Algebra di Boole e reti combinatorie

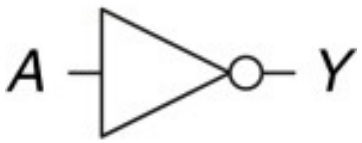
Porte logiche

- Un calcolatore può essere visto come un complesso sistema digitale che manipola e memorizza informazioni rappresentate in codice *binario*.
- I componenti digitali che costituiscono i mattoni fondamentali di un calcolatore sono le *porte logiche*
- Le porte logiche realizzano delle semplici operazioni che prendono uno o più input e producono un output
- Gli input sono indicati generalmente con le prime lettere dell'alfabeto A,B,C,D,… mentre gli output con le ultime X,Y,…
- Poiché gli input e gli output possono assumere in generale sia il valore 0 che 1 allora essi costituiscono delle variabili (dette variabili booleane)

Porta NOT

- La porta NOT restituisce in output il *complemento* dell'input:

NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

- Y è uguale a 1 se e solo se A non è uguale a 1: Y=1-A
- Altri simboli per NOT sono ¬A o ~A (usati soprattutto dai logici)

Porta AND

- La porta AND restituisce in output la *congiunzione* degli input:

AND

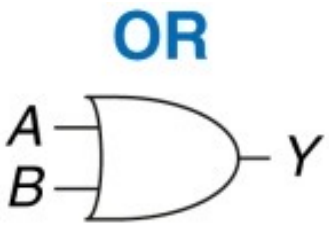
$Y = AB$

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

- Y è uguale a 1 se e solo se A e B sono entrambi uguali a 1
- L'operatore AND è anche rappresentato con A · B o A ∧ B

Porta OR

- La porta OR restituisce in output la *disgiunzione* degli input:



$$Y = A + B$$

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

- Y è uguale a 1 se e solo se A o B è uguale a 1
- L'operatore OR è anche rappresentato con A+B o A ∨ B

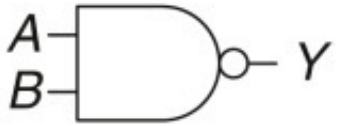
Altre porte logiche

XOR

$Y = A \oplus B$

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

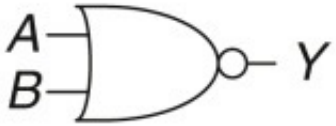
NAND



$Y = \overline{AB}$

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

NOR



$Y = \overline{A + B}$

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

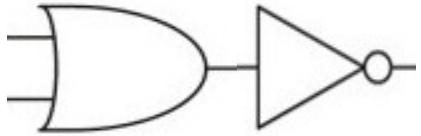
- XOR: Y = 1 se e solo se A oppure B è uguale a 1 (equiv. A≠B)
- NAND: Y è uguale a 1 se e solo se A o B non sono uguali a 1
- NOR: Y è uguale a 1 se e solo se A e B non sono uguali a 1

Altre porte logiche

- La porta NAND è un elemento circuitale che viene prodotto direttamente, tuttavia può essere ottenuto complementando una porta AND, ovvero mettendo *in serie* una porta AND e una NOT:

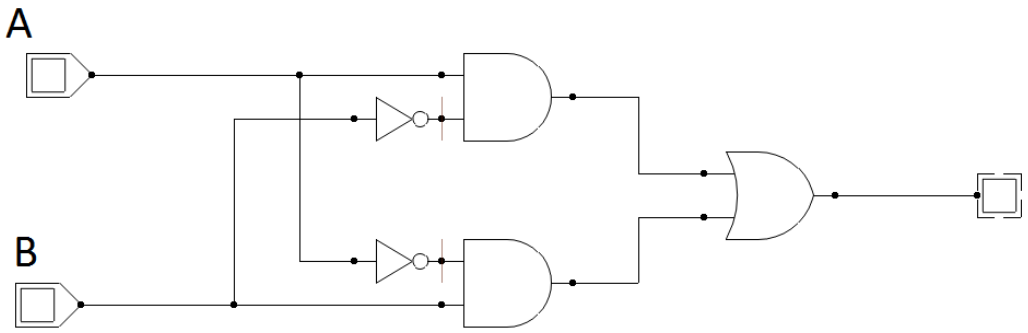
A	B	$A \cdot B$	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

- Analogamente la porta NOR si può ottenere mettendo *in serie* una porta OR e una NOT:



Altre porte logiche

- Anche la porta XOR può essere ottenuta mediante porte AND, OR e NOT.
  - XOR: Y = 1 sse A≠B
  - A≠B sse (A=1 e B=0) o (A=0 e B=1)
  - $Y=(A\cdot\overline{B})+(\overline{A}\cdot B)$



XNOR

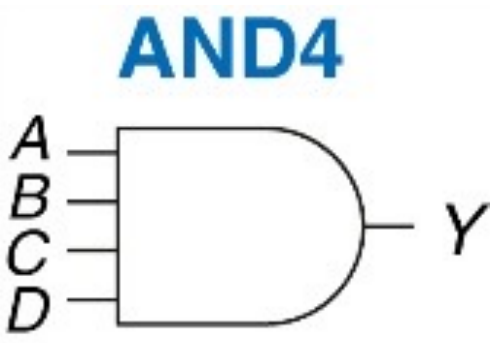
- Problema: quale è la tabella di verità della porta XNOR?

$$Y = \overline{A \oplus B}$$

A	B	$A \oplus B$	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Porte logiche con più linee di input

- Le porte logiche AND, OR, NAND,… possono avere anche più di 2 linee di ingresso.

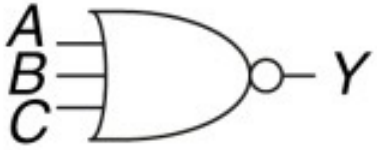


$Y = ABCD$

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

Porte logiche con più linee di input

NOR3



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

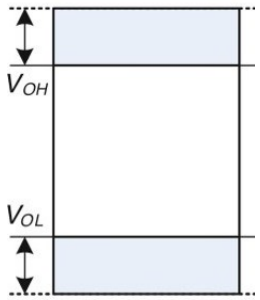
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Cosa c'è sotto?

- Finora abbiamo descritto le porte logiche in termini astratti in cui gli ingressi e le uscite assumono i valori binari 0 o 1.
- Chiaramente, per realizzare queste porte in concreto, questi valori devono corrispondere a una certa grandezza fisica.
- Nei calcolatori elettronici (questo laptop per intenderci) la grandezza fisica che *reifica* i valori logici 0 e 1 è il potenziale elettrico.
- Daremo per scontato che abbiate le nozioni minime di base su cosa sia il potenziale elettrico (o che almeno non mettiate le dita in una presa da 220V).
- Il valore logico 0 è rappresentato dal valore di potenziale di 0V (GRD)
- Il valore logico 1 è rappresentato da un valore di potenziale V_{DD} fornito dal generatore.
  - Fino agli anni ottanta V_{DD} = 5V, l'avvento di portatili, tablet e smarthphone ha reso necessario operare con valori di potenziale più bassi $V_{DD} \leq 1{,}5V$

Cosa c'è sotto?

- Notate che il potenziale elettrico è una grandezza fisica *continua* con cui rappresentiamo dei valori discreti (0 e 1) mediante le grandezze nominali GRD e V_{DD}.
- Come ogni sistema fisico reale, un circuito digitale è soggetto a del *rumore* che può alterare i valori nominali su cui operiamo. Per questo i componenti digitali hanno delle soglie di tolleranza GRD– V_{OL} e V_{OH}–V_{DD} .



- Per un corretto funzionamento di un circuito, il rumore non deve eccedere le soglie di tolleranza di tutti i suoi componenti.

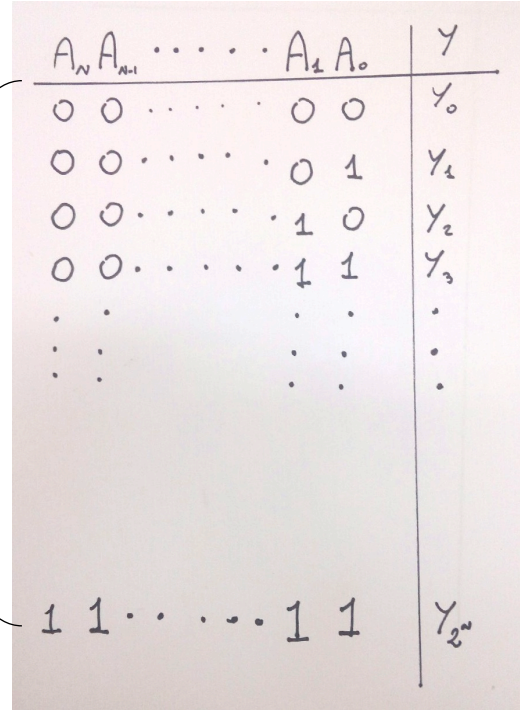
Funzioni booleane

- Le porte logiche esaminate finora costituiscono delle specifiche *funzioni booleane*

$$f:\{0,1\}^N \rightarrow \{0,1\}$$

- Domanda: quante funzioni booleane di N variabili esistono?

2^N righe



A_N	A_{N-1}	...	A_1	A_0	Y
0	0	...	0	0	Y_0
0	0	...	0	1	Y_1
0	0	...	1	0	Y_2
0	0	...	1	1	Y_3
⋮	⋮		⋮	⋮	⋮
1	1	...	1	1	Y_{2^N}

Una generica f assegna «liberamente» valori 0 o 1 alle Y_i
Quindi il numero di funzioni booleane di N variabili è pari al numero di parole binarie di lunghezza 2^N , ovvero:

$$2^{2^N}$$

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

AND

XOR

OR

NOR

XNOR

NAND

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Funzione costante
Y=0

Funzione costante
Y=1

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=A

Y=B

Y=$\bar{B}$

Y=$\bar{A}$

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=$A\bar{B}$

Y=$\bar{A}$B

Y= $\overline{\bar{A}\mathrm{B}}$ = A + $\bar{B}$ B≤A

Y= $\overline{\mathrm{A}\bar{B}}$ = $\bar{A} + B$ A≤B

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=A

Y= $\bar{B}$

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

Funzioni booleane di 2 variabili

Se le variabili sono $\mathbf{2}$ allora ottengo $\mathbf{2^4 = 16}$ possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=B

Y= $\bar{A}$

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

Funzioni booleane di 2 variabili

- Se le variabili sono 2 allora ottengo 16 possibili funzioni booleane

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

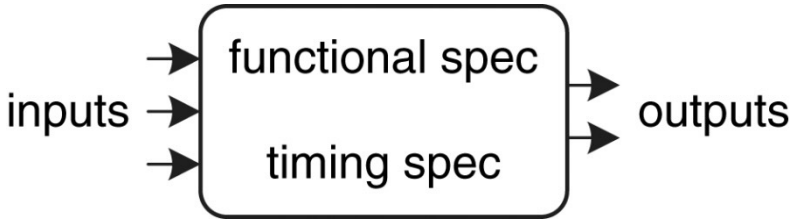
$Y = (A \leq B) \cdot (B \leq A)$ $A \equiv B$

$Y = B \leq A$

$Y = A \leq B$

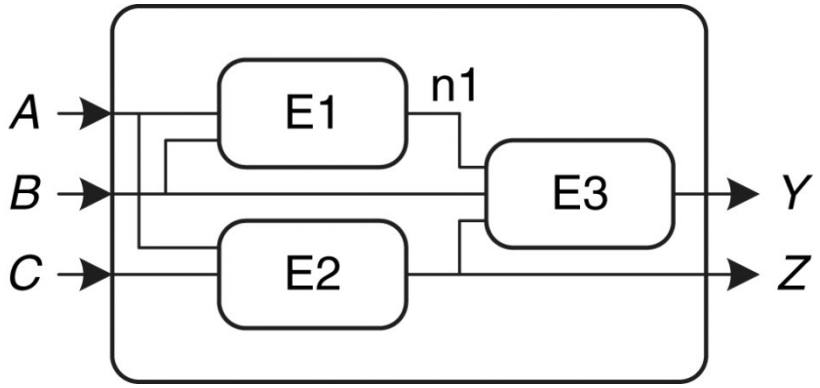
Circuiti digitali

- In generale un circuito digitale è una rete che elabora segnali discreti (rappresentati da variabili booleane). Prescidendo dalla sua configurazione interna, un circuito può essere visto come una *black-box* con
  - Uno o più input
  - Uno o più output
  - Una *specifica funzionale* che rappresenta la relazione fra input e output
  - Una *specifica temporale* che descrive il ritardo che intercorre affinché l segnali di input si propaghino nel circuito fino agli output.



Circuiti digitali

- La struttura interna di un circuito è composta da *elementi* e *nodi.*
- Un elemento è esso stesso un circuito digitale.
- Un nodo è una connessione che trasporta il segnale (e.g. cavo elettrico). I nodi si distinguono in nodi di input, output e interni
  - input: riceve il segnale dal mondo esterno
  - output: riporta il segnale al mondo esterno
  - Nodo interno: connette due elementi

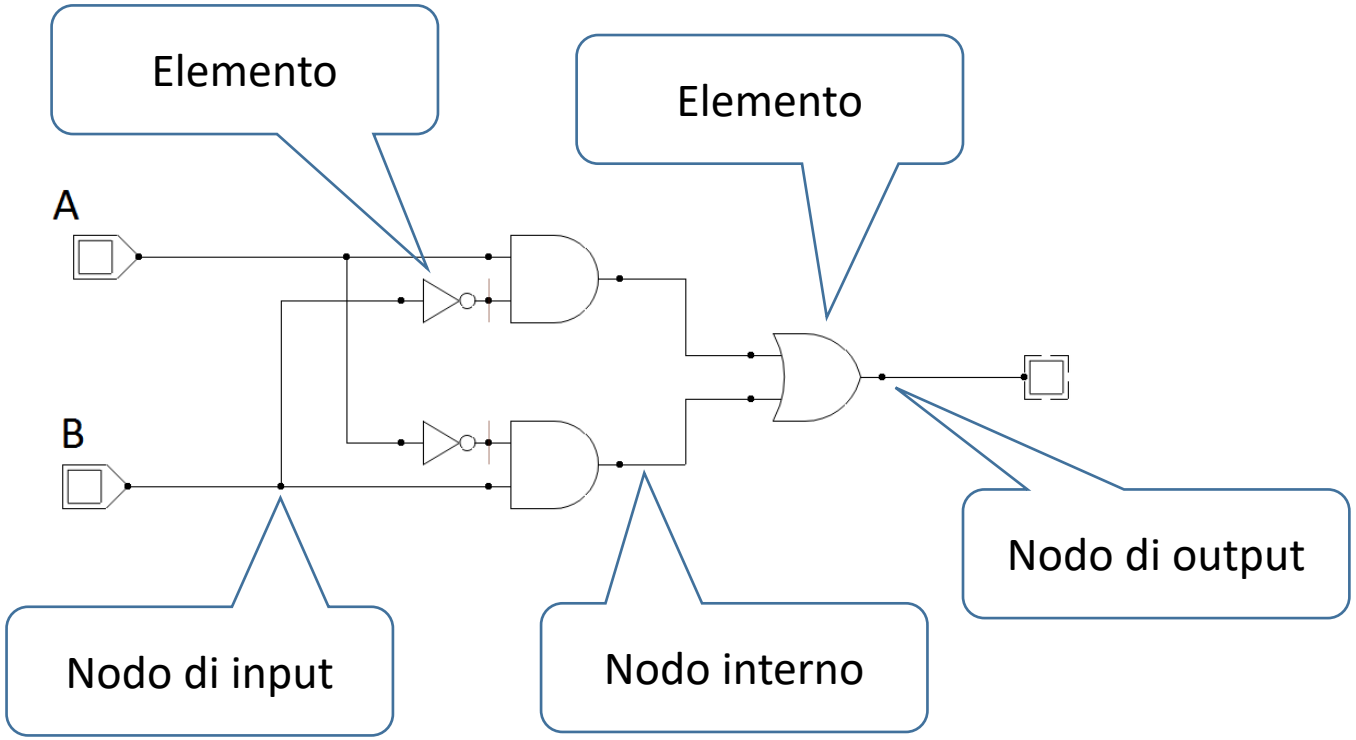


Reti combinatorie e sequenziali

- Vi sono due grandi categorie di circuiti digitali: le reti combinatorie e le reti sequenziali.
- In una rete combinatoria, i valori degli output dipendono esclusivamente dal valore corrente degli input (al netto dei ritardi di propagazione). In tal senso, le reti combinatorie si dicono memoryless, ovvero non hanno memoria della *"storia"* precedente del circuito
- In un rete sequenziale, invece, gli output dipendono non solo dal valore corrente degli input, ma anche dai valori precedenti. Si dice quindi che il circuito ha memoria.

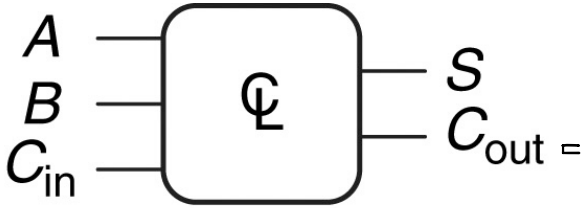
Reti combinatorie

- Le porte logiche viste finora sono un esempio di rete combinatoria.
- Un'altro esempio di rete combinatoria l'abbiamo visto in precedenza



Full adder

- Quando non siamo interessati alla struttura interna allora una rete combinatoria è descritta come segue

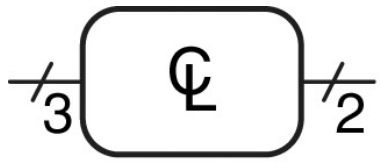


$$S = A \oplus B \oplus C_{in}$$
$$C_{out} = AB + AC_{in} + BC_{in}$$

- Questa rete combinatoria rappresenta un full adder di cui parleremo in seguito.
- Come avrete già capito, vi sono molti modi differenti per realizzare le funzioni che corrispondono al output S e C_{out}

Input e output multipli

- Per semplificare la rappresentazione grafica, si usa la notazione qui di fianco per indicare ingressi ed uscite multiple.
- In figura (a) un generico circuito combinatorio con 3 input e 2 output
- Il numero di input e output può essere omesso quando chiaro dal contesto o non rilevante. In figura (b) due generiche reti combinatorie in sequenza. Notate che il numero di output della prima rete deve essere uguale al numero di input della seconda.



(a)

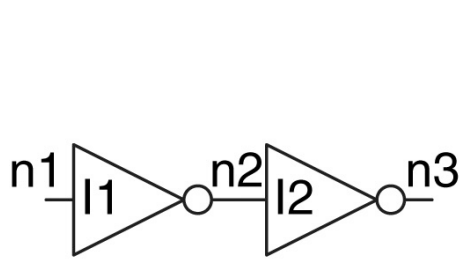
(b)

Regole di composizione di reti combinatorie

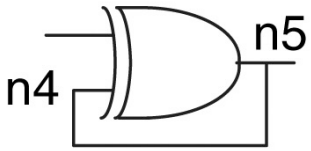
- Come abbiamo visto reti combinatorie possono essere composte per formare reti più grandi. Cosa ci garantisce che il risultato sia ancora una rete combinatoria? Ecco un insieme di regole che costituiscono delle condizioni sufficienti (ma non necessarie)
  - Ogni elemento è esso stesso una rete combinatoria
  - Ogni nodo che non è un input connette esattamente un output di un elemento
  - Il circuito non contiene cicli, ogni cammino interno alla rete visita un nodo al più una volta

Regole di composizione di reti combinatorie

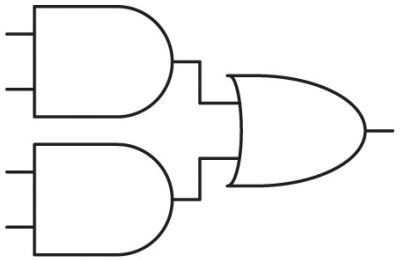
- Quali dei seguenti circuiti costituisce, secondo le regole di composizione, una rete combinatoria?



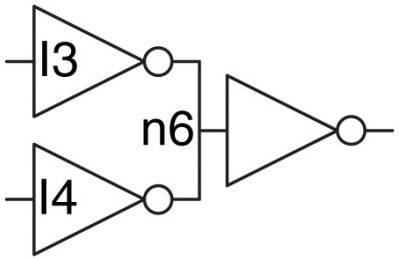
(a)



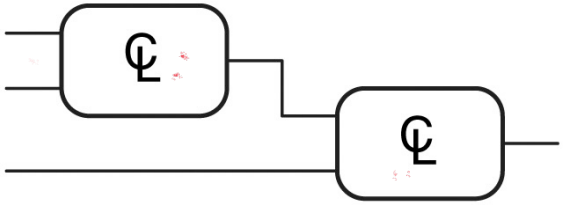
(b)



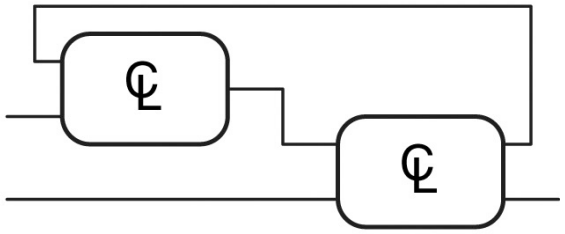
(c)



(d)



(e)



(f)

Tabella di verità e espressioni booleane

- Le specifiche funzionali di un circuito combinatorio sono descritte tramite tabelle di verità o espressioni booleane
- In generale, più espressioni booleane corrispondo alla stessa tabella di verità
  - Per esempio abbiamo visto che $\overline{A\bar{B}}$ e $\bar{A} + B$ corrispondono alla stessa tabella di verità, ovvero $\overline{A\bar{B}} = \bar{A} + B$.
- Come si ricava la tabella di verità di una data espressione?
- Viceversa, a partire da una tabella di verità, come si ricava una espressione corrispondente?

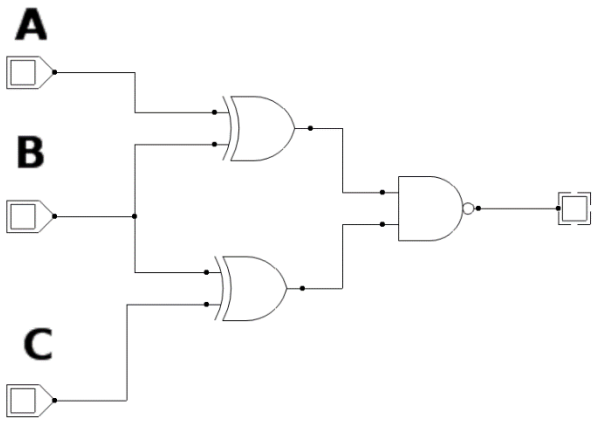
Dalle espressioni alle tabelle

- Il calcolo di una tabella di verità a partire una espressione booleana non è dissimile concettualmente dal calcolo di una espressione aritmetica. Si calcolano a ritroso le tabelle di verità delle sue sotto espressioni.
- Consideriamo l'espressione $(\bar{A}B)+(B\oplus C)$

A	B	C	$\bar{A}$	$\bar{A}\cdot B$	$B\oplus C$	$(\bar{A}B)+(B\oplus C)$
0	0	0	1	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	0	0	0	0

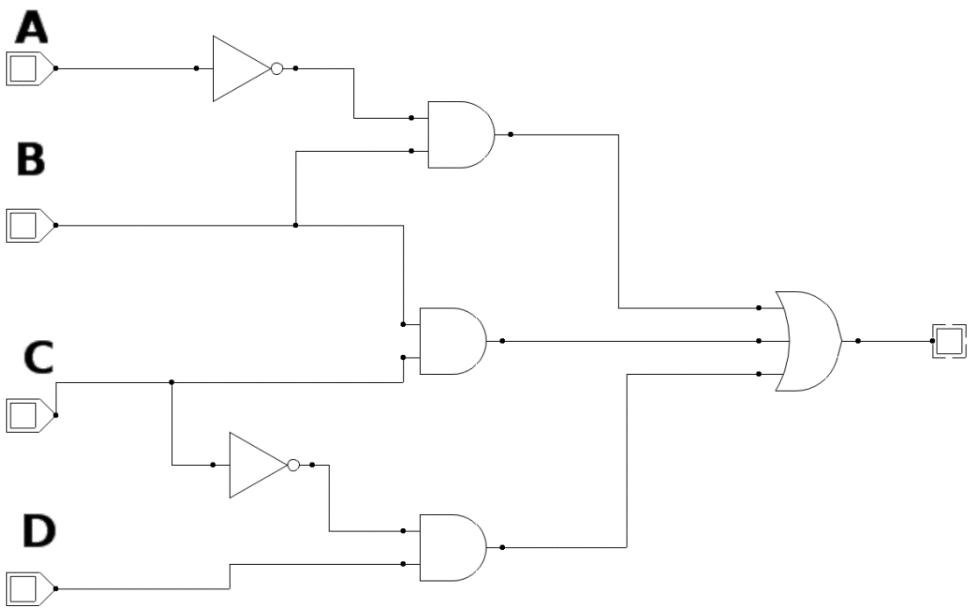
Esercizi

- Dimostrare che $(A \cdot \overline{B}) + (\overline{A} \cdot B)$ è equivalente alla porta XOR calcolandone la tabella di verità
- Scrivere l'espressione booleana e la relativa tabella di verità della rete:



Esercizi

- Scrivere l'espressione booleana e la relativa tabella di verità della rete:



Dalle tabelle alle espressioni

- ricavare da una tabella di verità un espressione booleana
- forma sintattica ben precisa: SOP (somma di prodotti)
- Definizioni preliminari:
  - Una variabile booleana A o la sua negata $\bar{A}$ sono detti litterali
  - Un prodotto (AND) di litterali è detto implicante: $\bar{A}$B , $\bar{A}$B$\bar{C}$, B sono tutti implicanti
  - Dato un insieme K di variabili booleane, un mintermine di K è un implicante che comprende un sola volta (positive oppure negate) tutte le variabili in K .
    - Ad esempio, se $\mathrm{K} = \{A, B, C\}$ allora $\bar{A}$B$\bar{C}$ è un mintermine per K. Invece, $\bar{A}$B è un implicante ma non un mintermine.
  - Analogamente, un maxtermine di K è una somma di litterali in cui occorrono tutte le variabili in K.
    - Ad esempio $\bar{A} + \bar{B} + \bar{C}$ è un maxtermine di $\mathrm{K} = \{A, B, C\}$.
- ordine di precedenza NOT →AND→OR
  - $\bar{A}$B$\bar{C}$ + CB = ($\bar{A}$B$\bar{C}$) + (CB)

La forma SOP

- Ognuna delle 2^N righe di una tabella di verità è caratterizzata da un mintermine

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=AB Y=$A\bar{B}$ Y=$\bar{A}$B Y= $\bar{A}\bar{B}$

La forma SOP

- I mintermini sono enumerati rigo dopo rigo a partire da 0,1 e così via. Quindi ogni mintermine è denotato dal numero binario della configurazione di input corrispondente.

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	0	$A\ B$	m_3

La forma SOP

- Quindi ad ogni tabella di verità corrisponde una espressione booleana ottenuta sommando tutti i mintermini per cui il valore dell'output Y è pari a 1

A	B	Y	minterm	minterm name
0	0	0	$\bar{A}\ \bar{B}$	m_0
0	1	1	$\bar{A}\ B$	m_1
1	0	0	$A\ \bar{B}$	m_2
1	1	1	$A\ B$	m_3

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

Y= $\Sigma(1,3)$

La forma SOP

- Consideriamo un esempio a tre variabili

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

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

Y= $\Sigma(0,4,5)$

La forma POS

- Una forma duale per rappresentare una funzione booleana è in forma POS (prodotto di somme).
- Ad ogni riga di una tabella di verità corrisponde un maxtermine che è uguale a 0 solo per quella riga

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=$A + B$

Y=$A + \bar{B}$

Y=$\bar{A} + B$

Y= $\bar{A} + \bar{B}$

La forma POS

- Anche i maxtermini sono enumerati come i mintermini.

A	B	Y	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

- La forma normale POS di una funzione booleana si ottiene come prodotto dei maxtermini per cui la funzione ritorna 0
  - Y=$(A + B) \cdot (\bar{A} + B)$
  - Y=$\prod(0,2)$

La forma POS

- Y=$(A+B)\cdot(\bar{A}+$B$)$

A B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0 0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0 1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Y=$(A+B)\cdot(\bar{A}+$B$)$ Y=$A+B$ Y=$\bar{A}+$B

Forme SOP & POS

SOP – sum-of-products

O	C	E	minterm
0	0	0	$\overline{O}\ \overline{C}$
0	1	0	$\overline{O}\ C$
1	0	1	$O\ \overline{C}$
1	1	0	$O\ C$

$$E = O\overline{C}$$
$$= \Sigma(2)$$

POS – product-of-sums

O	C	E	maxterm
0	0	0	$O + C$
0	1	0	$O + \overline{C}$
1	0	1	$\overline{O} + C$
1	1	0	$\overline{O} + \overline{C}$

$$E = (O + C)(O + \overline{C})(\overline{O} + \overline{C})$$
$$= \Pi(0, 1, 3)$$

Forme SOP & POS

- Se la tabella di verità ha pochi 1 allora la forma SOP è più succinta della forma POS
- Se la tabella di verità ha pochi 0 allora la forma POS è più succinta della forma SOP
- Nel caso in cui il numero di 0 e 1 è pressappoco lo stesso le due forme si equivalgono

Algebra di Boole

- Come abbiamo visto la medesima funzione può essere descritta da espressioni booleane distinte
- Alcune di queste possono essere più semplici di altre

A	B	Y	minterm	minterm name
0	0	0	$\bar{A}\ \bar{B}$	m_0
0	1	1	$\bar{A}\ B$	m_1
1	0	0	$A\ \bar{B}$	m_2
1	1	1	$A\ B$	m_3

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

Y= $(A + B) \cdot (\bar{A} + B)$

Y= B

- Come si fa con l'aritmetica, possiamo utilizzare un algebra per semplificare le espressioni

$$\frac{1}{x}(x + xy) \Longrightarrow \frac{x}{x}(1 + y) \Longrightarrow (1 + y)$$

Assiomi dell'algebra di Boole

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

Assiomi dell'algebra di Boole

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

- A1 e A1' ci dicono che il valore di una variabile booleana può essere 0 oppure 1

Assiomi dell'algebra di Boole

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

- A1 e A1' ci dicono che il valore di una variabile booleana può essere 0 oppure 1
- A2 e A2' definiscono l'operatore NOT (di fatto questi assiomi ripropongono la tabella di verità dell'operatore

Assiomi dell'algebra di Boole

	Axiom		Dual	Name
A1	$B = 0$ if $B \neq 1$	A1′	$B = 1$ if $B \neq 0$	Binary field
A2	$\overline{0} = 1$	A2′	$\overline{1} = 0$	NOT
A3	$0 \bullet 0 = 0$	A3′	$1 + 1 = 1$	AND/OR
A4	$1 \bullet 1 = 1$	A4′	$0 + 0 = 0$	AND/OR
A5	$0 \bullet 1 = 1 \bullet 0 = 0$	A5′	$1 + 0 = 0 + 1 = 1$	AND/OR

- A3, A4 e A5 definiscono l'operatore AND
- A3', A4' e A5' definiscono l'operatore OR
- Notate che ogni assioma «primato» si ottiene dal corrispondente non primato invertendo, da un lato, OR e AND e dall'altro gli 0 e 1. Questo è un principio generale detto principio di dualità.

Teoremi ad una variabile

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\overline{\overline{B}} = B$		Involution
T5	$B \bullet \overline{B} = 0$	T5′	$B + \overline{B} = 1$	Complements

Teorema B·1=B

Dimostrazione:

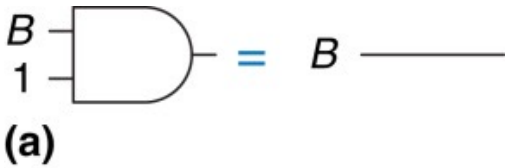
- Supponiamo B=1, allora B·1=1·1=1=B
- Supponiamo B=0, allora B·1=0·1=0=B

Teorema $B + \bar{B}$=1

Dimostrazione:

- Supponiamo B=1, allora $B + \bar{B}$=1+0=1
- Supponiamo B=0, allora $B + \bar{B}$=0+1=1

Corrispondenza hardware



(a)

(b)

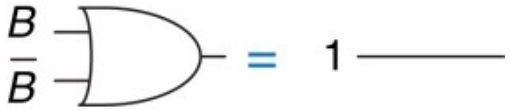
(a)

(b)

(a)

(b)

(a)



(b)

Teoremi più variabili

Number	Theorem	Name
T6	$B \bullet C = C \bullet B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	Distributivity
T9	$B \bullet (B+C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	Consensus

Teoremi più variabili: dualità

#	Theorem	Dual	Name
T6	$B \bullet C = C \bullet B$	$B+C = C+B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	$B + (C \bullet D) = (B+C)\ (B+D)$	Distributivity
T9	$B \bullet (B+C) = B$	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	$(B+C) \bullet (B+\overline{C}) = B$	Combining
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	$(B+C) \bullet (\overline{B}+D) \bullet (C+D) =$ $(B+C) \bullet (\overline{B}+D)$	Consensus

Principio di dualità: $+ \leftrightarrow \cdot \quad 1 \leftrightarrow 0$

Tecniche di dimostrazione

- Vi sono diversi metodi per dimostrare l'equivalenza di due espressioni.
  - Perfect induction: se le tabelle di verità di due espressioni coincidono allora le due espressioni sono equivalenti
  - usare assiomi e teoremi precedentemente provati per manipolare le espressioni fino ad ottenere espressioni uguali
  - Sfruttare la struttura delle espressioni e verificare che le tabelle di verità coincidono solo nei casi rilevanti

Perfect induction: commutativity

Number	Theorem	Name
T6	B•C = C•B	Commutativity

B	*C*	*BC*	*CB*
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Perfect induction: covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

B	*C*	*(B+C)*	*B(B+C)*
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Perfect induction: consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) =$ $(B \bullet C) + (\overline{B} \bullet D)$	Consensus

B	C	D	$BC + \overline{B}D + CD$	$BC + \overline{B}D$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

Limiti del perfect induction

- La tecnica del perfect induction è semplice ma «priva di intelligenza»
- Al crescere della lunghezza delle espressioni diventa sempre più laboriosa
- Al crescere delle variabili che occorrono nelle espressioni diventa estremamente più laboriosa
  - 4→16 checks
  - 5→32 checks
  - 6→64 checks
  - …

Proof teoretico: covering

Number	Theorem	Name
T9	B• (B+C) = B	Covering

B•(B+C)	= B•B + B•C	T8: Distributivity
	= B + B•C	T3: Idempotency
	= B•(1 + C)	T8: Distributivity
	= B•(1)	T2: Null element
	= B	T1: Identity

Proof teoretico: combining

Number	Theorem	Name
T10	$(B \bullet C) + (B \bullet \overline{C}) = B$	Combining

$B \bullet C + B \bullet \overline{C}$	$= B \bullet (C + \overline{C})$	T8: Distributivity
	$= B \bullet (1)$	T5': Complements
	$= B$	T1: Identity

Proof teoretico: consensus

Number	Theorem	Name
T11	$(B \bullet C) + (\overline{B} \bullet D) + (C \bullet D) = (B \bullet C) + (\overline{B} \bullet D)$	Consensus

$$B \bullet C + \overline{B} \bullet D + C \bullet D$$

$= BC + \overline{B}D + (CDB + CD\overline{B})$	T10: Combining
$= BC + \overline{B}D + BCD + \overline{B}CD$	T6: Commutativity
$= BC + BCD + \overline{B}D + \overline{B}CD$	T6: Commutativity
$= (BC + BCD) + (\overline{B}D + \overline{B}CD)$	T7: Associativity
$= BC + \overline{B}D$	T9': Covering

Per casi rilevanti

- Consideriamo T11

$$E: B \bullet C + \overline{B} \bullet D + C \bullet D$$

$$F: B \bullet C + \overline{B} \bullet D$$

Poiché $E = F + CD$, se F ritorna 1, allora E ritorna 1.
Occorre quindi dimostrare il contrario: se E ritorna 1, allora F ritorna 1.
Notate che E ritorna 1 sse (almeno) una fra BC, $B\overline{D}$, o CD ritornano 1.
Mostreremo allora che l'alternativa CD è superflua, ovvero che quando CD ritorna 1 almeno una $\bar{B}D$ e BC ritorna 1.
Supponiamo CD =1, per definizione questo vuol dire che C=1 e D=1. Quindi se B=0, allora $\bar{B}D = 1$, altrimenti BC=1.

Per casi rilevanti

B	C	D	$BC+\overline{B}D+CD$	$BC+\overline{B}D$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

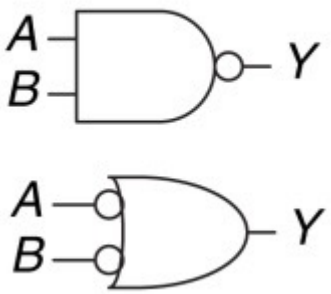
Teoremi di De Morgan

#	Theorem	Dual	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	$\overline{B_0 + B_1 + B_2 \ldots} = \overline{B_0} \bullet \overline{B_1} \bullet \overline{B_2} \ldots$	DeMorgan's Theorem

- La negata di un prodotto è uguale alla somma delle negate
- La negata di una somma è uguale al prodotto delle negate

Corrispondenza hardware

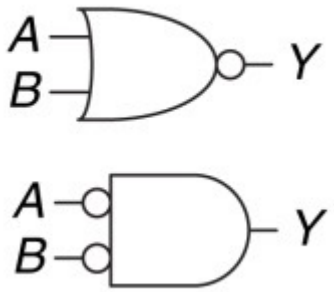
NAND



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

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

NOR



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

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

Teoremi di De Morgan

- I teoremi di De Morgan consentono di convertire un porta AND in una OR e viceversa

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

Esempio

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

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

$$= (\overline{A} \bullet (\overline{\overline{BD}})) + C$$

$$= (\overline{A} \bullet (BD)) + C$$

$$= \overline{A}BD + C$$

Altri tre teoremi fondamentali

- Sia E una espressione booleana che contiene una sotto espressione E_1 . Se $E_1 = E_2$, allora E=E' dove E' è ottenuto sostituendo E_1 con E_2.
  - **Esempio:** $BC + \overline{B}D + CD = BC + \overline{B}D$

    $(BC + \overline{B}D + CD)(FD + A) = (BC + \overline{B}D)(FD + A)$

- Se $E_1 = E_2$ allora $E'_1 = E'_2$. Dove E'_1 e E'_2 sono ottenuti sostituendo uniformemente una variabile booleana A con una qualsiasi espressione E.
  - **Esempio:** $A\overline{B} + AB = A$

    $AD\overline{C}\overline{B} + AD\overline{C}B = AD\overline{C}$

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

    $(CD + A)\overline{EF} + (CD + A)EF = (CD + A)$

- Se E=E' allora $\overline{E} = \overline{E'}$

Leggi di de Morgan e dualità

Alla luce del teorema precedente risulta chiaro che il principio di dualità è una conseguenza dei teoremi di De Morgan

- $B \bullet C = C \bullet B \rightarrow \overline{B} \bullet \overline{C} = \overline{C} \bullet \overline{B} \rightarrow \overline{\overline{B} \bullet \overline{C}} = \overline{\overline{C} \bullet \overline{B}} \rightarrow \overline{\overline{B}} + \overline{\overline{C}} = \overline{\overline{C}} + \overline{\overline{B}} \rightarrow B + C = C + B$
- $B \bullet 0 = 0 \rightarrow \overline{B} \bullet 0 = 0 \rightarrow \overline{\overline{B} \bullet 0} = \overline{0} \rightarrow \overline{\overline{B}} + \overline{0} = \overline{0} \rightarrow B + 1 = 1$

Teoremi di De Morgan e forme SOP/POS

I teoremi di De Morgan possono essere usati per ridurre una generica espressione E in forma SOP/POS senza «passare» per la tabella di verità

- Applica esaustivamente «De Morgan» per spingere la negazione nella struttura della formula
- Applica esaustivamente la proprietà distributiva dell'AND sull'OR (SOP)
- Applica esaustivamente la proprietà distributiva dell'OR sull'AND (POS)

Esempio

$$Y = \overline{(\overline{ACE}+\overline{D})} + B$$

$$= \overline{(\overline{ACE}+\overline{D})} \bullet \overline{B}$$

$$= (\overline{\overline{ACE}} \bullet \overline{\overline{D}}) \bullet \overline{B}$$

$$= ((\overline{\overline{AC}}+\overline{E}) \bullet D) \bullet \overline{B}$$

$$= ((AC+\overline{E}) \bullet D) \bullet \overline{B}$$

$$= (ACD + D\overline{E}) \bullet \overline{B}$$

$$= A\overline{B}CD + \overline{B}D\overline{E}$$

Da SOP a POS

I teoremi di De Morgan consentono anche di ricavare la forma POS a partire dalla SOP (e viceversa)

A	B	Y	$\bar{Y}$	minterm
0	0	0	**1**	$\bar{A}\ \bar{B}$
0	1	0	**1**	$\bar{A}\ B$
1	0	1	**0**	$A\ \bar{B}$
1	1	1	**0**	$A\ B$

$$\bar{Y} = \bar{A}\bar{B} + \bar{A}B$$

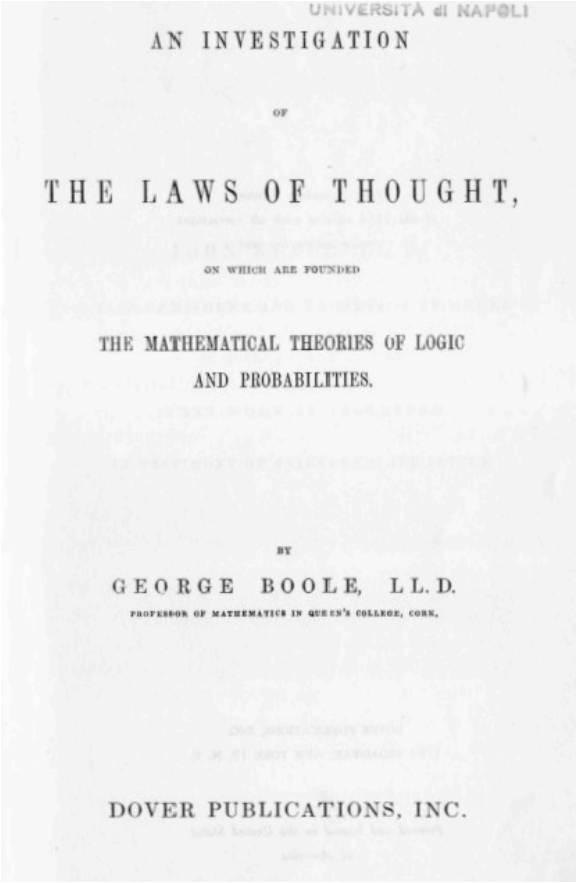
$$\bar{\bar{Y}} = \overline{\bar{A}\bar{B} + \bar{A}B} = \overline{\bar{A}\bar{B}} \cdot \overline{\bar{A}B} = (A + B) \cdot (A + \bar{B})$$

Completezza degli operatori

- Abbiamo visto che una qualsiasi funzione booleana $f:\{0,1\}^N \rightarrow \{0,1\}$ può essere espressa in forma SOP
- La forma SOP usa solo gli operatori NOT, AND, OR
- Questo vuol dire che questo insieme di operatori costituisce un insieme completo
- Per le leggi di de Morgan
  - $A \cdot B = \overline{\overline{A}+\overline{B}}$ (AND può essere espresso in termini di NOT e OR)
  - $A+B= \overline{\overline{A} \cdot \overline{B}}$ (OR può essere espresso in termini di NOT e AND)
- Quindi {NOT, AND, OR} è completo ma non minimale essendo {AND, NOT} e {OR, NOT} anche loro completi

Un po' di storia…

- L'algebra booleana prende nome dal matematico inglese George Boole
- L'oggetto di questa algebra non erano le porte logiche (che ancora non esistevano) ma *le leggi del pensiero e della deduzione*
- Infatti si può dare una interpretazione puramente logica delle espressioni booleane in cui
  - Le variabili corrispondono a delle proposizioni che possono essere vere (1) o false (0)
  - Gli operatori corrispondono alle relative preposizioni coordinate (e, o, non…)
  - «piove e ho l'ombrello» è vera sse «piove» e «ho l'ombrello» sono entrambe vere


UNIVERSITÀ di NAPOLI

AN INVESTIGATION

OF

THE LAWS OF THOUGHT,

ON WHICH ARE FOUNDED

THE MATHEMATICAL THEORIES OF LOGIC

AND PROBABILITIES.

BY

GEORGE BOOLE, LL.D.

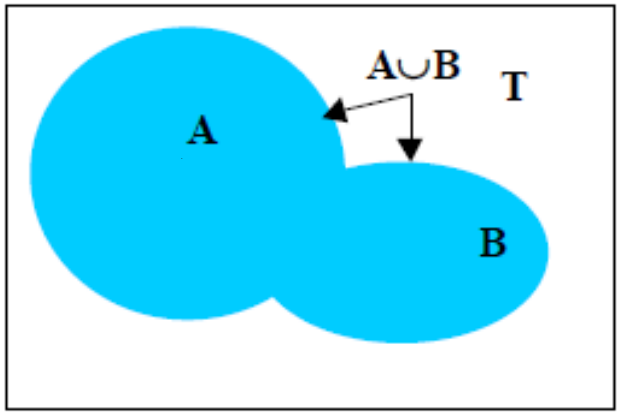
PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK.

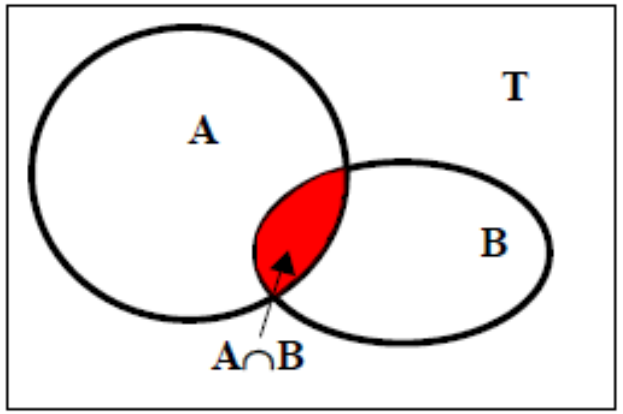
DOVER PUBLICATIONS, INC.

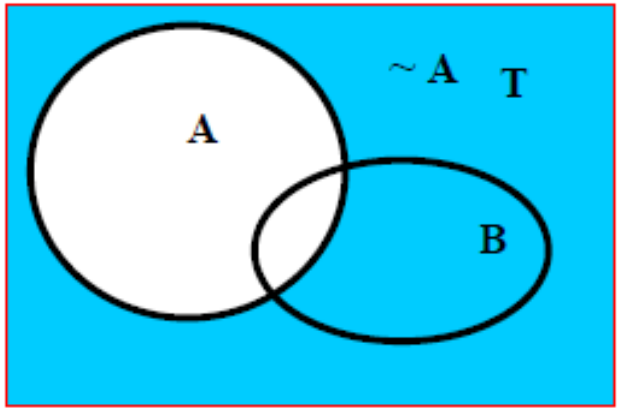
Operatori insiemistici

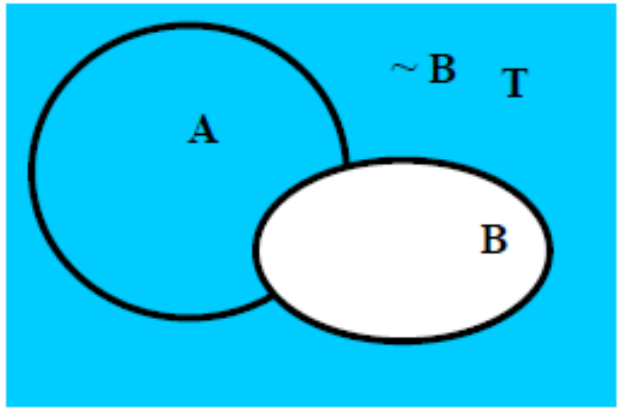
- Anche gli usuali operatori insiemistici di unione, intersezione e complemento formano un algebra di Boole
- T insieme universo
- $2^T=\{A \mid A \subseteq T\}$ insieme dei sottoinsiemi di T
- $\cap$ → AND
- $\cup$ → OR
- T/ → NOT
- 1 → T
- 0 → $\emptyset$

Operatori insiemistici

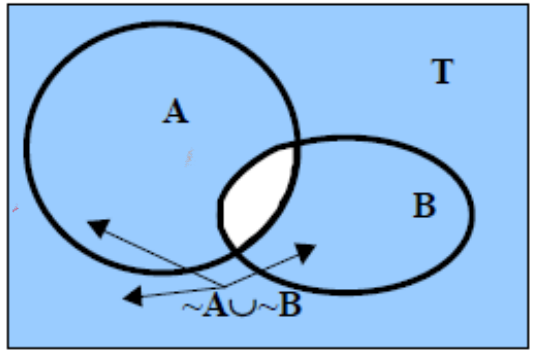






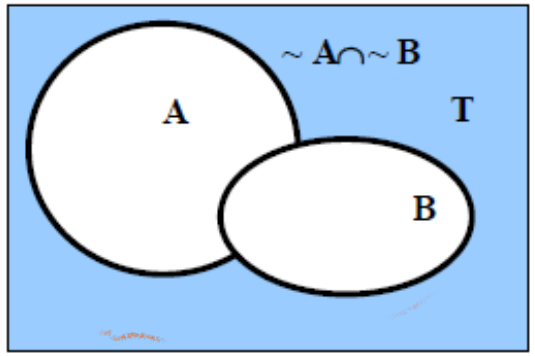


Operatori insiemistici



dal confronto con la figura relativa a **A∩B** si ricava

~A∪~B= ~ (A∩B)



dal confronto con la figura relativa a **A∪B** si ricava

~A∩~B= ~ (A∪B)

Esercizi

- Sfruttando i teoremi precedenti verificare le seguenti proprietà della porta XOR

$$a \oplus a = 0$$
$$a \oplus 0 = a$$
$$a \oplus 1 = \sim a, \text{ where } \sim \text{ is bit complement.}$$
$$a \oplus \sim a = 1$$
$$a \oplus b = b \oplus a \text{ (commutativity)}$$
$$a \oplus (b \oplus c) = (a \oplus b) \oplus c \text{ (associativity)}$$

Ricordarsi che, date due espressioni booleane E e F, $\mathrm{E} \bigoplus \mathrm{F} = E\bar{F} + \bar{E}F$

Esercizi

- Esercizi del libro “Digital Design and Computer Architecture – ARM edition”
  - 2.1
  - 2.2
  - 2.3
  - 2.4
- Verificare in maniera proof-teoretica che
  $\mathrm{a} \bigoplus \mathrm{b} = (\bar{a} + \bar{b})\ (a + b)$
- Dimostrare che la porta NAND è da sola un insieme completo (e minimale) di operatori booleani

$a \oplus a = 0$
$a \oplus 0 = a$
$a \oplus 1 = \sim a$, where $\sim$ is bit complement.
$a \oplus \sim a = 1$
$a \oplus b = b \oplus a$ (commutativity)
$a \oplus (b \oplus c) = (a \oplus b) \oplus c$ (associativity)
$a \oplus a \oplus a = a$
if $a \oplus b = c$, then $c \oplus b = a$ and $c \oplus a = b$.

$$- \quad a\bar{a} + \bar{a}a = a\bar{a} + a\bar{a} = a\bar{a} = 0$$

$$- \quad a\overline{(b\bar{c} + \bar{b}c)} + \bar{a}(b\bar{c} + \bar{b}c) = a(\overline{b\bar{c}} \cdot \overline{\bar{b}c}) + \bar{a}b\bar{c} + \bar{a}\bar{b}c =$$

$$= a\left[(\bar{b} + c) \cdot (b + \bar{c})\right] + \bar{a}b\bar{c} + \bar{a}\bar{b}c = a\left[\bar{b} \cdot (b + \bar{c}) + c(b + \bar{c})\right] +$$

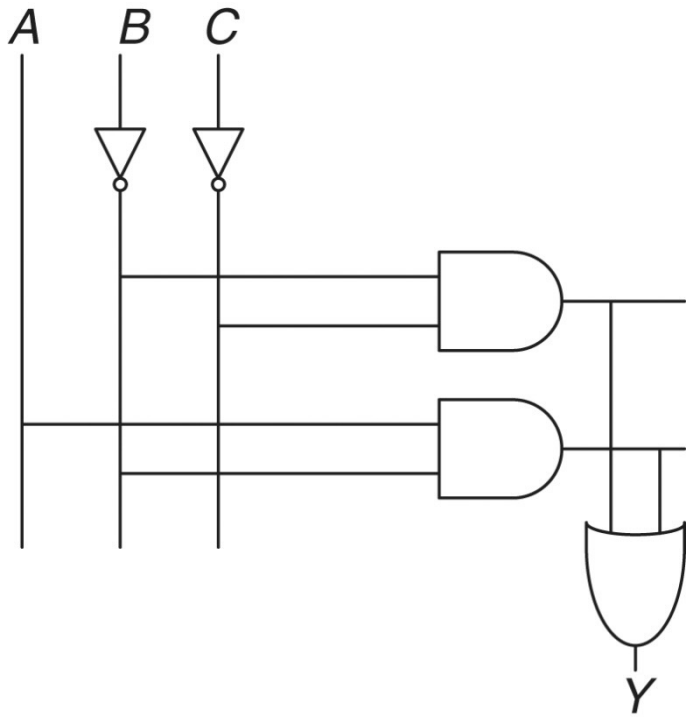
$$\bar{a}b\bar{c} + \bar{a}\bar{b}c$$

$$= a\bar{b}\bar{c} + abc + \bar{a}b\bar{c} + a\bar{b}c = (a\bar{b} + \bar{a}b)\bar{c} + (ab + \bar{a}\bar{b})c =$$

$$= (a\bar{b} + \bar{a}b)\bar{c} + \overline{(a\bar{b} + \bar{a}b)}\,c$$

Perché semplificare una espressione?

- Importanza della minimizzazione: utilizzare meno porte logiche
- Esercizio: mostrare che $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C = \bar{B}\bar{C} + A\bar{B}$



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

Esempio 1

$Y = A(AB + ABC)$

$= A(AB(1 + C))$	T8: Distributivity
$= A(AB(1))$	T2': Null Element
$= A(AB)$	T1: Identity
$= (AA)B$	T7: Associativity
$= AB$	T3: Idempotency

Esempio 2

$\mathbf{Y = \overline{A}BC + \overline{A}}$

$= \overline{A}$ T9’ Covering: $X + XY = X$

oppure

$= \overline{A}(BC + 1)$ T8: Distributivity

$= \overline{A}(1)$ T2’: Null Element

$= \overline{A}$ T1: Identity

Alcuni teoremi per semplificare espressioni

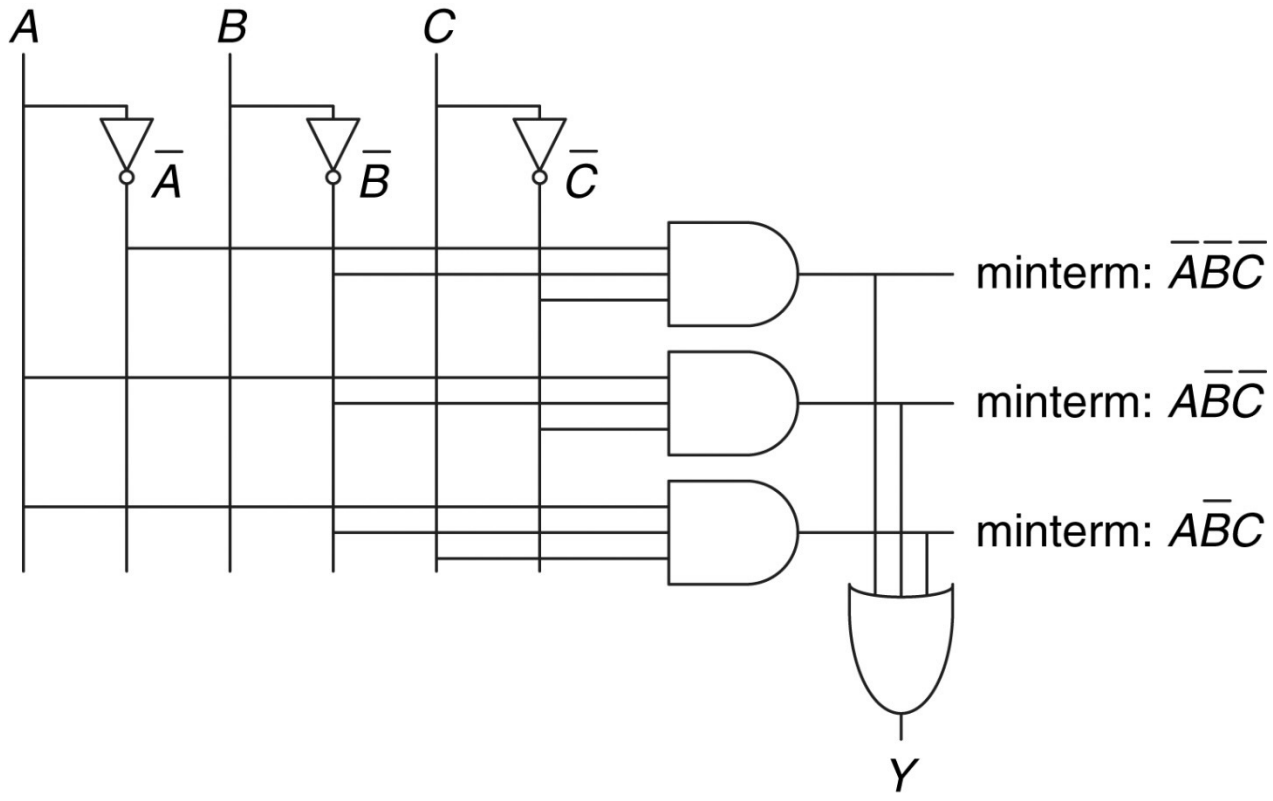
- Distributivity (T8, T8’) $B(C+D) = BC + BD$
  $B + CD = (B+C)(B+D)$
- Covering (T9’) $A + AP = A$
- Combining (T10) $P\overline{A} + PA = P$
- Expansion $P = P\overline{A} + PA$
  $A = A + AP$
- Duplication $A = A + A$
- “Simplification” theorem $P\overline{A} + A = P + A$
  $PA + \overline{A} = P + \overline{A}$

«Simplification theorem»

$$
\begin{aligned}
PA + \bar{A} &= PA + (\bar{\mathbf{A}} + \bar{\mathbf{A}}\mathbf{P}) && \text{T9' Covering} \\
&= PA + \mathbf{P}\bar{\mathbf{A}} + \bar{A} && \text{T6 Commutativity} \\
&= \mathbf{P}(\mathbf{A} + \bar{\mathbf{A}}) + \bar{A} && \text{T8 Distributivity} \\
&= P(\mathbf{1}) + \bar{A} && \text{T5' Complements} \\
&= P + \bar{A} && \text{T1 Identity}
\end{aligned}
$$

Schemi circuitali

- Ad ogni espressione booleana corrisponde in circuito combinatorio



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

Schemi circuitali SOP

- Le formule in forma SOP hanno degli schemi circuitali molto regolari:
  - Disegna una linea di input per ogni variabile che occorre positiva
  - Aggiungi delle linee con un NOT per ogni variabile che occorre negata
  - Per ogni mintermine disegna una porta AND e aggiungi in ingresso le linee che corrispondono ai relativi litterali
  - Collega tutte le uscite delle porte AND in un unico OR
- Per questo la forma SOP viene detta una logica a due livelli AND-OR

Semplificare formule SOP

- Per T10 $\mathrm{P}\bar{B} + \mathrm{P}B = P$ per ogni implicante P.
- Un implicante è detto *implicante primo* se non può essere combinato con altri implicanti della formula per ottenere un nuovo implicante con meno litterali.
- Una espressione SOP è minimale se tutti i suoi implicanti sono primi
- Minimizzazione: *ridurre il numero di implicanti e per ogni implicante ridurre il numero di litterali*

Semplificare formule SOP

- Nel minimizzare una SOP, può essere necessario «sdoppiare» un implicante allorché questo può essere ridotto in modi differenti.

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	
1	$\overline{B}\,\overline{C}(\overline{A}+A)+A\overline{B}C$	T8: Distributivity
2	$\overline{B}\,\overline{C}(1)+A\overline{B}C$	T5: Complements
3	$\overline{B}\,\overline{C}+A\overline{B}C$	T1: Identity

Non è minimale $A\bar{B}C$ e $A\bar{B}\bar{C}$ possono ridursi in $A\bar{B}$

Semplificare formule SOP

Step	Equation	Justification
	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	
1	$\overline{A}\,\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}\,\overline{C}+A\overline{B}C$	T3: Idempotency
2	$\overline{B}\,\overline{C}(\overline{A}+A)+A\overline{B}(\overline{C}+C)$	T8: Distributivity
3	$\overline{B}\,\overline{C}(1)+A\overline{B}(1)$	T5: Complements
4	$\overline{B}\,\overline{C}+A\overline{B}$	T1: Identity

Il numero di implicanti è lo stesso ma $A\overline{B}$ ha meno letterali di $A\overline{B}C$

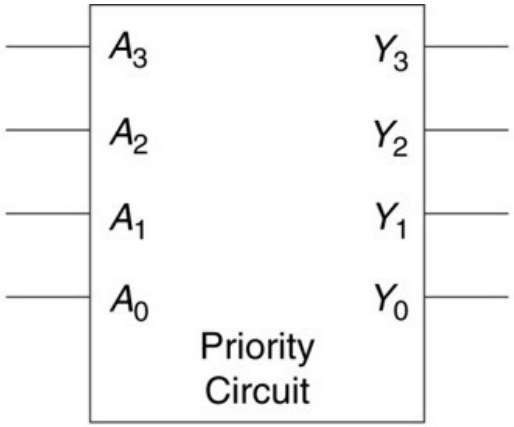
Semplificare le forme POS

- Come si semplifica una forma POS?
- La proprietà principale che si usa è il combining $(B+C) \cdot (B+\overline{C}) = B$
- Esempio:

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

Circuito a priorità

- I circuiti a priorità vengono utilizzati per assegnare una risorsa condivisa secondo un ordine di priorità fra chi ne fa richiesta
- A esempio posso avere 4 possibili richiedenti con priorità 3>2>1>0
- Gli input A_0 ,…, A_3 rappresentano le richieste della risorsa
- Gli output Y_0 ,…, Y_3 rappresentano a chi assegnata la risorsa, di volta in volta uno solo di essi sarà uguale a 1



Circuito a priorità

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Don’t cares

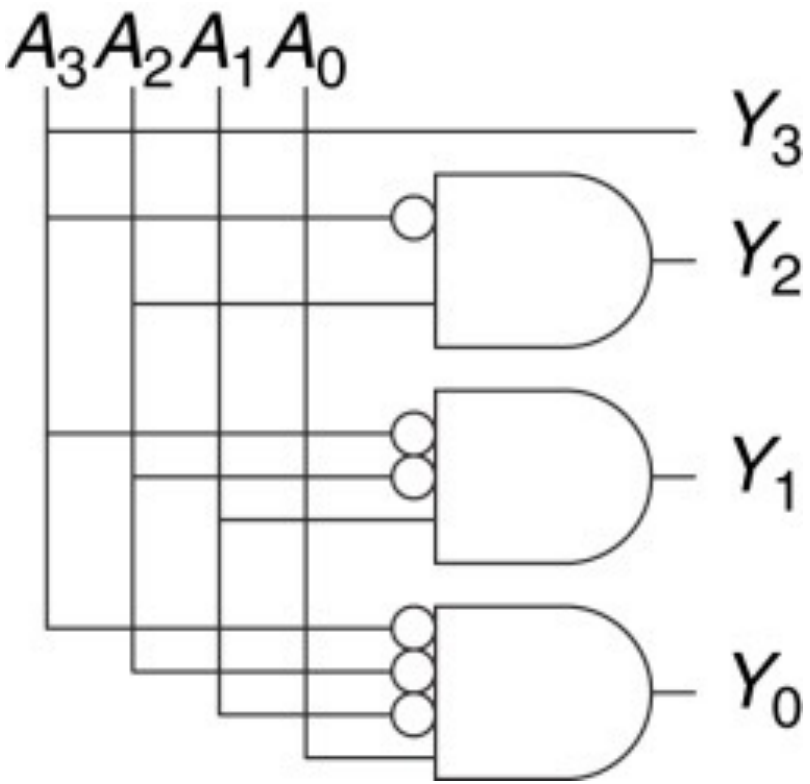
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

don’t cares

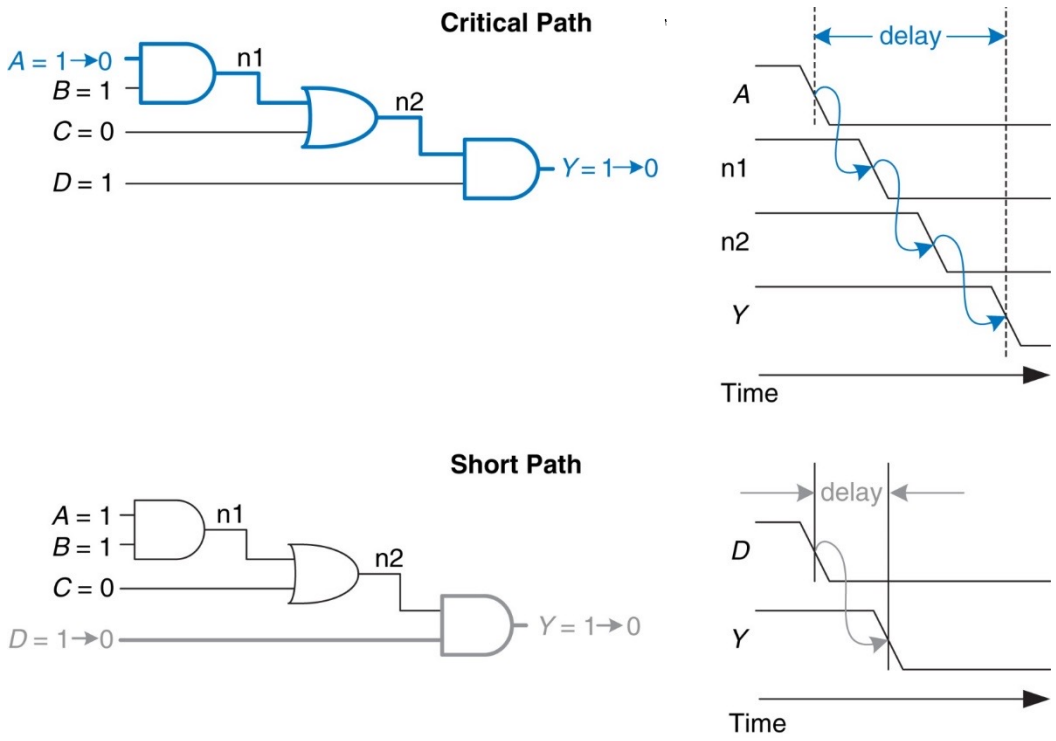
Don't cares

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0



Vantaggi delle SOP/POS

- La logica a 2 livelli della forma SOP presenta dei vantaggi, ad esempio, nei tempi di propagazione



Critical path: è il path da un input verso un output che attraversa il maggior numero di porte logiche e quindi induce il ritardo massimo nel circuito combinatorio

Limiti delle forme SOP/POS

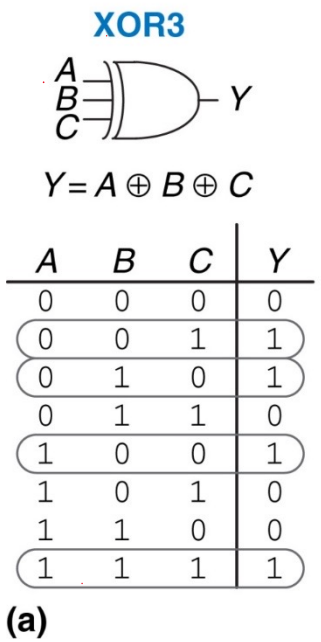
- Tuttavia alcune funzioni booleane, poste in forma SOP, sono estremamente poco succinte e quindi richiedono un numero considerevole di porte
- Presa la tabella di verità di funzione booleana di n variabili:
  - Se il numero di 1 nella colonna di output è piccolo allora la forma SOP è succinta
  - Se il numero di 0 nella colonna di output è piccolo allora la forma POS è succinta
  - Se il numero di 0 e 1 è più o meno lo stesso? Problema: avrò circa 2^{n-1} mintermini/maxtermini

Limiti delle forme SOP/POS

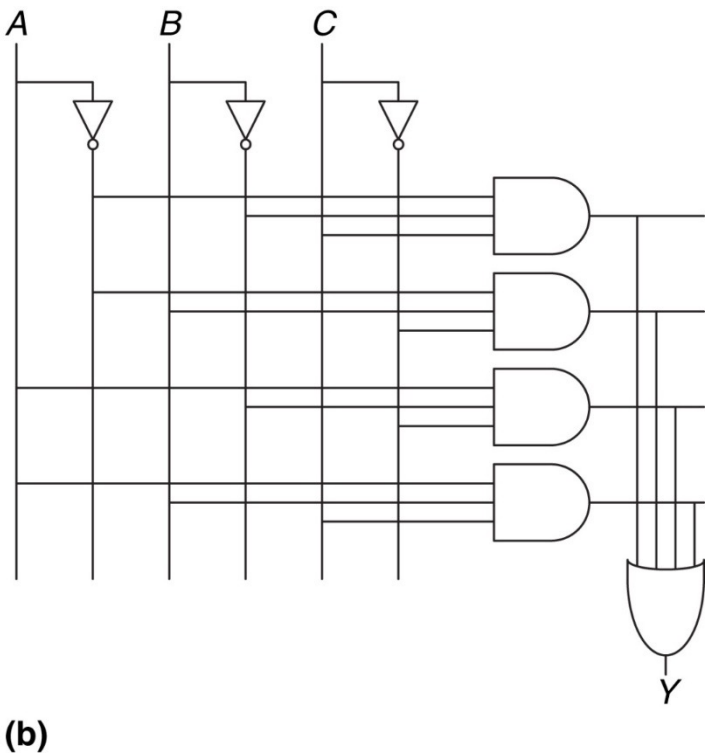
- Considerimo ad esempio uno XOR a più variabili
- Y=1 sse il numero di input uguali a 1 è dispari
- XOR3 in forma SOP

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

- XOR8 richiede 128 AND8 e un OR128

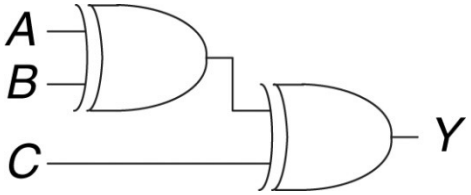


A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

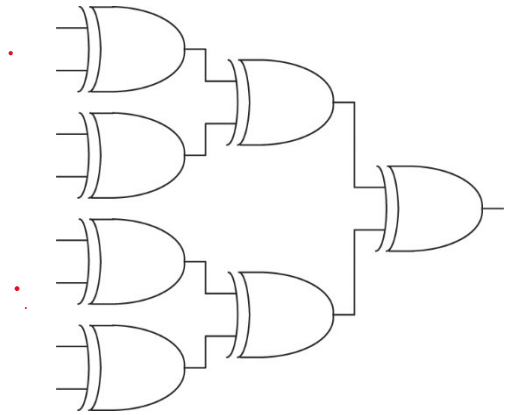


Logiche multilivello

- Per ridurre il numero di porte logiche a volte è necessario ricorrere ad una logica multilivello
- Ad esempio è facile verificare che $A \oplus B \oplus C = (A \oplus B) \oplus C$



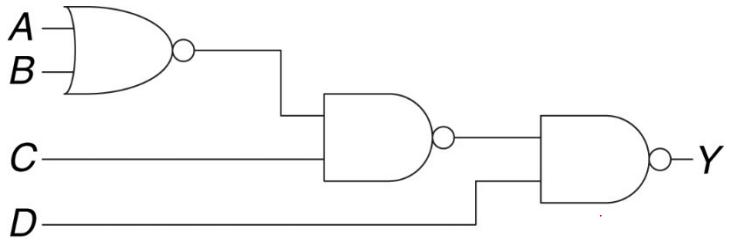
- Analogamente per XOR8:

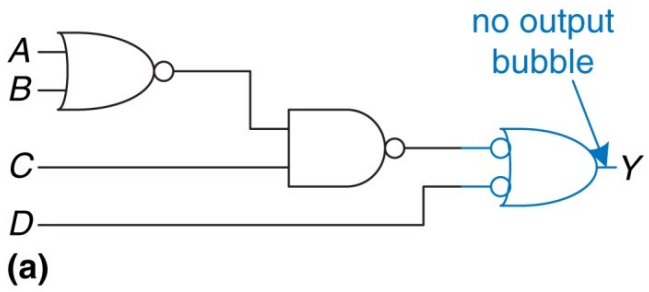


Bubble pushing

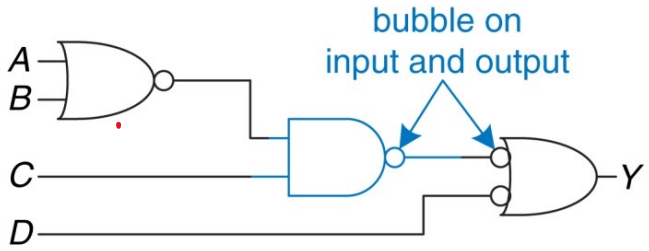
- Usando logiche multilivello e porte NAND/NOR a volte rende difficile capire quale funzione booleana un circuito realizza a causa delle molte negazioni annidate
- Per avere una espressione un po' più leggibile si possono applicare le leggi di De Morgan
- Il bubble pushing è una tecnica che applica sistematicamente le leggi di De Morgan e la legge della doppia negazione per eliminare le negazioni annidate
- Partendo dall'output Y si applicano a ritroso le leggi di De Morgan in modo che l'input e l'output di ogni nodo siano entrambi positivi o negati

Bubble pushing

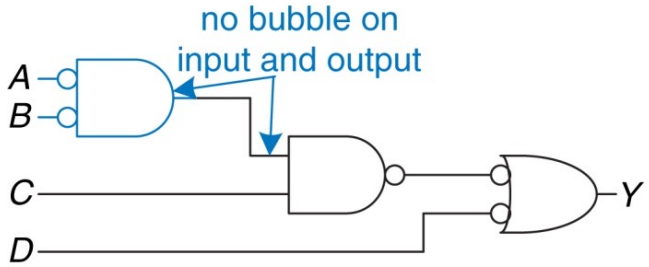




(a)

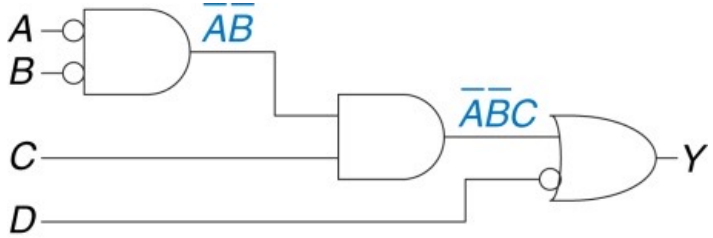


(b)



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

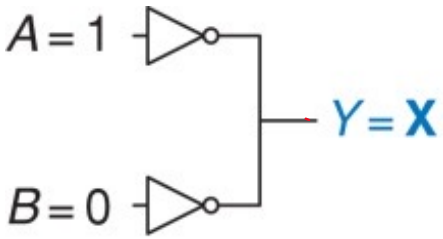
(c)



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

Valori illegali

- Considerate il circuito in figura. Quale sarà il valore di Y?

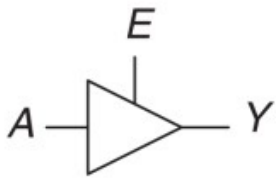


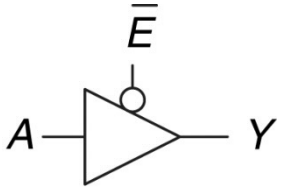
- Il valora del potenziale è portato ad essere contemporaneamente sia 0 che V_{DD}
- Questo tipo di configurazioni vengono dette *illegali*. Il valore di Y dipenderà dalla natura fisica delle porte.
  - Stabile ma indeterminato
  - Oscillare fra i valori 0 e V_{DD}
  - Avere un valore intermedio
- Configurazioni del genere possono indurre ad una forte dissipazione che riscalda e danneggia il circuito (corto circuito)

Stato di alta impedenza

- Un nodo, oltre ad essere in uno stato 0/1, può anche essere in un stato Z *floating* o di alta impedenza (alta impedenza=no passaggio di corrente)
- Ad esempio un ingresso di una porta logica non collegato è nello stato Z
- Gli stati di alta impedenza vengono utilizzati per disconnettere una parte di un circuito dal resto. Per questo si utilizzano i tristate buffer

Tristate Buffer



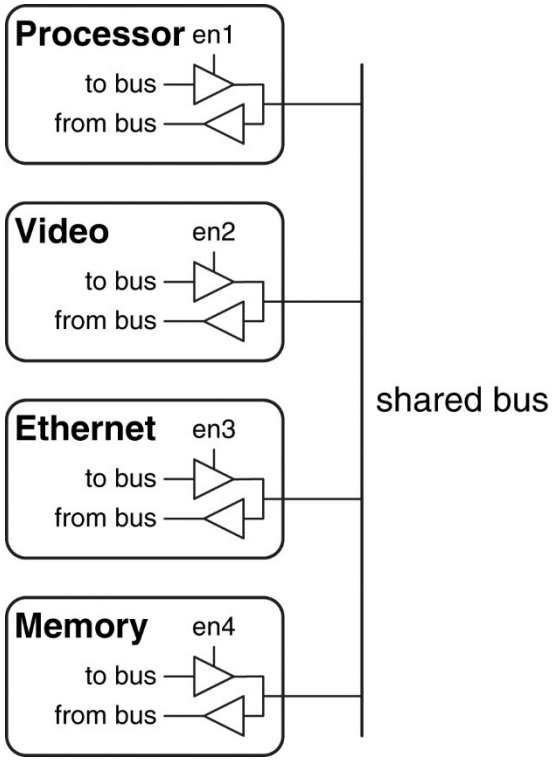


E	A	Y
0	0	Z
0	1	Z
1	0	0
1	1	1

$\overline{E}$	A	Y
0	0	0
0	1	1
1	0	Z
1	1	Z

Bus condiviso

- I tristate buffers sono usati in bus che connettono diversi chip
- Nell'esempio processore, scheda video e controller ethernet devono poter comunicare con la memoria centrale.
- Tuttavia, per evitare stati illegali, solo un componente alla volta può «immettere» segnali sul bus
- Gli altri componenti quindi devono essere temporaneamente disconnessi tramite un tristate buffer



Esercizi

- Esercizi 2.13 – 2.15 – 2.16 – 2.17 – 2.25 – 2.26 – 2.27

Mappe di Karnaugh

- Le mappe di Karnaugh sono un metodo per semplificare espressioni booleane in forma SOP
- In realtà non introducono tecniche di semplificazione nuove, sono semplicemente un espediente grafico che consente di rilevare più facilmente implicati che possono essere semplificati
- Quindi alla base delle mappe di Karnaugh c'è il solito principio:

$$PA + P\overline{A} = P$$

Mappe di Karnaugh

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

(a)

C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

(b)

C \ AB	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

(c)

C \ AB	00	01	11	10
0	1	0	0	0
1	1	0	0	0

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

Mappe di Karnaugh

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

(a)

Gray code

Y \ AB, C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

(b)

Y \ AB, C	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

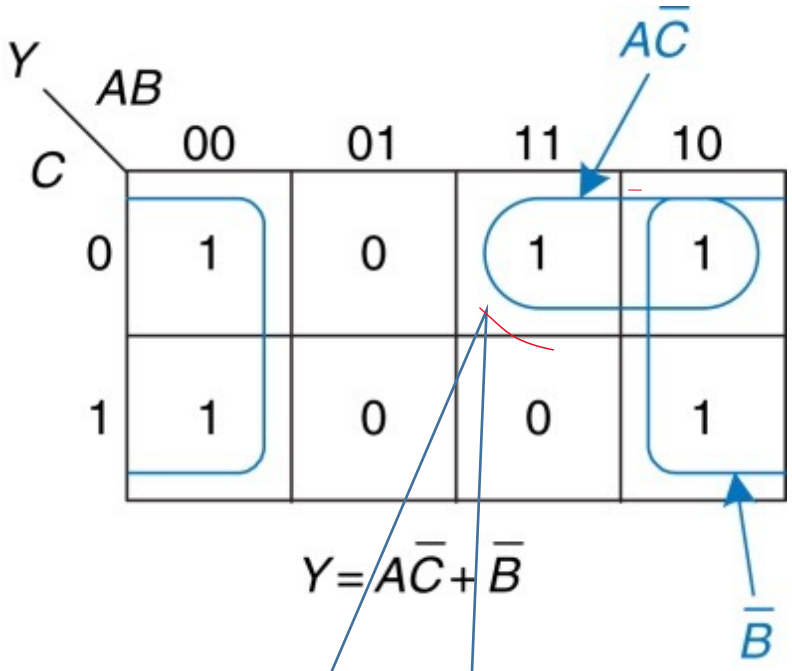
(c)

Y \ AB, C	00	01	11	10
0	1	0	0	0
1	1	0	0	0

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

Y \ AB, C	00	01	11	10
0	1	0	1	1
1	1	0	0	1

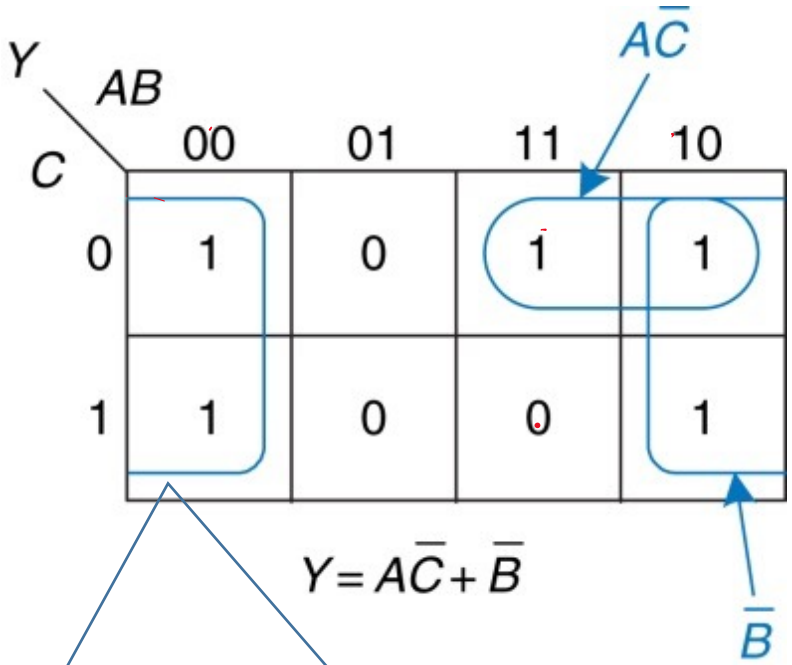
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



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

$$AB\bar{C} + A\bar{B}\bar{C} = A\bar{C}(B + \bar{B}) = A\bar{C}$$

Y \ AB	00	01	11	10
C = 0	1	0	1	1
C = 1	1	0	0	1

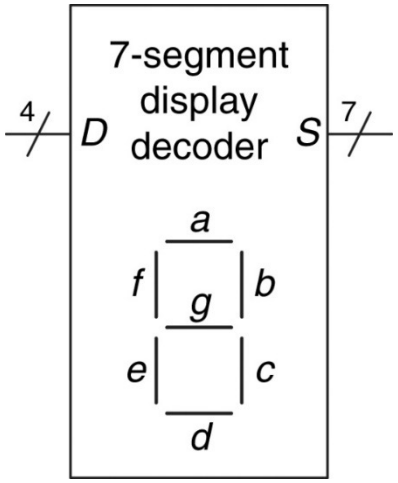


$$\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C = \bar{A}\bar{B}(\bar{C} + C) + A\bar{B}\,(\bar{C} + C) = \bar{A}\bar{B} + A\bar{B} = (\bar{A} + A)\,\bar{B} = \bar{B}$$

Regole di minimizzazione con K-maps

- Usare il minimo numero di *bolle* necessari per ricoprire tutti gli 1
- Tutte le caselle sottese ad una bolla devono contenere un 1
- Ogni bolla ricopre un blocco di caselle che è una potenza di 2: 1,2,4,…
  - In una bolla le variabili che non assumono valore costante devono assumere entrambi i valori 0 e 1
- Ogni bolla deve essere la più larga possibile
- Una bolla può «trasbordare» i bordi della K-map
  - La K-map ha una struttura *toroidale*
- Una casella può essere ricoperta da più bolle se questo permette un numero inferiore di bolle

Display a 7 segmenti



Display a 7 segmenti

$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

Display a 7 segmenti

S_a ($D_{1:0}$ \ $D_{3:2}$)	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	0

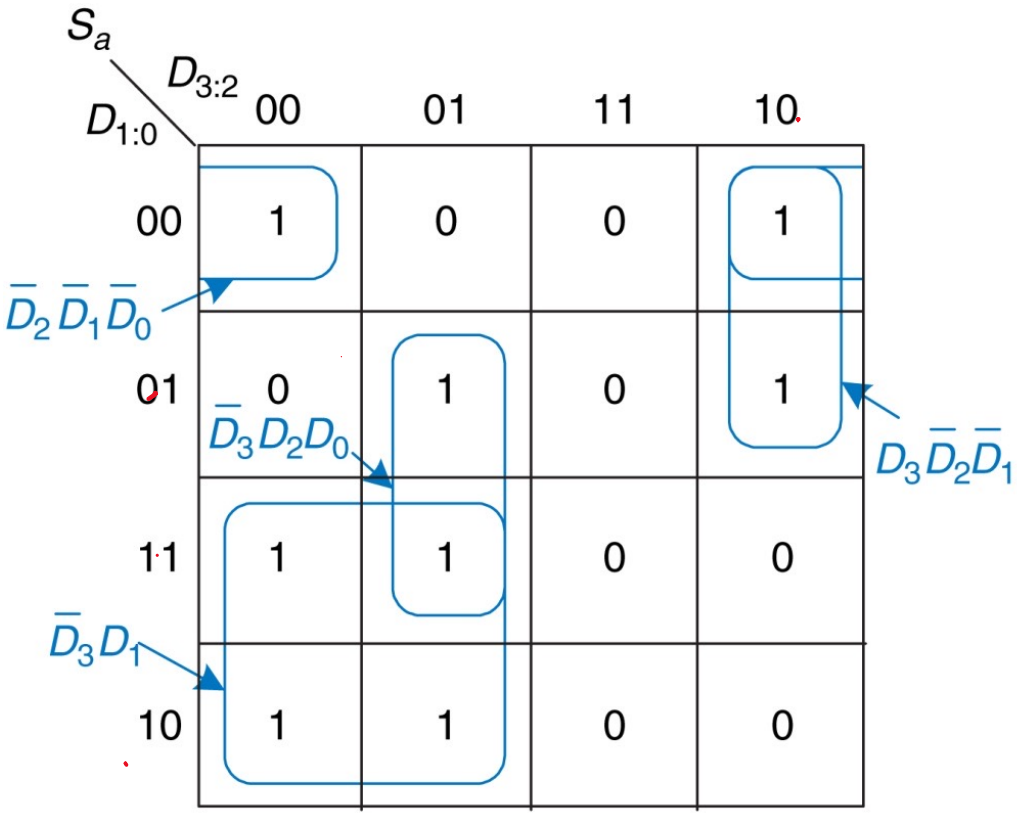
$D_{3:0}$	S_a	S_b	S_c	S_d	S_e	S_f	S_g
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	1	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1
others	0	0	0	0	0	0	0

Display a 7 segmenti

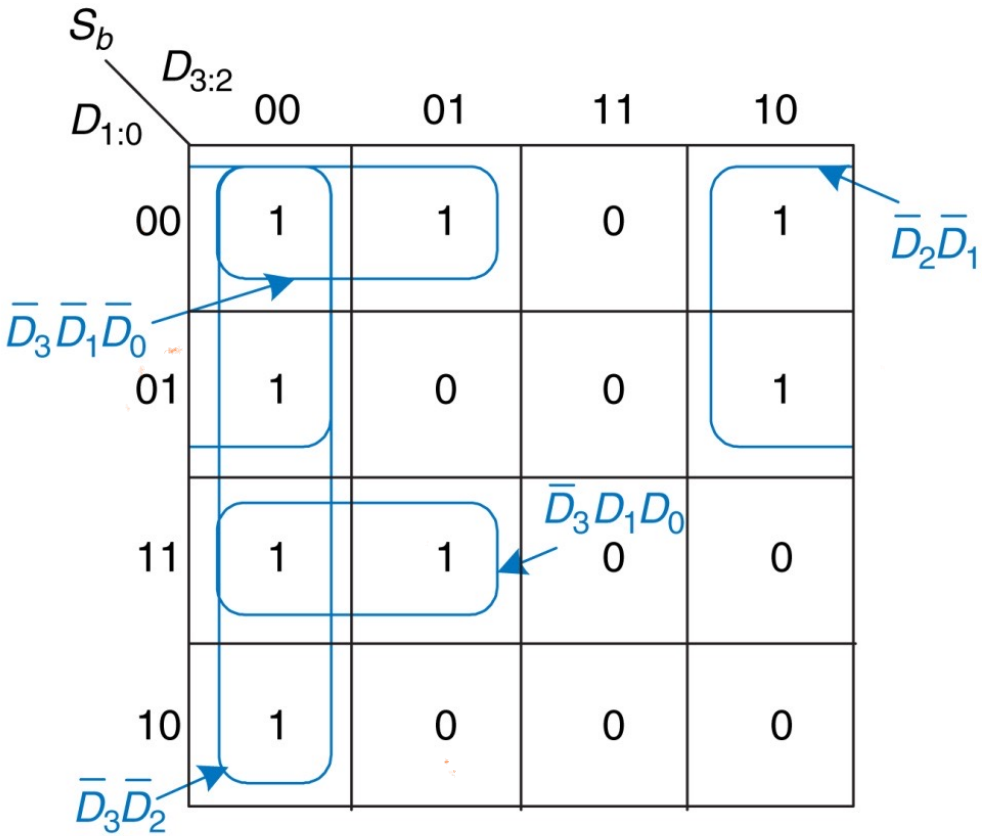
S_a $D_{1:0}$ \ $D_{3:2}$	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	0

S_b $D_{1:0}$ \ $D_{3:2}$	00	01	11	10
00	1	1	0	1
01	1	0	0	1
11	1	1	0	0
10	1	0	0	0

Display a 7 segmenti



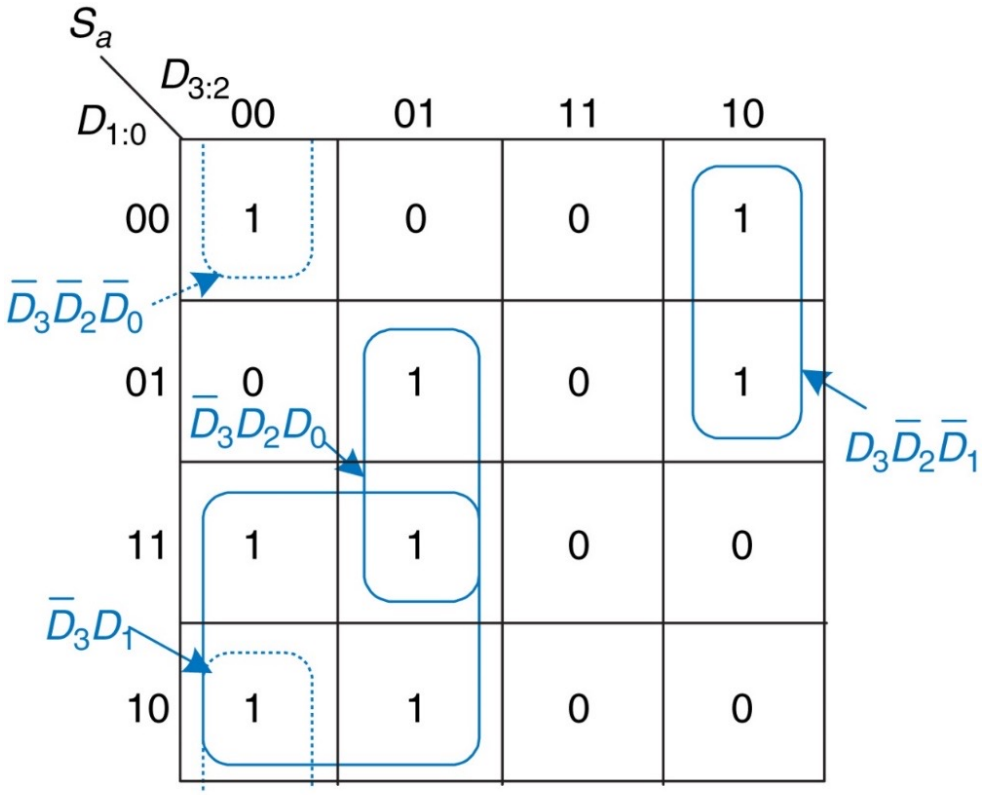
$$S_a = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_2\overline{D}_1\overline{D}_0$$



$$S_b = \overline{D}_3\overline{D}_2 + \overline{D}_2\overline{D}_1 + \overline{D}_3D_1D_0 + \overline{D}_3\overline{D}_1\overline{D}_0$$

Display a 7 segmenti

In generale, vi possono essere diverse forme minime:



$$S_a = \overline{D}_3D_1 + \overline{D}_3D_2D_0 + D_3\overline{D}_2\overline{D}_1 + \overline{D}_3\overline{D}_2\overline{D}_0$$

Don’t care in output

- Abbiamo visto nei circuiti a priorità come si possano utilizzare degli input «don’t care» per avere una rappresentazione della tabella di verità più succinta
- Valori «don’t care» (X) si possono avere anche in output allorché per una data configurazione degli input il valore dell’output è irrilevante (non ci interessa se sia 0 o 1)
- Ad esempio nel display a sette segmenti per gli input «illegali» 10-15 l’output può essere di tipo X
- Poiché non ci interessa se un valore X corrisponde in realtà ad uno 0 o un 1 allora nella minimizzazione di una espressione possiamo scegliere che valore dargli come più opportuno.
- In particolare in una K-map sostituiremo i valori X con degli 1 se questo consente di avere un numero inferiore di cerchi o cerchi più larghi

K-map con valori «don't care»

S_a \ $D_{3:2}$ / $D_{1:0}$	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	1	X	X
10	1	1	X	X

$$S_a = D_3 + D_2 D_0 + \bar{D}_2 \bar{D}_0 + D_1$$

S_b \ $D_{3:2}$ / $D_{1:0}$	00	01	11	10
00	1	1	X	1
01	1	0	X	1
11	1	1	X	X
10	1	0	X	X

$$S_b = \bar{D}_2 + D_1 D_0 + \bar{D}_1 \bar{D}_0$$

Esercizi su K-maps

AB / CD	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	1

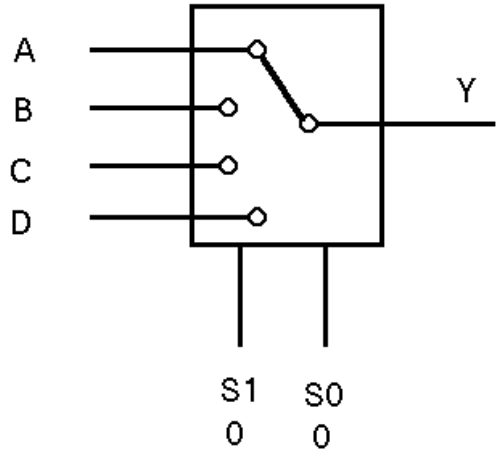
AB / CD	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	1	1	1	1
10	1	0	0	1

AB / CD	00	01	11	10
00	1	0	0	0
01	0	1	1	1
11	1	1	1	1
10	1	0	0	0

AB / CD	00	01	11	10
00	0	1	1	0
01	1	0	0	1
11	1	0	0	1
10	0	1	1	0

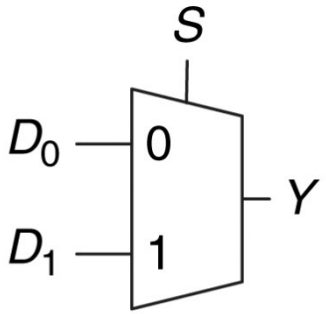
Multiplexer

- Un multiplexer è sostanzialmente un selettore di linea

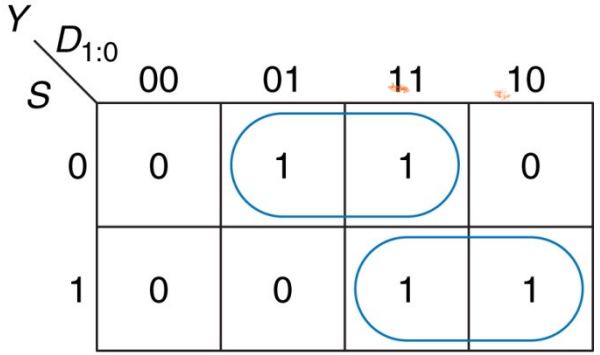


- In generale è costituito da N ingressi (dove N è una potenza di 2), 1 uscita, e $\log_2 N$ linee di selezione che indicano a quale ingresso deve corrispondere l'output

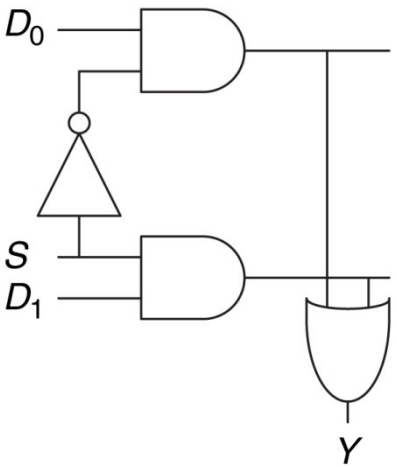
Multiplexer 2:1

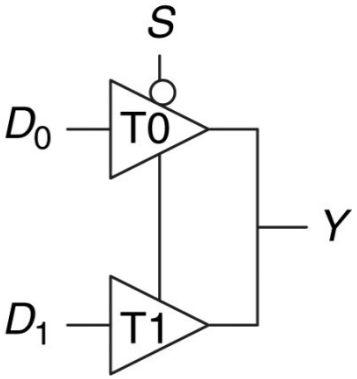


S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



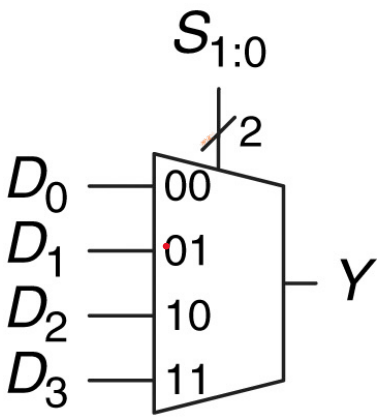
$$Y = D_0\overline{S} + D_1 S$$

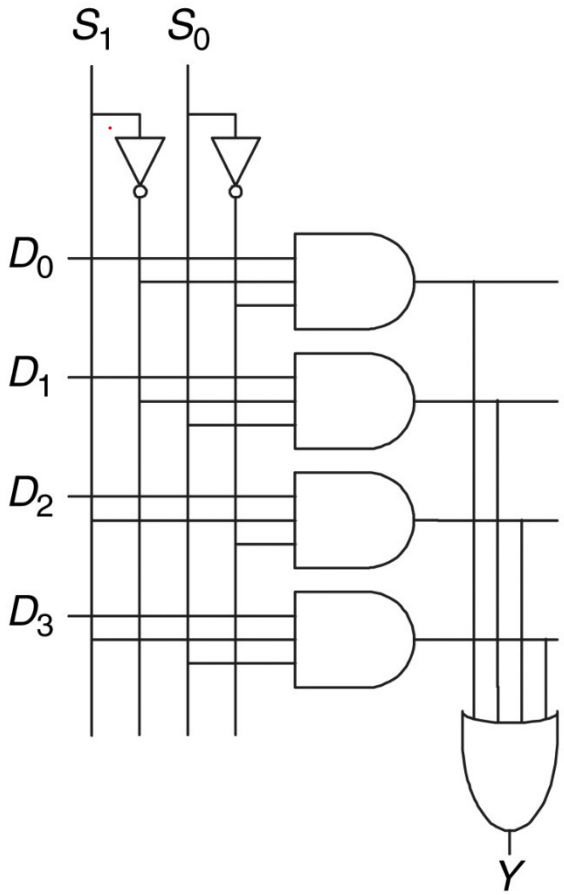




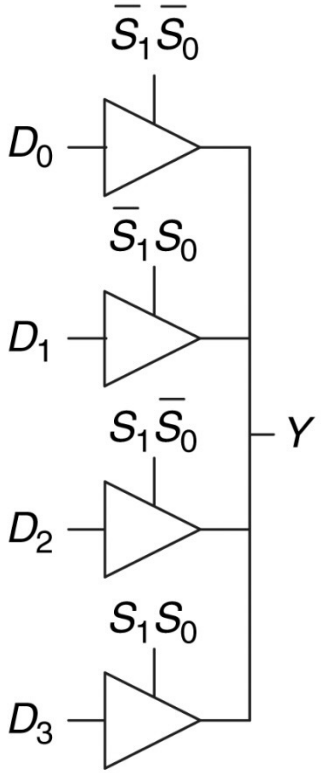
$$Y = D_0\overline{S} + D_1 S$$

Multiplexer 4:1

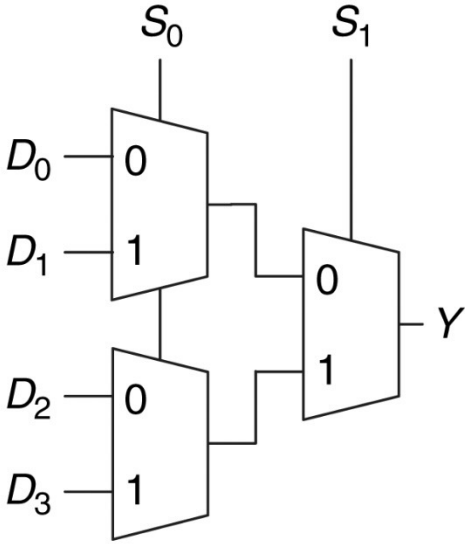




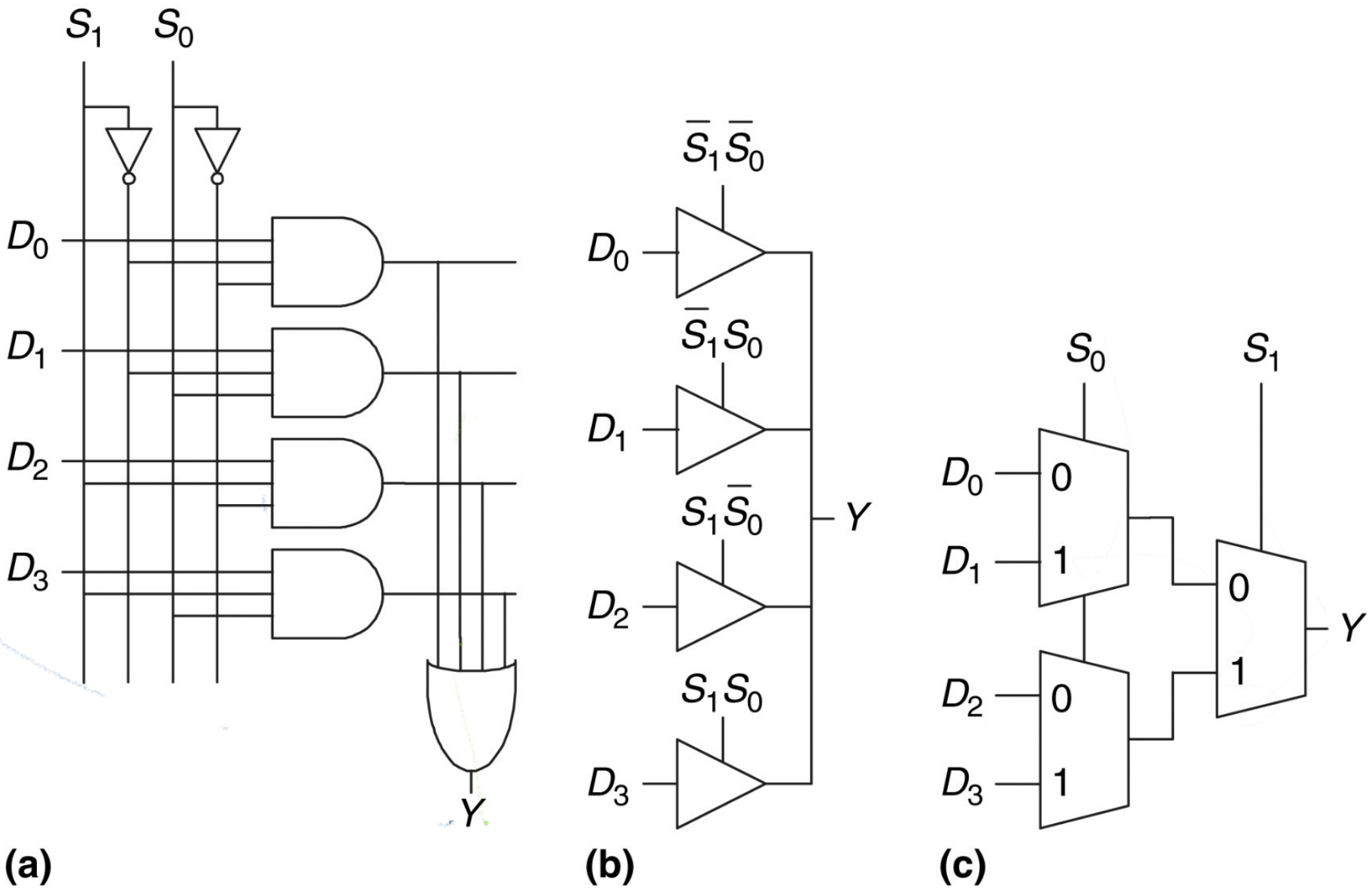
(a)



(b)



(c)

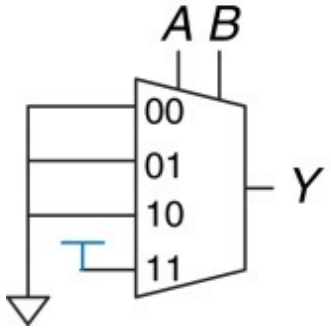

S_1
S_0
D_0
D_1
D_2
D_3
Y
(a)
$\overline{S}_1\overline{S}_0$
$\overline{S}_1 S_0$
$S_1\overline{S}_0$
$S_1 S_0$
D_0
D_1
D_2
D_3
Y
(b)
S_0
S_1
D_0
D_1
D_2
D_3
0
1
0
1
0
1
Y
(c)

Sintetizzare funzioni booleane con mux

- I multiplexer possono essere usati anche per sintetizzare delle funzioni booleane
- Sintetizzare una funzione di m variabili con un mux a 2^m linee è molto semplice: le variabili saranno linee di selezione. Data una certa configurazione delle variabili, la linea di ingresso corrispondente sarà posta al valore della funzione in quella configurazione
- Di fatto le linee di ingresso riproducono la tabella di verità della funzione

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

$Y = AB$

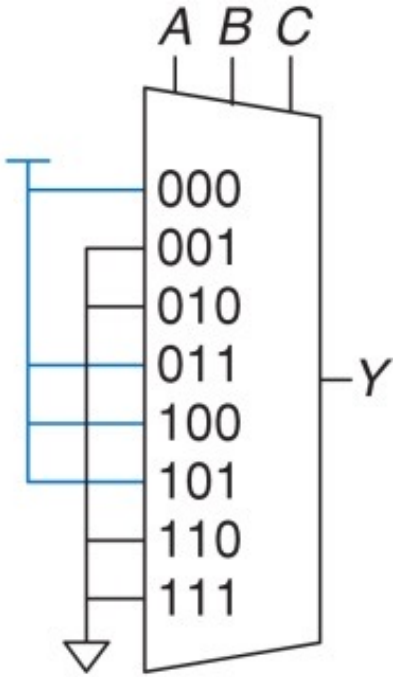


Sintetizzare funzioni booleane con mux

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

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

(a)

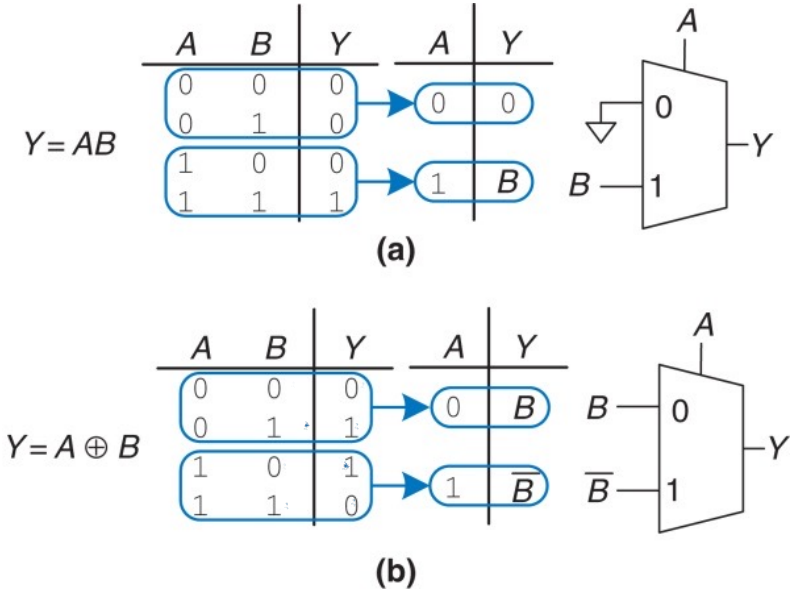


(b)

Y= Σ(0,3,4,5)

Sintetizzare funzioni booleane con mux

- E' possibile utilizzare un mux con 2^{m-1} ingressi per sintetizzare un funzione ad m variabili
- Le prime m-1 variabili saranno linee di selezione, mentre le linee di ingresso possono essere poste a 0,1, oppure all'ultima variabile (positiva o negata)



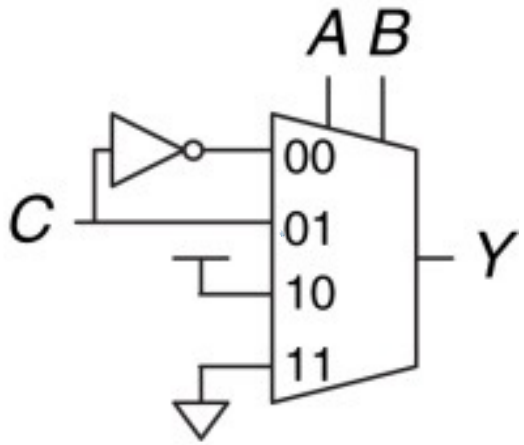
Sintetizzare funzioni booleane con mux

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

(a)

A	B	Y
0	0	$\overline{C}$
0	1	C
1	0	1
1	1	0

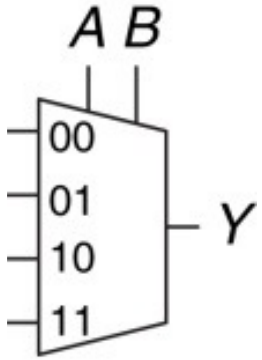
(b)



(c)

Sintetizzare funzioni booleane con mux

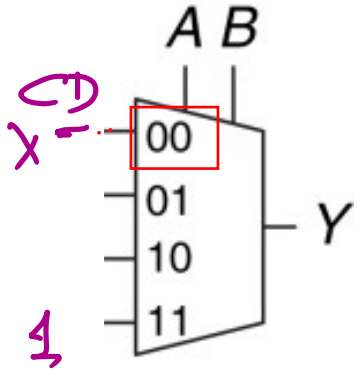
Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili



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

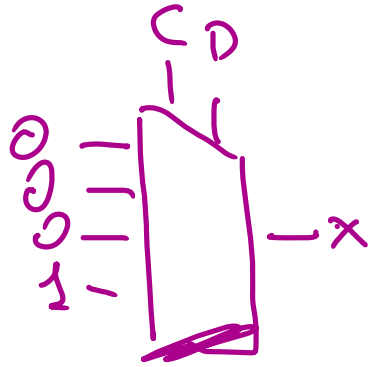
Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili



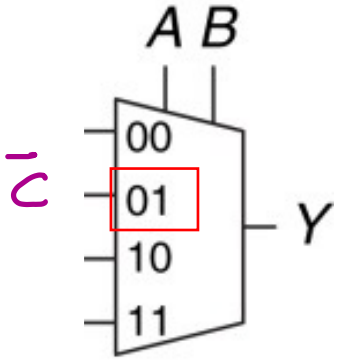
A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

CD



Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

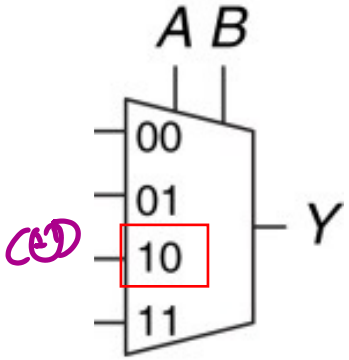


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

$\bar{C}$

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili

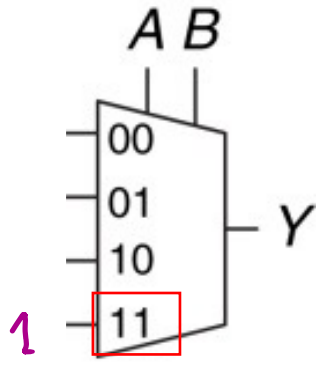


A B C D	Y
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

$C \oplus D$

Sintetizzare funzioni booleane con mux

Supponiamo di avere mux 4:1 e di voler sintetizzare una funzione di 4 variabili



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

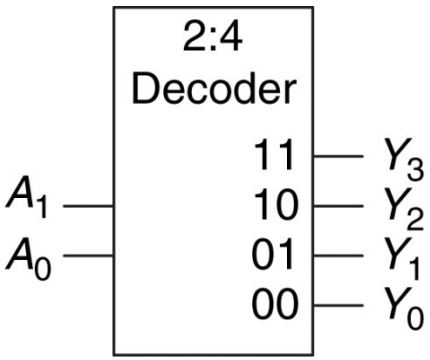
1

Esercizi

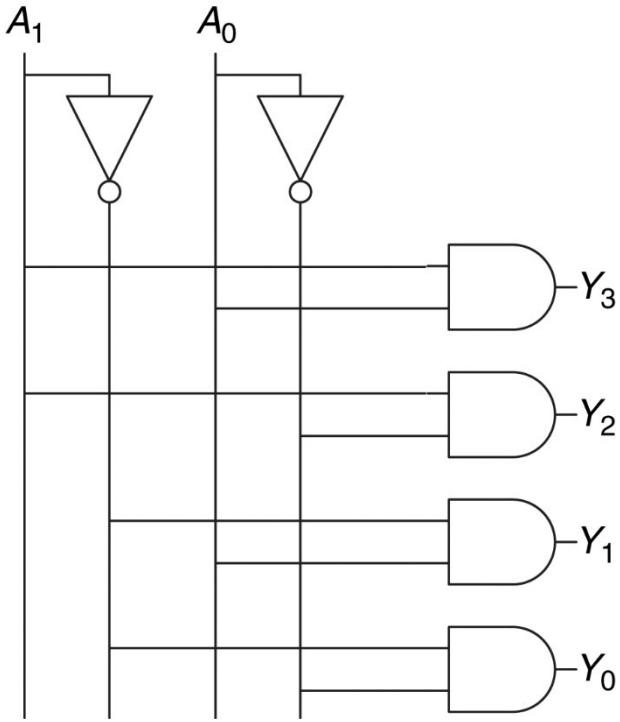
- Sintetizzare con multiplexer 4:1 la funzione di 4 variabili $\Sigma(0,5,9,10,13)$
- Sintetizzare con multiplexer 4:1 la funzione di 4 variabili $\Sigma(1,2,3,12,13)$
- Sintetizzare con multiplexer 4:1 la funzione di 4 variabili $\Sigma(0,1,2,3,5,15)$
- Sintetizzare con multiplexer 4:1 la funzione di 4 variabili $\Sigma(1,2,6,7,8,13)$

Decoder

- Un decoder ha N linee di ingresso e 2^N linee di uscita
- se m è numero rappresentato dagli input allora solo l'm-esima linea di uscita è pari a 1 mentre tutte le altre sono a 0

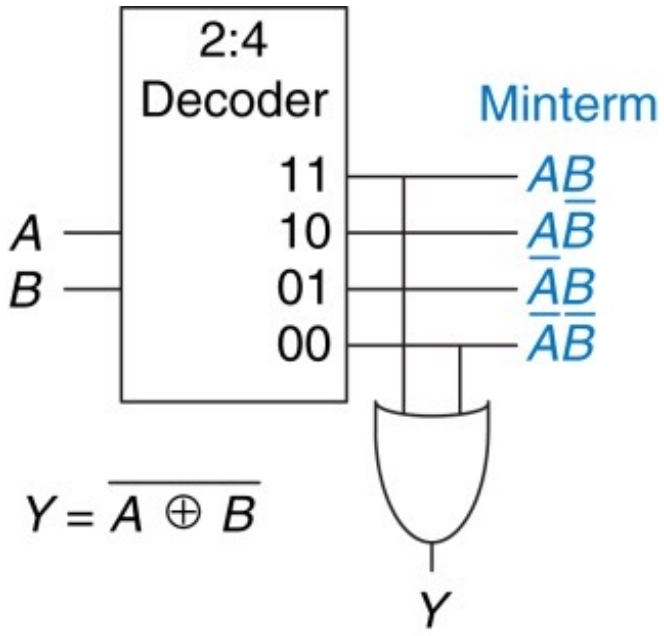


A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0



Sintetizzare funzioni booleane con decoder

- Anche i decoder possono essere usati per sintetizzare funzioni booleane
- Basta mettere in OR tutte e solo le linee di uscita che occorrono nella sigma-espressione della funzione da sintetizzare



$Y = \Sigma(0, 3)$

Esercizi

AB CD	00	01	11	10
00	X	0	0	X
01	0	1	X	0
11	1	1	1	0
10	1	X	0	1

AB CD	00	01	11	10
00	X	0	0	X
01	0	1	X	0
11	1	1	1	0
10	1	X	0	1

A*C+B*D*+BD

AB CD	00	01	11	10
00	X	0	0	X
01	1	1	X	X
11	1	0	0	1
10	0	X	0	1

AB CD	00	01	11	10
00	X	0	0	X
01	1	1	X	X
11	1	0	0	1
10	0	X	0	1

C*D+AB*+B*D

AB CD	00	01	11	10
00	X	0	0	X
01	1	0	X	X
11	1	0	0	1
10	1	X	0	1

AB CD	00	01	11	10
00	X	0	0	X
01	1	0	X	X
11	1	0	0	1
10	1	X	0	1

B*

A cosa è uguale la espressione(A+C)(A*+B)(B+C)

(A). (A+B)(A+C)

(B). (A+B)(A*+C)

(C). (A+C)(A+B*)

(D). (A+C)(A*+B)

(E). (A+C*)(A+B*)

A cosa è uguale la espressione(A+C)(A*+B)(B+C)

(A). (A+B)(A+C)

(B). (A+B)(A*+C)

(C). (A+C)(A+B*)

(D). (A+C)(A*+B)

(E). (A+C*)(A+B*)

Indicare il numero di funzioni booleane (distinte) di quattro variabili che soddisfano la condizione: $f(0,B,C,D)=0$, per qualsiasi valore di B,C e D.

Indicare il numero di funzioni booleane (distinte) di quattro variabili che soddisfano la condizione: f(0,B,C,D)=0, per qualsiasi valore di B,C e D.

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

$2^8=256$

Indicare il numero di funzioni booleane (distinte) di quattro variabili che soddisfano la condizione: se $A>D$ allora $f(A,B,C,D)=1$.

Indicare il numero di funzioni booleane (distinte) di quattro variabili che soddisfano la condizione: se A>D allora f(A,B,C,D)=1.

ABCD	Y
0000	X
0001	X
0010	X
0011	X
0100	X
0101	X
0110	X
0111	X
1000	1
1001	X
1010	1
1011	X
1100	1
1101	X
1110	1
1111	X

$2^{12}=4096$

Ridurre la seguente espressione booleana in forma SOP minima:

$$\overline{(A \cdot \bar{C}) \oplus (B + D)} + A\bar{C}\bar{D}$$

Ridurre la seguente espressione booleana in forma SOP minima:

$$\overline{(A \cdot \bar{C}) \oplus (B + D)} + A\bar{C}\bar{D} = (A \cdot \bar{C}) \cdot (B + D) + \overline{(A \cdot \bar{C})} \cdot \overline{(B + D)} + A\bar{C}\bar{D} = (A \cdot \bar{C}) \cdot (B + D) + (\bar{A} + C) \cdot \bar{B} \cdot \bar{D} + A\bar{C}\bar{D} = AB\bar{C} + A\bar{C}D + \bar{A}\bar{B}\bar{D} + \bar{B}C\bar{D} + A\bar{C}\bar{D} = A\bar{C} + \bar{B}\bar{D}$$

AB CD	00	01	11	10
00	1	0	1	1
01	0	0	1	1
11	0	0	0	0
10	1	0	0	1

Usando solo dei multiplexer 2:1 e i segnali di ingresso $A, B, C, \bar{A}, \bar{B}, \bar{C}, 0,1$ disegnare un circuito equivalente alla espressione booleana

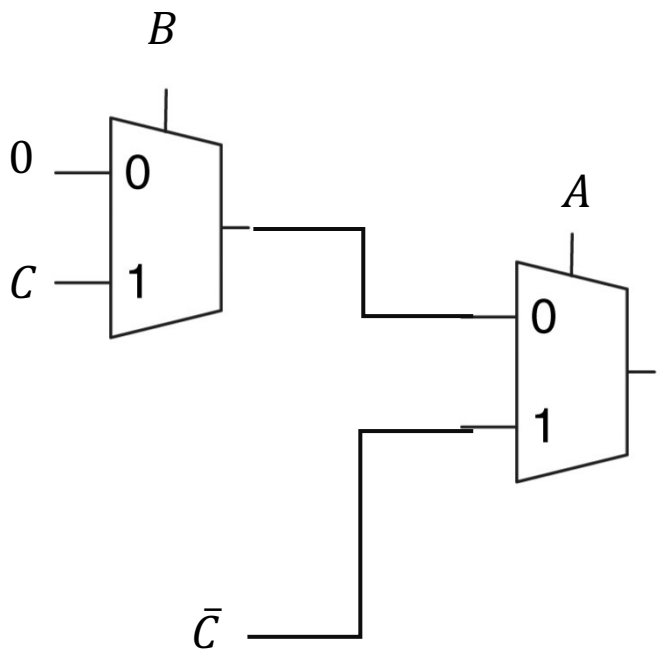
$$(A \oplus C) \cdot (A + B)$$

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

Usando solo dei multiplexer 2:1 e i segnali di ingresso $A, B, C, \bar{A}, \bar{B}, \bar{C}, 0,1$ disegnare un circuito equivalente alla espressione booleana

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

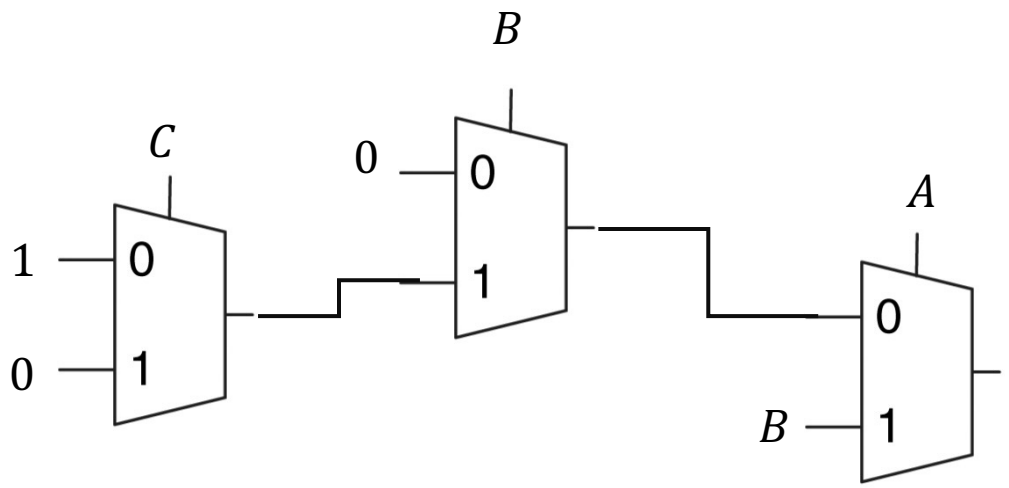
$$(A \oplus C) \cdot (A + B)$$



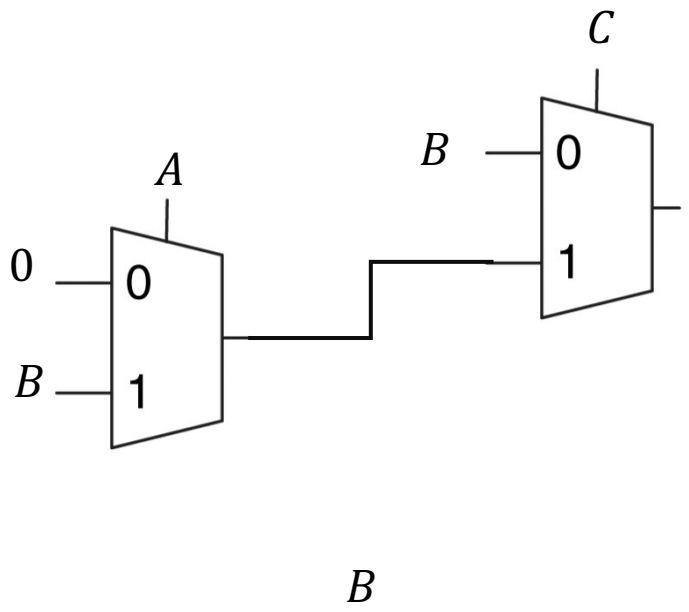
Usando solo 2 multiplexer 2:1 e i segnali di ingresso $A, B, C, 0, 1$ disegnare un circuito equivalente alla espressione booleana

A B C	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

$$(B \cdot \bar{C}) + (A \cdot B)$$



Usando solo 2 multiplexer 2:1 e i segnali di ingresso $A, B, C, 0,1$ disegnare un circuito equivalente alla espressione booleana

A B C	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

$$(B \cdot \bar{C}) + (A \cdot B)$$