Parte II :
Analisi e sintesi dei circuiti combinatori

2.1 Introduzione

Una rete combinatoria é un circuito elettronico digitale in grado di calcolare in modo automatico una funzione binaria di una o più variabili booleane.
Lo schema generale di una rete combinatoria é il seguente:

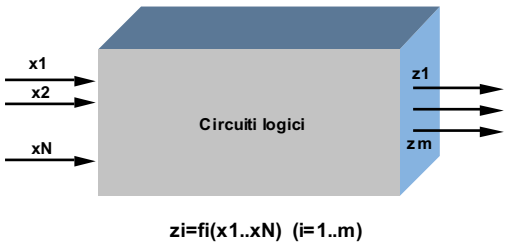


Figura 2.1 *Generico schema di rete combinatoria*

I valori delle uscite in ogni istante sono univocamente determinati dai valori presenti contemporaneamente sugli ingressi.

Una rete combinatoria si compone di elementi di calcolo logico elementare detti **porte**. Questi elementi verranno descritti nel paragrafo successivo.
Una **funzione booleana FB** una descrizione estensiva del funzionamento di una rete combinatoria, data dall'insieme delle coppie ordinate
$((a_1..a_N), b_i)$ i=1...m con ai,bi ∈ **B**: (0,1)
rappresentate tramite tabelle (per ognuno dei 2^N possibili input binari esprime il corrispondente output di f).
$f : \{ 0 , 1 \}^N \longrightarrow \{ 0 , 1 \}$

Graficamente:

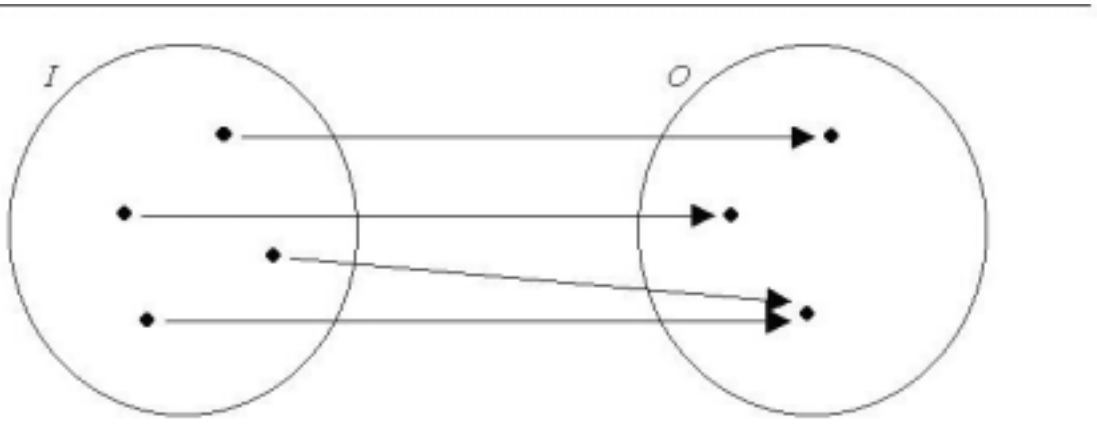


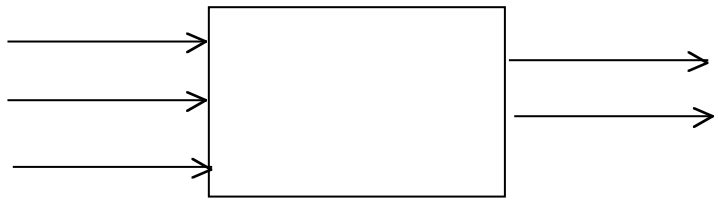
Figura 3.1 Trasformazione di un insieme di numeri binari.

La *funzione booleana* di un circuito digitale pu essere rappresentato mediante una **tabella di verit** o **true table** (**TT**) che determina completamente il legame funzionale fra ingressi e uscite.

Dati n ingressi, una TT elenca ordinatamente nella parte sinsitra tutte le 2^n combinazioni di valori delle variabili booleane di ingresso. Nella parte destra, nella cella di posizione (i,j) si indica il valore assunto dalla variabile booleana di uscita z_j in corrispondenza alla sequenza di valori assunti dalle variabili di ingresso nella riga i della tabella.

Ad esempio, una descrizione completa del comportamento di una rete combinatoria a 3 ingressi e due uscite data da:

x3 x2 x1	z2z1
0 0 0	0 0
0 0 1	0 1
0 1 0	0 0
0 1 1	0 1
1 0 0	0 0
1 0 1	1 1
1 1 0	0 1
1 1 1	1 1



Il comportamento di un circuito combinatorio pu essere equivalentemente descritto da un espressione booleana. Come gi detto nel capitolo 1, una EB descritta come segue:

Dato un insieme numerabile di variabili *Var* a valori binari, le espressioni booleane EB su di esso sono definite induttivamente da :

- $0, 1 \in$ EB
- $\forall\, v \in Var \quad v \in$ EB
- se $E_1, E_2 \in$ EB allora $E_1 + E_2$, $E_1 \cdot E_2$, $E_1 \in \overline{\text{EB}}$

dove $\overline{\text{E}}$ è ottenuta applicando opportunemente le leggi di De Morgan e ricordando che:

- $\overline{0} = 1$ e $\overline{1} = 0$
- $\overline{\overline{v}} = v$

Teor.: *- per ogni espressione booleana esiste un'unica funzione booleana equivalente*
- per ogni funzione booleana esistono infinite espressioni booleane equivalenti

Infine, una rete pu essere descritta disegnando lo **schema circuitale (SC)**. Uno schema circuitale un collegamento di circuiti elementari detti **porte** tramite linee.
Identifichiamo tre tipi di linee:

- **linee di ingresso**, etichettate con i nomi delle variabili booleane di ingresso
- **linee di uscita**, etichettate con i nomi delle variabili di uscita
- **linee interne**, ciascuna delle quali collega l'uscita di una porta con l'ingresso di un'altra porta

Lo schema circuitale di una rete combinatoria deve soddisfare i seguenti requisiti:

¥ Ogni ingresso di ogni porta presente nella rete deve essere collegato ad una linea di ingresso oppure ad una linea interna.

- L'uscita di ogni porta presente deve essere collegata ad una linea di uscita oppure ad una linea interna.
- Il collegamento di porte tramite linee non deve dare luogo a cicli.

Un esempio mostrato in figura:

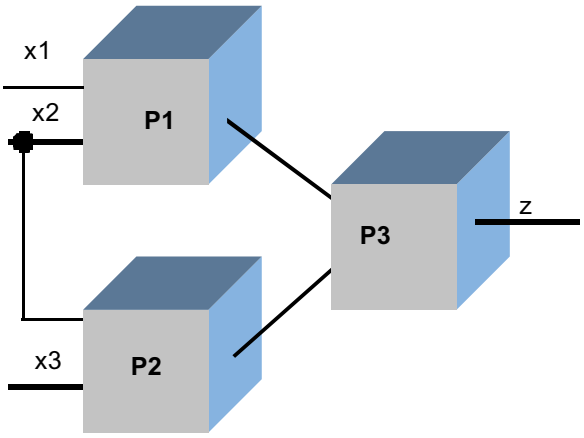


Figura 2.2 *Generico schema circuitale*

In figura 2, le porte logiche sono rappresentate da generici quadrati. Nel paragrafo successivo analizziamo i vari tipi di porte logiche.

Riassumendo, una rete combinatoria pu essere rappresentata mediante una funzione booleana FB, mediante infinite espressioni booleane EB equivalenti, mediante infiniti schemi circuitali equivalenti.

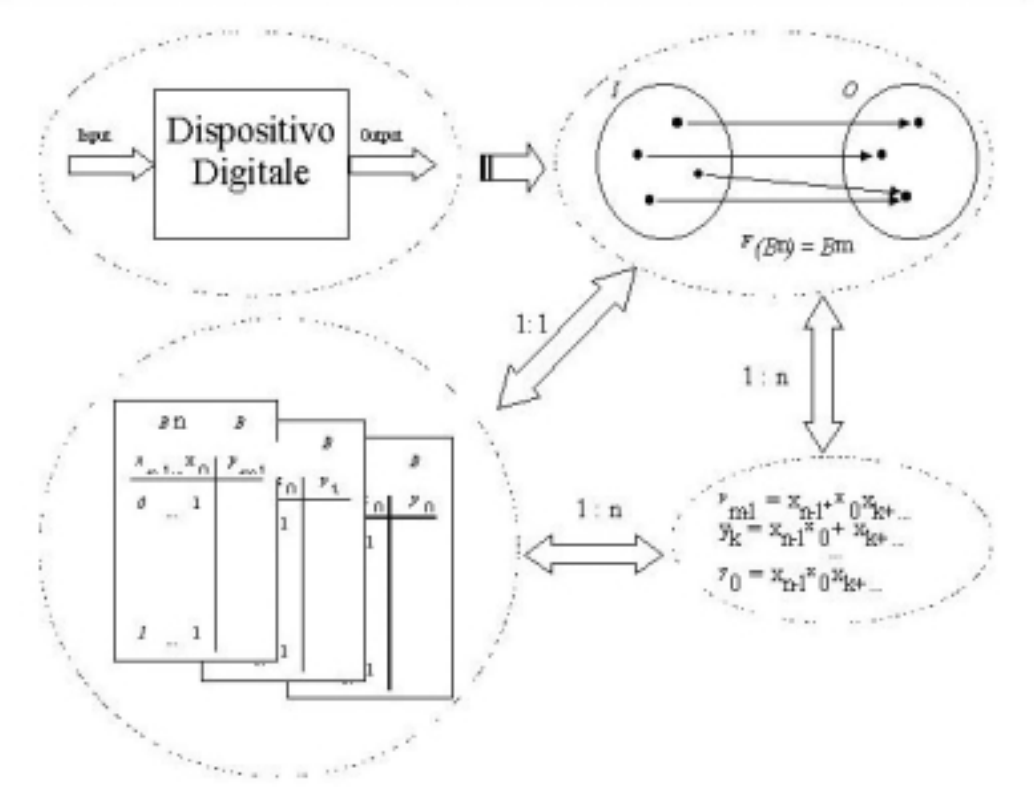


Figura 3.4 Relazione tra elaborazione di numeri binari, funzioni, tabelle delle verità ed espressioni.

Nel seguito, vedremo come passare dall'una all'altra forma di rappresentazione di una rete combinatoria.

2.2 Porte logiche elementari

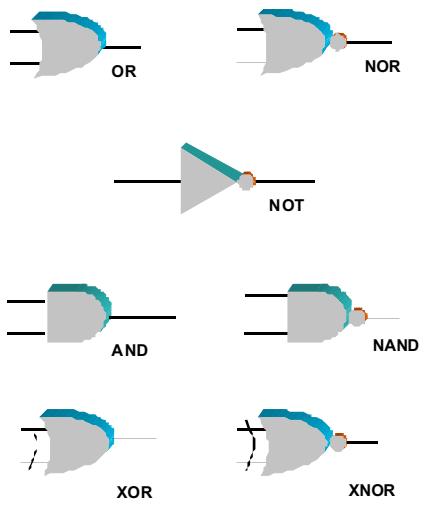


Figura 2.3 *Porte logiche elementari*

In figura 3 sono riportati i simboli grafici usati per rappresentare i 7 tipi di porte elementari. Di seguito, riportiamo la EB (con varie simbologie) e la TT per ciascuna delle 7 porte. Si noti che il simbolo XOR (un + circondato da un cerchio) usato da alcuni come simbolo dell'OR. In questi appunti tuttavia lo useremo per indicare XOR.

OR

X1 X0	Y
0 0	0
0 1	1
1 0	1
1 1	1

Y= X1 **or** X0
Y= X1 + X0 , Y= X1 ∨ X0

NOR

X1 X0	Y
0 0	1
0 1	0
1 0	0
1 1	0

Y= X1 **nor** X0
Y= $\overline{X1 + X2}$
Y =¬ (XI+X0)

NOT (INVERTER)

X0	Y
0	1
1	0

Y= **not**X0 ; Y=$\overline{X0}$; Y= ¬ X

AND

X1 X0	Y
0 0	0
0 1	0
1 0	0
1 1	1

Y=X1 **and** X0
Y=X1 ¥ X0
Y=X1X0, Y=X1∧X0

NAND

X1 X0	Y
0 0	1
0 1	1
1 0	1
1 1	0

Y=X1 **nand** X0

$Y = \overline{X1X0}$

Y= ¬ (X1X0)

X-OR

X1 X0	Y
0 0	0
0 1	1
1 0	1
1 1	0

Y=X1**xor**X0 ; Y=X1⊕X0

X-NOR

X1 X0	Y
0 0	1
0 1	0
1 0	0
1 1	1

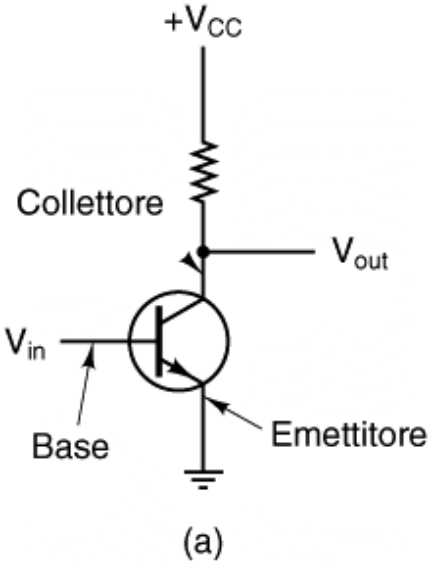
Y= X1**xnor**X0

$Y = \overline{X1 \quad X0}$

Y=¬ (X1⊕X0)

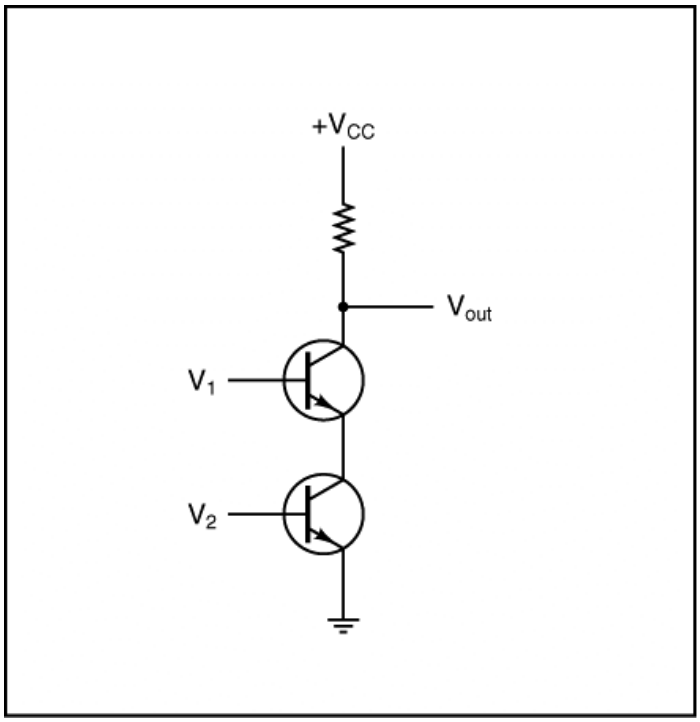
L'analisi a livello fisico dei circuiti digitali non è materia di questo corso, ma piuttosto di corsi di elettronica. Tuttavia si mostra qui un esempio di implementazione fisica di porte NOT e NAND.

Porta NOT



(a)

Porta NAND



2.2.1 Porte logiche a più ingressi

La figura mostra lo schema di equivalenza per porte logiche a pi ingressi. L'estensione pu essere effettuata per mezzo di operatori logici dello stesso tipo messi opportunamente in cascata, solo se la funzione logica gode della propriet associativa. Questo vero per porte AND, OR , XOR e XNOR, ma non per porte logiche NAND e NOR. Un esempio mostrato in figura.

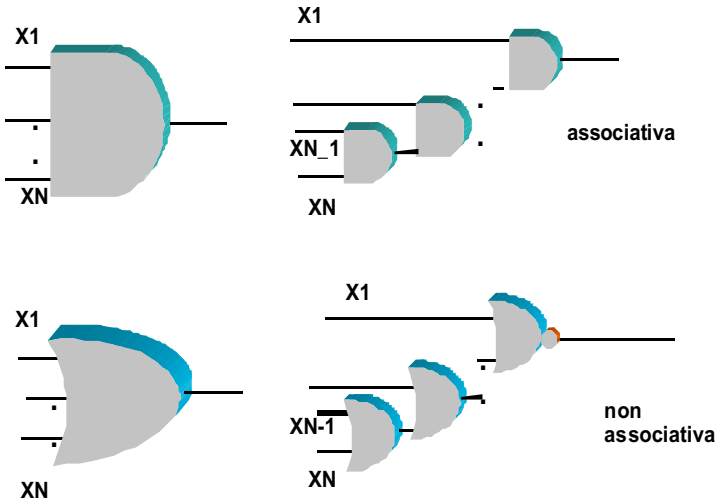
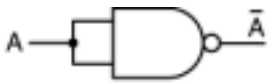
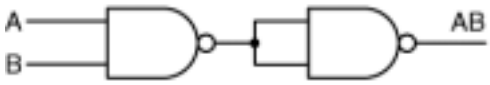


Figura 2.4 *Schema di equivalenza per porte logiche a pi ingressi*

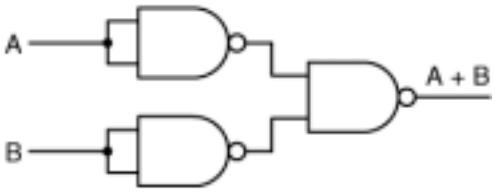
Universalità delle porte NAND



equivale a un NOT

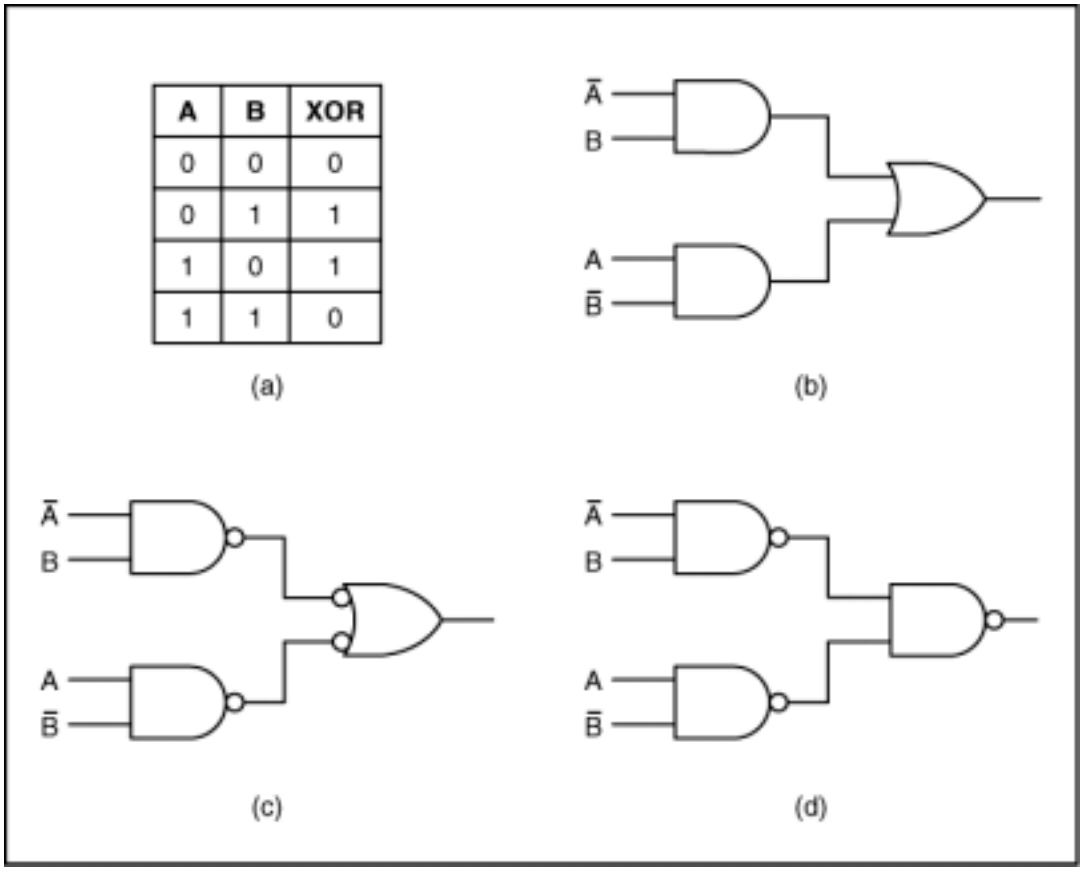


equivale a una porta AND



equivale a una porta OR

Realizzazione di un circuito con un solo tipo di porta, esempio:



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

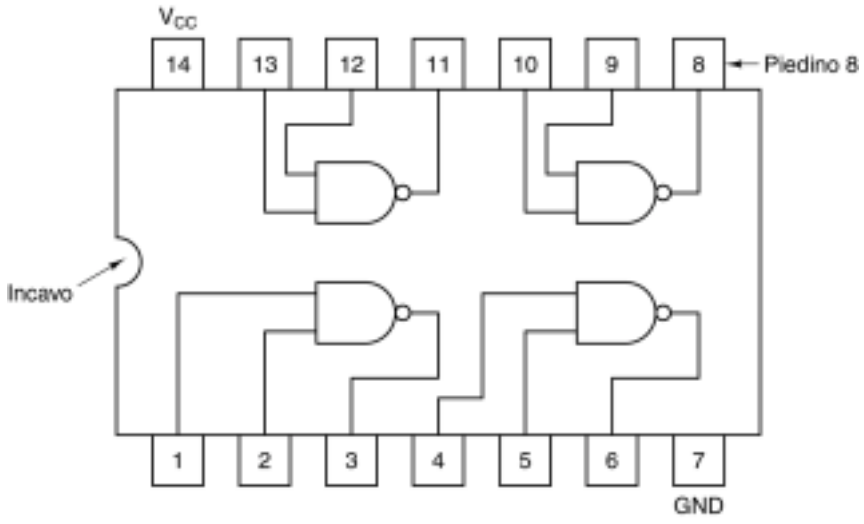
(a)

(b)

(c)

(d)

Uso: ottimizzare utilizzo integrati



2.3 Equivalenza fra le varie forme di rappresentazione del funzionamento di un circuito combinatorio

Nel paragrafo 1 abbiamo visto come una rete combinatoria possa essere rappresentata in forme simboliche diverse: la FB (ovvero la true table TT), la EB, e lo SC.
In questo paragrafo, vedremo come passare dall'una all'altra forma di rappresentazione.

2.3.1 Dalla TT alla EB: forme canoniche congiuntive e disgiuntive.

Consideriamo un circuito combinatorio con n variabili di ingresso, $X_1..X_n$. Denotiamo ogni occorrenza di una singola variabile, sia in forma semplice X_i che complementata $\overline{X_i}$, col nome di *letterale*.
Ad ogni riga della tabella di verit , costituita da una sequenza di n valori booleani, $b_{n-1}..b_0$, associamo una sequenza di letterali nel seguente modo:
se b_i=0, facciamo corrispondere a bi il letterale $\overline{X_i}$,
se b_i=1 facciamo corrispondere a bi il letterale X_i .
Il prodotto logio (**and**) degli n letterali cos ricavato si chiama *mintermine*.

Ad esempio, alla stringa 001 corrisponde il mintermine $\bar{x}_2\bar{x}_1x_0$

Viceversa, se ad ogni riga della tabella di verit associamo una sequenza di letterali nel seguente modo:
se b_i=0, facciamo corrispondere a bi il letterale X_i ,
se b_i=1 facciamo corrispondere a bi il letterale $\overline{X_i}$,
la somma (or) degli n letterali cos ottenuti prende il nome di *maxtermine*.

Ad esempio, alla stringa 001 corrisponde il maxtermine $x_2 + x_1 + \bar{x}_0$

La somma logica (or) dei mintermini corrispondenti alle righe della TT in cui la variabile booleana di uscita 1 prende il nome di **forma canonica disgiuntiva FCD** della funzione booleana del circuito.
Il prodotto (and) dei maxtermini corrispondenti alle righe della TT in cui la variabile booleana di uscita 0 prende il nome di **forma canonica congiuntiva FCC** della funzione boolana del circuito.

Esempio:

Data la tabella di verit :

X2X1X0	Y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	1

ricaviamo i mintermini per le righe 001, 101 e 111.

La forma canonica disgiuntiva data da:

$Y=\overline{X_2}\,\overline{X_1}X_0 + X_2\overline{X_1}X_0 + X_2X_1X_0$

La *forma normale disgiuntiva* **FND** o **SOP** (Sum of Products) una generalizzazione della precedente. In una SOP i prodotti non sono necessariamente mintermini.
Applicando le regole dell'algebra booleana alla EB dell'esempio precedente, otteniamo
$Y = (\overline{X_2} + X_2)\overline{X_1}X_0 + X_2X_1X_0 = \overline{X_1}X_0 + X_2X_1X_0$

che appunto, una forma FND poich il prodotto $\overline{X_1}X_0$ non un mintermine.
Studieremo in fase di sintesi come ottenere la FND ottima di un circuito.

I maxtermini per la TT precedente vanno ricavati per le righe 000 010 011 100 110.
La forma canonica congiuntiva data da:

$Y = (X_2 + X_1 + X_0)(X_2 + \overline{X_1} + X_0)(X_2 + \overline{X_1} + \overline{X_0})(\overline{X_2} + X_1 + X_0)(\overline{X_2} + \overline{X_1} + X_0)$

La forma canonica congiuntiva si pu analogamente ottenere ricavando la forma canonica disgiuntiva della funzione complemento di quella data, e quindi complementando il risultato.
Ad esempio, la forma canonica disgiuntiva della funzione complemento di quella la cui TT mostrata sopra data dalla:

$\overline{Y} = \overline{X_2}\,\overline{X_1}\,\overline{X_0} + \overline{X_2}X_1\overline{X_0} + \overline{X_2}X_1X_0 + X_2\overline{X_1}\,\overline{X_0} + X_2X_1\overline{X_0}$

complementando ancora si ottiene (applicando DeMorgan):

$\overline{\overline{Y}} = \overline{\overline{X_2}\,\overline{X_1}\,\overline{X_0} + \overline{X_2}X_1\overline{X_0} + \overline{X_2}X_1X_0 + X_2\overline{X_1}\,\overline{X_0} + X_2X_1\overline{X_0}} =$

$(\overline{\overline{X_2}\,\overline{X_1}\,\overline{X_0}})(\overline{\overline{X_2}X_1\overline{X_0}})(\overline{\overline{X_2}X_1X_0})(\overline{X_2\overline{X_1}\,\overline{X_0}})(\overline{X_2X_1\overline{X_0}}) =$

$(X_2 + X_1 + X_0)(X_2 + \overline{X_1} + X_0)(X_2 + \overline{X_1} + \overline{X_0})(\overline{X_2} + X_1 + X_0)(\overline{X_2} + \overline{X_1} + X_0) = Y$

Le due forme sono quindi equivalenti.
La forma normale congiuntiva FNC o POS (Product of Sum) una generalizzazione della precedente. In una FNC i prodotti non sono necessariamente maxtermini.

2.3.2 Passare dalla EB alla FB o TT

Per ottenere la TT dalla EB di una rete combinatoria occorre trasformare la EB in modo da ottenere una forma canonica congiuntiva o disgiuntiva, oppure una sua generalizzazione FCC o FCD.
Ad esempio, data l'espressione:
$Y = X_1(\overline{X_0} + X_2X_0)$
sviluppando i prodotti e semplificando, si ottiene una forma FCD:
$Y = X_1\overline{X_0} + X_2X_1X_0$
Ad ogni termine T_i della FCD, associare una stringa binaria $R_i = b_{n-1}..b_0$ nel seguente modo:
se il letterale $\overline{X_i}$ porre $b_i=0$,
se il letterale X_i porre $b_i=1$,
se X_i non compare in T_i, porre $b_i=X$, dove col simbolo X (o d, o -) indichiamo la condizione di indifferenza "*dont'car*e" ossia indifferentemente uno 0 o un 1.

Nell'esempio sopra, otteniamo le seguenti stringhe: R_1= X10 ed R_2=111.
A questo punto costruiamo la parte sinistra della tabella di verit , TT, e nella parte destra, poniamo un 1 in corrispondenza a tutte le righe uguali o implicate dalle stringhe R_i. Nell'esempio, la stringa R_1 contiene le stringhe 110 e 010, dato che X_2 pu essere sia 0 che 1 (dont'care). La TT della rete combinatoria considerata sar pertanto:

X2X1X0	Y
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	0
1 0 0	0
1 0 1	0
1 1 0	1
1 1 1	1

Analogamente, da una EB in forma canonica congiuntiva o FNC, si pu ottenere una TT nel seguente modo:
ad ogni prodotto P_i della POS, associare una stringa binaria R_i= $b_{n-1}..b_0$ nel seguente modo:
se il letterale $\overline{X_i}$ porre b_i=1,
se il letterale X_i porre b_i =0,
se X_i non compare in T_i, porre b_i=X, dove col simbolo X indichiamo "dont'care" ossia indifferentemente uno 0 o un 1.
Nella tabella di verit TT poniamo uno 0 in corrispondenza a tutte le righe uguali o implicate dalle stringhe R_i.
In un certo senso, intuitivo comprendere come una forma disgiuntiva elenchi le combinazioni di valori delle variabili booleane che portano ad 1 la variabile di uscita, mentre la forma congiuntiva elenchi le combinazioni che portano a 0 il valore della variabile di uscita.

Esempio : derivare la FCD della funzione che dà 1 *sse* riceve un numero pari di 1 in input (si consideri una funzione booleana triargomentale)

La forma tabellare di f è :

x_1	x_2	x_3	f
0	0	0	1
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

da cui FCD è: $\bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$

2.3.4 Dalla EB allo SC

Per ricavare lo schema circuitale SC di una rete combinatoria da una EB, conviene ancora partire dalla forma canonica congiuntiva o disgiuntiva, oppure una sua generalizzazione FNC o

FND. Questo tuttavia non strettamente necessario, se non si desidera ottenere un circuito minimizzato, problema del quale ci occuperemo in fase di studio dei problemi di sintesi.
Assegnata dunque una EB, costruiamo una rappresentazione gerarchica degli operatori booleani, partendo dai pi esterni.
Ad esempio, data la EB (non in forma canonica): $Y = X_2X_1(X_3 + \overline{X_0}X_2)$
possiamo riscrivere la EB evidenziando l annidamento degli operatori booleani come segue:
AND(AND(X_2,X_1), OR(X3,AND(NOT(X0),X2))
(notate che OP(x_1,x_2) , dove OP un operatore booleano, sta per x_1OPx_2)

Lo schema ad albero equivalente mostrato in figura 2.5 :

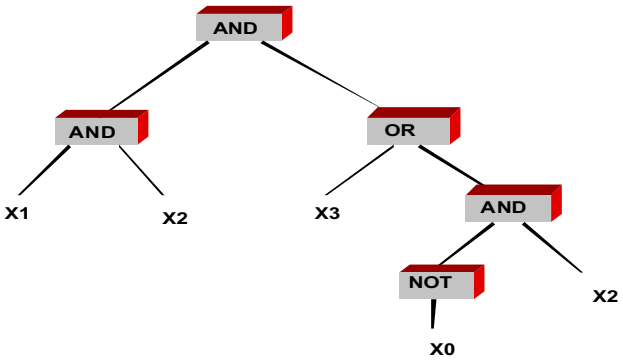


Figura 2.5 *Rappresentazione gerarchica di una EB*

A questo punto, il passaggio allo schema circuitale immediato:
- i terminali dell'albero sono le variabili booleane di ingresso
- i nodi vengono associati alle corrispondenti porte logiche elementari
- gli archi vengono associati alle linee interne della rete.

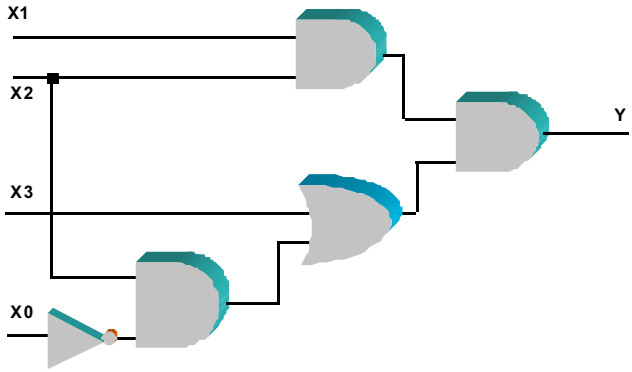


Figura 2.6 *Schema circuitale della EB*: $Y = X_2X_1(X_3 + \overline{X_0}X_2)$

2.4 Analisi di reti combinatorie

Come gi accennato nella Parte I, il problema di **analisi** consiste nell'esame della rappresentazione schematica dei comportamenti di un circuito, al fine di determinare le relazioni di causa/effetto fra segnali di ingresso e segnali di uscita. Dunque, l'obiettivo di ricavare, dallo schema circuitale, una rappresentazione in termini di EB o TT delle relazioni ingresso/uscita Poich sappiamo, dal paragrafo precedente, come ottenere una TT dalla EB, ci limiteremo ad introdurre un metodo per ottenere la EB dallo schema circuitale. Un possibile procedimento percorrere il cammino inverso rispetto al metodo precedentemente introdotto per ottenere un SC dalla EB.
- Dallo SC, per ogni linea di uscita, tracciamo il percorso inverso delle linee interne fino a raggiungere le linee di ingresso, ottenendo un albero gerarchico ai cui simboli circuitali

corrispondono nodi etichettati, alle linee di collegamento corrispondono gli archi, e le cui foglie, o simboli terminali, sono i segnali di ingresso raggiunti.

- Dall'albero passiamo alla sua forma compatta parentesizzata, e quindi costruiamo l'espressione booleana corrispondente, partendo dai simboli logici pi interni. Ad esempio:

AND(AND(X_2,X_1), OR(X3,AND(NOT(X0),X2))

AND(AND(X_2,X_1), OR(X3,AND($\overline{X_0}$,X2))

AND(AND(X_2,X_1), OR(X3,$\overline{X_0}X_2$)

AND(X_2X_1 , $X_3 + \overline{X_0}X_2$)

$X_2X_1(X_3 + \overline{X_0}X_2)$

Una modalit alternativa consiste nel suddividere lo SC in **stadi**. Uno stadio S_i una sezione verticale del circuito tale che ogni linea di ingresso di uno stadio attraversa al pi una porta logica. Le linee di ingresso dello stadio S_1 sono i segnali di ingresso della rete, mentre le linee di uscita dello stadio S_{n-1} sono i segnali di uscita della rete. Il valore di n pari al numero massimo di porte attraversabili quando si percorre il circuito dalle linee di ingresso fino all'uscita. Le linee di uscita del generico S_i coincidono con le linee di ingresso dello stadio successivo, S_{i+1}. Il metodo consiste, partendo dallo stadio S_1 e procedendo verso destra, nel calcolare le funzioni combinatorie delle uscite di ogni stadio S_i in funzione dei suoi ingressi. In figura 2.7 mostrato un esempio.

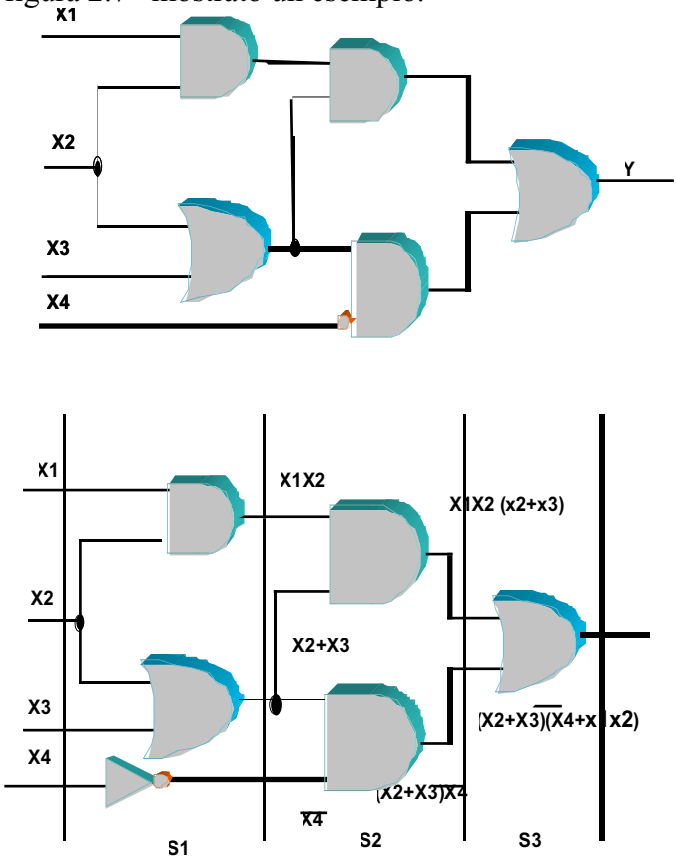


Figura 2.7 *Analisi di una rete logica con il metodo degli stadi*

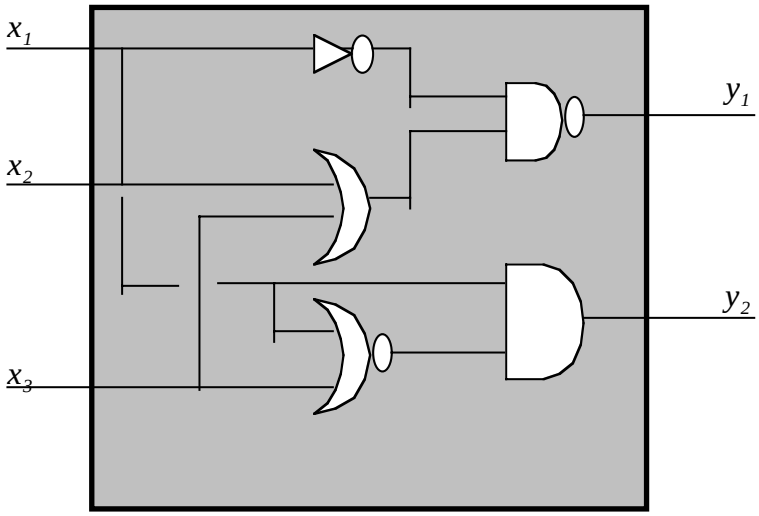
ESERCIZI

Teor.: *- per ogni rete logica ad m uscite esiste un'unica m-pla di espressioni booleane equivalenti*

- per ogni m-pla di espressioni booleane esiste un'unica rete logica ad m uscite equivalente

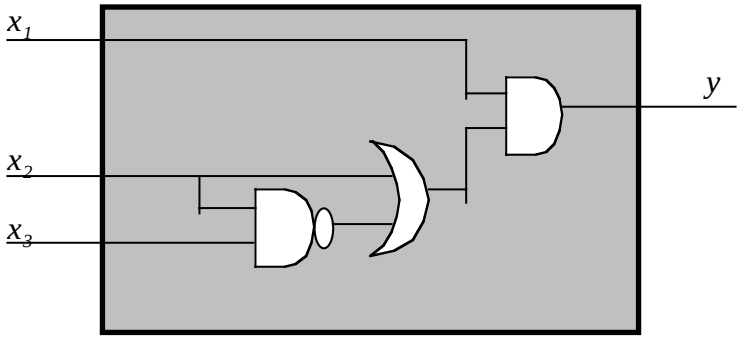
Esempio : la coppia di espressioni $< x_1 \cdot \overline{(x_2 + x_3)}\ ,\ \overline{(x_3 + x_1) \cdot \overline{x_1}} >$

ha come circuito equivalente



Tramite regole dell'algebra booleana è possibile semplificare reti combinatorie.

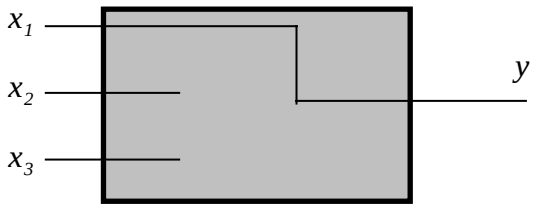
Esempio : il circuito



corrisponde all'espressione booleana

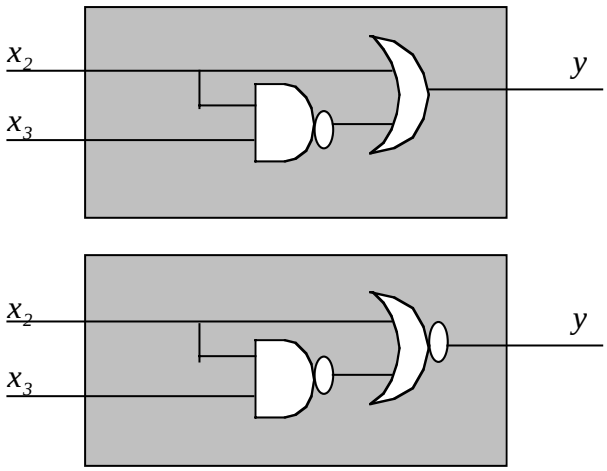
$$(\ \overline{(x_2 \cdot x_3)}\ +\ x_2\ ) \cdot x_1\ =\ (\ \overline{x}_2\ +\ \overline{x}_3\ +\ x_2\ ) \cdot x_1\ =\ 1 \cdot x_1\ =\ x_1$$

avendo usato De Morgan, il fatto che $x_2 + \overline{x}_2 = 1$, il fatto che qualunque numero sommato ad 1 dà 1 e che 1 è l'elemento neutro per $\cdot$. Pertanto il precedente circuito equivale a



OSS. : è possibile avere reti tautologiche (che ad ogni assegnazione delle variabili di input danno sempre 1) che reti contraddittorie (che ad ogni assegnazione danno sempre 0).

Esempio: le reti seguenti sono rispettivamente tautologica e contraddittoria



CIRCUITI ALL – NAND E ALL – NAND

Le porte NAND e NOR sono chiamate universali perché, tramite esse, si possono simulare tutte le altre porte.

Esempio :

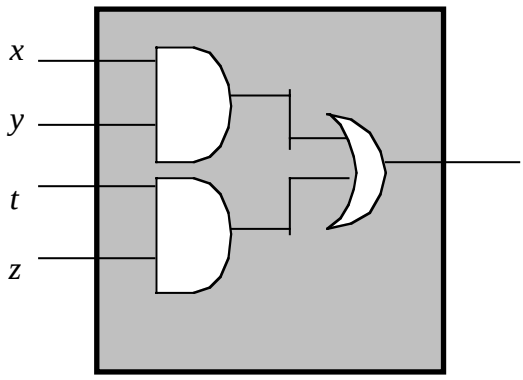
- ***NOT*** $(x) = $ ***NAND*** (x , x)
  Infatti $x = x \wedge x$ da cui $\neg x = \neg (x \wedge x)$

- ***AND*** $(x , y) = $ ***NAND*** $($ ***NAND*** $(x , y) ,$ ***NAND*** $(x , y))$
  Infatti $x \wedge y = \neg (\neg (x \wedge y)) = \neg (\neg (x \wedge y) \wedge \neg (x \wedge y))$

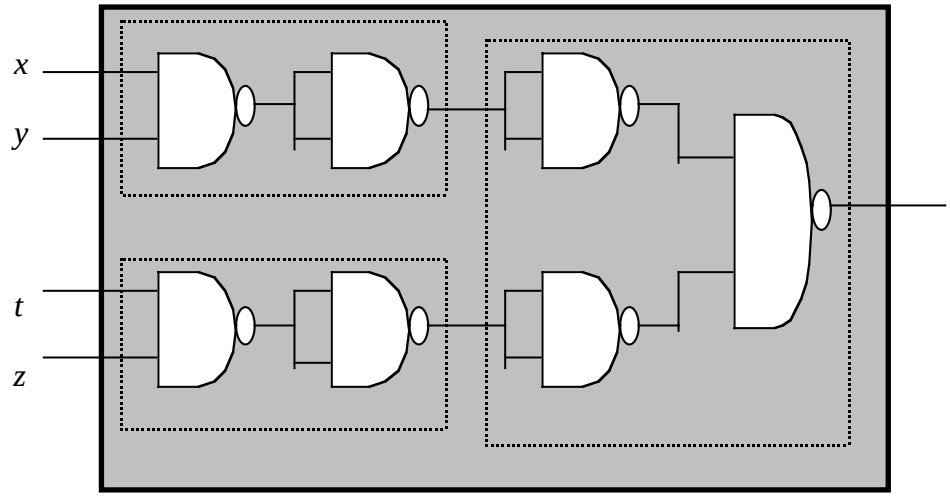
- ***OR*** $(x , y) = $ ***NAND*** $($ ***NAND*** $(x , x) ,$ ***NAND*** $(y , y))$
  Infatti $x \vee y = \neg (\neg x \wedge \neg y) = \neg (\neg (x \wedge x) \wedge \neg (y \wedge y))$

Quindi un qualsiasi circuito può essere realizzato usando solo porte ***NAND*** o porte ***NOR*** .

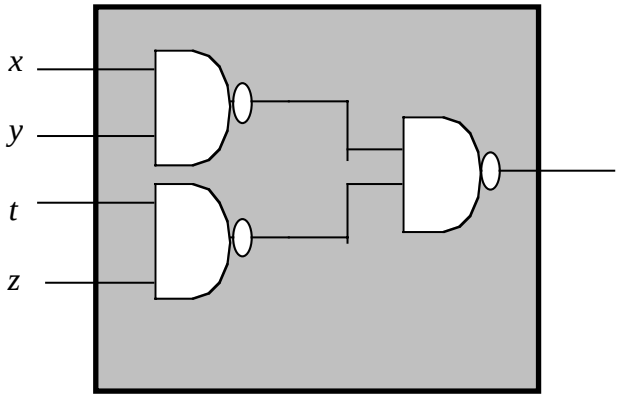
Esempio : trasformare in ALL – NAND il seguente circuito



Sostituendo porta a porta si ottiene il circuito



Osservando però che $(x \textbf{ NAND } x) \textbf{ NAND } (x \textbf{ NAND } x) = x$ [1] si ottiene



ESERCIZI DA SVOLGERE

1 Si dimostrino, facendo uso esclusivamente degli assiomi dell'algebra di Boole, le seguenti leggi:
   a) $\overline{x + y} = \bar{x} \cdot \bar{y}$ e $\overline{x \cdot y} = \bar{x} + \bar{y}$
   b) $1 + x = 1$
   c) *unicità dell'elemento complementare*
   d) $x + x = x$ e $x \cdot x = x$ (proprietà di idempotenza)

[1] Poich $\neg(\neg(x \wedge x) \wedge \neg(x \wedge x)) = (x \wedge x) \vee (x \wedge x) = x \vee x = x$.

2 Scrivere la FCD e FCC delle seguenti funzione booleane :

a) la funzione che dà 1 se la concatenazione dei quattro input x_1 , x_2 , x_3 , x_4 è la rappresentazione binaria di un numero naturale multiplo di 3.
b) la funzione che ha come input quattro valori binari e dà 1 se riceve almeno tre 1
c) la funzione che, presi 3 booleani, dà 1 se il numero di zeri è dispari

3 Per ognuna delle funzioni dell'esercizio precedente scrivere almeno due FND e due FNC diverse dalle FCD e FCC.

4 Scrivere i circuiti combinatori per le seguenti espressioni booleane e, se possibile, semplificarli usando regole dell'algebra di Boole:

a) $\overline{(x_1 + (x_2 \cdot x_3))} + x_2 \cdot ((\overline{x_4 \cdot x_2}) + \overline{x_3})$

b) $x_1 \cdot \overline{x_2} \cdot (x_1 + \overline{((x_3 \cdot x_2) + (x_1 \cdot x_2))})$

c) $\overline{x_2} + ((\overline{x_1 \cdot x_3}) + (x_3 \cdot \overline{x_2})) \cdot x_2$

5 Dimostrare come gli operatori ***NOT*** , ***AND*** , ***OR*** , ***NAND*** e ***XOR*** possano venir simulati usando esclusivamente porte ***NOR*** . Dimostrare poi come gli operatori ***NOR*** e ***XOR*** possano venir simulati usando esclusivamente porte ***NAND*** .

6 La regola $(x \textbf{ NAND } x) \textbf{ NAND } (x \textbf{ NAND } x) = x$ (usata nell'Esercizio 11) serve per semplificare un circuito ALL – NAND e corrisponde alla regola $\neg (\neg x) = x$. Utilizzando nozioni del corso di Logica Matematica, trovare ulteriori regole di semplificazione per circuiti ALL – NAND e ALL – NOR .

7 Tradurre le seguenti espressioni booleane in circuiti ALL – NAND e in circuiti ALL – NOR ed eventualmente semplificare i risultati ottenuti :

a) $(x_1 + x_2) \cdot \overline{x_2} + x_1$

b) $x_2 \cdot \overline{(x_1 + x_3)} \cdot (x_2 + \overline{x_3})$

2.5 Sintesi di reti combinatorie

La sintesi di una rete combinatoria implica il passaggio dalla specifica funzionale del circuito allo schema circuitale. Il procedimento di sintesi pu essere sviluppato in tre fasi:

1 Dalla specifica funzionale si ricava la funzione booleana rappresentata in forma tabulare (TT)
2 Dalla TT si ricava una espressione minima, o EB minima della TT
3 Dalla EB minima si ricava lo schema circuitale

Il passo 3 stato gi descritto nel paragrafo 2. In questo paragrafo ci occuperemo del problema di minimizzazione di una funzione booleana.

Il problema di ricavare una espressione minima deriva dalla necessit di **ridurre il numero di porte logiche** necessarie per realizzare una rete combinatoria. Con l'avvento dei circuiti integrati su scala media, alta e molto alta (MSI, LSI e VLSI) questa esigenza meno sentita, soprattutto in termini di costi. Il costo di una porta logica aggiunta pressoch nullo. Tuttavia, ridurre il numero di porte attraversate da un segnale logico ha ancora una sua rilevanza in termini di *specifiche temporali*. Il tempo di risposta di una rete combinatoria dipende dal numero di porte logiche attraversate: ridurre tale numero pu avere effetti importanti in termini di prestazioni.

2.5.1 Le mappe di Karnaugh

La tecnica di minimizzazione basata sul metodo delle mappe di Karnaugh pu essere studiata sul Fummi et al. par. 4.2

Si sa che per ogni FB esistono infinite EB equivalenti. Se ne vuol trovare la *minima* rispetto ad una qualche misura (tipicam. il numero di operatori). Infatti, tradotta l'EB in un circuito, si ha che :

- meno porte $\rightarrow$ minor costo (benché attualmente il costo di un componente elementare sia pressoché nullo)
- meno porte $\rightarrow$ minor tempo di attraversamento (di solito)

M. di K. $\Rightarrow$ metodo semplice ma buono solo fino a funzioni di 4 variabili. Trova una minima FND

Si basano sul fatto che $x \cdot y \cdot z + x \cdot y \cdot z = x \cdot y$

Le mappe consentono di identificare velocemente gli addendi della FCD così semplificabili.

Mappe :

$f(x, y)$

x \ y	0	1
0	p_{00}	p_{01}
1	p_{10}	p_{11}

$f(x, y, z)$

x \ $y\ z$	00	01	11	10
0	p_{000}	p_{001}	p_{011}	p_{010}
1	p_{100}	p_{101}	p_{111}	p_{110}

$f(x, y, z, t)$

$x\ y$ \ $z\ t$	00	01	11	10
00	p_{0000}	p_{0001}	p_{0011}	p_{0010}
01	p_{0100}	p_{0101}	p_{0111}	p_{0110}
11	p_{1100}	p_{1101}	p_{1111}	p_{1110}
10	p_{1000}	p_{1001}	p_{1011}	p_{1010}

Algoritmo:

- trasforma f in FCD.
- $p_i = \begin{cases} 1 & \text{se } minterm(p_i) \in \text{FCD}(f) \\ 0 & \text{altrimenti.} \end{cases}$
  (dove, per esempio, per $f(x,y,z)$ si ha che $P_2 = minterm(p_2) = minterm(p_{010}) = \bar{x} \cdot y \cdot \bar{z}$)
- raggruppa tutti gli 1 adiacenti con un insieme minimale di blocchi.

  (N.B.: le mappe sono da pensare curvate su di sé (*toro*) e quindi ogni elemento della mappa ha 4 elementi adiacenti [2])
  I blocchi prendono il nome di **implicanti**. Un blocco deve avere un numero di celle pari a 2^i (i=0,1,2..). I **primi implicanti** sono blocchi di dimensioni massima. I **primi implicanti essenziali** sono il numero minimo di blocchi di dimensione massima che coprono tutti gli 1 della tabella.
- considera la FND che ha per ogni blocco del ricoprimento un addendo ottenuto come il prodotto delle variabili che assumono lo stesso valore su tutti gli elementi del blocco. Le variabili saranno affermate se assumono 1 , negate altrimenti.[3]

[2] Per esempio, per $f(x,y,z,t)$, l elemento p_{0101} adiacente a p_{0001} , p_{0100} , p_{0111} e p_{1101} ; cos come p_{0000} adiacente a p_{0001} , p_{0100} , p_{1000} e p_{0010} .

[3] Per esempio, per $f(x,y,z,t)$, se il blocco copre due 1 adiacenti allora l addendo corrispondente sar il prodotto di 3 letterali, se il blocco copre quattro 1 adiacenti allora si avr il prodotto di 2 letterali, se il blocco copre otto 1 allora si avr un solo letterale

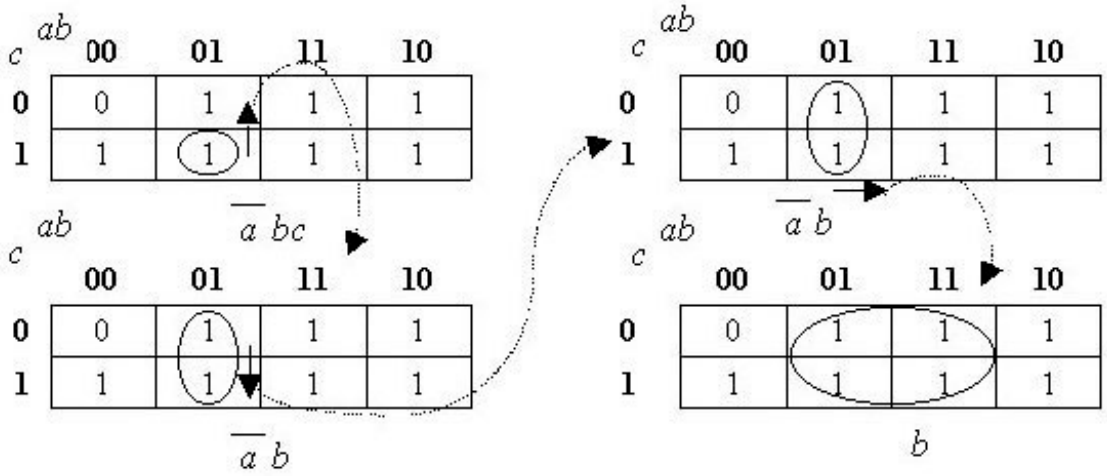


Figura 4.7 Espansione del mintermine $\overline{a}\,bc$ nel cubo b.

Esempio (da Fummi et al.)

Tabella 4.1 Tabella delle verità di una funzione $f(B^4) \rightarrow B$.

x	*y*	*z*	*v*	*o*		*x*	*y*	*z*	*v*	*o*
0	0	0	0	0		1	0	0	0	0
0	0	0	1	1		1	0	0	1	1
0	0	1	0	0		1	0	1	0	0
0	0	1	1	0		1	0	1	1	1
0	1	0	0	1		1	1	0	0	0
0	1	0	1	1		1	1	0	1	0
0	1	1	0	1		1	1	1	0	1
0	1	1	1	1		1	1	1	1	1

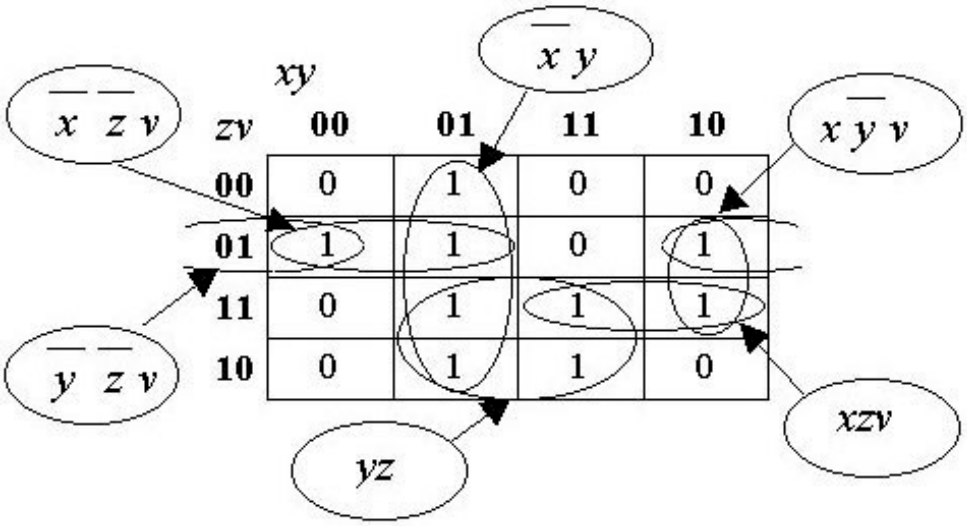


Figura 4.9 Mappa di Karnaugh per la funzione di Tabella 4.1.

Altri esempi:

Esempio 1: si trovi la minima FND per la funzione seguente

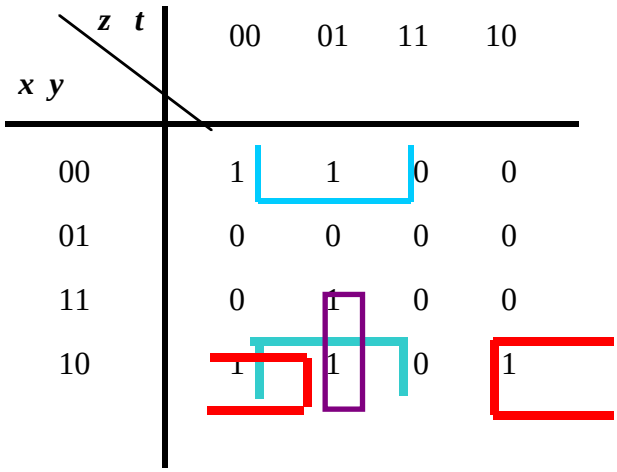
x	*y*	*z*	*t*	*f*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Anzitutto osserviamo che

$$\text{FCD}(f) = P_0 + P_1 + P_8 + P_9 + P_{10} + P_{13}$$

che richiede $4_{\text{porte NOT}} + 18_{\text{porte AND}} + 5_{\text{porte OR}} = 27_{\text{porte}}$

Ricaviamo la mappa:



E' possibile ricavare 3 primi implicanti essenziali, da cui

$$FND_{min} = \bar{y} \cdot \bar{z} + x \cdot \bar{z} \cdot t + x \cdot \bar{y} \cdot t$$

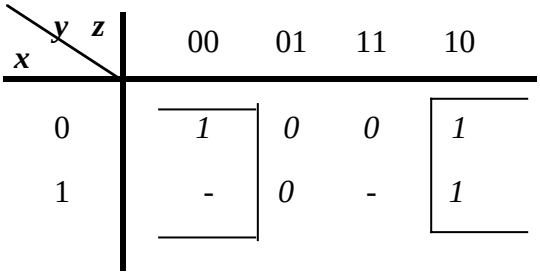
che richiede solo $3_{\text{porte NOT}} + 5_{\text{porte AND}} + 2_{\text{porte OR}} = 10_{\text{porte}}$

Il metodo funziona anche per funzioni booleane parziali (N.B.: una FB parziale è una funzione che non è definita su tutte i 2^n possibili input [4]). Con le mappe di Karnaugh i valori indefiniti possono venir considerati 0 o 1 a seconda di quale assegnamento dà il ricoprimento migliore.

Esempio 2: si trovi la minima FND per la funzione seguente

x	*y*	*z*	*f*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	-
1	0	1	0
1	1	0	1
1	1	1	-

La mappa di K. è



da cui $E_{min} = \bar{z}$

N.B.: è risultato migliore assegnare al primo '-' il valore 1 mentre al secondo il valore 0; ciò origina una EB t.c. $E_{min}(100) = 1$ e $E_{min}(111) = 0$. L'arbitrarietà di tale scelta è giustificata dal fatto che la funzione non è definita su tali input e quindi possiamo assumere che non si presenteranno mai alla funzione (e pertanto il comportamento della funzione su di essi è irrilevante).

[4] Una funzione parziale pu in realt essere vista come una funzione totale $f : \{0, 1\}^n \to \{0, 1, -\}$ dove - detto *don t care* ed esprime il fatto che la funzione non definita sull input corrispondente.

2.5.1 Sintesi di reti con circuiti MSI LSI

Come accennato nel paragrafo precedente, ridurre il numero di porte logiche in un circuito combinatorio ha effetti rilevanti in termini di specifiche temporali, mentre meno rilevante, nella maggior parte dei casi, il vantaggio in termini di costi. Per questo motivo, possibile utilizzare componenti integrati su scala media, larga e molto larga per realizzare reti combinatorie senza dover cablare su una schede singole porte logiche.
Nel seguito illustreremo tre possibili soluzioni: queste soluzioni comportano, in generale, un sottoutilizzo di porte logiche ma mantengono in linea di massima le stesse prestazioni in termini di tempo di risposta della rete (ovviamente, a parit di tecnologia).

Realizzazione di reti combinatorie mediante Multiplexers

Un **multiplexer** (MPX) una rete combinatoria con N ingressi, una uscita, e $\log_2 N$ segnali di controllo. In ogni istante t, l'uscita Y uguale al valore di uno ed uno solo degli ingressi, x_i. Il valore di i determinato dai segnali di controllo, che attivano uno fra N interruttori.
Un MPX costituito da un insieme di N porte AND, che funzionano da interruttori, e da un decodificatore.

Un **decodificatore** un dispositivo combinatorio con $\log_2 N$ ingressi ed N uscite, tale che, quando la combinazione dei valori di ingresso assume il valore decimale i, la porta di uscita i-esima uguale ad 1 e tutte le altre sono uguali a zero.
In base a queste specifiche verbali, si deduce che la TT di un decodificatore nxN, dove $n=\log_2 N$, la seguente:

xn... x2 x1	zN…z2z1
0... 0 0	0..... 1
0... 0 1	0..... 10
0... 1 0	0....100
0... 1 1	0..1000
.....	.....
1... 0 0	0001..0
1... 0 1	001....0
1... 1 0	01.....0
1... 1 1	10.....0

La figura mostra uno schema simbolico e uno schema circuitale di un MPX Nx1 e di un decodificatore 2x4.

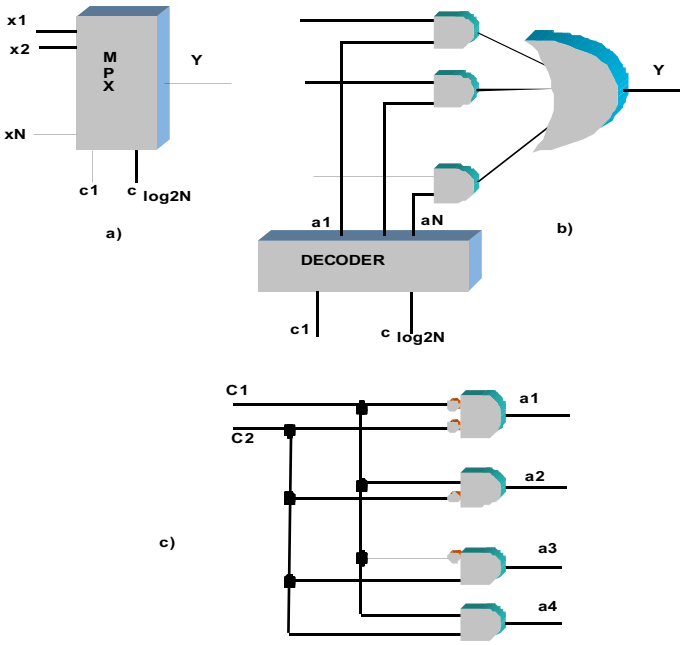


Figura 2.8 *Schema simbolico e circuitale di un Multiplexer Nx1*

MPX e decoders sono circuiti utili per la realizzazione di varie funzioni digitali (alcune delle quali vedremo in seguito). In questo paragrafo ci interessa mostrare come sia possibile realizzare un circuito combinatorio C, le cui specifiche siano fornite in termini di tabella di verit . Il procedimento molto semplice: data una funzione booleana di n variabili ed una uscita Y, si utilizza un MPX Nx1, con n=log$_2$N. Gli n segnali di ingresso della funzione booleana vengono collegati agli ingressi di controllo del MPX, mentre gli N ingressi principali vengono individualmente cablati al valore "0" o "1" (corrispondenti ad esempio ai valori della massa e dell'alimentazione), secondo quanto specificato dalla TT. Ad esempio, data la TT:

x3 x2 x1	Y
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	0

possiamo realizzare la rete combinatoria corrispondente come mostrato in figura.
Ad esempio, se la configurazione delle variabili di ingresso 001, il decodificatore nel MPX produce la 8-upla 00000010, collegando l'uscita Y con la seconda porta di ingresso del MPX, che a sua volta collegata al valore 1, come richiesto dalle specifiche.

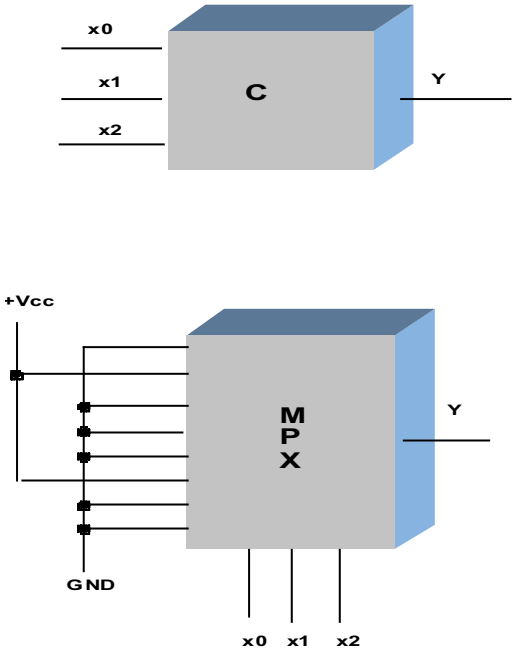


Figura 2.9 *Schema di implementazione di un circuito combinatorio* ***C*** *a 3 ingressi e 1 uscita mediante un* ***MPX*** *8x1*

Realizzazione di reti combinatorie mediante PLA

Un PLA o Programmable Logic Array un dispositivo integrato *semi-custom*, ovvero un dispositivo la cui realizzazione implica dei processi standard (cio indipendenti dall'applicazione) ed una fase finale *ad-hoc* realizzata secondo le specifiche fornite dal committente del circuito.
Un PLA una rete combinatoria integrata su scala medio-alta, con n ingressi, m uscite, e tre stadi interni: uno stadio di inversione dei segnali di ingresso, una matrice di porte AND ed una matrice di porte OR.
Un PLA consente di implementare espressioni booleane in forma FND.
La figura mostra lo schema a blocchi e lo schema circuitale per il caso di 4 ingressi e 4 uscite. Durante il processo standard di realizzazione del lay-out del circuito integrato, non vengono realizzate le connessioni fra stadi. Questo processo viene completato sulla base delle specifiche fornite dal committente. le specifiche vanno fornite in termini di funzioni FND.

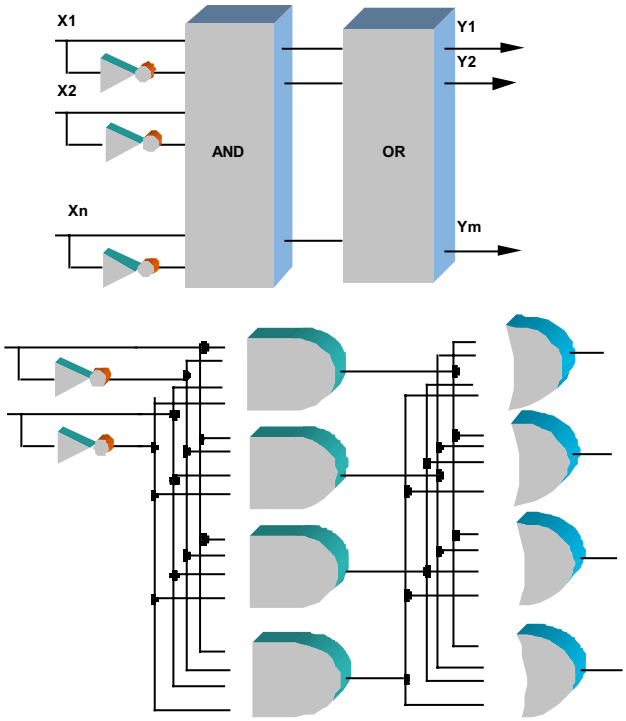


Figura 2.10 *Schema circuitale di un PLA a 4 ingressi e 4 uscite*

La successiva figura mostra lo schema simbolico di un PLA, corrispondente allo schema circuitale della precedente figura. Le righe in alto simboleggiano i segnali di ingresso, diretti o negati. Le colonne simboleggiano le porte AND. Le righe in basso simboleggiano le porte OR. Un pallino all'incrocio fra una riga ed una colonna simboleggia la realizzazione di una connessione. In figura (b), i pallini sulle colonne indicano (da sinistra verso destra) le seguenti connessioni sulle porte AND:

$X1\overline{X2}$, $\overline{X}1$, $\overline{X}1X2$, $\overline{X}1\overline{X2}$

Le espressioni booleane realizzate sono:

Y1 = $X1\overline{X2} + \overline{X}1$
Y2 = $\overline{X}1X2 + \overline{X}1\overline{X2}$
Y3 = $\overline{X}1 + \overline{X}1X2$
Y4 = $X1\overline{X2} + \overline{X}1\overline{X2}$

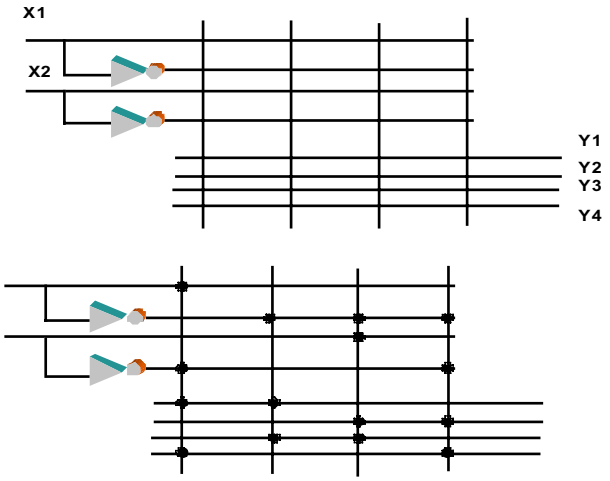


Fig 2.11 *Schema sintetico del PLA che realizza le EB precedenti*

Realizzazione di reti combinatorie mediante ROM

Una ROM (Read Only Memory) un dispositivo integrato su scala alta o molto alta (LSI-VLSI). Come mostrato in figura, una ROM un dispositivo con n ingressi (dette linee di indirizzamento) ed m uscite (dette linee dati). All'interno del dispositivo, le linee $A_1..A_n$ selezionano, tramite un decodificatore, una fra $N=2^n$ righe di una matrice Nxm. La selezione della riga i-esima della matrice (a_i) consente di leggere, su ciascuna delle colonne di uscita O_j il valore 0 o 1 stabilmente memorizzato nella cella di coordinate (i,j). In figura, un pallino nero simboleggia la memorizzazione di un 1, l'assenza del pallino simboleggia uno 0.
La memorizzazione di un valore booleano in una cella ROM permanente, ovvero, non pu essere cancellato.
In figura mostrato un possibile metodo di memorizzazione. Se si desidera memorizzare un 1 in posizione (i,j) , si pone all'incrocio della riga e colonna corrispondenti un dispositivo detto *diodo*. Un diodo un dispositivo elettronico che consente il transito di un segnale in una sola direzione, da sinistra verso destra se il diodo posizionato come in figura. Se il decoder seleziona la porta di uscita a_i, il valore 1 presente sulla corrispondente linea del decoder viene trasferito sulla linea di uscita Oj, se il diodo connesso come in figura.

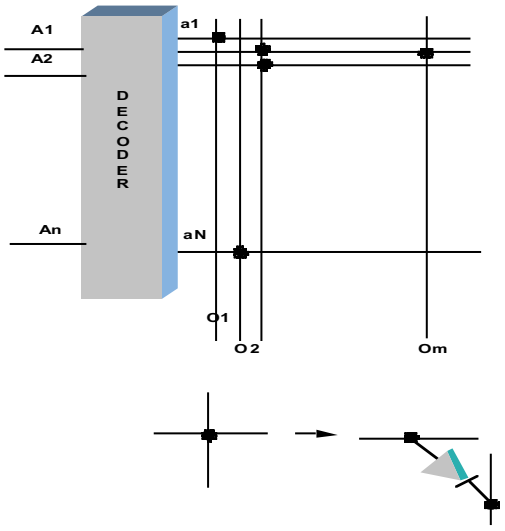


Figura 2.12 *Schema di una possibile implementazione di una ROM*

Una ROM pu essere usata per realizzare un insieme di funzioni booleane le cui specifiche siano fornite in termini di tabelle di verit TT.
La realizzazione molto semplice: basta "copiare" la parte destra della tabella di verit (che una matrice Nxm, dove N sono le possibili combinazioni delle n variabili di ingresso, ed m sono le funzioni booleane di uscita) nella matrice della ROM.
Ad esempio, se sono assegnate le seguenti specifiche:

$A_3A_2A_1$	O_2O_1
0 0 0	0 0
0 0 1	0 1
0 1 0	0 0
0 1 1	0 1
1 0 0	0 0
1 0 1	1 1
1 1 0	0 1
1 1 1	1 1

lo schema della ROM che le implementa mostrato in figura (i codici in uscita al decoder indicano la combinazione delle variabili $A_3A_2A_1$ che portano ad 1 la riga corrispondente).

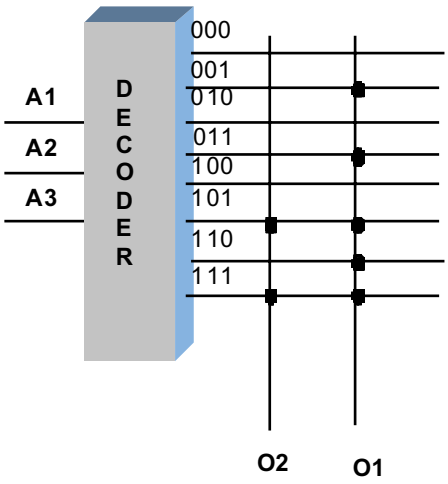


Figura 2.13 *Schema della ROM relativa alla precedente TT*

Paragone fra le soluzioni considerate

L'utilizzo di componenti integrati su scala bassa per la realizzazione di reti combinatorie conveniente solo nel caso in cui il numero di porte logiche da utilizzare dell'ordine di 10. Nello scegliere l'espressione booleana minimizzata *E* che realizza una certa funzione booleana espressa come TT, occorre minimizzare lo spreco di porte logiche. A questo scopo, pu essere conveniente usare il teorema di DeMorgan per trasformare *E* in modo tale da poter ottimizzare l'uso degli integrati a disposizione.
La realizzazione di una rete combinatoria tramite un MPX conveniente per reti di media scala. Occorre tener presente i seguenti vantaggi e svantaggi:

- (+) il componente pu essere riutilizzato
- (-) ogni componente realizza una sola funzione booleana (al pi)

PLA e ROM consentono una facile realizzazione di reti combinatorie su scala larga e molto larga. Occorre tener presente i seguenti vantaggi e svantaggi:

- (-) il componente non pu essere riutilizzato, in quanto la realizzazione delle specifiche a cura del fabbricante del circuito integrato. Nel caso di ROM, esistono tipi di ROM (EPROM) che consentono - disponendo di opportuna apparecchiatura- la riscrittura della matrice.
- (+) ogni componente realizza pi funzioni booleane.

2.6 Esempi di reti combinatorie “notevoli”

Nel seguito vedremo alcuni esempi di sintesi di circuiti combinatori "notevoli", ovvero di uso comune. Introdurremo questi circuiti seguendo la metodologia di sintesi vista nel paragrafo 5.

2.6.1 Comparatore aritmetico

Problema: date due stringhe binarie di n bit A e B rappresentanti due numeri naturali, dire se

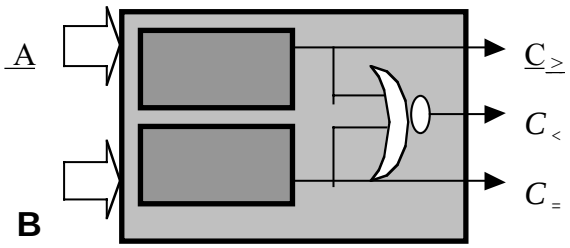
A > B , A = B , A < B

OSS.1 : è possibile solo una delle condizioni appena presentate!!

Vogliamo un circuito del tipo

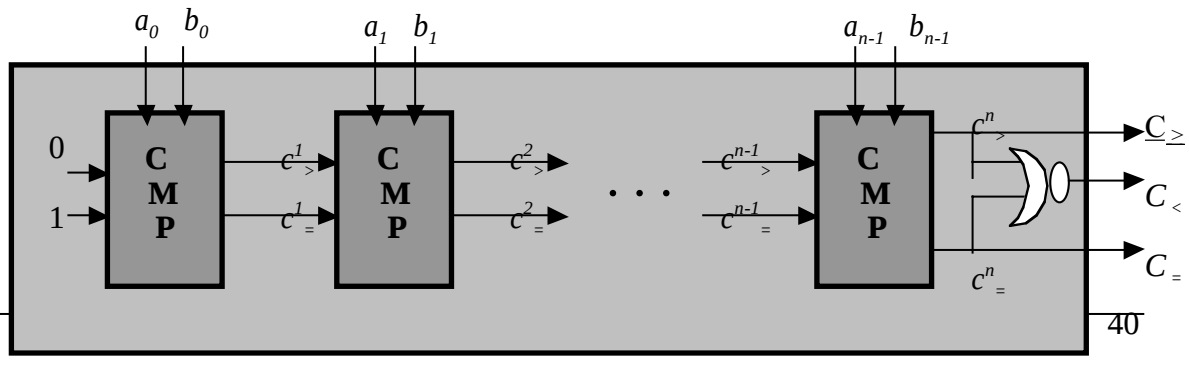
tale che :
- $C_> = 1$ sse $A > B$
- $C_< = 1$ sse $A < B$
- $C_= = 1$ sse $A = B$

OSS.2: per l’OSS.1 si ha che $C_< = C_> \; \textbf{NOR} \; C_=$ e pertanto basterà trovare i circuiti per calcolare $C_>$ e $C_=$, da cui

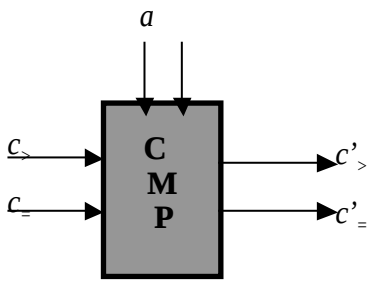


L’idea è quella di costruire un circuito incrementale, cioè un circuito formato da n celle circuitali elementari di confronto messe in cascata; per fare ciò usiamo dei risultati parziali così definiti:

Per ogni $i = 1, \dots, n$
- $c^i_> = 1$ sse $a_{i-1} \dots a_0 > b_{i-1} \dots b_0$
- $c^i_= = 1$ sse $a_{i-1} \dots a_0 = b_{i-1} \dots b_0$



A)REALIZZAZIONE della CELLA ELEMENTARE di COMPARAZIONE (CMP)



La tabella di verità per **CMP** è

a	*b*	$c_>$	$c_=$	$c'_>$	$c'_=$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	-	-
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	-	-
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	-	-
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	-	-

N.B.: in corrispondenza degli input con $c_> = c_= = 1$ sono stati messi due '-' poiché, per OSS.1, tale situazione non si presenterà mai.

A.1) REALIZZAZIONE del SOTTO-CIRCUITO per $c'_>$

La mappa di K. è

a b \ $c_>$ $c_=$	00	01	11	10
00	0	0	-	1
01	0	0	-	0
11	0	0	-	1
10	1	1	-	1

da cui $E_{min} = c_> \cdot a + c_> \cdot \bar{b} + a \cdot \bar{b}$

A.2) REALIZZAZIONE del SOTTO-CIRCUITO per $c'_=$

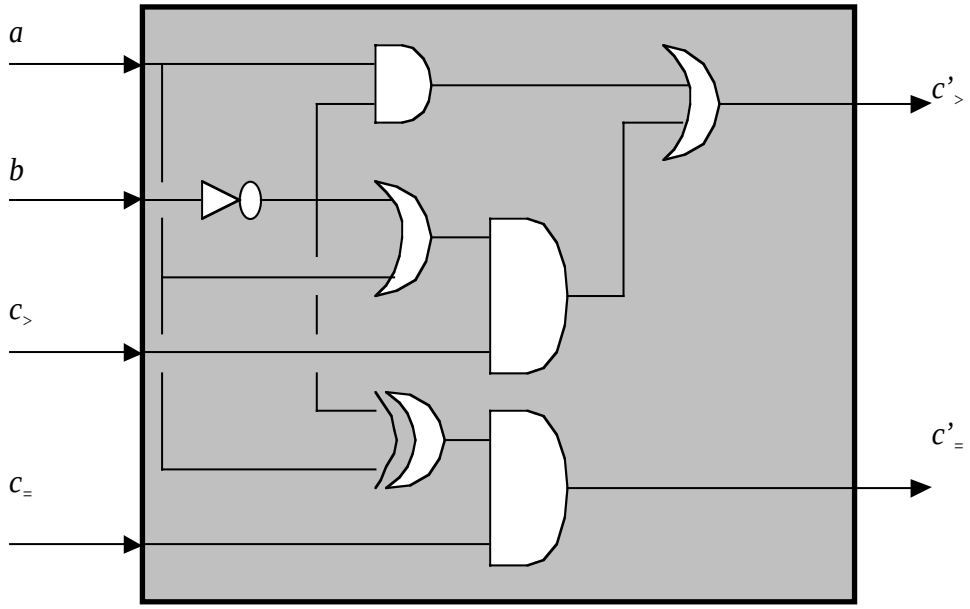
La mappa di K. è

$a\ b$ \ $c_> c_=$	00	01	11	10
00	0	1	-	0
01	0	0	-	0
11	0	1	-	0
10	0	0	-	0

da cui $E_{min} = c_= \cdot a \cdot b + c_= \cdot \bar{a} \cdot \bar{b}$

che fattorizzata porta a $E'_{min} = c_= \cdot (a \cdot b + \bar{a} \cdot \bar{b}) = c_= \cdot (a \ \mathbf{XOR} \ \bar{b})$

A.3) DISEGNO del CIRCUITO



A.4) COMPARATORE di INTERI

Il circuito concettualmente resta lo stesso; bisogna però notare che, lavorando con interi, bisogna tener conto dei segni e dei valori assoluti. In particolare:

- *se A e B sono entrambe non-negativi :* vedi pagg. precedenti
- *se A e B sono entrambe negativi :* gli output del circuito ottenuto nelle pagg. precedenti vanno complementati
- *se A è non-negativo e B è negativo :* si ha sempre $C_> = 1$, $C_< = C_= = 0$
- *se A è negativo e B è non-negativo :* si ha sempre $C_> = 0$, $C_> = C_= = 0$

Per il caso in cui gli interi sono espressi in complemento a 2, guardatevi la soluzione del seguente esercizio, proposto come esercizio di esame a Settembre 2004 :

Esercizio: Progettare una rete logica combinatoria chr confronta due interi A e B in complemento a 2, ciascuno a 3 bit (valori da –4 a +3) fornisce su tre linee L E e G ciascuna delle quali è 1 in corrispondenza dei 3 possibili risultati di un confronto fra A e B. In particolare:
L=1 se A<B, E=1 se A=B, G=1 se A>B

Soluzione:
Ricordate che, in complemento a due, un intero N si esprime mediante la:

$$N = -2^{n-1} + \sum_{i=0..n-2} b_i \, 2^i$$

$$b_i \in \{0,1\}$$

Quindi, con numeri di tre bit, si ha:

N_{10}	N_{Ca2}
+3	011
+2	010
+1	001
0	000
-1	111
-2	110
-3	101
-4	100

Ad es, 100 = $-1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = -4$

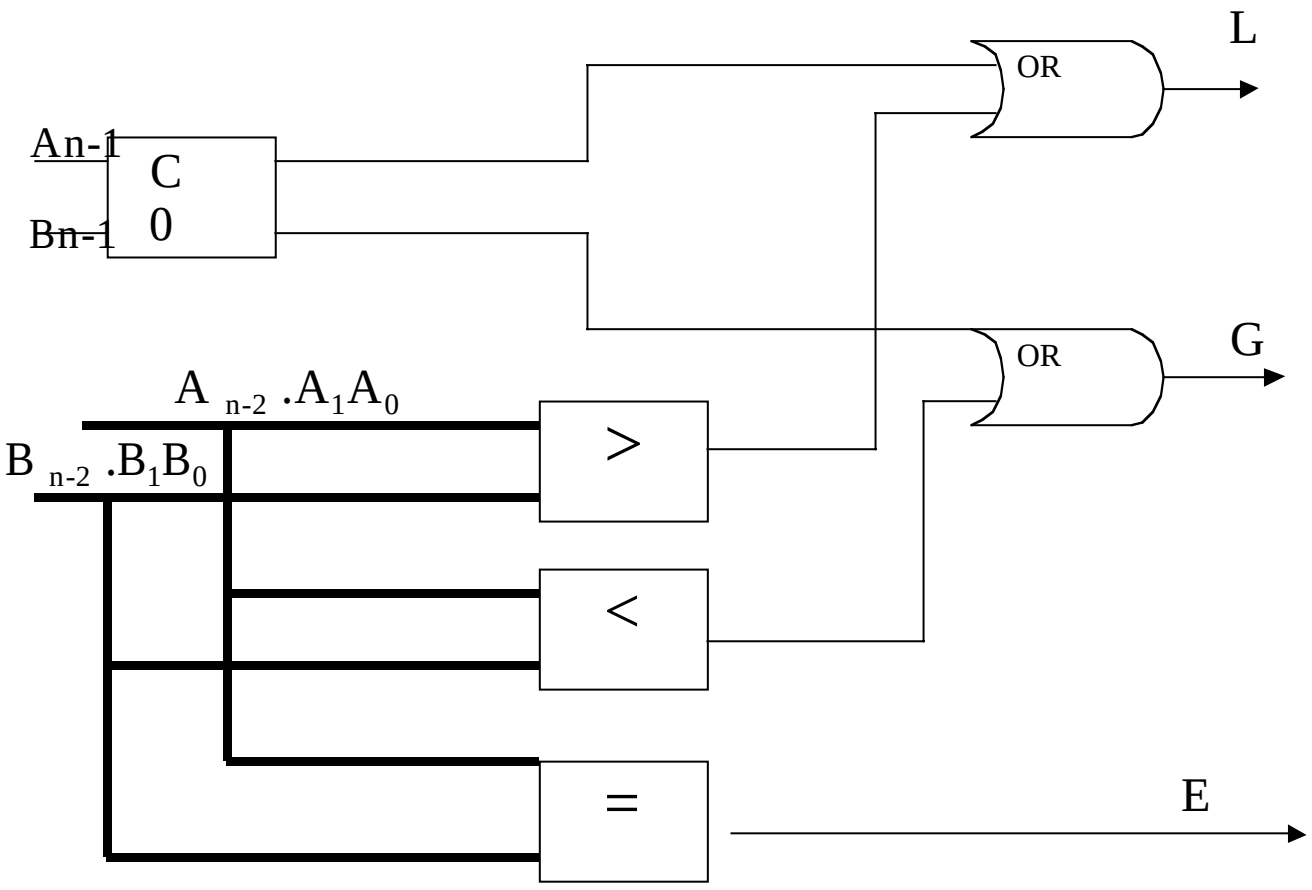
Per risolvere l'esercizio, si consideri che il confronto dei bit più significativi delle stringhe A e B può portare in alcuni casi ad una decisione, senza dover confrontare il resto delle stringhe. Infatti, indicando con An-1 e Bn-1 gli MSB di A e B (nel caso generale di stringhe di n bit), si ha:

A_{n-1} B_{n-1}	L E G	L1 G1
00	? ? ?	00
01	1 0 0 B<A	10
10	0 0 1 A<B	01
11	? ? ?	00

Indichiamo con C0 il circuito combinatorio la cui tabella di verità è qui sopra riportata, e con L1 e G1 le relative uscite (la colonna centrale non fa parte della tabella).
Se il primo confronto non produce una decisione, si deve passare al confronto delle due sottostringhe A_{n-2} .A_1A_0 e B_{n-2} ..B_1B_0. Tale confronto a questo punto può utilizzare i normali circuiti di paragone, utilizzati per i numeri naturali. Per come sono rappresentati i numeri in Ca2, un circuito di paragone "standard" può confrontare anche numeri negativi, infatti ad es. –4 (100) è minore di –1 (111).
Per risolvere il problema in maniera generale, cioè indipenentemente dal numero di bit delle stringhe A e B (è possibile una soluzione ad-hoc per il caso di confronto di stringhe di due bit, mediante un circuito combinatorio avente in ingresso la stringa $A_1A_0B_1B_0$), possiamo usare 3 circuiti combinatori diversi:
C1 un comparatore di maggioranza stretta, che produce un output Y1=1 se A>B
C2 un comparatore di minoranzaa stretta che produce un output Y2=1 se A<B
C3 un rilevatore di uguaglianza, che produce Y3=1 se A=B

C1, C2 e C3 ricevono in ingresso solo gli n-1 bit meno significativi delle stringhe n-arie A e B. C1 e C2 si realizzano mediante una cascata di moduli "comparatori di

maggioranza" ciascuno dei quali riceve il risultato della comparazione del modulo precedente (ti-1) e produce un output per il comparatore seguente (ti). Il valore dei ti sarà uno se le sottostringhe confrontate fino al modulo i producono il risultato $A_{i,i-1..0} > B_{i,i-1..0}$.



2.6.2 Sommatore Parallelo a n bit

Specifica: un sommatore binario realizza la somma aritmetica fra due stringhe di n bit, $A_1..A_n$ $B_1..B_n$

Ricavare la funzione booleana. Concentriamoci sull'esecuzione del generico passo i della somma di due stringhe. La somma aritmetica fra i bit A_i e B_i dipende dal riporto della somma dei bit A_{i-1} B_{i-1}. Analogamente, il riporto della somma fra A_i e B_i influenzer la somma fra A_{i+1} B_{i+1}.

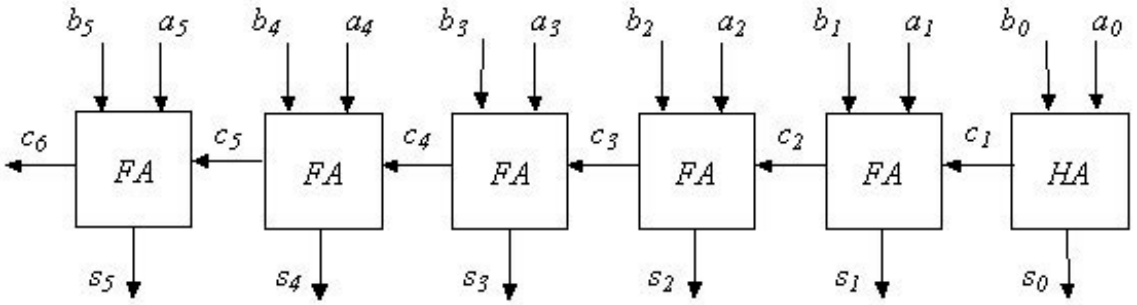


Figura 9.2 Sommatore a propagazione di riporto.

Indicando con C_{i-1} il riporto (*carry*) della somma fra A_{i-1} B_{i-1} e con C_i quello della somma fra A_i e B_i , avremo la seguente tabella di verit per il modulo che esegue la somma fra A_i e B_i.

$C_{i-1}B_iA_i$	Sum_iC_i
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

Ricavare una espressione booleana minima.

Dalla tabella di verit , applicando il metodo di Karnaugh, otteniamo (si ricordi che $\oplus$ usato come il simbolo dell'XOR):

$$Sum_i = A_i \oplus B_i \oplus C_i$$
$$C_i = A_iC_{i-1} + B_iC_{i-1} + A_iB_i$$

Ottenere lo schema circuitale dall'espressione minimizzata.
Dettagli sul progetto del sommatore sono sul libro Fummi et al. par 9.3.1

2.6.3 Incrementatore

Specifica: un incrementatore un dispositivo che riceve in ingresso n bit $A_1..A_n$ che rappresentano il numero A, e produce in uscita n bit che rappresentano il numero A+1. Il caso che consideriamo quello di un numero intero positivo. Quando $A=2^n$, il risultato dell'

incremento pu essere un segnale di errore (e dunque avremo bisogno di un output aggiuntivo) oppure una stringa di zero (incrementatore modulo 2^n).

Ricavare la funzione booleana: Nel secondo caso, la tabella di verit si ricava facilmente dalla specifica (consideriamo il caso di n=3):

$A_2A_1A_0$	$B_2B_1B_0$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0

Ricavare una espressione booleana minima.

Dalla tabella di verit , applicando il metodo di Karnaugh, otteniamo:

$B_0 = \overline{A_0}$

Lasciamo per esercizio il calcolo di B_1 e B_2, e la derivazione dello schema circuitale.
Per esercizio, si possono anche ridisegnare i circuiti ricavati per comparatore, sommatore e incrementatore usando PLA, ROM o MPX.