

UNIVERSITÀ DEGLI STUDI DI SALERNO

Fondamenti di Informatica

Algebra di Boole e Circuiti Logici

Prof. Christian Esposito

Corso di Laurea in Ingegneria Meccanica e Gestionale (Classe I)

A.A. 2016/17

L'Algebra di Boole – 1/4

- **Un po' di storia**

- Il matematico inglese George Boole nel 1847 fondò un campo della matematica e della filosofia chiamato **logica simbolica**
- Shannon per primo applicò la logica simbolica ai circuiti nel 1939

- **L'algebra di Boole è caratterizzata da**

- **Variabili booleane (o binarie):** variabili i cui valori possono essere 0 oppure 1
 - Ma anche: vero/falso, on/off, si/no
- **Operazioni (o funzioni) booleane:** funzioni i cui input ed output sono variabili booleane

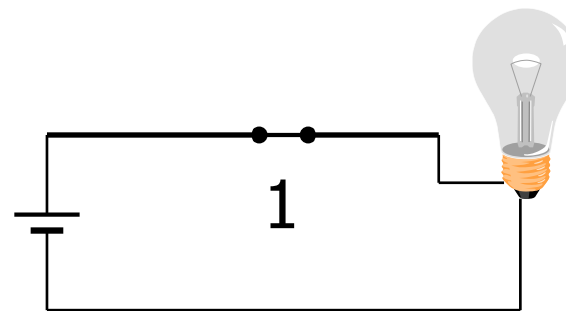
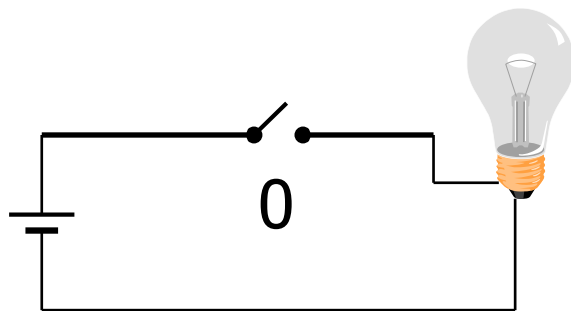
- **Relazione con i circuiti logici**

- Si studia l'algebra booleana poiché le sue funzioni sono isomorfe ai circuiti digitali: un circuito digitale può essere espresso tramite un'espressione booleana e viceversa
 - Le variabili booleane corrispondono a segnali
 - Le funzioni booleane corrispondono a circuiti



L'Algebra di Boole – 2/4

- Come variabili contempla solo due costanti: **0** e **1** (**falso** e **vero**)
 - Corrispondono a due stati che si escludono a vicenda
 - Possono descrivere lo stato di apertura o chiusura di un generico contatto o di un circuito a più contatti



- Sulle variabili booleane si definiscono le funzioni (od operazioni) **AND, OR, NOT**
 - Ed altre definite a partire da esse

L'Algebra di Boole – 3/4

- Le operazioni **AND** e **OR** sono **operazioni binarie** (agiscono su due operandi), l'operazione **NOT** è **unaria**
- Nella valutazione delle espressioni booleane esiste una **relazione di precedenza fra gli operatori** NOT, AND e OR, nell'ordine in cui sono stati elencati
- Per alterare tale relazione bisogna usare le parentesi
 - Talvolta usate solo per maggiore chiarezza

L'Algebra di Boole – 4/4

- Gli **operatori** dell'algebra booleana possono essere **rappresentati e descritti in vari modi**
 - Spesso sono descritti semplicemente come AND, OR e NOT
 - **Tavole di verità**
 - Nella descrizione dei circuiti appaiono sotto forma di **porte logiche**

Operatore (o funzione) OR

- **Somma logica (OR):** il valore della somma logica è il simbolo **1** se il valore di almeno uno degli operandi è il simbolo **1**

Tavola di verità

x_1	x_2	$F(x_1, x_2) = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

- In generale, date n variabili binarie, la loro somma logica (OR) è data da

$$x_1 + x_2 + \dots + x_n = \begin{cases} 1 & \text{se almeno una } x_i \text{ vale } 1, \text{ con } 1 \leq i \leq n \\ 0 & \text{se } x_1 = x_2 = \dots = x_n = 0 \end{cases}$$

Operatore OR: Possibili Rappresentazioni

- $x \mid y$ <- Usato in MATLAB
- $\text{or}(x, y)$ <- Usato in MATLAB
- $x \# y$
- $x \text{ or } y$
- $x + y$
- $x \cup y$
- $x \vee y$

Operatore (o funzione) AND

- **Prodotto logico (AND):** il valore del prodotto logico è il simbolo 1 se il valore di tutti gli operandi è il simbolo 1

x_1	x_2	$F(x_1, x_2) = x_1 \times x_2$
0	0	0
0	1	0
1	0	0
1	1	1

- In generale, date n variabili binarie indipendenti, il loro prodotto logico (AND) è dato da

$$x_1 \times x_2 \times \dots \times x_n = \begin{cases} 0 & \text{se almeno una } x_i \text{ vale } 0, \text{ con } 1 \leq i \leq n \\ 1 & \text{se } x_1 = x_2 = \dots = x_n = 1 \end{cases}$$

Operatore AND: Possibili Rappresentazioni

- $x \& y$ <- Usato in MATLAB
- $\text{and}(x, y)$ <- Usato in MATLAB
- $x \text{ and } y$
- $x \wedge y$
- $x \cap y$
- $x \times y$
- $x * y$
- xy

Operatore (o funzione) NOT

- **Operatore di negazione (NOT):** inverte il valore della costante su cui opera
 - Noto anche come **inverter**

$$\begin{array}{ll} \overline{0} = 1 & \overline{\overline{0}} = 0 \\ \overline{1} = 0 & \overline{\overline{1}} = 1 \end{array}$$

- In generale, la negazione di una variabile x è

$$\begin{array}{l} \bar{x} = 0 \text{ se } x = 1 \\ \bar{x} = 1 \text{ se } x = 0 \end{array}$$

- L'elemento $\bar{x} = \text{NOT}(x)$ viene detto **complemento** di x
 - Il complemento è unico

Operatore NOT: Possibili Rappresentazioni

- $y = \sim x$ <- Usato in MATLAB
- $y = \text{not}(x)$ <- Usato in MATLAB
- $y = !x$
- $y = \text{not } x$
- $y = x'$
- $y = \neg x$
- $y = \bar{x}$

Algebra di Boole: Alcune Identità

Funzione AND	Funzione OR	Funzione NOT
$0 \times 0 = 0$	$0 + 0 = 0$	$x + \bar{x} = 1$
$0 \times 1 = 0$	$0 + 1 = 1$	$x \times \bar{x} = 0$
$1 \times 0 = 0$	$1 + 0 = 1$	$\bar{\bar{x}} = x$
$1 \times 1 = 1$	$1 + 1 = 1$	
$x \times 0 = 0$	$x + 0 = x$	
$0 \times x = 0$	$0 + x = x$	
$x \times 1 = x$	$x + 1 = 1$	
$1 \times x = x$	$1 + x = 1$	
$x \times x = x$	$x + x = x$	

**Legge
dell'idempotenza**

Algebra di Boole: Proprietà e Leggi

Proprietà Commutativa $x_1x_2 = x_2x_1$ $x_1 + x_2 = x_2 + x_1$	Leggi di Assorbimento $x_1 + x_1x_2 = x_1$ $x_1(x_1 + x_2) = x_1$
Proprietà Distributiva $x_1(x_2 + x_3) = x_1x_2 + x_1x_3$ $x_1 + (x_2x_3) = (x_1 + x_2)(x_1 + x_3)$	Leggi di De Morgan $\overline{x_1 + x_2} = \bar{x}_1 \times \bar{x}_2$ $\overline{x_1 \times x_2} = \bar{x}_1 + \bar{x}_2$
Proprietà Associativa $x_1(x_2x_3) = (x_1x_2)x_3$ $x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$	Altre Note $x_1 + \bar{x}_1x_2 = x_1 + x_2$ $x_1(\bar{x}_1 + x_2) = x_1x_2$

Leggi di De Morgan – 1/4

- Il complemento di una somma di variabili è uguale al prodotto dei complementi delle variabili

- Il complemento di due o più variabili poste in OR è uguale all'AND dei complementi delle singole variabili

Leggi di Assorbimento

$$x_1 + x_1x_2 = x_1$$
$$x_1(x_1 + x_2) = x_1$$

Leggi di De Morgan

$$\overline{x_1 + x_2} = \bar{x}_1 \times \bar{x}_2$$
$$x_1 \times x_2 = \overline{\bar{x}_1 + \bar{x}_2}$$

Proprietà Associativa

$$x_1(x_2x_3) = (x_1x_2)x_3$$
$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$$

Altre Note

$$x_1 + \bar{x}_1x_2 = x_1 + x_2$$
$$x_1(\bar{x}_1 + x_2) = x_1x_2$$

Leggi di De Morgan – 2/4

- Il complemento di un prodotto di variabili è uguale alla somma dei complementi delle variabili

- Il complemento di due o più variabili poste in AND è equivalente all'OR dei complementi delle singole variabili

Leggi di Assorbimento

$$x_1 + x_1x_2 = x_1$$
$$x_1(x_1 + x_2) = x_1$$

Leggi di De Morgan

$$\overline{x_1 + x_2} = \bar{x}_1 \times \bar{x}_2$$
$$\overline{x_1 \times x_2} = \bar{x}_1 + \bar{x}_2$$

Proprietà Associativa

$$x_1(x_2x_3) = (x_1x_2)x_3$$
$$x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3$$

Altre Note

$$x_1 + \bar{x}_1x_2 = x_1 + x_2$$
$$x_1(\bar{x}_1 + x_2) = x_1x_2$$

Leggi di De Morgan – 3/4

- Osservazione: $\bar{\bar{x}} = x$ (Eq. 1)
- Legge 1 di De Morgan: $\overline{x_1 + x_2} = \bar{x}_1 \times \bar{x}_2$ (Eq. 2)
- Utilizzando (Eq. 1) posso scrivere (Eq. 2) come segue: $\overline{\overline{x_1 + x_2}} = \overline{\bar{x}_1 \times \bar{x}_2}$
- Utilizzando ancora (Eq. 1) ottengo che $x_1 + x_2 = \overline{\bar{x}_1 \times \bar{x}_2}$
- L'OR fra x_1 e x_2 può essere espresso in termini delle sole operazioni AND e NOT
 - Ogni volta che in un'espressione booleana troviamo un OR, lo possiamo sostituire con la appropriata combinazione di AND e NOT
 - Ogni espressione può essere espressa in termini delle sole due operazioni logiche AND e NOT

Leggi di De Morgan – 4/4

- **Osservazione:** $\bar{\bar{x}} = x$ (Eq. 1)
- **Legge 2 di De Morgan:** $\overline{x_1 \times x_2} = \bar{x}_1 + \bar{x}_2$ (Eq. 3)
- Utilizzando (Eq. 1) posso scrivere (Eq. 3) come segue: $\overline{\overline{x_1 \times x_2}} = \overline{\bar{x}_1 + \bar{x}_2}$
- Utilizzando ancora (Eq. 1) ottengo che $x_1 \times x_2 = \overline{\bar{x}_1 + \bar{x}_2}$
- L'AND fra x_1 e x_2 può essere espresso in termini delle sole operazioni OR e NOT
 - Ogni volta che in un'espressione booleana troviamo un AND, lo possiamo sostituire con la appropriata combinazione di OR e NOT
 - Ogni espressione può essere espressa in termini delle sole due operazioni logiche OR e NOT

Alcune Osservazioni

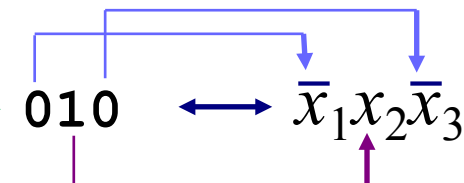
- Identità, proprietà e leggi viste fino ad ora sono generalmente applicate nelle trasformazioni di funzioni booleane in altre equivalenti, ma di più facile realizzazione circuitale
- Dalle leggi di **De Morgan** si evince che la scelta delle funzioni OR, AND e NOT, come funzioni primitive, è ridondante

Funzioni Logiche (o Booleane)

– 1/5

- Date n variabili booleane indipendenti x_1, x_2, \dots, x_n , queste possono assumere 2^n **configurazioni** distinte
 - Ad esempio per $n = 3$ si hanno 8 configurazioni
- Ogni riga (**in rosso**) mostra il valore restituito a partire da una particolare configurazione dell'input
- Una configurazione specifica è individuata univocamente da un AND di tutte le variabili, dove quelle corrispondenti ai valori 0 compaiono negate
 - **Prodotto fondamentale o prodotto minimo (minterm)**

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Funzioni Logiche (o Booleane)

– 2/5

Configurazioni	x_1	x_2	x_3	$F(x_1, x_2, x_3)$
$\bar{x}_1 \bar{x}_2 \bar{x}_3$	0	0	0	0
$\bar{x}_1 \bar{x}_2 x_3$	0	0	1	0
$\bar{x}_1 x_2 \bar{x}_3$	0	1	0	0
$\bar{x}_1 x_2 x_3$	0	1	1	1
$x_1 \bar{x}_2 \bar{x}_3$	1	0	0	0
$x_1 \bar{x}_2 x_3$	1	0	1	1
$x_1 x_2 \bar{x}_3$	1	1	0	1
$x_1 x_2 x_3$	1	1	1	1

- 011 indica tra le $2^3=8$ configurazioni possibili, quella in cui x_1 vale 0, x_2 vale 1 e x_3 vale 1
- Questa configurazione si scrive semplicemente con il prodotto $\bar{x}_1 x_2 x_3$
- Se in una configurazione una variabile compare con 1, si assume il valore diretto, se invece compare con uno 0, si assume il valore negato

Funzioni Logiche (o Booleane)

– 3/5

- Una variabile y è **funzione** delle n variabili indipendenti x_1, x_2, \dots, x_n , quando esiste un criterio che fa corrispondere in modo univoco ad ognuna delle 2^n configurazioni di x un determinato valore y (ovviamente 0 o 1)

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

- Una rappresentazione esplicita di una funzione è la tavola di verità, in cui si elencano tutte le possibili combinazioni di x_1, x_2, \dots, x_n , con associato il valore di y

$$y = x_1 + x_2$$



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

Funzioni Logiche (o Booleane)

– 4/5

- Si può specificare l'output di ogni funzione booleana esprimendo, tramite un'espressione booleana, quali combinazioni delle variabili di input determinano l'output 1

- Più precisamente, per passare dalla rappresentazione mediante tavola di verità alla notazione tramite espressione booleana è necessario

1. Identificare tutte le righe della tavola di verità che danno 1 in output
2. Per ogni riga con un 1 in output, scrivere la configurazione delle variabili che la definiscono
3. Collegare tramite OR tutte le configurazioni ottenute

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3$$

Funzioni Logiche (o Booleane)

– 5/5

- $F(x_1, x_2, x_3) = \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 x_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3$

- Con l'uso dei minterm possiamo determinare l'espressione algebrica di una funzione booleana a partire dalla tavola di verità
- L'espressione algebrica trovata si chiama **forma canonica** della funzione e si ottiene con uno sviluppo in minterm
 - **Una somma (OR) di prodotti (AND)**
- Se un minterm assume valore 1 anche la funzione F assume il valore 1

x_1	x_2	x_3	$F(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Esempio 1: la Funzione Exclusive OR (XOR) – 1/2

- Il **comportamento** della funzione **Exclusive OR** può essere descritto come segue
 - *$F = \text{“L’output deve essere 1 (vero) se solo uno dei suoi input è 1, ma non se entrambi gli input sono 1”}$*
- Questo può essere **rifrasato** come segue
 - *$F = \text{“L’output è 1 se } (x_1 \text{ OR } x_2) \text{ è 1, AND se } (x_1 \text{ AND } x_2) \text{ sono NOT 1 (falso)”}$*
- *Che può essere scritto come*
 - $F = (x_1 + x_2) \times \overline{(x_1 x_2)}$


Esempio 1: la Funzione Exclusive OR (XOR) – 2/2

- La funzione XOR verifica la disuguaglianza di due variabili

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Si può scrivere la funzione come somma logica (OR) delle configurazioni corrispondenti agli 1

- L'espressione come somma di prodotti è quindi


$$XOR(x_1, x_2) = \bar{x}_1 \times x_2 + x_1 \times \bar{x}_2$$

Forma canonica: somma di prodotti (OR di AND)

N.B. tutte le funzioni logiche si possono scrivere in questa forma

Esempio 2: dalla Tavola di Verità alla Funzione

<i>x</i>	<i>y</i>	<i>z</i>	<i>F</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

- **Problema:** date tre variabili booleane (x, y, z), si scriva la funzione F che vale 1 quando solo due di esse hanno valore 1

Si può scrivere la funzione come somma logica (OR) delle configurazioni corrispondenti agli 1

$$F(x, y, z) = \bar{x}yz + x\bar{y}z + xy\bar{z}$$

Forma canonica: somma di prodotti (OR di AND)

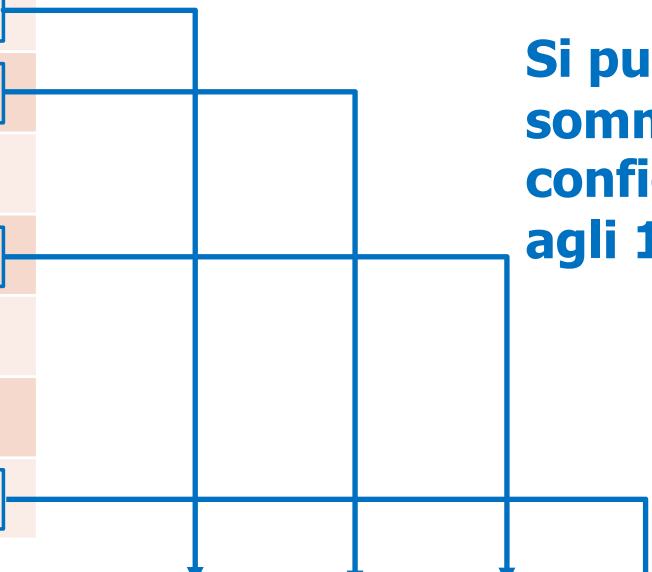
N.B. tutte le funzioni logiche si possono scrivere in questa forma

Esempio 3: dalla Tavola di Verità alla Funzione

x	y	z	F
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

- **Problema:** date tre variabili booleane (x, y, z) , si scriva la funzione F che vale 1 quando il numero di 1 è dispari

Si può scrivere la funzione come somma logica (OR) delle configurazioni corrispondenti agli 1


$$F(x, y, z) = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

Forma canonica: somma di prodotti (OR di AND)

N.B. tutte le funzioni logiche si possono scrivere in questa forma

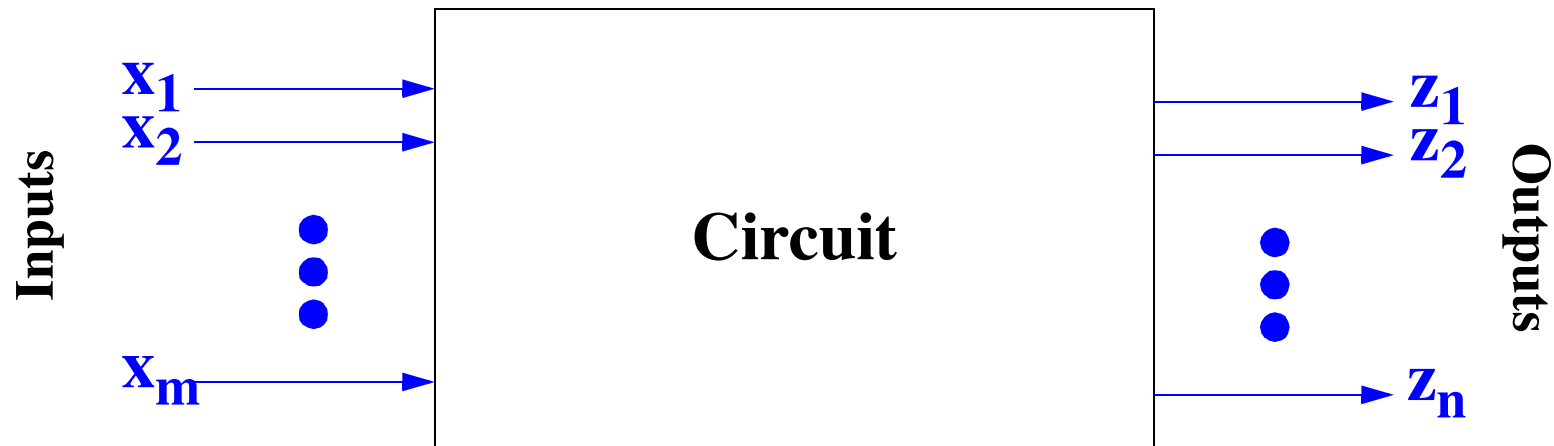
Esempio 4: dalla Funzione alla Tavola di Verità

- Vediamo un esempio per la funzione

- $F = x \times \overline{(y + z)}$

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Circuito Logico



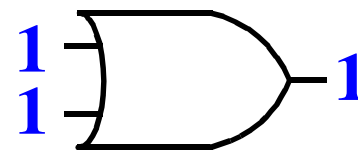
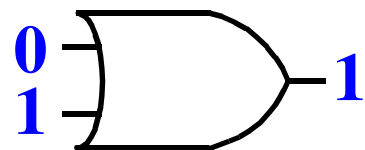
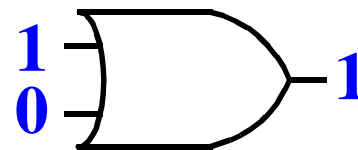
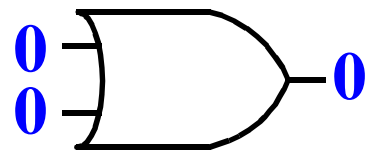
- Il **cuore** di un **sistema digitale** è il **circuito logico** digitale
 - Progettato a partire da **porte logiche**
 - Collegate tra loro per formare **circuiti più grandi**
 - Combinati per realizzare circuiti di grande importanza pratica nell'architettura del computer

Porte Logiche

- Building block utilizzati per creare circuiti digitali
- Qualsiasi circuito può essere implementato usando solo porte logiche elementari (AND, OR e NOT)
 - Le cose si fanno complicate quando si hanno numerosi input ed output
- Dispositivi elettronici che implementano semplici funzioni booleane
- Ciascuna porta ha il proprio simbolo logico che permette a funzioni complesse di essere rappresentate mediante un diagramma logico
- La funzione di ciascuna porta può essere rappresentata da una tabella di verità o utilizzando la notazione booleana

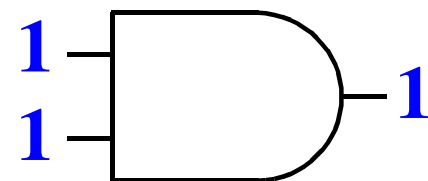
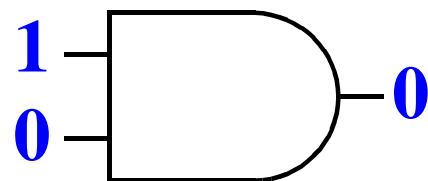
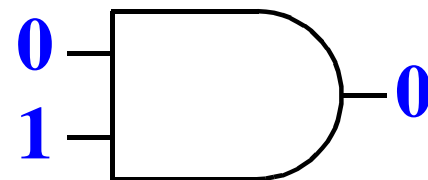
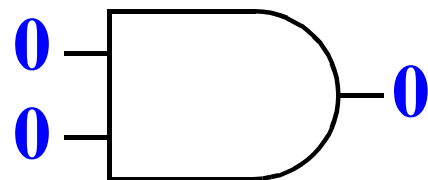
Funzione OR: Tavola di Verità e Porta Logica

x_1	x_2	$x_1 \text{ OR } x_2$
0	0	0
0	1	1
1	0	1
1	1	1



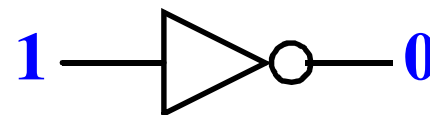
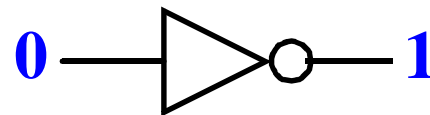
Funzione AND: Tavola di Verità e Porta Logica

x_1	x_2	$x_1 \text{ AND } x_2$
0	0	0
0	1	0
1	0	0
1	1	1



Funzione NOT: Tavola di Verità e Porta Logica

x	$NOT\ x$
0	1
1	0



Porta NAND



(a) Circuit symbol

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

(b) Truth table

$$C = \overline{A \cdot B}$$

(c) Boolean expression

Porta NOR



(a) Circuit symbol

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

(b) Truth table

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

(c) Boolean expression

Porta XOR



(a) Circuit symbol

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

(b) Truth table

$$C = A \oplus B$$

(c) Boolean expression

Porta Exclusive NOR



(a) Circuit symbol

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

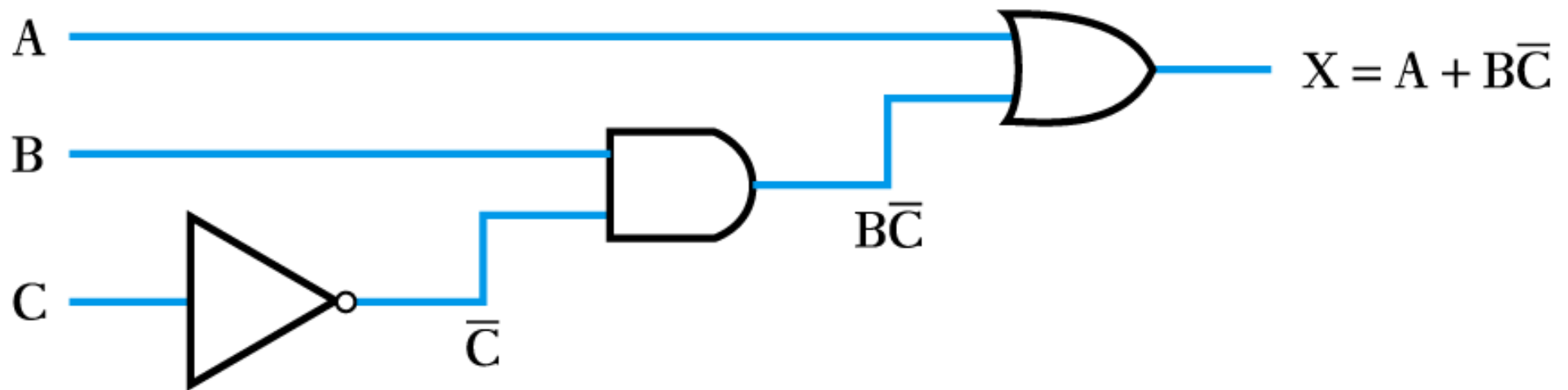
(b) Truth table

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

(c) Boolean expression

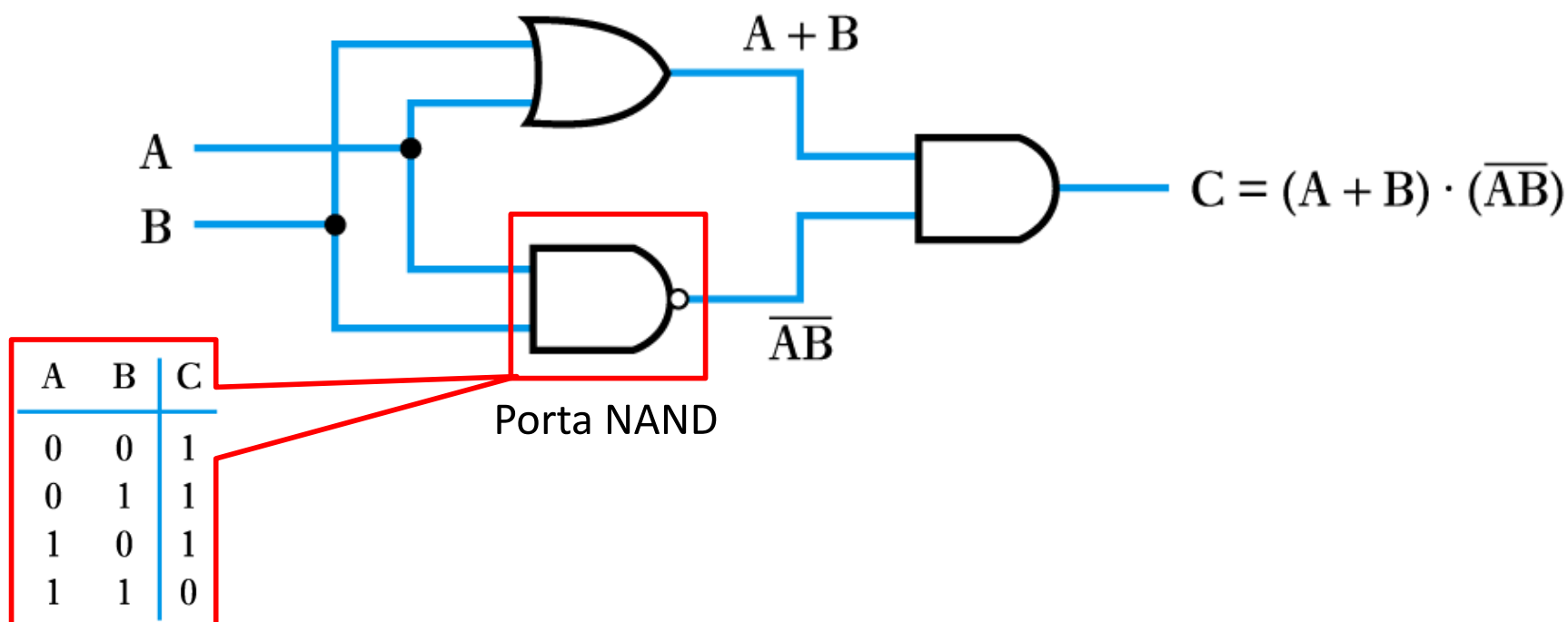
Esempio 5: dalla Funzione al Circuito

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



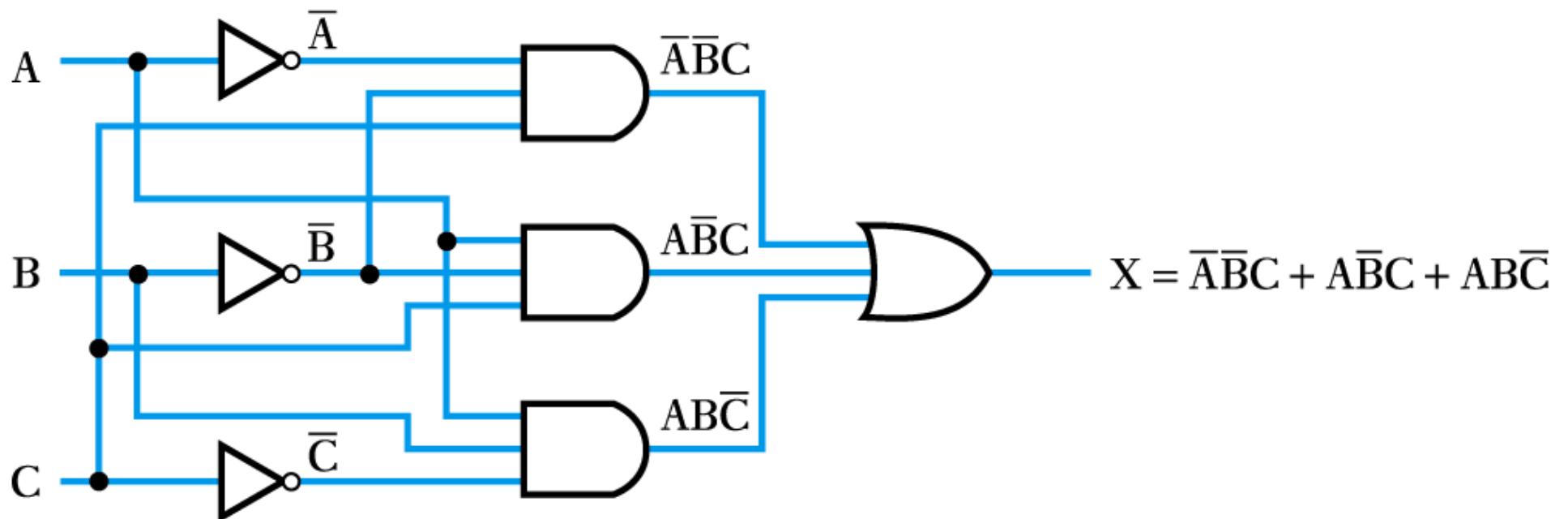
Esempio 6: dalla Funzione al Circuito

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

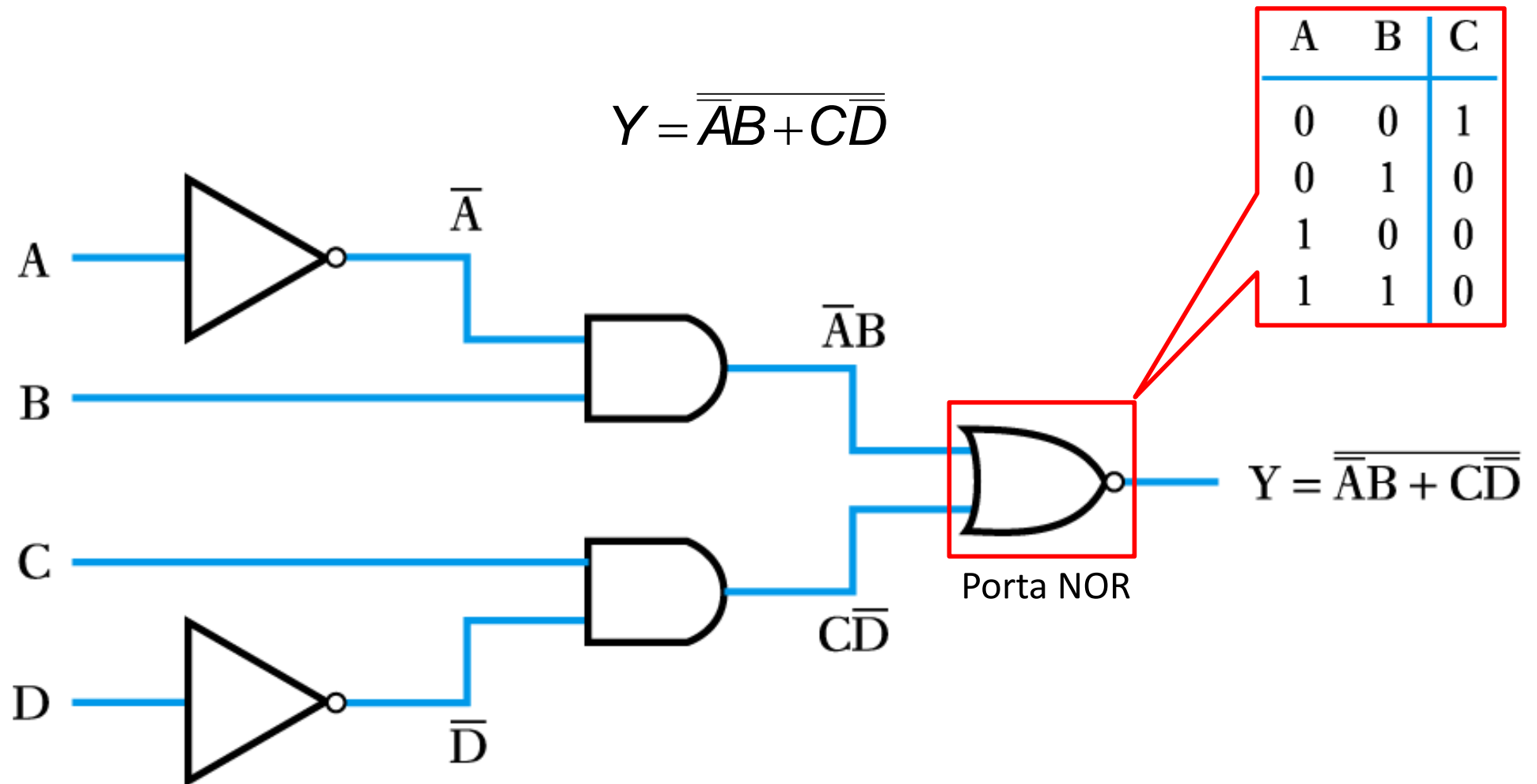


Esempio 7: dalla Funzione al Circuito

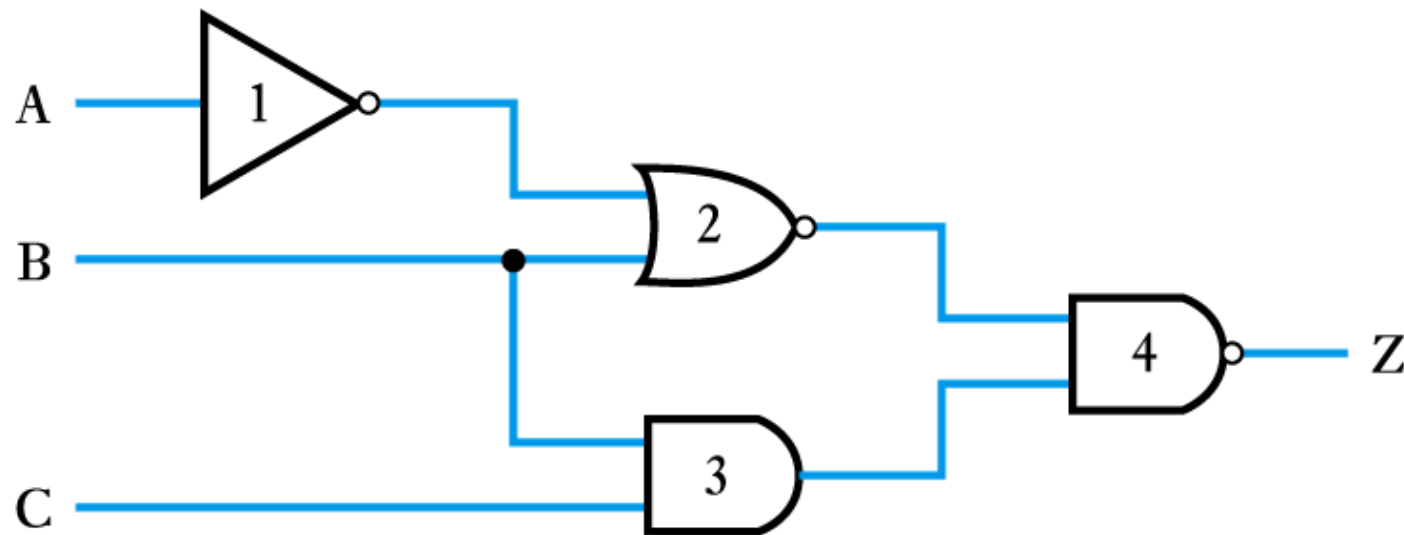
$$X = \bar{A}\bar{B}C + A\bar{B}C + AB\bar{C}$$



Esempio 8: dalla Funzione al Circuito

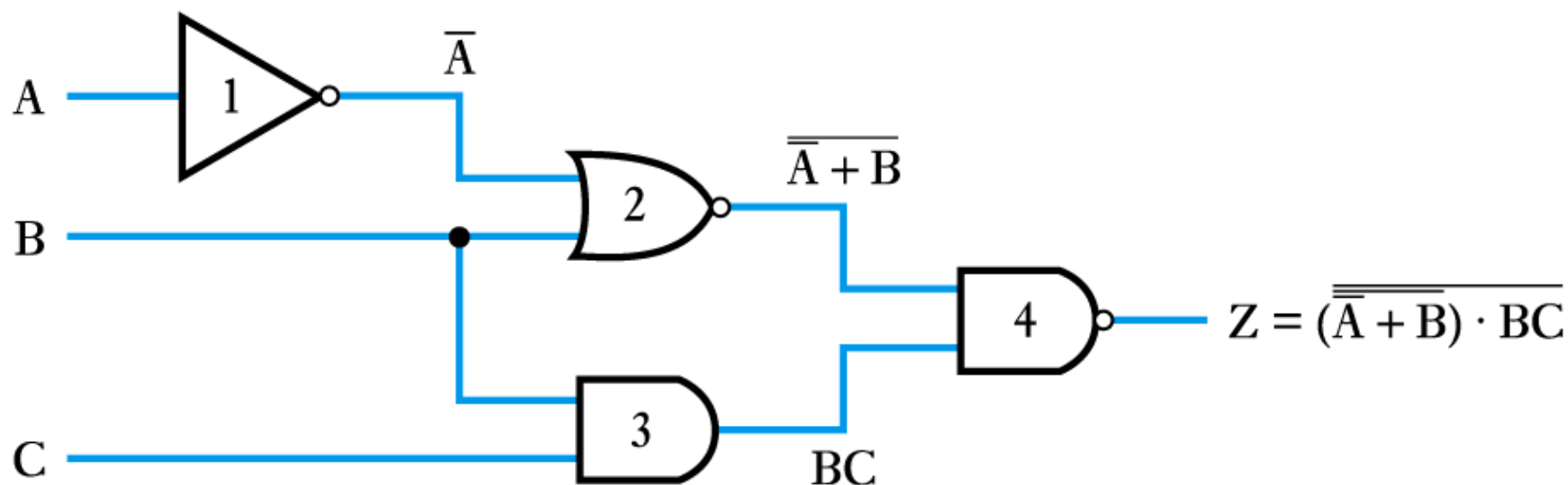


Esempio 9: dal Circuito alla Funzione – 1/2



Esempio 9: dal Circuito alla Funzione – 2/2

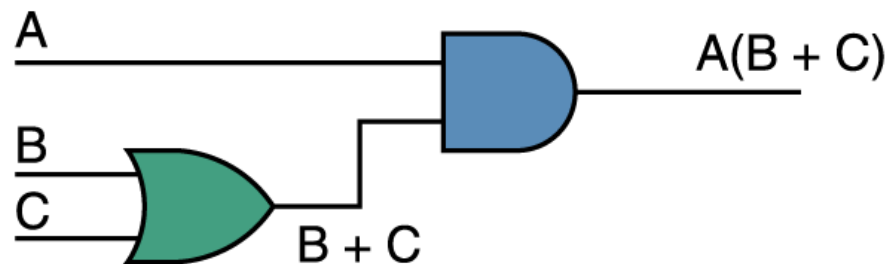
- Procedere progressivamente dagli input verso l'output aggiungendo a turno le espressioni logiche all'output di ciascuna porta logica



Esempio 10: Funzione \Rightarrow Tavola di Verità \Rightarrow Circuito

- Si consideri la seguente funzione: $A(B + C)$

A	B	C	$B + C$	$A(B+C)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



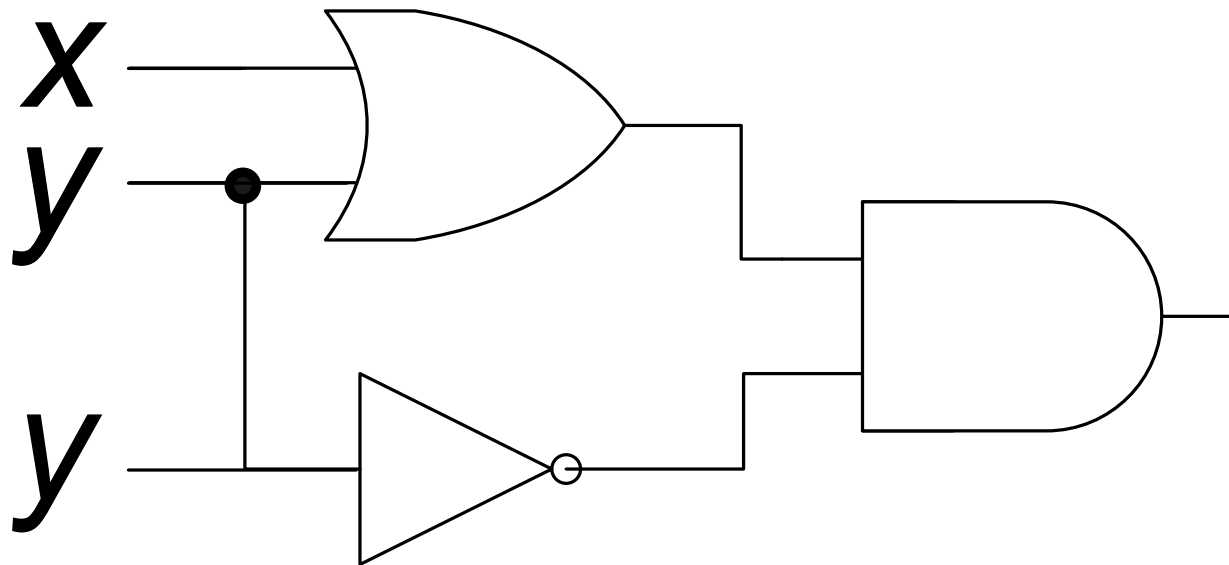
Ricapitolando...

- Abbiamo visto che una **funzione logica** (ma anche un **circuito logico**) può essere **espressa in due modi**
 - **Tavola di Verità**
 - **Porte Logiche**
- Perché abbiamo bisogno di tutte queste diverse rappresentazioni?
 - Alcune sono più facili di altre per cominciare a progettare un circuito
 - Di solito si comincia con la tavola di verità
 - Si deriva un'espressione booleana da essa (magari esemplificata)
 - Si trasforma l'espressione booleana in un circuito

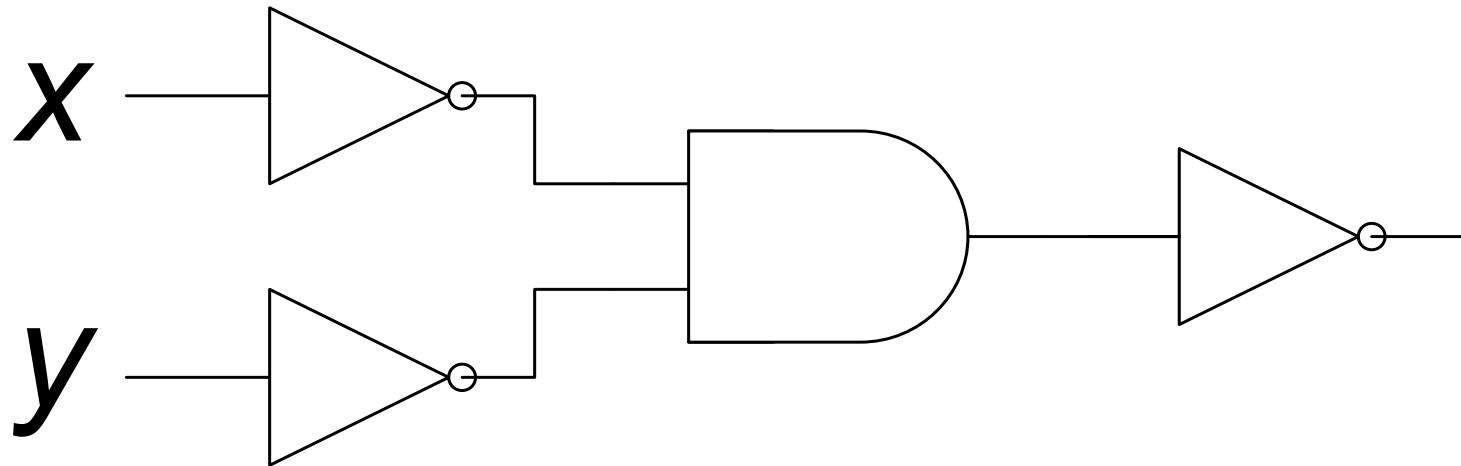
Esercizio 1: determinare la funzione espressa dalla seguente tavola di verità

A	B	C	X
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	1
1	1	1	0

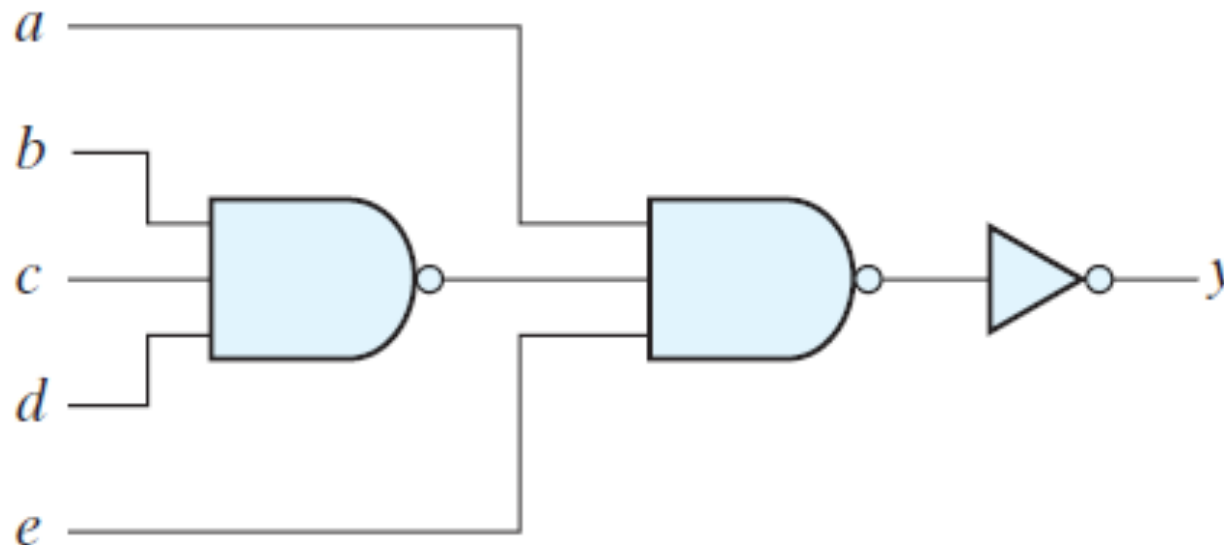
Esercizio 2: trovare l'output del seguente circuito (tavola di verità e funzione)



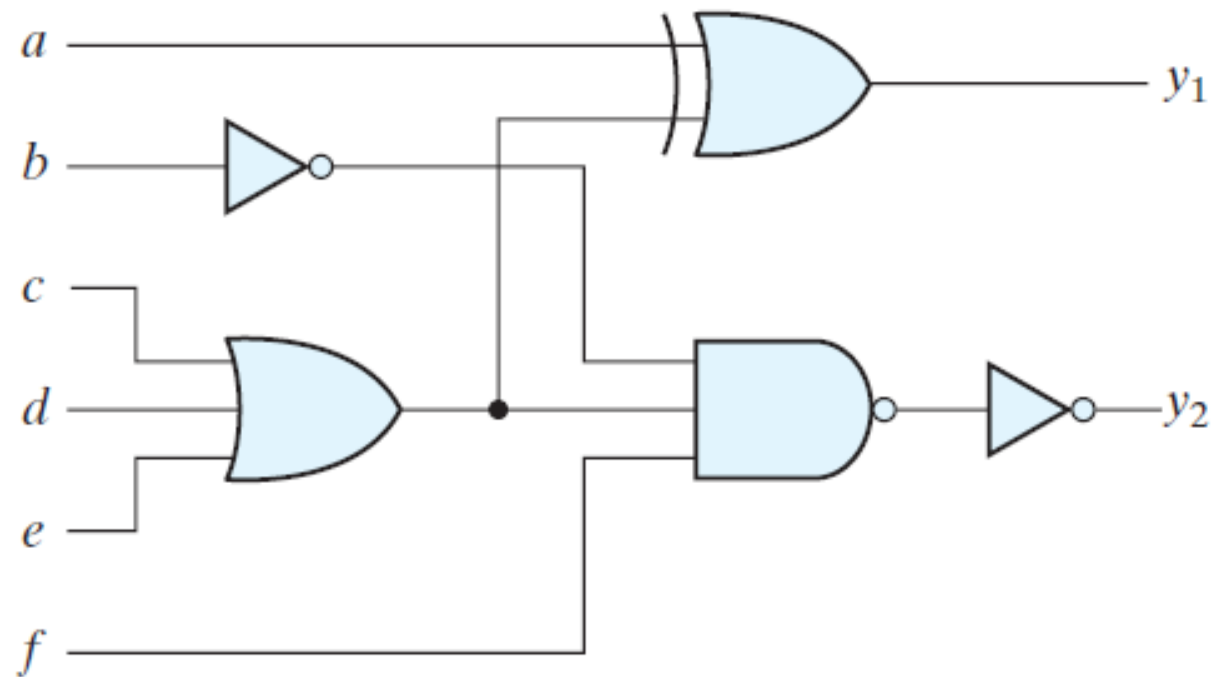
Esercizio 3: trovare l'output del seguente circuito (tavola di verità e funzione)



Esercizio 4: trovare l'output del seguente circuito (tavola di verità e funzione)



Esercizio 5: trovare l'output del seguente circuito (tavola di verità e funzione)



Esercizio 6: progettare il circuito per ciascuna delle seguenti espressioni

- $\bar{x} + y$
- $\overline{(x + y)}x$
- Scrivere la funzione XOR usando AND, OR e NOT

Riferimenti

- **Libro di testo**

- Capitolo 3
 - Paragrafo 4

- **Altri riferimenti**

- <http://www.di.unito.it/~piccolo/teach/AA1516/Lezioni/Lezione2.pdf>
- <http://liceocuneo.it/basteris/wp-content/uploads/sites/3/CIRCUITI20DIGITALI1.pdf>
- <http://bias.csr.unibo.it/maltoni/arc/Dispense/LogicaDigitale.pdf>
- http://people.unipmn.it/bobbio/DIDATTICA/ARCH1_00/ALDISP_00/varbol00.pdf