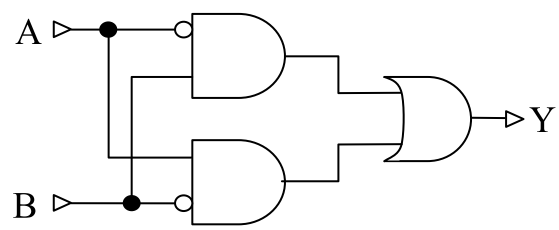
Esercitazione del 16/03/2006 - Soluzioni

Rappresentazioni possibili per una funzione logica:

- **circuito logico:**



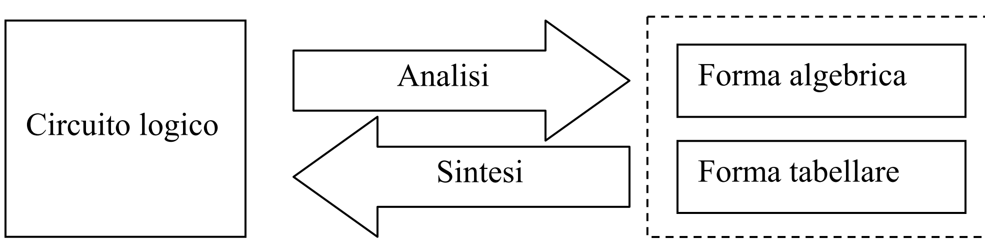
- **forma tabellare** (tabella lookup):

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

- **formula algebrica:**

Y= (NOT A)B OR NOT(B) A

Sintesi e analisi di circuiti logici:



Precedenza degli operatori logici:
(L04 -I circuiti binari: definizione delle funzioni logiche, p.31)

In assenza di parentesi, AND ha la priorità sull'OR ed il NOT su entrambi:

NOT > AND > OR

Principio di dualità:
(L04 -I circuiti binari: definizione delle funzioni logiche, p.34)

Il duale di una funzione si ottiene sostituendo:
AND con OR,OR con AND, 0 con 1 ed 1 con 0.

Proprietà generali dell'Algebra Booleana:
(L04 -I circuiti binari: definizione delle funzioni logiche, p.32)

0. Doppia Inversione	$\sim(\sim x) = x$	
1. Identità:	$1\,x = x$	$0 + x = x$
2. Elemento nullo:	$0\,x = 0$	$1 + x = 1$
3. Idempotenza:	$x\,x = x$	$x + x = x$
4. Inverso:	$x \sim x = 0$	$x + \sim x = 1$
5. Commutativa:	$x\,y = y\,x$	$x + y = y + x$
6. Associativa:	$(x\,y)\,z = x\,(y\,z)$	$(x + y) + z = x + (y + z)$
7. Distributiva:	$x\,(y + z) = x\,y + x\,z$	$x + y\,z = (x + y)\,(x + z)$
8. Assorbimento:	$x\,(x + y) = x$	$x + x\,y = x$
9. De Morgan:	$\sim(x\,y) = \sim x + \sim y$	$\sim(x + y) = \sim x \; \sim y$

Forma Canonica SOP *(Sum Of Product)* :

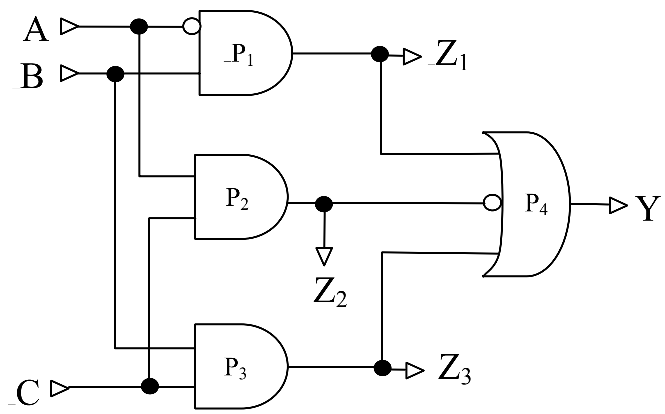
Implicante → prodotto di variabili per cui una funzione vale 1
Mintermine → Implicante contenente tutte le variabili della funzione

Forma **SOP** di F → $F = \sum_{i=1}^{Q} m_j$ dove m_j è il *j*-esimo mintermine della funzione F

Ex: $\mathbf{A}$ XOR $\mathbf{B} = \overline{\mathbf{A}}\,\mathbf{B} + \mathbf{A}\,\overline{\mathbf{B}}$

Massimo Marchi – marchi@dsi.unimi.it

1. **Ricavare la forma tabellare , la prima forma canonica e la forma algebrica del seguente circuito semplificando dove possibile.**



Costruite una tabella con una riga per ogni possibile combinazione degli ingressi ed una colonna con i risultati intermedi per l'uscita di ogni porta logica, ev. aggiungendo ulteriori colonne per il risultato negato (come ~Z_2 nell'esempio).

AND

OR

A	**B**	**C**	~A	Z_1=~AB	Z_2=AC	Z_3=BC	~Z_2	**Y**=Z_1+~Z_2+Z_3
0	0	~~0~~	~~1~~	0	0	0	1	1
0	0	~~1~~	~~1~~	0	0	0	1	1
0	1	0	1	1	0	0	1	1
0	1	1	1	1	0	1	1	1
1	0	0	0	0	0	0	1	1
1	0	1	0	0	1	0	0	0
1	1	0	0	0	0	0	1	1
1	1	1	0	0	1	1	0	1

Nota: *una porta AND dà come risultato 0 quando uno dei suoi ingressi è 0. Per calcolare una colonna risultato di una AND allora è comodo procedere nel seguente modo: per ogni termine si identificano le celle a 0 e si pone a 0 la cella risultato corrispondente. Alla fine le celle ancora vuote si pongono ad 1. Analogamente il metodo duale può essere applicato alle porte OR: prima si identificano le celle risultato a 1 corrispondenti ai termini posti uguale a 1 quindi si completano le celle ancora vuote con 0.*

Ex. Si consideri il calcolo di Z_1. Il termine ~A va a zero per le ultime quattro configurazioni. Ne segue che Z_1 può essere posto a zero per le corrispondenti celle. Analogamente B è uguale a zero per le prime due configurazioni e per la 5^{ta} e la 6^{ta} . Ne segue che Z_1 può essere messo a 0 anche per le prime due celle. Le restanti celle saranno obbligatoriamente uguali ad 1.

Nota: Il numero di mintermini nella forma canonica SOP è pari al numero di 1 nella colonna risultato. Viceversa il numero di maxtermini presenti nella seconda forma canonica POS è uguale al numero di 0 presenti. In questo caso quindi sarebbe più conveniente in termini di compattezza di descrizione usare la seconda forma canonica POS al posto della forma SOP.

Calcoliamo la forma algebrica partendo dal circuito logico:

Nota: E' comodo ricostruire la formula partendo dalle uscite e risalendo poi il circuito fino agli ingressi.

NOTA: Per ogni passaggio di semplificazione è indicato il punto di lavoro con un evidenziazione gialla e il risultato ottenuto con una zona sottolineata nella riga successiva.

Y =	← *Sviluppo la porta* P_4
= (Z_1 + ~Z_2 + Z_3)	← *Sviluppo le porte* P_1, P_2 *e* P_3
= (~AB + ~(AC) + BC)	← *8b De Morgan:* ~(**xy**) = ~**x** + ~**y**
= (~AB + ~A + ~C + BC)	← *6b Assorbimento:* (~**x**)**y** + (~**x**) = (~**x**)
= (~A + ~C + BC)	

Dim: **x** + ~**xy** = (**x** + **xy**) + ~**xy** ←*8b*
= ***x*** + (***xy*** + ~***xy***) ←*6b*
= ***x*** + ***y(x*** + ~***x)*** ←*7a*
= ***x*** + ***y1*** ←*4b*
= ***x*** + ***y*** ←*1a*

Vale la seguente proprietà: **x** + ~**xy** = **x** + **y**

Questo *non è altro che l'unico maxtermine della seconda forma canonica.*

= (~A + ~C + BC)	← *Applico:* **x** + ~**xy** = **x** + **y**
= (~A + ~C + B)	← *8b De Morgan*
= ~ (A~BC)	

Questo *non è altro che l'unico mintermine della prima forma canonica della funzione ottenuta negando quella data.*

Calcoliamo la forma canonica partendo dalla tabella e passando per la forma SOP:

La prima forma canonica si ottiene sommando i mintermini della funzione risultato. Ad ogni combinazione degli ingressi per cui la funzione vale 1 corrisponde un mintermine.

Y = ~A~B~C + ~A~BC + ~AB~C + ~ABC + A~B~C + AB~C + ABC

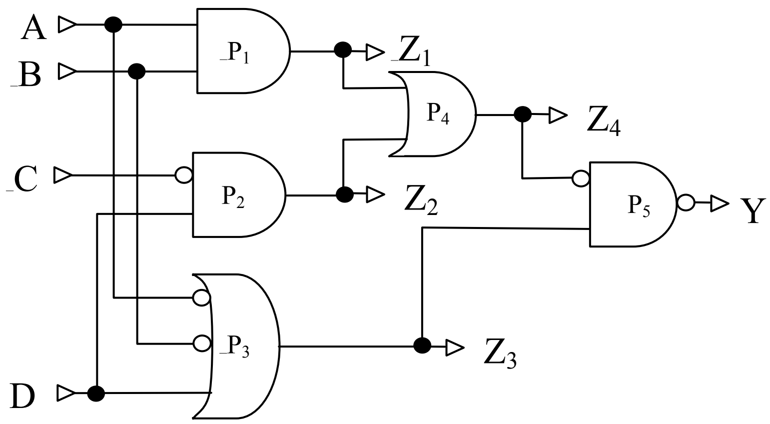
Semplifichiamo applicando varie volte le regole 7a e 4b:

x y + ~***x y*** = (***x*** + ~***x***) ***y*** = ***1 y*** = ***y***

Questo corrisponde ad individuare di volta in volta degli *implicanti* (**y**) sempre più piccoli della funzione data

: Y = ~A~B~C + ~A~BC + ~AB~C + ~ABC + A~B~C + AB~C + ABC	
Y = ~A~B + ~AB~C + ~ABC + A~B~C + AB~C + ABC	
Y = ~A~B + ~AB + A~B~C + AB~C + ABC	
Y = ~A~B + ~AB + A~C + ABC	
Y = ~A + A~C + ABC	← *6b Associativa:* **A**(**w**)+**A**(**z**) = **A**[(**w**) +(**z**)]
Y = ~A + A(~C + BC)	← **x** + ~**xy** = **x** + **y**: **x**=*~C* , *~***x***=~(~C)=C*
Y = ~A + A(~C + B)	← **x** + ~**xy** = **x** + **y**: **x**=*~A* , ***y***=*(~C+B)*
Y = ~A + ~C + B = ~ (A~BC)	← *Vedi sopra*

2. **Ricavare la forma tabellare , la prima forma canonica e la forma algebrica del seguente circuito semplificando dove possibile.**



Forma tabellare:

A	B	C	D	~A	~B	~C	Z_1= AB	Z_2=~CD	Z_3=~A~BD	Z_4= Z_1+ Z_2	~Z_4	Z_3~Z_4	Y
0	0	0	0	1	1	1	0	0	1	0	1	1	0
0	0	0	1	1	1	1	0	1	1	1	0	0	1
0	0	1	0	1	1	0	0	0	1	0	1	1	0
0	0	1	1	1	1	0	0	0	1	0	1	1	0
0	1	0	0	1	0	1	0	0	1	0	1	1	0
0	1	0	1	1	0	1	0	1	1	1	0	0	1
0	1	1	0	1	0	0	0	0	1	0	1	1	0
0	1	1	1	1	0	0	0	0	1	0	1	1	0
1	0	0	0	0	1	1	0	0	1	0	1	1	0
1	0	0	1	0	1	1	0	1	1	1	0	0	1
1	0	1	0	0	1	0	0	0	1	0	1	1	0
1	0	1	1	0	1	0	0	0	1	0	1	1	0
1	1	0	0	0	0	1	1	0	0	1	0	0	1
1	1	0	1	0	0	1	1	1	1	1	0	0	1
1	1	1	0	0	0	0	1	0	0	1	0	0	1
1	1	1	1	0	0	0	1	0	1	1	0	0	1

Nota: Dalla tabella si possono ricavare suggerimenti sul come semplificare la formula data. Ex: il gruppo di **1** *corrispondenti alle configurazioni per cui A e B valgono* **1** *suggerisce di cercare di derivare l'implicante* **AB** *durante la semplificazione della SOP.*

.Stesso discorso vale per gli **1** *presenti quando* **CD** = **01** *che suggerisce di cercare di derivare l'implicante ~***CD.**

Calcoliamo la forma algebrica partendo dal circuito logico:

$$Y = \overline{(\overline{Z_4} + Z_3)} = \overline{\overline{(Z_1 + Z_2)}\ (\overline{A}+\overline{B}+D)}$$ ← *Sviluppo le porte del circuito*

$$= \overline{\overline{(AB + \overline{C}D)}\ (\overline{A}+\overline{B}+D)}$$ ← *9a De Morgan*

$$= \underline{\overline{\overline{(AB + \overline{C}D)}}} + \underline{\overline{(\overline{A}+\overline{B}+D)}}$$ ← *0 Doppia Negazione*

$$= (\underline{AB + \overline{C}D}) + \overline{(\overline{A}+\overline{B}+D)}$$ ← *9b De Morgan*

$$= (AB + \overline{C}D) + (\ \underline{\overline{(\overline{A}+\overline{B})}\ \overline{D}}\ )$$ ← *9b De Morgan*

$$= (AB + \overline{C}D) + (\overline{\overline{A}}\ \ \overline{\overline{B}}\ \overline{D})$$ ← *0 Doppia Negazione*

$$= (AB + \overline{C}D + (\underline{AB}\overline{D})$$ ← *5b Commutativa*

$$= AB + \underline{AB\overline{D} + C\overline{D}}$$ ← *8b Assorbimento*

$$= \mathbf{\underline{AB} + \overline{C}D}$$

Nota: Questa è la somma dei due implicanti indviduati in tabella!!! Ci sono solo loro perché insieme coprono tutti gli **1** *della tabella (vedi mappe di karnaugh)*

Quando compaiono diversi livelli di negazione conviene applicare ripetutamente Demorgan e Doppia Negazione

Calcoliamo la forma canonica partendo dalla tabella e passando per la forma SOP:

Y = ~A~B~CD + ~AB~CD + A~B~CD + AB~C~D + AB~CD + ABC~D + ABCD

Semplifichiamo applicando varie volte: ***x y*** + ~***x y*** = ***y***

Y = ~A~B~CD + ~AB~CD + A~B~CD + AB~C~D + AB~CD + ABC~D + ABCD

= ~A~B~CD + ~AB~CD + A~B~CD + AB~C~D + AB~CD + <u>ABC</u>

= ~A~B~CD + ~AB~CD + A~B~CD + AB~C + ABC

= ~A~B~CD + ~AB~CD + A~B~CD + <u>AB</u>

= <u>~B~CD</u> + ~AB~CD + AB

Nello sviluppo e semplificazione tramite le SOP conviene come primo passaggio individuare degli implicanti dellafunzione data. Questo si può ottenere individuando iterativamente coppie di termini uguali eccetto che per un letterale che compare sia negato che non negato.

Applico sul termine **AB** la regola di 8b, ***x* = *x* + *x y*,**
in modo da creare un termine **AB~CD** da utilizzare nella semplificazione successiva:

= ~B~CD + ~AB~CD + AB

= ~B~CD + ~AB~CD + <u>AB + AB~CD</u> ← 7a e 4b: ***x y*** + ~***x y*** = ***y***

= ~B~CD + <u>B~CD</u> + AB ← idem

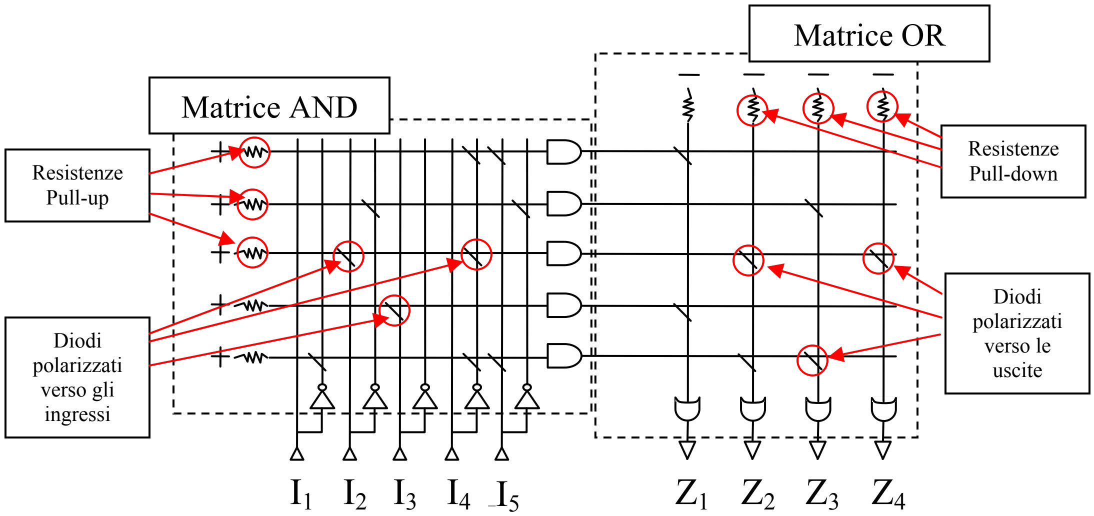
= ~CD + AB ← *5b Commutativa*

= **AB + ~CD**

Massimo Marchi – marchi@dsi.unimi.it

PLA : Programmable Logic Array

Sono costituite da due sezioni: una matrice AND che può codificare un certo numero di implicanti ed una matrice OR che mette insieme gli implicanti ottenuti. Praticamente



PLA a 5 ingressi e 4 uscite; può sintetizzare 5 implicanti diversi (corrispondenti alle AND presenti nella matrice AND).

realizzano funzioni attraverso forme SOP. Vengono usate per codificare funzioni con un alto numero di ingressi ed uscite ma con una descrizione in termini di implicanti non troppo complicata (la matrice AND in genere permette di codificare solo un sotto-insieme di possibili implicanti) e con implicanti utilizzati in più funzioni di uscita (un implicante può venire riutilizzato per più funzioni di uscita). Le AND e le OR sono realizzate con diodi opportunamente orientati e polarizzati. Questo sistema permette tempi di commutazione rapidi ed praticamente indipendenti dalla complessità della funzione codificata. Per programmare la PLA occorre bruciare fisicamente i diodi che non interessano e preservare quelli che realizzano le funzioni volute.

I cinque implicanti realizzati nella PLA dell'esempio sono (dall'alto al basso):

$$Y_1 = \sim I_4 I_5 \quad Y_2 = \sim I_2 \sim I_5 \quad Y_3 = I_2 \sim I_4 \quad Y_4 = I_3 \quad Y_5 = \sim I_1 \sim I_4 I_5$$

Le quattro funzioni realizzate dalla PLA dell'esempio sono:

$$Z_1 = Y_1 + Y_4 = \sim I_4 I_5 + I_3$$

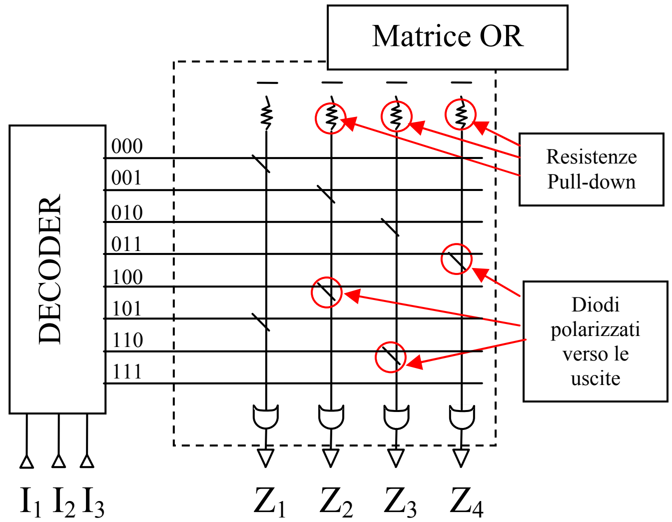
$$Z_2 = Y_3 + Y_5 = I_2 \sim I_4 + \sim I_1 \sim I_4 I_5$$

$$Z_3 = Y_2 + Y_5 = \sim I_2 \sim I_5 + \sim I_1 \sim I_4 I_5$$

$$Z_4 = Y_3 = I_2 \sim I_4$$

ROM: Read-Only Memory.

Sono realizzate attraverso un decodificatore di indirizzi ed una matrice OR. Praticamente realizzano funzioni logiche attraverso la forma tabellare.



Viene usata per sintetizzare funzioni notevolmente complesse per cui è necessario calcolare rapidamente il risultato. La programmazione avviene come per le PLA bruciando opportunamente i diodi nella matrice OR. Il decoder decodifica la configurazione degli ingressi ed attiva l'uscita corrispondente. Per ogni configurazione è quindi possibile decidere se la funzione deve valere 0 (diodo bruciato) oppure 1 (diodo presente). Il tempo di commutazione in questo caso è pressoché uguale a quello necessario per decodificare gli ingressi.

Le funzioni realizzate dalla ROM nell'esempio è:

I_1	I_2	I_3	Z_1	Z_2	Z_3	Z_4
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1
1	0	0	0	1	0	0
1	0	1	1	0	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	0