprietà commutativa

(1.2b) $A \cdot B = B \cdot A$ " "

(1.3a) $A \cdot (B+C) = A \cdot B + A \cdot C$ " distributiva

(1.3b) $A+(B \cdot C) = (A+B) \cdot (A+C)$ " "

(1.4a) $A+0 = A$ esistenza dell'elemento neutro della +

(1.4b) $A \cdot 1 = A$ " " " del .

(1.5a) $A+\overline{A} = 1$ esistenza dell'elemento opposto

(1.5b) $A \cdot \overline{A} = 0$ " "

Si è assunto implicitamente l'uso delle parentesi e la proprietà riflessiva, simmetrica e transitiva dell'uguaglianza. Si conviene inoltre che l'operazione di prodotto ha precedenza su quella di somma e che si iniziano le operazioni dal livello di parentesi più interno.

Una funzione booleana è una espressione che combina mediante le operazioni di somma e prodotto un numero finito di variabili, ciascuna delle quali può assumere come valore uno degli elementi dell'insieme I.

Un teorema dell'algebra booleana è una asserzione che può essere dimostrata con l'uso degli assiomi.

Si noti che gli assiomi sono presentati a coppie, in modo tale che in ciascuna coppia un assioma può essere derivato dall'altro scambiando l'operazione di somma con quella di prodotto. Vale in generale un "principio di dualità": da un teorema già dimostrato si può ottenere un nuovo teorema scambiando + con $\bullet$ e ogni variabile col suo complemento. In effetti le variabili qui restano inalterate poiché in

un teorema le variabili hanno significato generico (tra l'altro, possono rappresentare anche intere espressioni). I teoremi principali dell'algebra booleana sono riportati in un successivo paragrafo.

Il principio di dualità può essere applicato anche ad una funzione booleana, ottenendo una nuova espressione (detta duale) della *stessa* funzione. Va sottolineato che in una funzione le variabili hanno significato *specifico* (rappresentano cioè segnali fisici), quindi sia la funzione che le variabili vanno esplicitamente sostituite col proprio complemento.

L'algebra booleana ha trovato applicazione per lo studio formale di varie scienze, per esempio la logica proposizionale, che studia le regole del ragionamento sulla base della verità o falsità di asserzioni non equivoche. In questo caso, l'insieme I contiene due soli elementi, "vero" e "falso". In questo ambito si può far rientrare lo studio dei circuiti a commutazione, nei quali cioè gli stati di entrata e di uscita possono assumere due soli valori, rappresentati simbolicamente con 1 e 0.

1.4 - I TEOREMI DELL'ALGEBRA BOOLEANA

Si elencano alcuni teoremi dell'algebra booleana che si avrà occasione di richiamare in seguito. I teoremi sono dimostrabili sulla base degli assiomi, o di un teorema già dimostrato, o per dualità. Vengono trascritti accoppiati per dualità. Il simbolo del prodotto è omesso là dove non necessario.

(1.6) $A+A=A$ $AA=A$

(1.7) $A+1=1$ $A\cdot 0=0$

(1.8) $A+B+C=A+(B+C)$ $ABC=A(BC)$ (propr. associativa)

(1.9) $A(A+B)=A$ $A+AB=A$

(1.10) $A(\overline{A}+B)=AB$ $A+\overline{A}B=A+B$

(1.11) $\overline{A}(A+B)=\overline{A}B$ $\overline{A}+AB=\overline{A}+B$

(1.12) $(A+B)(A+\overline{B})=A$ $AB+A\overline{B}=A$

(1.13) $(A+B)(\overline{A}+C)=\overline{A}B+AC$

(1.14) $\overline{\overline{A}}=A$

(1.15) $\overline{AB}=\overline{A}+\overline{B}$ $\overline{A+B}=\overline{A}\,\overline{B}$ (teoremi di De Morgan)

Si fa ora qualche esempio di dimostrazione.

Dimostriamo che $A+A=A$.

Per la proprietà simmetrica dell'uguaglianza, basta dimostrare che $A=A+A$. Allora:

$$A = A+0 = A+A\overline{A} = (A+A)(A+\overline{A}) = (A+A)1 = A+A$$

(1.4a) (1.5b) (1.3b) (1.5a) (1.4b)

Sotto ogni segno di eguaglianza è indicato l'assioma che lo giustifica.

Dimostriamo che $\overline{A+B}=\overline{A}\,\overline{B}$.

Basta far vedere che $A+B$ e $\overline{A}\,\overline{B}$ sono uno il complemento dell'altro, cioè (assioma 1.5)

$(A+B)\,\overline{A}\,\overline{B}=0$

$(A+B)+\overline{A}\,\overline{B}=1$

Infatti

$(A+B)\cdot\overline{A}\,\overline{B}=A\,\overline{A}\,\overline{B}+B\overline{A}\,\overline{B}=0+0=0$

(1.3a) (1.5b) (1.4a)

$(A+B)+\overline{A}\,\overline{B} = (A+B+\overline{A})(A+B+\overline{B}) = (1+B)(1+A) = 1\cdot 1 = 1$

(1.3b) (1.5a) (1.12) (1.4b)

I teoremi di De Morgan, che sono particolarmente importanti, possono essere estesi al caso di n variabili:

(1.16) $\overline{A_1 + A_2 + A_3 + \ldots + A_n} = \overline{A}_1 \overline{A}_2 \ldots \overline{A}_n$

(1.17) $\overline{A_1 A_2 A_3 \ldots\ldots A_n} = \overline{A}_1 + \overline{A}_2 + \ldots + \overline{A}_n$

La dimostrazione avviene col metodo dell'induzione finita: cioè, il teorema (già dimostrato per il caso di due variabili) viene assunto vero per il caso di k (>2) variabili e si dimostra quindi che esso è anche vero per k+1 variabili; data l'arbitrarietà di k, segue che il teorema è vero per ogni n. Allora:

$\overline{A_1 + A_2} = \overline{A}_1 \overline{A}_2$ già dimostrato

$\overline{A_1 + A_2 + \ldots A_k} = \overline{A}_1 \overline{A}_2 \ldots \overline{A}_k$ è supposto vero

Si ha:

$\overline{A_1 + A_2 + \ldots + A_{k+1}} = \overline{(A_1 + A_2 + \ldots + A_k) + A_{k+1}}$ (per la 1.8)

$= (\overline{A_1 + A_2 + \ldots A_k})\,\overline{A}_{k+1}$ (per la 1.15)

$= (\overline{A}_1 \overline{A}_2 \ldots \overline{A}_k)\,\overline{A}_{k+1}$ assunto

$= \overline{A}_1 \overline{A}_2 \ldots \overline{A}_{k+1}$ (per la 1.8)

1.5 - FORMA CANONICA DI UNA FUNZIONE BOOLEANA

Un prodotto di variabili o loro complementi si chiama "*termine prodotto*". Esempio

$$A B \overline{C} D$$

Analogamente, si definisce un "*termine somma*":

$$A + B + \overline{C}$$

Poiché le uniche operazioni logiche su variabili, disponibili sia in forma vera che negata, sono somma e prodotto, una funzione booleana può sempre essere scritta sotto forma di "somma di termini prodotto" (S.d.P.), oppure, dualmente, sotto forma di "prodotto di termini somma" (P.d.S.). Per esempio, la funzione

$$Y = A(B+C)+D$$

con la proprietà' distributiva (1.3a) diventa:

$$Y = AB + AC + D$$

che è in forma S.d.P.

Con la proprietà' distributiva (1.3b) la stessa Y diventa

$$Y = (A+D)(B+C+D)$$

che è in forma P.d.S.

Se si applica successivamente la (1.3b) alla espressione S.d.P. si giunge a

$$Y = (A+D)(A+D+C)(A+B+D)(B+C+D)$$

che è ancora in forma P.d.S.

L'esempio qui riportato fa vedere come la stessa funzione Y può assumere molte espressioni diverse, manipolando l'espressione algebrica con teoremi ed assiomi. Tuttavia, nella manipolazione delle funzioni logiche conviene spesso riferirsi ad una espressione standard, chiamata "canonica".

Un termine (somma o prodotto) è canonico quando in esso appaiono esplicitamente tutte le variabili, o in forma vera o in forma negata.

Una funzione è in forma canonica quando tutti i suoi termini sono canonici.

Per esempio, nella funzione

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

solo il primo termine è canonico.

Una funzione in forma S.d.P. diventa canonica moltiplicandola per i termini

$$(A+\overline{A}),\ (B+\overline{B}),\ (C+\overline{C}),...$$

essendo A,B,C... le variabili da cui Y dipende. Questi termini valgono 1, per la (1.5a), e per la (1.4b) la funzione non cambia dopo la moltiplicazione.

Come esempio, canonizziamo la funzione precedente.

Moltiplicando per $(A+\overline{A})$ e applicando la proprietà distributiva (1.3a) si ha

$$Y = A\overline{A}BC + \overline{A}\,\overline{A}BC + A\overline{B}C + \overline{A}\,\overline{B}C + AA\overline{C} + A\overline{A}\,\overline{C}$$

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

Come si vede, i termini che già includevano la variabile A non vengono modificati, e per loro la moltiplicazione risulta superflua. Tenendo presente questa osservazione, il procedimento si snellisce molto.

Moltiplicando ora Y per $(B+\overline{B})$, si ha

$$Y = \overline{A}BC + A\overline{B}C + \overline{A}\,\overline{B}C + AB\overline{C} + A\overline{B}\,\overline{C}$$

È inutile procedere alla terza moltiplicazione, poiché la funzione è ormai canonica.

Una funzione in forma P.d.S. diventa canonica sommando ad essa i termini

$$A\overline{A},\ B\overline{B},\ C\overline{C},...$$

Questi termini valgono 0, per la (1.5b) e, per la (1.4a) la funzione non cambia. Esempio:

$Y = (A+B)(A+\overline{C})$

Sommando $B\overline{B}$, per la proprietà distributiva (1.3b) si ha

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

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

Sommando $C\overline{C}$ e procedendo come sopra, si completa la canonizzazione.

Dai due procedimenti illustrati si deduce una osservazione conclusiva che suggerisce un metodo rapido per la canonizzazione: se un termine (somma o prodotto) già contiene la variabile rispetto alla quale si sta canonizzando, esso resta inalterato; in caso contrario, da tale termine ne vengono generati due, uno contenente la variabile in forma vera, l'altro in forma negata.

1.6 - ANALISI DI UNA FUNZIONE BOOLEANA

L'analisi di una funzione booleana consiste nello scrivere la sua tavola della verità.

La tavola della verità è costruita elencando tutti gli stati logici distinti che le variabili possono assumere (stati di entrata della rete), e indicando, a fianco di ciascuno stato di entrata, lo stato logico assunto dall'uscita.

Se il numero di variabili è n, il numero di stati distinti di entrata (e quindi di righe della tavola) è 2^n: si tratta infatti delle disposizioni con ripetizione di due oggetti (0 e 1) su n posti. È facile convincersi che tali disposizioni si ottengono scrivendo in codice binario a n bit i numeri interi da 0 a 2^n-1. In fig. 1.8 è mostrata la tavola per funzioni di tre variabili.

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Fig.1.8

Una volta identificati gli stati delle variabili, si può procedere a determinare, riga per riga, il corrispondente valore dell'uscita Y sostituendo alle variabili il valore indicato nella riga. In pratica, la procedura può essere snellita di molto facendo ricorso ad opportuni teoremi, limitandosi a cercare le righe in cui la funzione vale 1 oppure quelle in cui vale 0.

Se la funzione è scritta come S.d.P., conviene limitarsi a cercare i casi in cui la funzione assume valore 1, applicando il teorema 1.7a all'intera espressione e il teorema 1.4b ai singoli termini prodotto: basta quindi cercare le righe in cui i singoli termini prodotto valgono 1.

Consideriamo per esempio la funzione, tabulata in Fig. 1.8

$$Y = A + B\overline{C}$$

Per il teorema (1.7a), Y vale 1 se

- A vale 1 (sottinteso: indipendentemente dal valore assunto da B e da C): di conseguenza, Y assume valore 1 nelle righe 5-8 della tavola.

oppure se

- $B\overline{C}$ vale 1 (sottinteso: indipendentemente dal valore logico assunto da A): per il teorema 1.4b, questo accade quando B vale 1 e C vale 0. Di conseguenza, Y assume valore 1 nelle righe 3 e 7 della tavola.

Si potrebbero anche cercare le righe in cui la funzione assume valore 0, utilizzando il teorema1.4a: ma occorrerebbe trovare le righi in cui tutti i termini della somma assumono valore 0, il che è chiaramente molto meno agevole della procedura precedente.

Se la funzione è scritta in forma P.d.S., conviene limitarsi a cercare i casi in cui la funzione assume valore 0, applicando il teorema 1.7b all'intera funzione e 1.4a ai singoli termini somma: basta quindi limitarsi a cercare le righe in cui i singoli termini somma valgono 0.

1.7 - SINTESI DI UNA FUNZIONE BOOLEANA

La sintesi consiste nel ricavare l'espressione algebrica di una funzione, una volta nota la tavola della verità.

I procedimenti di sintesi sono due, uno duale dell'altro, e danno la funzione in forma canonica come S.d.P. o P.d.S., rispettivamente.

Col primo procedimento, ogni riga della tavola della verità in cui Y vale 1 genera un termine prodotto canonico, nel quale le variabili appaiono in forma vera se nella riga valgono 1, in forma negata se valgono 0. La funzione cercata è la somma dei termini prodotto così ottenuti.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig.1.9

Col secondo procedimento, ogni riga della tavola della verità in cui Y vale 0 genera un termine somma canonico, nel quale le variabili appaiono in forma vera se nella riga valgono 0, in forma negata se nella riga valgono 1. La funzione cercata è il prodotto dei termini somma così ottenuti.

Così, per esempio, dalla tavola della verità di **Fig. 1.9**, col primo procedimento si ha:

$$Y = \overline{A}\,B + A\,\overline{B}$$

Col secondo procedimento si ha:

$$Y = (A + B)(\overline{A} + \overline{B})$$

1.8 - IMPLEMENTAZIONE DI UNA FUNZIONE. LOGICHE A NAND E A NOR

L'implementazione consiste nella costruzione della rete relativa ad una funzione logica. Per far questo, occorre disporre di "elementi di circuito" che meccanizzino ciascuno dei tre operatori logici. Ogni elemento di circuito è costituito a sua volta da una rete, nota come "porta logica" (gate) e oggi disponibile in forma integrata. La struttura di tale rete, la sua complessità e le sue prestazioni, dipendono dalla tecnologia usata nella costruzione del circuito integrato. A ciascuna tecnologia corrisponde una "famiglia logica", a sua volta divisa in sottofamiglie. Su questo torneremo ampiamente nel seguito.

Commercialmente, è oggi disponibile un vasto assortimento di "moduli integrati" (o, più semplicemente, "integrati") contenenti al loro interno sia semplici porte logiche (in numero varia bile a seconda del tipo di porta) sia reti estremamente complesse (per esempio, un microprocessore). Tali integrati sono costruiti secondo standard ben definiti, sia meccanici (la struttura più diffusa è quella 'dual-in-line, DIP) che elettrici. Ogni integrato è identificato da una sigla, costituita da una parte letterale (caratteristica del costruttore) e da una alfanumerica, standard. Così, per esempio, il 7408 è un integrato della famiglia TTL standard che contiene quattro porte AND a due entrate (per logica positiva); il 74LS04 è un integrato della famiglia TTL LS che contiene sei porte NOT, ecc. La mappa interna dell'integrato, cioè come i piedini del modulo sono collegati alle porte interne, è reperibile sui cataloghi del costruttore. Un esempio è dato tra le figure.

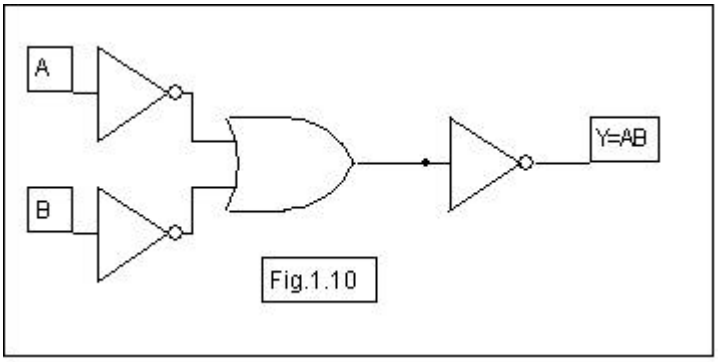


Fig.1.10

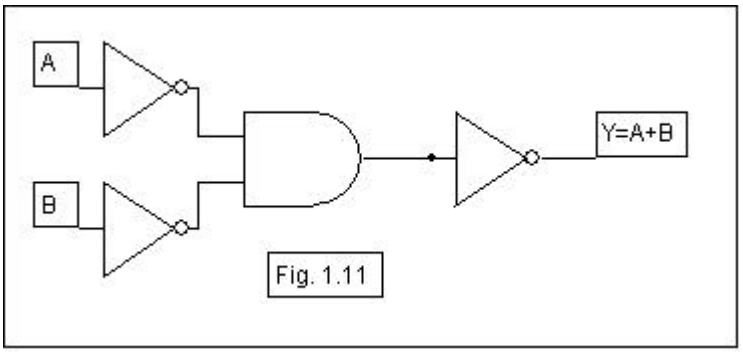


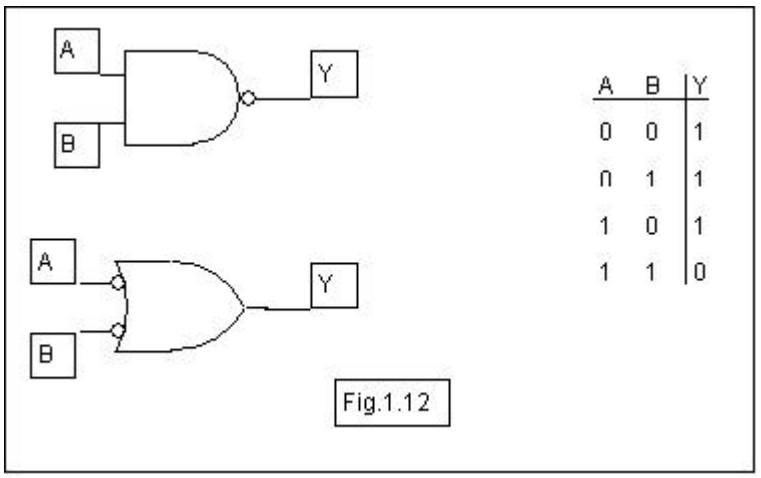
Fig. 1.11

Le porte logiche verranno per ora trattate come "scatole nere", delle quali interessano soltanto le proprietà terminali (cioè la tavola della verità); della loro struttura interna ci occuperemo al momento opportuno.

C'è da chiedersi, ora, quanti elementi di circuito siano necessari e sufficienti per realizzare una qualunque rete logica. Si è già detto che *è sufficiente* disporre di tre tipi di porte logiche, quanti sono

gli operatori logici. In realtà, con il teorema di De Morgan è facile vedere che le tre operazioni logiche possono essere implementate con due sole porte logiche. Consideriamo, infatti la porta AND a due entrate, la cui funzione è

$$Y = AB$$



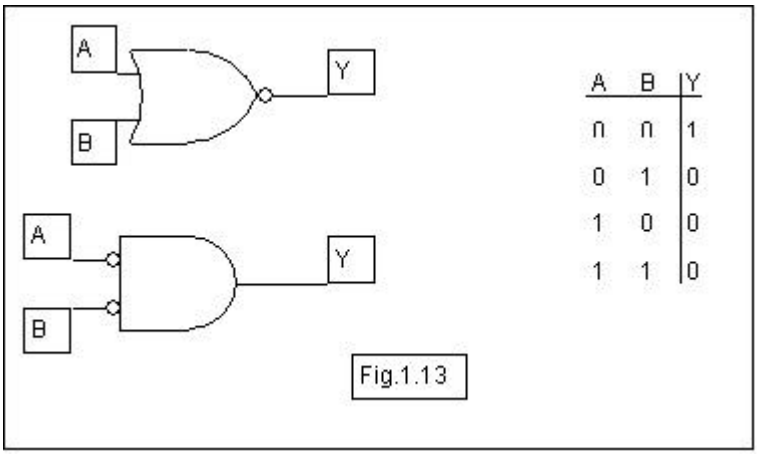
Con il teorema 1.14 e con quello di De Morgan, la stessa Y può essere scritta

$$Y = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$$

La rete relativa a tale espressione è in **Fig. 1.10**. Si conclude che la porta AND a due (o, in generale, a n) entrate può essere realizzata mediante una porta OR a due (n) entrate e tre (n+1) inverter. [Questa stessa conclusione può essere raggiunta ricordando che un AND per logica positiva è anche un OR per logica negativa].

Analogamente, è possibile vedere che la porta OR a n entrate è realizzabile mediante la porta AND a n entrate e n+1 inverter, **Fig. 1.11**.

Pertanto, *per realizzare i tre operatori logici (e quindi, per costruire qualunque rete logica) è necessario e sufficiente disporre di due soli tipi di porte logiche*: la porta OR e la porta NOT, oppure la porta AND e la porta NOT.



Si può allora pensare di inglobare le due porte in una unica rete in modo da usare un solo elemento di circuito. Le soluzioni oggi adottate sono due (una duale dell'altra). La prima è l'AND seguito da inverter, cui si dà il nome di NAND; la seconda è l'OR seguito da inverter, cui si dà il nome di NOR.

Nel primo caso la funzione realizzata è

$$Y = \overline{AB} = \overline{A} + \overline{B}$$

Tavola della verità del NAND e simboli circuitali corrispondenti alle due espressioni algebriche sono in **Fig. 1.12**.

Nel secondo caso, la funzione realizzata è

$$Y = \overline{A + B} = \overline{A}\,\overline{B}$$

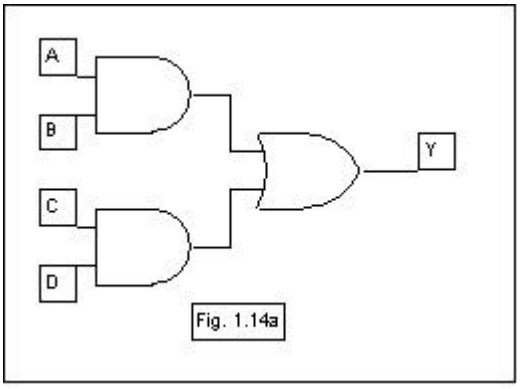
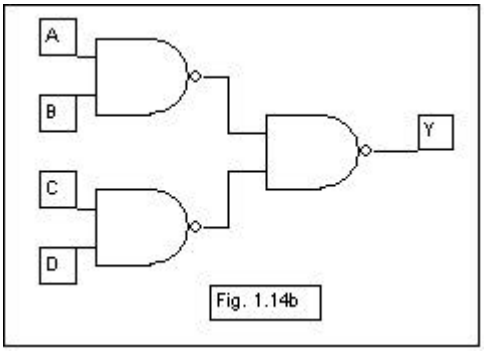


Tavola della verità del NOR e simboli circuitali sono in **Fig. 1.13**.

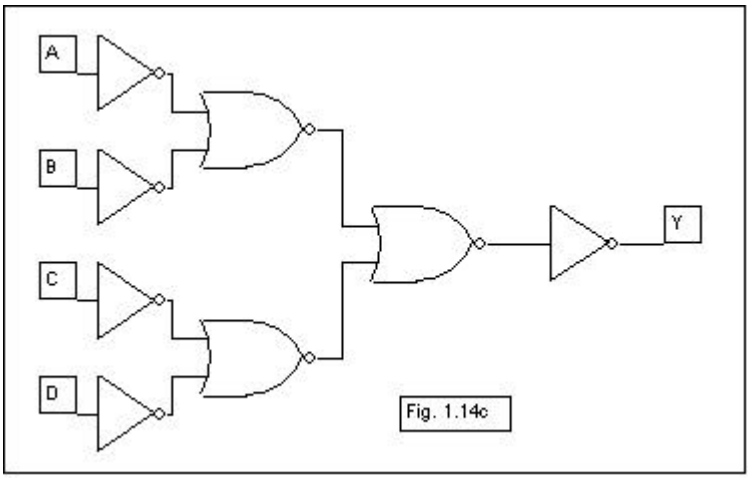
È facile convincersi che un NAND per logica positiva (negativa) è anche un NOR per logica negativa (positiva).

Da un NAND o da un NOR si ha un inverter collegando insieme le entrate (prima e ultima riga della tavola della verità). Allora appare chiaro che disponendo di soli NAND, o di soli NOR, è possibile realizzare le tre operazioni logiche, ed è quindi possibile costruire qualunque rete logica. Il NAND, o il NOR, ha quindi la funzione di *porta logica universale*.



Le reti, oggi, vengono costruite in logica a NAND o a NOR, anziché in logica AND-OR-INVERTER (AOI).

Poiché le regole di sintesi di una funzione, studiate in precedenza, forniscono una espressione algebrica (in forma P.d.S. oppure S.d.P.) direttamente implementabile in logica AOI, occorre comprendere come convertire l'espressione algebrica se si vuole una realizzazione in logica a NAND o in logica a NOR.



Ancora una volta, il problema è risolto con il teorema di De Morgan.

Si abbia per esempio la funzione

$$Y = AB + CD$$

la cui rete, in logica AOI, è in **Fig. 1.14a**.

Con i teoremi 1.14 e di De Morgan, si ha

$$= \overline{\overline{AB} + \overline{CD}} = \overline{\overline{AB}}\,\overline{\overline{CD}}$$

che è realizzabile con soli NAND, **Fig. 1.14b**.

Con gli stessi teoremi, si ha anche

$$\overline{\overline{AB}} + \overline{\overline{CD}} = \overline{\overline{A} + \overline{B}} + \overline{\overline{C} + \overline{D}} = \overline{\overline{\overline{A} + \overline{B}} + \overline{\overline{C} + \overline{D}}}$$

che è realizzabile con soli NOR, **Fig. 1.14c**.

Si noti il gran numero di inverter necessari nella versione a NOR. In generale, se l'espressione di Y è in forma S.d.P., la soluzione a NAND è più conveniente di quella a NOR. Viceversa, se la Y è in forma P.d.S.

1.9 - CARATTERISTICHE DELLE FAMIGLIE LOGICHE

Si è già detto che le famiglie logiche si distinguono per la struttura circuitale e per i componenti che vengono adoperati per costruire la porta base della famiglia (per esempio, il NAND). Questo fa sì che le prestazioni ottenibili da una rete logica siano fortemente dipendenti dalla famiglia cui appartengono i componenti usati per costruire la rete stessa. Pertanto, il progettista prima di passare alla realizzazione della rete deve decidere quale famiglia usare, in vista della particolare applicazione cui la rete è destinata. Questa decisione si basa su alcuni parametri comparativi delle famiglie logiche che è bene conoscere subito, e che possono essere illustrati indipendentemente dalla struttura interna delle porte, la cui descrizione è rimandata più avanti, quando verrà anche chiarito il significato delle sigle che distinguono le varie famiglie.

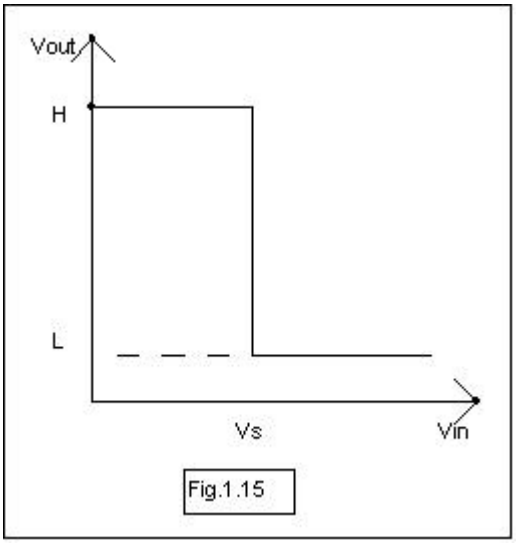
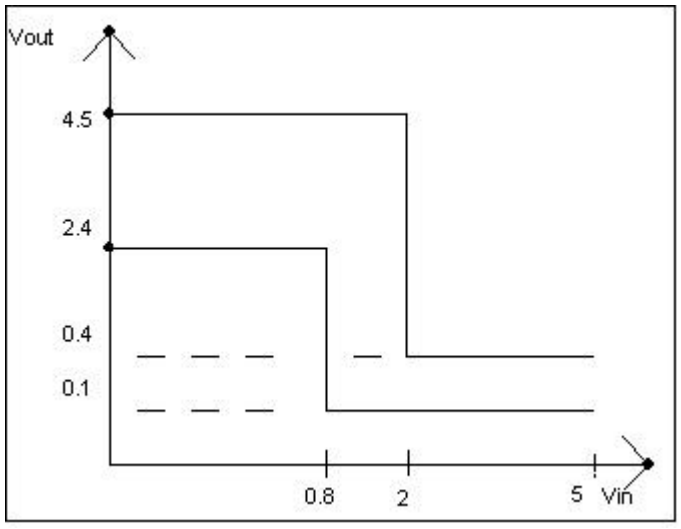


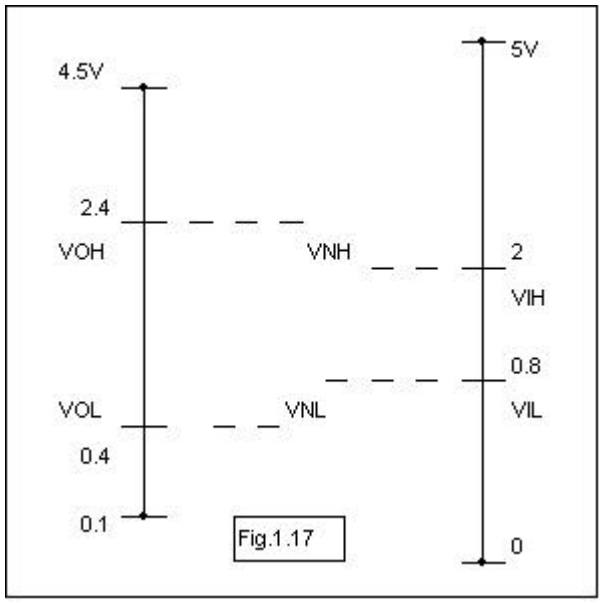
Fig.1.15

Scegliere i componenti di una famiglia piuttosto che di un'altra è spesso legato all'applicazione specifica della rete da realizzare: per esempio, è del tutto ovvio che, dal punto di vista della massima frequenza di commutazione, i problemi che pone la logica di controllo di una lavatrice sono diversi da quelli che si incontrano nel realizzare una scala di conteggio per impulsi con frequenza di ripetizione di 200 MHz. Infatti, nel primo caso la logica può essere implementata anche con interruttori elettromeccanici, che permettono frequenze di commutazione di pochi Hz (sufficienti, tuttavia, per l'uso specifico); nel secondo caso, occorre ricorrere a componenti della famiglia ECL, che hanno la adeguata velocità di commutazione. Come altro esempio, si pensi ai problemi di potenza a bordo di un satellite artificiale: qui è problematico non solo generare la potenza necessaria al funzionamento dell'elettronica a bordo, ma anche dissipare il calore sviluppato dal consumo di energia: l'unico modo è l'irraggiamento, per cui occorre adoperare integrati a bassissima dissipazione di potenza, quali quelli della famiglia CMOS. Questo problema, in generale, è molto meno sentito in apparecchiature a terra, ove è sempre possibile ricorrere al condizionamento degli ambienti.

Descriviamo, ora, gli elementi comparativi delle famiglie logiche. Per comodità, gli esempi numerici si riferiranno alla famiglia TTL standard, che a tutt'oggi è la più diffusa.

Occorre, anzitutto, definire i livelli logici. A tale scopo, possiamo anticipare che la struttura circuitale di una porta è quella di un amplificatore ad alto guadagno. Se si misura la sua caratteristica ingresso-uscita si ha, per un inverter, la curva di **Fig. 1.15** (il guadagno $\Delta V_{out}/\Delta V_{in}$ è stato per semplicità supposto infinito e la transizione è stata tracciata verticale). In questa figura, si è indicato con H e L la tensione più alta e quella più bassa, rispettivamente, misurata in uscita. Si è poi indicato con Vs la tensione di entrata a cui avviene la commutazione. Ripetendo la misura su altri esemplari di inverter, si trova che sia V_s, sia le tensioni di livello alto, H, che di livello basso, L, di uscita cambiano. Ciò è dovuto alle tolleranze di costruzione. Se si fa la misura su un gran numero di inverter TTL, si ottiene la famiglia di caratteristiche di trasferimento. In **Fig. 1.16,** le curve marcate sono le caratteristiche estreme della famiglia. Queste curve mostrano che i livelli logici di uscita da una porta TTL possono variare (a vuoto) nell'intervallo 0.1-0.4 volt per il livello basso, e 2.4-4.5 volt per il livello alto. In altri termini, scegliendo a caso una porta TTL, i livelli logici L e H che si misureranno alla sua uscita cadranno nei suddetti intervalli, rispettivamente (non accadrà mai di misurare una tensione compresa fra 0.4 e 2.4 volt).





Analogamente, dal lato entrata, i due livelli logici non sono discriminati da una tensione V_s ben definita per qualunque porta: le curve di **Fig. 1.16** evidenziano che se scegliamo a caso una porta TTL, possiamo affermare in anticipo che essa riconoscerà sicuramente come livello basso le tensioni di entrata comprese nell'intervallo 0 - 0.8 volt, e riconoscerà sicuramente come livello alto le tensioni comprese nell'intervallo 2 - 5 volt. Le tensioni intermedie, cioè fra 0.8 e 2 volt,

non sono utilizzabili, nel senso che non è garantito che una porta TTL scelta a caso riconosca in modo non equivoco una tensione in tale intervallo: per esempio, un livello di 1.5 volt all'ingresso potrà essere riconosciuto come livello basso da alcuni esemplari e come livello alto da altri.

La **Fig. 1.17** riassume quanto ora detto. Sulla stessa figura sono indicati alcuni simboli caratteristici, di cui è bene conoscere il significato.

V_{IL}: massima tensione riconosciuta in entrata come livello basso

V_{IH}: minima tensione riconosciuta in entrata come livello alto

V_{OL}: massima tensione misurata in uscita nello stato basso

V_{OH}: minima tensione misurata in uscita nello stato alto

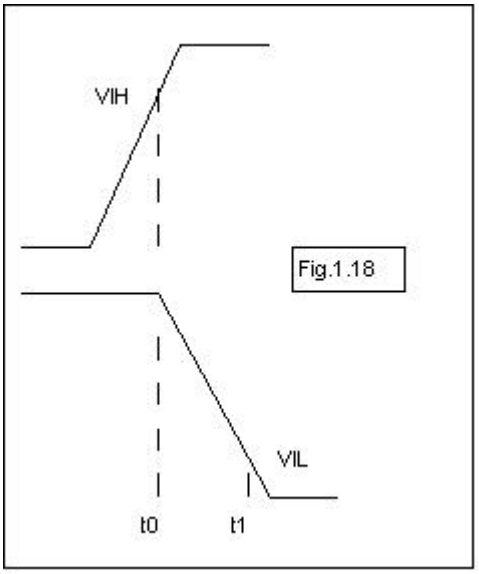


Fig.1.18

Un primo parametro caratterizzante per una famiglia logica è *l'immunità al rumore*, cioè la capacità della porta di sopportare un disturbo sovrapposto al livello logico di entrata senza che quest'ultimo venga interpretato in modo errato. L'immunità al rumore viene espressa dal costruttore mediante i "*margini di rumore*", così definiti:

$$V_{NH} = V_{OH} - V_{IH}$$

$$V_{NL} = V_{IL} - V_{OL}$$

rispettivamente per il livello alto e per il livello basso. Tali margini, per la famiglia TTL, sono entrambi eguali a 0.4 volt, come appare dalla **Fig. 1.17**.

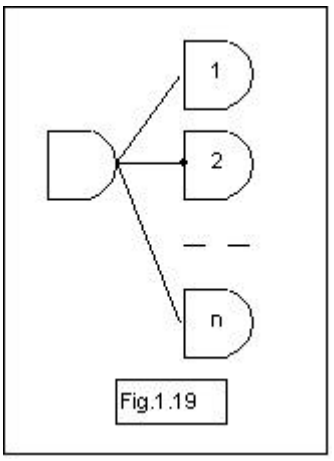


Fig.1.19

Un secondo parametro caratteristico di una famiglia è il *ritardo di propagazione*, t_{pd}, nella porta base. A tale ritardo è strettamente legata la massima velocità di commutazione consentita in una rete. Infatti, è intuitivo che per evitare errori nello stato di uscita di una rete è necessario che l'intervallo fra due transizioni di entrata sia almeno pari al ritardo di propagazione totale della rete. Il ritardo di propagazione di una porta viene definito, **Fig. 1.18**, come l'intervallo di tempo fra l'istante t_0 in cui l'entrata ha raggiunto un livello sicuramente riconoscibile come nuovo stato e l'istante t_1 in cui la transizione di uscita ha raggiunto anch'essa un

livello sicuramente riconoscibile come nuovo stato. In generale, il ritardo di propagazione è diverso a seconda del tipo di transizione (H->L, oppure L->H).

Il terzo parametro caratteristico è il *fan-out*, che può essere definito come segue. Consideriamo una porta P di una famiglia e colleghiamo alla sua uscita, Y, n ingressi di porte appartenenti alla stessa famiglia, **Fig. 1.19**. Come si vedrà nel Capitolo IV, per mantenere l'ingresso di una porta nello stato alto è necessario fornire corrente a tale ingresso; il valore massimo di tale corrente è indicato con I_{IH}, e nella famiglia TTL è di 40 μA. Per mantenere un ingresso nello stato basso, è invece necessario assorbire corrente da esso; il massimo valore di tale corrente, I_{IL}, è di 1.6 mA nella famiglia TTL. I valori I_{IH} e I_{IL} sono quelli che assicurano che i livelli logici saranno riconosciuti in modo corretto da una porta scelta a caso.

Ovviamente, l'uscita Y della porta P può assorbire o erogare una corrente finita, i cui valori minimi sono I_{OL} e I_{OH} (rispettivamente 16 mA (per il livello L) e 0.4 mA (per il livello H) nella famiglia TTL). È chiaro che, volendo mantenere i livelli logici corretti, n non può essere arbitrariamente grande. Il suo massimo valore sarà $n=I_{OH}/I_{IH}$ per il livello alto, e $n'=I_{OL}/I_{IL}$ per il livello basso. Tale massimo valore si chiama *fan-out*: nella famiglia TTL, esso è 10 per entrambi i livelli.

Un altro elemento di paragone fra le varie famiglie può essere la *tensione di alimentazione* richiesta. Alcune famiglie (per esempio, la TTL) richiedono una tensione fissa con stretta tol- leranza (+5 volt, +-5%). La famiglia ECL richiede sia una tensione positiva che una negativa (oppure solo negativa), il che può risultare piuttosto scomodo. Altre famiglie accettano senza problemi alimentazioni variabili in un ampio intervallo (per esempio le porte CMOS accettano tensioni comprese tra 2 e 15 volt).

Per quel che riguarda, infine, la *potenza dissipata per gate*, staticamente essa è strettamente legata alla tecnologia costruttiva della famiglia, come si vedrà, e può variare in condizioni statiche da meno di 1 microwatt per gate, come nella famiglia CMOS, a decine di milliwatt per gate in altre famiglie. Per ora, possiamo sottolineare che alla potenza dissipata è legato il ritardo di propagazione. Infatti, le capacità distribuite nel circuito sono abbastanza definite; se si riduce la corrente, per ridurre la dissipazione, occorre più tempo per caricare le capacità fino alla soglia di commutazione V ($V = Q/C = It/C$), e quindi aumenta il ritardo di propagazione. Talvolta, viene considerato come fattore di merito il prodotto fra la potenza dissipata dalla porta e il suo ritardo di propagazione: tale prodotto ha le dimensioni di un lavoro, e si misura in picoJoule (pJ).

La tavola di **Fig. 1.20** riassume alcune caratteristiche per le più diffuse famiglie logiche.

CARATTERISTICHE MEDIE, PORTA BASE									
FAMIGLIA	**TTL**					**ECL**	**CMOS**		
SOTTOFAMIGLIA	**STANDARD**	**LS**	**ALS**	**S**	**AS (FAST)**	**ECL**	**STANDARD**	**HC (HCT)**	**AC (FACT)**
Margine di rumore, V	**0.4**	**0.4**	**>0.4**	**>0.4**	**>0.4**	**0.15**	**1**	**1**	**1.25**
Tpd, nsec	**9**	**8**	**4**	**3**	**2**	**<2**	**50 (su 50 pF)**	**9 (“)**	**3 (“)**
Fclock, MHz	**25**	**33**	**34**	**95**	**125**	**>200**	**8**	**60**	**>100**
Potenza statica, mW	**15**	**3**	**2**	**25**	**12**	**>25**	**.000001**	**.0001**	**.0001**
Potenza a 1 MHz, su 50 pF, mW							**1**	**1**	**1**
Fan out	**10**	**20**	**20**	**10**	**>40**	**20**	**2(su LS)**	**10 (“)**	**60 (“)**
Vcc, V	**5**	**5**	**5**	**5**	**5**	**-5**	**3-15**	**2-6**	**2-6**
VIH, V	**2**	**2**	**2**	**2**	**2**		**3.5**	**3.15**	
VIL, V	**0.8**	**0.8**	**0.8**	**0.8**	**0.8**		**1.5**	**0.9**	
VOH, V	**2.4**	**2.7**	**Vcc-2**	**2.7**	**Vcc-2**		**4.5**	**4.4**	
VOL, V	**0.4**	**0.4**	**0.5**	**0.5**	**0.5**		**0.5**	**0.3**	

Fig. 1.20

1.10 - CRITERI REALIZZATIVI DI UNA RETE LOGICA

Conviene ora osservare che una funzione logica può essere rappresentata mediante varie espressioni algebriche: infatti, si è già visto che partendo da una di tali espressioni è possibile ricavarne numerose altre con l'uso di teoremi e assiomi, senza che la tavola della verità della funzione cambi.

Il problema è quale espressione conviene scegliere nella realizzazione.

È ragionevole attendersi che la rete logica da realizzare soddisfi le tre condizioni seguenti:

- abbia il minimo costo possibile: cioè la rete deve richiedere il minimo numero di circuiti integrati (costo dei componenti) ed il minimo numero di collegamenti (costo di realizzazione).
- Abbia la massima velocità operativa: questo equivale a chiedere che sia minimo l'intervallo di tempo fra l'istante t_0 in cui una variabile di entrata cambia stato logico e l'istante t_1 in cui l'uscita cambia, di conseguenza, stato. Poiché ogni porta logica è caratterizzata da un ritardo di propagazione, questo equivale a chiedere che i segnali attraversino il minor numero possibile di porte logiche per arrivare all'uscita della rete.
- Funzioni in modo affidabile: cioè lo stato dell'uscita in corrispondenza di un assegnato stato delle entrate deve essere in ogni istante corretto, cioè quello previsto dalla tavola della verità.

Purtroppo, queste tre condizioni sono quasi sempre tra loro incompatibili, nel senso che l'ottimizzazione di una delle tre condizioni spesso contrasta con quella delle altre due: per esempio, la versione di minimo costo di una rete non sempre coincide con quella che ha la massima velocità operativa, né è detto che funzioni in maniera affidabile.

Nei paragrafi seguenti illustreremo i metodi per realizzare separatamente le tre esigenze, evidenziando le incompatibilità fra le medesime.

1.11 - MINIMIZZAZIONE DI UNA FUNZIONE LOGICA

I metodi noti per la minimizzazione del costo mirano a minimizzare il numero P di porte AND e OR necessarie per l'implementazione (costo del materiale), ed il numero E delle interconnessioni, pari al numero di entrate alle porte (costo del montaggio); il numero degli inverter viene ridotto successivamente, come si dice più avanti. Poiché P e E sono indipendenti, come fattore di merito della rete, dal punto di vista della minimizzazione, si può assumere il prodotto PE.

Data una equazione algebrica in forma qualunque, si può verificare che P è uguale al numero di termini prodotto o somma contenenti più di una lettera, più uno; E è uguale al numero di termini prodotto o somma contenenti più di una lettera, più il numero di lettere che compaiono nell'equazione.

La ricerca della espressione minima deve ovviamente avvenire con operazioni lecite, quindi mediante l'uso di teoremi o assiomi dell'algebra booleana. In effetti, il teorema che appare adatto allo scopo è (1.12) che permette di ridurre sia P che E: due termini (prodotto o somma) differenti per una variabile vengono sostituiti con uno solo, costituito dalla parte invariante dei due termini stessi.

Due metodi basati sulla applicazione sistematica di questo teorema, che permettono di giungere rapidamente alla soluzione, verranno illustrati nei paragrafi seguenti. La soluzione viene fornita come S.d.P. o come P.d.S. e si vedrà fra poco che essa è anche la versione più veloce della rete. Tuttavia, può accadere che, ricorrendo ad altri teoremi, si possa ridurre ulteriormente il prodotto PE: questo in ogni caso va a scapito della velocità della rete.

Va precisato che i metodi di minimizzazione hanno una giustificazione logica, cioè si fondano sui teoremi dell'algebra booleana; pertanto non possono tener conto delle particolarità costruttive delle porte logiche. Per esempio, nel paragrafo 1.15 si fa vedere come è possibile talvolta implementare una funzione con risparmio di porte, sfruttando la proprietà di alcune di esse di avere le uscite direttamente collegabili insieme (logica cablata).

Inoltre, è opportuno sottolineare che, poiché le porte sono disponibili non come singolo elemento, ma a gruppi di due, tre, quattro,… in un unico contenitore, la minimizzazione del prodotto PE per una rete realizzata in "logica sparsa" (cioè connettendo insieme circuiti integrati distinti) ha significato quando quest'ultima è vista come facente parte di una rete di più ampie dimensioni, nella quale, quindi, è ragionevole supporre che gli integrati saranno utilizzati pressoché al 100%. Per reti di

piccole dimensioni, espressioni della stessa funzione con prodotti PE diversi possono richiedere lo stesso numero di integrati.

Con lo sviluppo della tecnologia di integrazione su larga scala (realizzazione di reti logiche complesse su singolo circuito integrato) il costo del prodotto finale non può più essere valutato semplicemente in termini di prodotto PE: infatti il costo dipende essenzialmente dall'area di silicio occupata. Naturalmente, anche in questo caso la minimizzazione ha grande interesse, perché permette di realizzare reti più complesse sulla stessa superficie di silicio.

Prima di passare ad illustrare i metodi di minimizzazione, conviene anticipare come essi funzionano. Essi applicano il teorema (1.12) a tutte le possibili coppie di termini (non si può sapere a priori quale coppia darà la soluzione più conveniente) il massimo numero di volte possibile. I termini ottenuti alla fine di questa operazione (quando il teorema non è più applicabile) costituiscono *il set irriducibile di implicanti primi*. Da questo set va successivamente estratto il sub-set che costituisce la soluzione cercata (espressione minimale della funzione). In generale, ci saranno più soluzioni possibili (cioè più sub-set), e la scelta finale va fatta sulla base di altri criteri, per es. la minimizzazione degli inverter.

In seguito considereremo esempi particolarmente semplici; ma possono verificarsi casi in cui la ricerca della soluzione minimale non è banale. Per una trattazione più completa, si veda [1].

1.12 - IL METODO DI QUINE

Questo metodo si basa, come già detto, sulla applicazione sistematica del teorema (1.12). Per la prima fase, cioè la ricerca del set irriducibile di implicanti primi, si comincia con lo scrivere la funzione da minimizzare in forma canonica, per esempio S.d.P., e si trascrivono i termini in una tabella.

Si comincia quindi a confrontare ciascun termine con tutti i successivi, cercando quelli che differiscono da esso per una variabile.

Il fatto che un termine venga utilizzato in più confronti non deve suscitare perplessità: infatti, ciò è da un lato consentito dal teorema (1.6), secondo il quale un termine può essere ripetuto quante volte si vuole nella espressione algebrica senza che questa cambi; d'altro lato ciò risulta indispensabile in quanto, come già detto, non si può sapere a-priori quale coppia darà il risultato più conveniente.

Ad ogni coppia trovata si applica il teorema (1.12), compilando così una seconda tavola con i termini semplificati.

Lo stesso procedimento viene applicato alla seconda tavola, compilandone una terza. Si procede così fino a quando il teorema (1.12) non è più applicabile.

Un termine al quale il teorema non è più applicabile è un implicante primo.

L'illustrazione del metodo risulterà più chiara procedendo con un esempio.

Consideriamo la funzione

$$Y=\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,BCD + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C\,\overline{D} + AB\overline{C}\,\overline{D} + AB\,\overline{C}\,D + ABC\,\overline{D} + ABCD$$

La prima tavola è in **Fig. 1.21a**. Il termine (1) può essere accoppiato con (8), generando il termine ABC; con (9), generando il termine ABD; con (10), generando il termine BCD: tutti trascritti nella seconda tavola. Il termine (2) può essere accoppiato con (3), con (6) e con (7); ecc.

Nella seconda tavola si procede nello stesso modo, come illustrato nella figura.

Nella terza tavola ci sono solo implicanti primi.

Il set irriducibile degli implicanti primi, ora ottenuto, è probabilmente ridondante, se si ricorda che è stato ricavato facendo tutti i confronti possibili.

Occorre ora identificare la (o le) somma minimale, *che è costituita dagli implicanti primi necessari e sufficienti a 'coprire' la funzione* (con ciò si intende dire che, partendo da tali implicanti primi, deve essere possibile, applicando a rovescio il teorema, ricostruire i termini canonici originari). A tale scopo, si costruisce una tavola, come in **Fig. 1.21d**, che ha come indici di riga gli implicanti primi, e come indici di colonna i numeri d'ordine dei termini canonici originali, cioè quelli nella prima tavola. In ogni riga, vengono quindi marcate le caselle in corrispondenza dei termini canonici che hanno generato l'implicante primo indicato sulla riga (ultima colonna di fig. 1.21c). Dalla tavola appare chiaro che l'implicante primo BD è certamente necessario, perché solo da esso si può risalire (applicando il teorema a rovescio) ai termini canonici (5) e (10). Lo stesso dicasi per $\overline{B}\,\overline{D}$ (solo da esso si può risalire a (3) e a (4)). Entrambi, però, non sono sufficienti a coprire la funzione (mancano i termini (7) e (8)): per ottenere la somma minimale è necessario aggiungere ad essi o l'implicante primo AB, ottenendo la soluzione

1	ABCD	**1'**	ABC	1,8	**1''**	AB	1',13',2',12'	1,7,8,9
2	$A\,\overline{B}\,\overline{C}\,\overline{D}$	**2'**	ABD	1,9	**2''**	BD	2',10',3',9'	1,5,9,10
3	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	**3'**	BCD	1,10	**3''**	$\overline{B}\,\overline{D}$	4',8',5',7'	2,3,4,6
4	$\overline{A}\,\overline{B}C\,\overline{D}$	**4'**	$\overline{B}\,\overline{C}\,\overline{D}$	2,3	**4''**	$A\,\overline{D}$	5',6',11',12'	2,6,7,8
5	$\overline{A}\,B\,\overline{C}\,D$	**5'**	$A\,\overline{B}\,\overline{D}$	2,6	**Fig.1.21c**			
6	$A\,\overline{B}C\,\overline{D}$	**6'**	$A\overline{C}\,\overline{D}$	2,7				
7	$AB\overline{C}\,\overline{D}$	**7'**	$\overline{A}\,\overline{B}\,\overline{D}$	3,4				
8	$ABC\,\overline{D}$	**8'**	$\overline{B}C\,\overline{D}$	4,6				
9	$AB\,\overline{C}\,D$	**9'**	$B\,\overline{C}\,D$	5,9				
10	$\overline{A}\,BCD$	**10'**	$\overline{A}\,BD$	5,10				
Fig.1.21a		**11'**	$AC\,\overline{D}$	6,8				
		12'	$AB\,\overline{D}$	7,8				
		13'	$AB\,\overline{C}$	7,9				
		Fig.1.21b						

-1.34-

$Y = AB + BD + \overline{B}\,\overline{D}$

oppure l'implicante primo $A\overline{D}$, ottenendo la soluzione

$Y = A\overline{D} + BD + \overline{B}\,\overline{D}$

Entrambe le soluzioni in questo caso hanno lo stesso costo e richiedono lo stesso numero di inverter; ma naturalmente questo non accade sempre. Per una discussione del caso più generale in cui non ci siano implicanti primi necessari, si veda [1].

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
AB	**X**						**X**	**X**	**X**	
BD	**X**				**X***				**X**	**X***
$\overline{B}\,\overline{D}$		**X**	**X***	**X***		**X**				
$A\overline{D}$		**X**				**X**	**X**	**X**		

Fig. 1.21d

1.13 - LA MAPPA DI KARNAUGH

Questa mappa è un rettangolo suddiviso in un numero di caselle pari al numero di righe della tavola

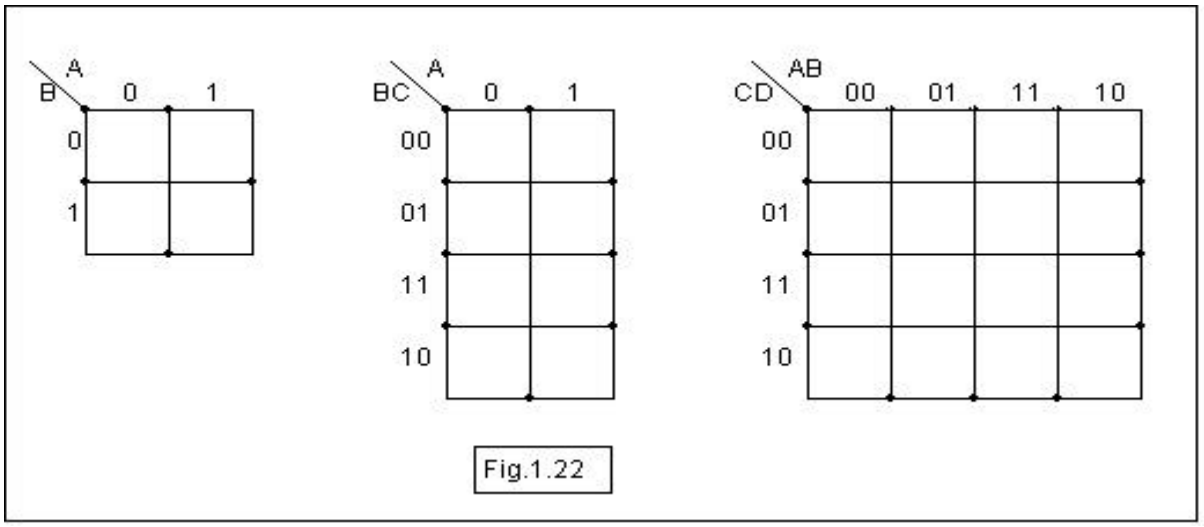


Fig.1.22

della verità della funzione Y che si vuole minimizzare, e può essere considerata un altro modo di scrivere la tavola della verità stessa.

Ogni casella viene associata ad una riga della tavola della verità in modo tale che due caselle adiacenti (cioè aventi un lato in comune) corrispondano a righe della tavola che differiscono per il valore di una sola variabile. L'associazione è inoltre tale che le due caselle alle estremità della stessa riga o della stessa colonna risultino adiacenti (cioè, è come se la mappa fosse chiusa a cilindro sia in orizzontale che in verticale).

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

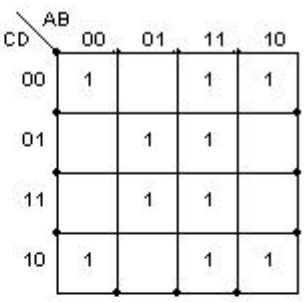


Fig.1.23

La soluzione per funzioni di due, tre, quattro variabili è in **Fig. 1.22**. I valori delle variabili nella tavola della verità sono opportunamente riportati come indici di riga e di colonna della mappa cosicché, scelta una casella, le sue coordinate individuano la riga della tavola della verità cui essa è associata. Nella casella andrà trascritto lo stato corrispondente dell'uscita Y.

La **Fig. 1.23** riporta tavola della verità e mappa di Karnaugh per la funzione di 4 variabili già considerata per illustrare il metodo di Quine. Consideriamo funzioni sintetizzate come S.d.P.; è conveniente in questo caso riportare nella mappa solo gli 1.

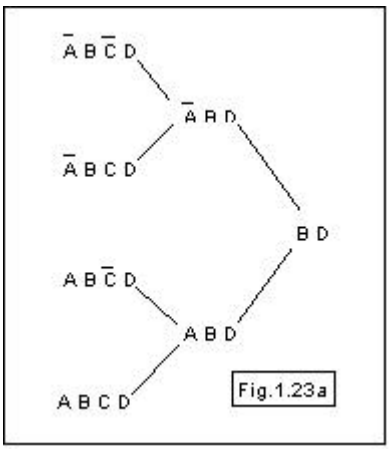


Fig.1.23a

Per come è strutturata la mappa, segue che se si considerano due caselle adiacenti contenenti un 1, esse corrispondono a due termini prodotto canonici differenti per una variabile, ai quali quindi può essere applicato il teorema (1.12). La mappa di K. ha quindi il pregio di mettere visualmente in evidenza le coppie a cui è applicabile il teorema. Ma non basta.

Ricordiamo che l'applicazione del teorema (1.12) ad una coppia di termini canonici, ciascuno di n variabili (prima tabella di Quine), produce un termine di n-1 variabili (seconda tabella di Quine). Può accadere che l'applicazione del teorema a due coppie di termini canonici, ciascuno di n variabili, produca due termini di n-1 variabili ai quali sia ancora applicabile il teorema, producendo infine un unico termine di n-2 variabili (terza tabella di Quine). Un esempio è in Fig. 1.23a.

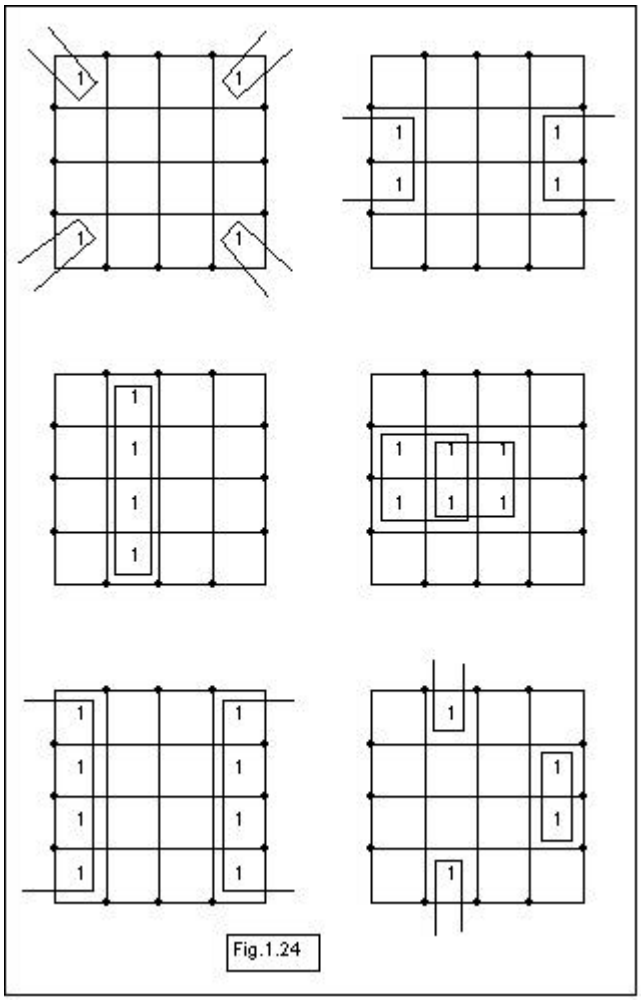


Fig.1.24

In sintesi, può accadere che partendo da un gruppo di 4 (2^2) termini, ciascuno di n variabili, si possa ricavare, applicando successivamente il teorema, un unico termine di n-2 variabili. Se si osserva la topologia delle caselle corrispondenti a questi 4 termini nella mappa di Fig. 1.23, si vede che ognuna delle 4 caselle è adiacente ad altre due del gruppo. Questa è una proprietà generale della mappa di K.: possiamo cioè affermare che se 2^k termini canonici di n variabili sono

tali che applicando successivamente il teorema (1.12) si giunge ad un implicante primo, di n-k variabili, allora le caselle corrispondenti nella mappa sono disposte in modo tale che ciascuna è adiacente ad altre k.

A differenza del metodo di Quine, la mappa di K. permette di individuare questi gruppi immediatamente, rendendo molto celere la minimizzazione.

Alcuni esempi di raggruppamenti validi sono in **Fig. 1.24**.

Da un gruppo si ottiene un implicante primo *quando il gruppo è distinto* (un gruppo di 2^k caselle si dice distinto quando non è interamente contenuto in un gruppo di 2^{k+1} caselle), *cioè ha la massima dimensione possibile* (infatti, per quanto sopra detto, questo significa che il teorema sarà applicato il massimo numero di volte ai termini del gruppo).

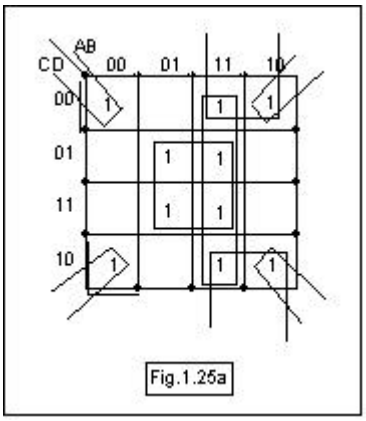


Fig.1.25a

Operativamente, è conveniente cominciare la ricerca degli implicanti primi a partire dai gruppi distinti di maggior dimensione; naturalmente, tutti gli 1 devono entrare a far parte di almeno un gruppo. Gli implicanti primi così ottenuti costituiscono il set irriducibile. La **Fig. 1.25a** mostra tale set per la tavola di **Fig. 1.23**.

L'espressione minimale della funzione è ottenuta considerando gli implicanti primi necessari e sufficienti a includere tutti gli 1 della mappa (cioè quelli necessari e sufficienti a coprire la Y). Questa (o queste) espressione viene trovata selezionando dapprima i gruppi che hanno almeno un 1 non compreso negli altri gruppi (questi 1 corrispondono agli X asteriscati nella Fig. 1.21d); poi si aggiungono altri gruppi fino a coprire la mappa (cioè fino ad includere tutti gli 1). L'analogia con il metodo di Quine è evidente.

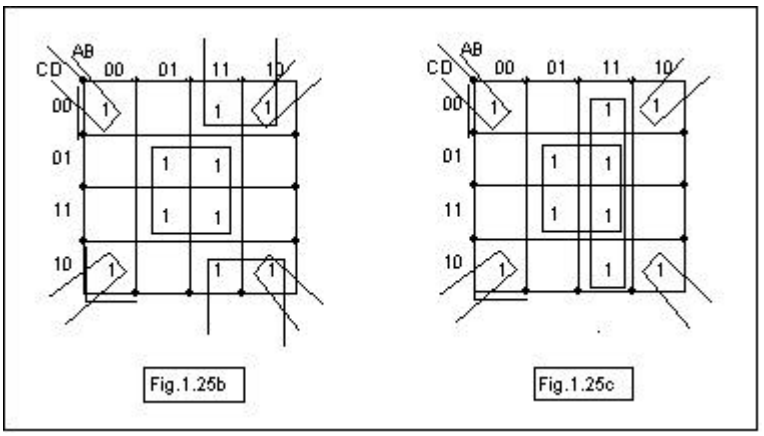
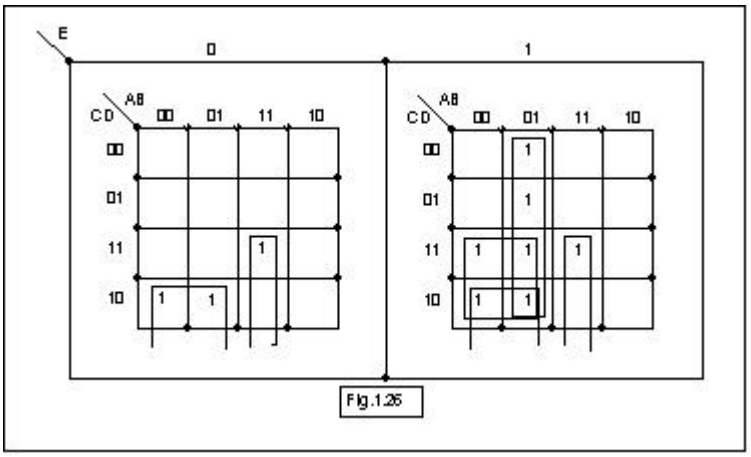


Fig.1.25b Fig.1.25c

Le **Fig. 1.25b,c** mostrano le due soluzioni per la funzione dell'esempio, ovviamente eguali a quelle già ottenute con Quine.



Consideriamo ora un esempio di mappa di K. per una funzione di 5 variabili. Occorrono 32 caselle e si ricorre a due mappe di 16,a loro volta contenute in una mappa da 2, come mostrato nell'esempio di **Fig. 1.26**. La quinta variabile assume valore logico 0 in una mappa e 1 nell'altra. Allora, è evidente che sono adiacenti anche due caselle, appartenenti ciascuna ad una delle due mappe da 16, le quali coinciderebbero qualora le mappe venissero sovrapposte.

Si potrebbe estendere l'uso della mappa di K. anche a funzioni di più di 5 variabili, ma il metodo diventa macchinoso ed è preferibile ricorrere al metodo di Quine.

Consideriamo infine il caso in cui la tavola della verità non sia specificata in modo completo: ciò può accadere o perché una particolare combinazione di variabili non si presenta mai nel problema in esame, oppure quando tale combinazione, pur presentandosi, non ha influenza sul funzionamento della rete. Si dice che quello stato di entrata "non importa", N.I..

L'uscita della rete in corrispondenza di uno stato N.I. è pertanto arbitrario, e può essere opportunamente scelto come 1 o come 0 così da ottenere una soluzione di minor costo. Nella mappa di K. la scelta migliore appare in genere immediata, dopo un rapido esame della mappa stessa. Nel caso del metodo di Quine, invece, occorre minimizzare la funzione sia assegnando valore 1 allo stato N.I. che assegnando 0. Solo confrontando i risultati a posteriori si può dire quale scelta è più conveniente. Esempi verranno visti nel seguito.

Se si desidera la soluzione minimale sintetizzata come P.d.S., si usa il procedimento duale cioè si considerano gli 0 della mappa. Il procedimento è identico.

1.14 - LIVELLI DI UNA RETE LOGICA E RITARDO DI PROPAGAZIONE

Il numero massimo di porte AND e OR che un segnale di ingresso alla rete deve attraversare per giungere all'uscita è il numero di livelli della rete.

Le funzioni scritte sotto forma P.d.S. o S.d.P. danno luogo a reti a due livelli: questo è anche il numero minimo di livelli per una funzione logica non banale.

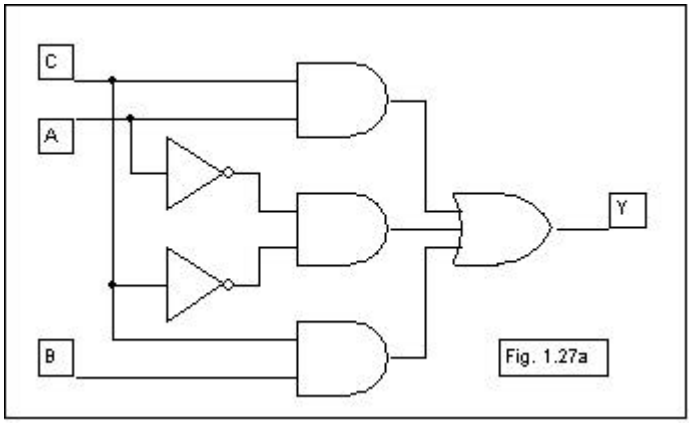


Fig. 1.27a

Poiché ogni porta logica è caratterizzata da un ritardo di propagazione del segnale, si comprende che la versione a due livelli di una rete è anche la più veloce.

I metodi di minimizzazione visti in precedenza danno una espressione minima a due livelli, che è quindi anche la soluzione più veloce. Come già detto, può accadere che una ulteriore elaborazione algebrica della funzione riduca ulteriormente il costo, ma a spese della velocità. Si consideri per esempio la funzione

$$Y = AC + \overline{A}\,\overline{C} + BC$$

La rete corrispondente è in **Fig. 1.27a**. Con la proprietà associativa si ha

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

la cui rete, a tre livelli, è in **Fig. 1.27b**. Se si considera il costo delle due soluzioni, nel primo caso si ha PE = 36, nel secondo caso PE = 32.

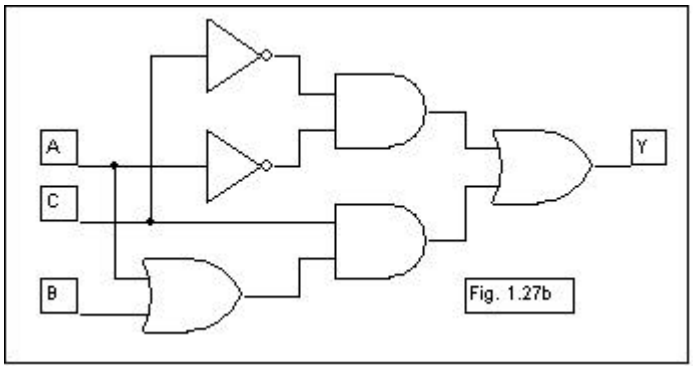


Fig. 1.27b

1.15 - L'AFFIDABILITÀ DI UNA RETE LOGICA

Una rete logica è affidabile se lo stato dell'uscita è in ogni istante quello previsto dalla sua tavola della verità.

La non affidabilità è imputabile alla differenza nei tempi di propagazione dei segnali di entrata attraverso diversi percorsi nella rete per giungere alla stessa uscita. Questo può far sì che l'uscita assuma un valore logico errato, temporaneamente (nel caso di reti combinatorie) o anche definitivamente (nel caso di reti sequenziali, delle quali si parla nel Capitolo III). Questo fenomeno viene chiamato alea.

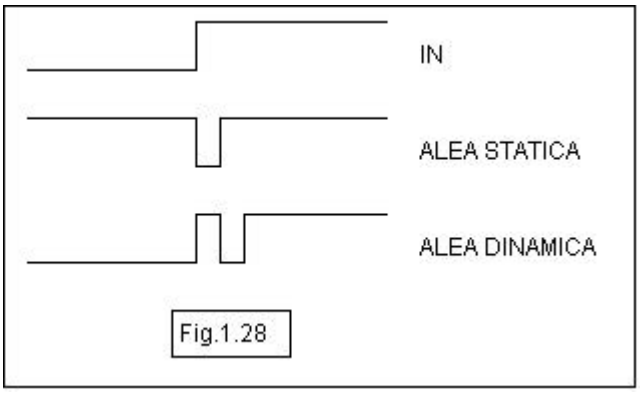


Fig.1.28

Consideriamo per ora le reti combinatorie e limitiamoci per semplicità all'alea associata al cambiamento di stato di una sola variabile di entrata: nel caso di cambiamento contemporaneo di più variabili, è più difficile prevedere e correggere l'alea.

Si distinguono due tipi di alea.

Una rete presenta alea statica quando l'uscita, pur dovendo restare nello stesso stato logico in seguito al cambiamento di una variabile di entrata, transita per breve tempo nell'altro stato.

Una rete presenta alea dinamica quando l'uscita, pur dovendo cambiare stato in seguito al cambiamento di una variabile di entrata, "rimbalza" una o più volte tra i due stati prima di fermarsi definitivamente nello stato corretto.

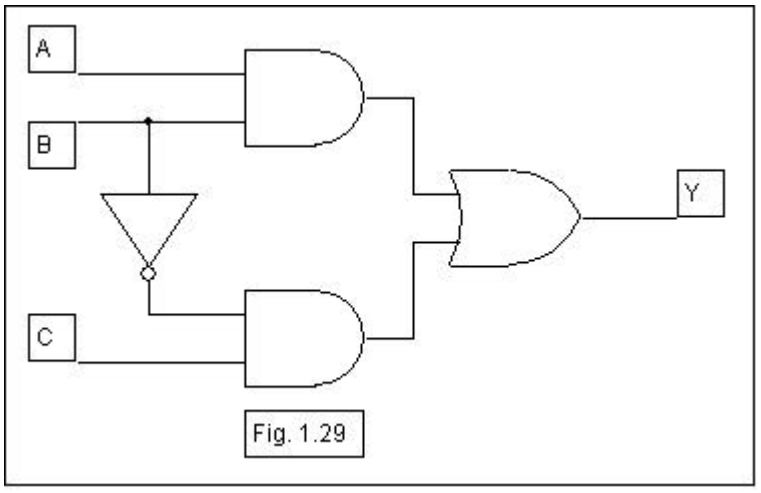


Fig. 1.29

Il fenomeno è illustrato in **Fig. 1.28**: come si osserva, in entrambi i casi la rete assume, sia pure temporaneamente, un valore logico errato.

Cominciamo a considerare l'alea statica.

Condizione necessaria affinché il cambiamento di stato di una varia-

bile di entrata provochi alea statica è che tale cambiamento influenzi la stessa uscita attraverso due percorsi diversi differenti per una inversione.

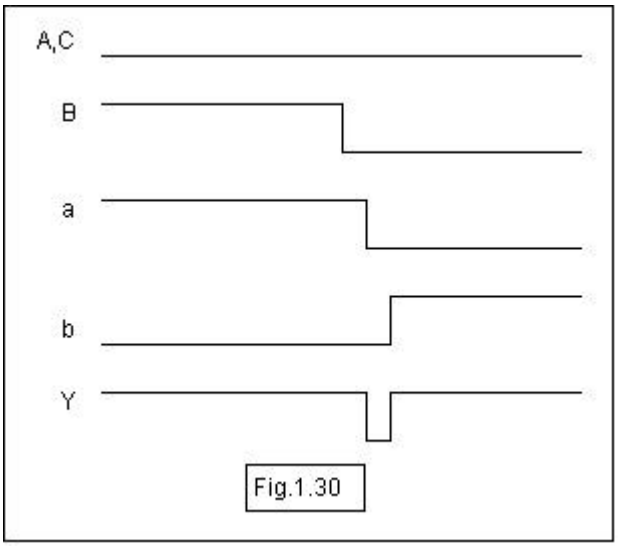


Fig.1.30

In **Fig. 1.29** è disegnata la rete relativa alla funzione

$$Y = AB + \overline{B}C$$

mentre in **Fig. 1.30** è disegnato il diagramma temporale relativo alla transizione 1->0 di B (mentre A=C=1), tenendo conto del ritardo di propagazione attraverso le porte, assumendolo eguale a δt per ciascuna porta (per semplicità di disegno, si considera 0 il ritardo dell'OR). Come si può osservare, a causa della presenza dell'inverter le due entrate all'OR non cambiano stato contemporaneamente, e questo fa sì che l'uscita, che dovrebbe restare a 1 anche dopo la transizione di B, assuma transitoriamente il valore 0, errato. Si può anche verificare che la transizione 0-1 di B non dà problemi.

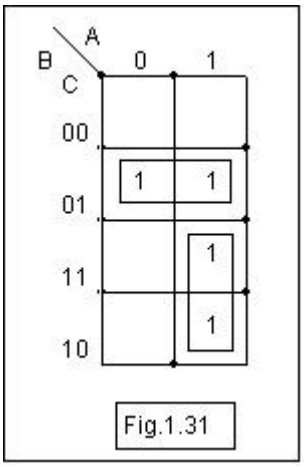


Fig.1.31

La condizione su esposta non è tuttavia sufficiente, come si può facilmente comprendere: infatti il ritardo di propagazione non è costante per tutte le porte, anche nell'ambito della stessa famiglia logica (a causa delle tolleranze di costruzione), ed è fortemente variabile da famiglia a famiglia. Pertanto, può accadere che la distribuzione dei ritardi lungo i due percorsi sia, casualmente, tale da neutralizzare l'alea.

La possibilità che si instauri un'alea statica nella rete in esame è prevedibile già osservando l'espressione algebrica della funzione: questa richiede che Y resti nello stato 1 quando, per effetto del cambiamento di stato di B, entrambi i termini prodotto devono cambiare valore logico. A causa della inevitabile presenza di un inverter, è presumibile che il cambiamento non avvenga contemporaneamente.

La possibilità che si presenti alea statica si traduce nella seguente topologia della mappa di Karnaugh della funzione, **Fig. 1.31**: il cambiamento di stato di B che genera l'alea statica corrisponde ad un salto fra due caselle adiacenti, appartenenti a due raggruppamenti diversi.

È facile convincersi che non possono esserci alee per transizioni all'interno dello stesso raggruppamento: queste transizioni, infatti, non comportano alcun cambiamento di valore logico del termine generato da quel raggruppamento.

Ci si può anche convincere, osservando l'esempio precedente, che una rete sintetizzata come somma di prodotti non può presentare alea se l'uscita è 0 e, dopo il cambiamento di una variabile di entrata (diciamo B) essa deve restare nello stato 0: questo perché ogni termine prodotto che contiene B non può essere eguale a 0 a causa di B, ma deve necessariamente includere qualche altra variabile (in forma vera o in forma complementata) che è 0: poiché per ipotesi solo B cambia stato, l'uscita resta a 0.

Naturalmente, può esserci alea anche se la funzione viene sintetizzata come prodotto di somme: in questo caso, si potrà avere alea statica se la funzione deve restare a 0 quando, per il cambiamento di stato di una variabile, i due termini somma devono cambiare stato contemporaneamente. La mappa di Karnaugh può ancora essere usata per evidenziare l'alea.

L'alea statica può essere eliminata aggiungendo alla funzione un termine che, senza alterare la tavola della verità, "costringa" la rete a restare nello stato corretto quando la variabile incriminata cambia stato. Nell'esempio considerato, tale termine è AC: nella mappa di Karnaugh esso serve a congiungere i due raggruppamenti iniziali, facendo ora sì che la transizione di B avvenga nell'ambito di uno stesso raggruppamento, appositamente creato. È possibile dimostrare che se l'alea viene eliminata considerando la forma S.d.P. della funzione, anche la forma P.d.S. della stessa funzione non presenta alea.

È, comunque, evidente che l'eliminazione dell'alea statica comporta l'introduzione di un termine

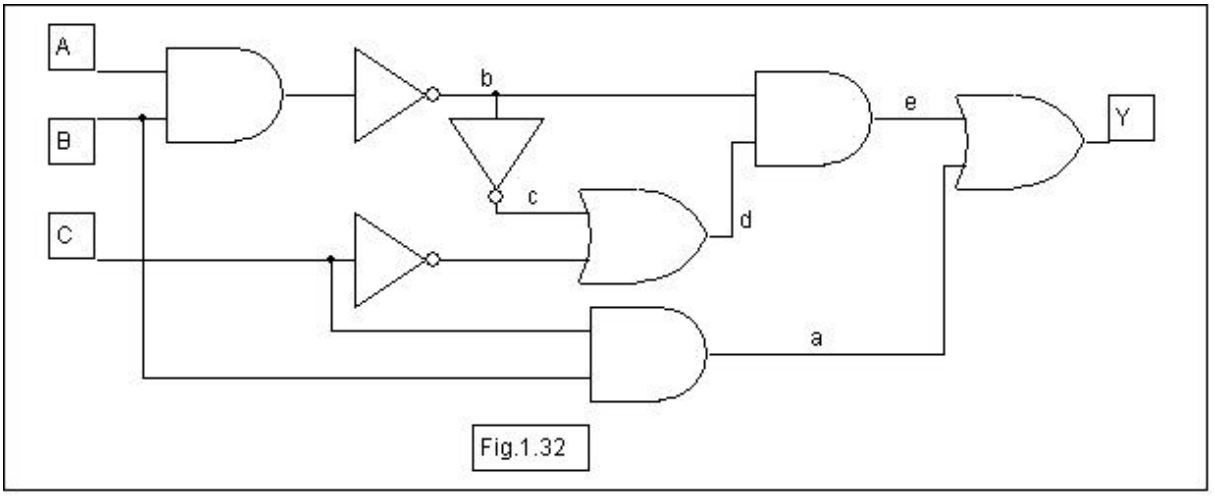


Fig.1.32

ridondante, cioè ha come contropartita la rinuncia alla forma minima della funzione.

Consideriamo ora l'alea dinamica.

Condizione necessaria affinché il cambiamento di stato di una variabile di entrata provochi alea dinamica è che tale cambiamento influenzi la stessa uscita attraverso almeno tre percorsi diversi, differenti per una inversione.

Questa situazione si può presentare, tipicamente, quando nella espressione algebrica della funzione compaiono termini raccolti a fattore comune, come accade quando si vuole diminuire il costo di una rete applicando la proprietà associativa.

Si consideri (come esempio costruito appositamente) la rete di **Fig. 1.32**, relativa alla funzione

$$Y = BC + \overline{AB}\,(AB + \overline{C})$$

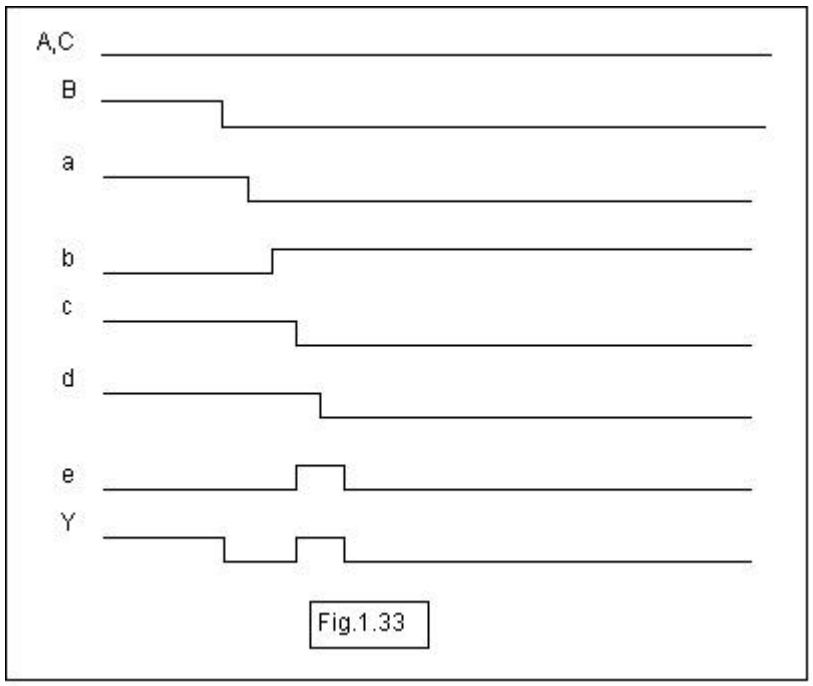


Fig.1.33

La variabile B soddisfa la condizione necessaria per il verificarsi dell'alea dinamica. In **Fig. 1.33** è considerata la transizione 1->0 di B, mentre A=C=1, nell'ipotesi che tutte le porte abbiano lo stesso ritardo (si considera 0 il ritardo dell'OR di uscita).

Naturalmente, anche per l'alea dinamica non è possibile dare condizioni sufficienti per il suo verificarsi: i ritardi reali delle porte usate possono essere tali da neutralizzarla.

L'eliminazione dell'alea dinamica avviene modificando l'espressione algebrica, rifattorizzandola o, meglio, eliminando la fattorizzazione (con conseguente aumento del costo).

Conviene infine osservare che il problema dell'alea nelle reti combinatorie può essere irrilevante in alcune applicazioni, data la transitorietà del fenomeno: è il caso, per esempio, se l'uscita della rete è utilizzata per pilotare un display a LED. Se invece la stessa rete dovesse pilotare un contatore di impulsi, l'alea darebbe luogo a un conteggio errato: in questo caso è imperativo eliminarla.

1.16 - LOGICA CABLATA

La struttura interna di una porta logica, vista dall'uscita, può essere schematizzata come in **fig. 1.34**

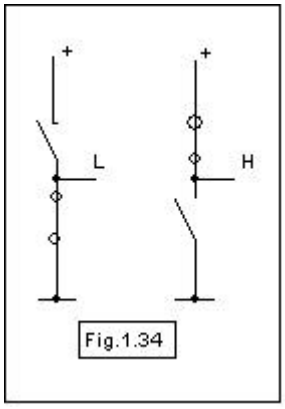


Fig.1.34

(i due switches sono nella realtà due transistor, **Fig. 1.35a**: lo switch chiuso corrisponde ad un transistor saturato, quello aperto ad un transistor interdetto). È evidente che non è possibile collegare insieme le uscite di più porte, senza che questo comporti inconvenienti (per esempio, eccessiva dissipazione di potenza, o transienti sulle linee di alimentazione).

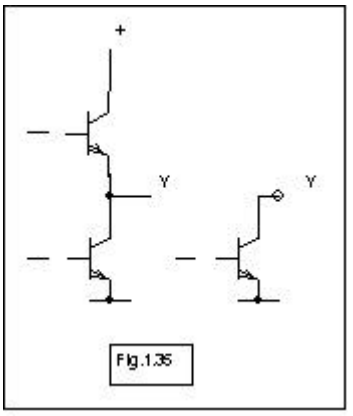


Fig.1.35

Esistono particolari versioni di porte logiche che hanno una struttura di uscita diversa. Le porte 'open-collector' hanno la struttura di uscita schematizzata in **fig. 1.35b**: in questo modo, è possibile collegare insieme le uscite di più porte Y_1, Y_2,... e aggiungere dall'esterno una resistenza R tra l'uscita comune Y e l'alimentazione (resistenza di pull-up), come mostrato in **Fig. 1.36**. Si può vedere che sull'uscita comune Y è realizzato l'AND per logica positiva tra Y_1, Y_2,...:

$$Y = Y_1 \cdot Y_2 \cdot \ldots$$

Infatti, se l'uscita della porta Y_i è nello stato basso (che è lo 0 in logica positiva, interruttore chiuso), il terminale di uscita comune Y è cortocircuitato a massa. Pertanto, basta che una delle uscite Y_i sia nello stato basso perché il terminale comune Y sia nello stato basso: e questa è proprio la funzione AND per logica positiva (o, se si preferisce, la funzione OR per logica negativa). Pertanto, viene realizzata la funzione logica AND senza materialmente usare una porta, ma con un cablaggio. Si parla di AND "cablato" (wired-AND), oppure di OR cablato (wired-OR).

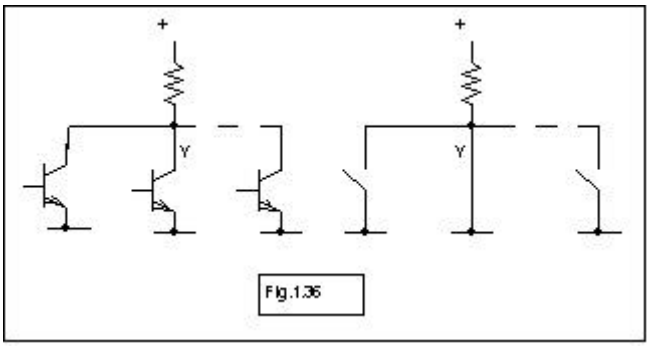
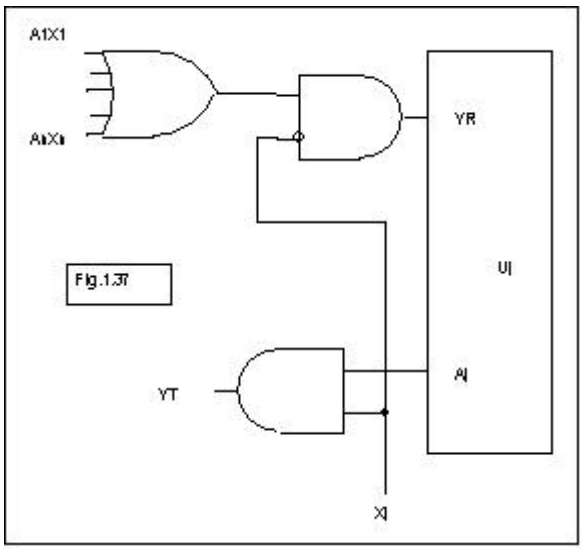


Fig.1.36

La logica cablata è normalmente usata nella architettura dei calcolatori, perché permette alle varie periferiche di comunicare fra loro e con l'unità centrale trasmettendo o ricevendo dati su un unico insieme di linee ('bus'), con risparmio di hardware.

Per un corretto scambio dell'informazione, occorre che:

- le porte in wired-or siano abilitate a trasmettere una alla volta sul bus
- ci sia un *'arbitro'* che gestisca il controllo del sistema, decidendo di volta in volta quale porta abilitare alla trasmissione
- il valore logico dell'uscita comune Y sia considerato valido solo in sincronismo con il segnale di abilitazione.



Per esempio, supponiamo che n unità u_1, u_2, ….. u_j, … u_n di un sistema di elaborazione (tipicamente, un calcolatore) debbano scambiare fra loro informazioni. Per semplicità, supponiamo che l'informazione sia costituita da parole di 1 bit. Usando una logica convenzionale, ciascuna unità dovrebbe avere una struttura di ingresso-uscita del tipo mostrato in **Fig. 1.37**. X_j è il bit di abilitazione che controlla se l'unità deve ricevere dati ($X_j = 0$) oppure li deve trasmettere ($X_j = 1$).

La funzione implementata per la trasmissione è

$$YT = A_j X_j$$

essendo A_j il bit da trasmettere.

L'unità Uj può ricevere informazione da una delle altre n-1 unità, in accordo con la funzione

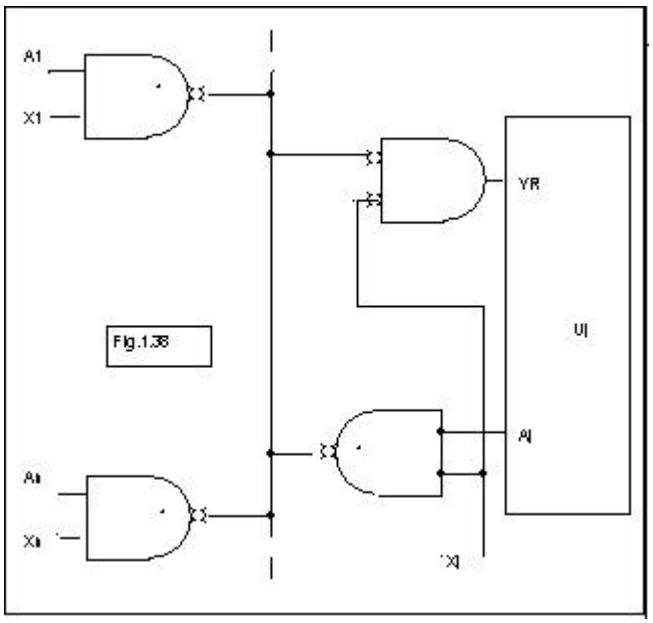
$$YR = (A_1X_1+A_2X_2+....+A_{j-1}X_{j-1} + A_{j+1}X_{j+1} +.....+ A_nX_n)\,\overline{X_j}$$

Se si riscrive questa espressione col teorema di De Morgan, si ha

$$= \overline{\overline{A_1X_1}\;\overline{A_2X_2}.....\overline{A_nX_n}}\;\overline{X_j}$$

Se si utilizzano NAND con uscita open-collector per realizzare i prodotti $\overline{A_jX_j}$, lo schema di comunicazione per il bit può allora essere modificato come in **Fig. 1.38**: la porta OR di Fig. 1.37 è stata sostituita da un filo di collegamento (per semplicità, non è mostrata la resistenza di pullup).

La **Fig. 1.39** mostra un esempio per tre unità. L'unità A può trasmettere un bit sulla linea di bus mostrata, quando il *'bus arbitrator'* abilita la porta A_1 ad accedere al bus fornendo il segnale X_A -> H, e ponendo nel contempo X_B e X_C -> L: in coincidenza con l'abilitazione di A a trasmettere, le unità B e C sono abilitate a ricevere il bit, qualora siano interessate. Le porte A_1, B_1, C_1 devono essere del tipo open-collector.

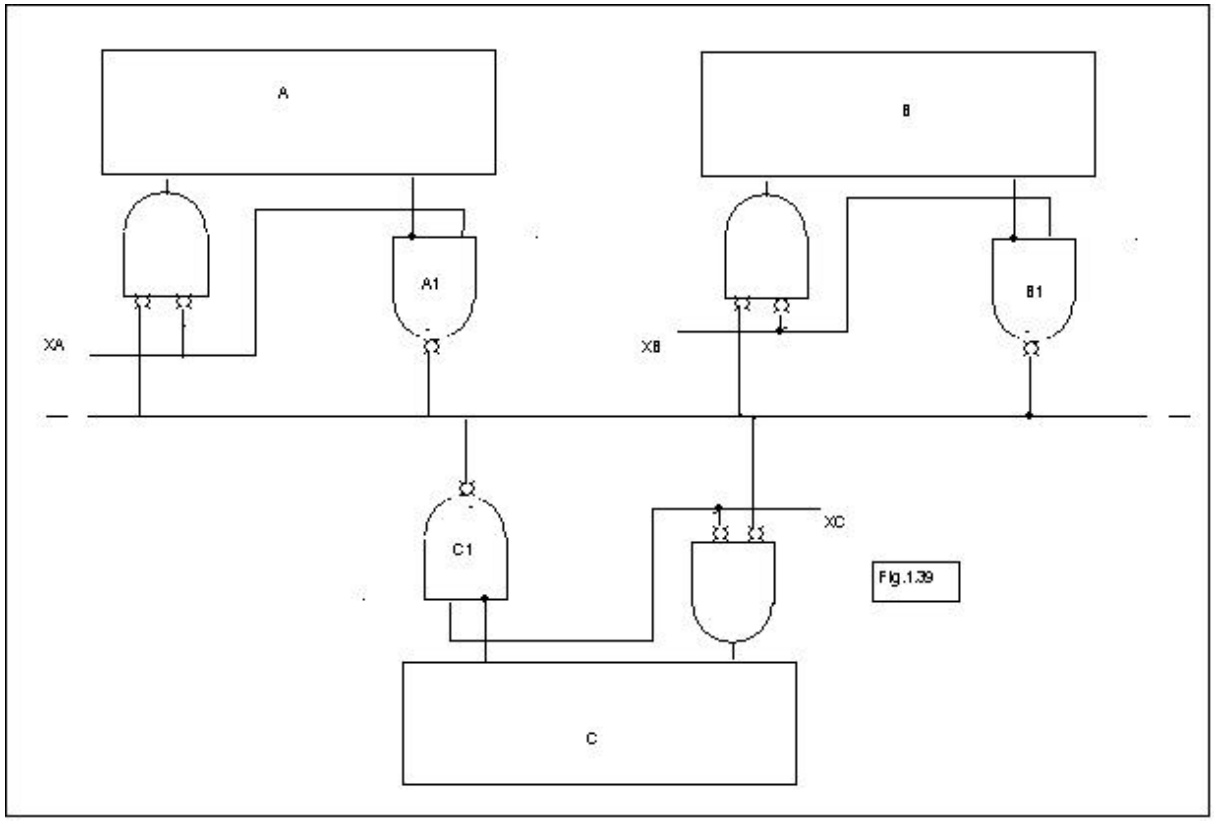


Se l'informazione è costituita da più bit, la struttura precedente va replicata per ogni bit.

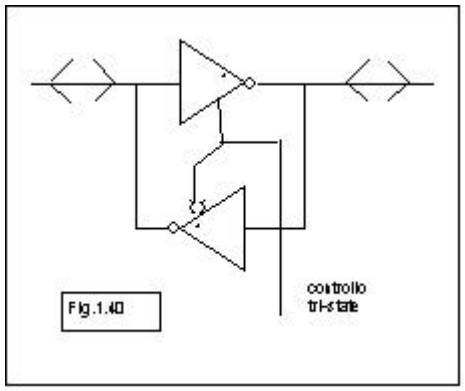
Si noti la semplicità dell'hardware di interfaccia ingresso-uscita, che tra l'altro non dipende dal numero delle unità che accedono al bus, data l'assenza dell'OR, e può quindi essere standardizzato.

La struttura open-collector ha un inconveniente: la velocità delle commutazioni L->H e H->L sulle linee di bus è determinata dalle costanti di tempo di carica e di scarica della capacità C presente tra la linea di bus e massa (il cui valore è proporzionale al numero di porte connesse alla linea). La costante di tempo di scarica (transizione H->L) è determinata dalla resistenza di saturazione di un transistor (switch chiuso) che è di qualche decina di ohm ed è pertanto breve. La costante di tempo di carica (transizione L->H) è invece molto più lunga, determinata dalla resistenza R di pull-up esterna che è tipicamente dell'ordine del kOhm.

Il problema è stato risolto con l'introduzione delle porte a tre stati logici, o 'tri-state'. Queste porte hanno, oltre agli usuali terminali di entrata e di uscita, un terminale di controllo che agisce come segue. Supponendo, per es., che il controllo tri-state sia attivo alto, quando esso è L la porta esegue la propria funzione logica in accordo con la tavola della verità, e lo stato di uscita è realizzato secondo lo schema degli switches illustrato in **Fig. 1.34**. Quando invece il controllo tri-state è H, l'uscita della porta va nel 'terzo stato logico', corrispondente in fig. 1.34 a *entrambi gli switches aperti*: il terminale di uscita della porta è ora flottante ed il terzo stato logico viene perciò chiamato stato alta impedenza (Z).



Il collegamento wired-OR di porte tri-state avviene senza necessità di resistenza di pull-up, e funziona come segue. Le porte collegate siano tutte inizialmente nello stato Z; quando il controllo tri-state abilita una (e una sola) delle porte, allora se l'uscita di tale porta è per es. L, anche l'uscita comune Y viene forzata nello stato L, la capacità del nodo si scarica rapidamente e non c'è nessun problema di cortocircuito dell'alimentazione. Se invece l'uscita della porta va H, anche Y andrà H; la capacità viene altrettanto rapidamente caricata, a differenza delle porte open-collector.



La tecnologia tri-state permette di semplificare ulteriormente l'accesso al bus, come mostrato in **Fig. 1.40**: per la proprietà della porta tri-state, le due linee di comunicazione di ciascuna unità con il bus, Fig. 1.39, possono ora essere sostituite da una sola linea bidirezionale (ciascuna unità del sistema, ad un certo istante, o trasmette o riceve).

La coppia di porte di accesso ad una linea di bus mostrata in Fig. 1.40 è disponibile su singolo integrato (bus transceiver, per es. 75160).

BIBLIOGRAFIA

[1] Edwards, The principles of switching circuits. MIT Press, 1973

[2] I. Mendolia, U. Torelli, Elettronica digitale e dispositivi logici. Hoepli

[3] E. McCluskey, Logic design principles. Prentice-Hall

[4] H. Taub, D. Shilling, Digital integrated electronics. McGraw-Hill