Fondamenti di Informatica

Algebra di Boole e Circuiti Logici

Prof. Fedeli Massimo

IIS Fermi Sacconi Cpia

L'Algebra di Boole – 1/4

Un po' di storia

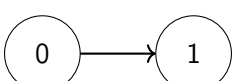
- Il matematico inglese **George Boole** nel 1847 fondò un campo della matematica e della filosofia chiamato **logica simbolica**
- **Shannon** per primo applicò la logica simbolica ai circuiti nel 1939

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, sì/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 – 3/4

- 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 – 4/4

- 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

- 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

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

$$x_1+x_2+\ldots+x_n = \begin{cases} 1 & \text{se almeno una } x_i \text{ vale 1} \\ 0 & \text{se } x_1 = x_2 = \ldots = x_n = 0 \end{cases}$$

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

Operatore OR: Possibili Rappresentazioni

- x | y ← Usato in MATLAB
- or(x, y) ← Usato in MATLAB
- x # y
- x 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

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

$$x_1 \cdot x_2 \cdot \ldots \cdot x_n = \begin{cases} 0 & \text{se almeno una } x_i \text{ vale } 0 \\ 1 & \text{se } x_1 = x_2 = \ldots = x_n = 1 \end{cases}$$

Tavola di verità

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

Operatore AND: Possibili Rappresentazioni

- x & y ← Usato in MATLAB
- and(x, y) ← Usato in MATLAB
- x 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**

In generale, la negazione di una variabile x è:

$$\bar{x} = \begin{cases} 0 & \text{se } x = 1 \\ 1 & \text{se } x = 0 \end{cases}$$

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

Proprietà

$$\bar{0} = 1$$
$$\bar{1} = 0$$
$$\bar{\bar{x}} = x$$

Operatore NOT: Possibili Rappresentazioni

- $y = \tilde{}x$ ← 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

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$
$$x \times 0 = 0$$
$$0 \times x = 0$$
$$x \times 1 = x$$
$$1 \times x = x$$

$x \times x = x$

Funzione OR

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$
$$x + 0 = x$$
$$0 + x = x$$
$$x + 1 = 1$$
$$1 + x = 1$$

$x + x = x$

Funzione NOT

$$x + \bar{x} = 1$$
$$x \times \bar{x} = 0$$
$$\bar{\bar{x}} = x$$

Legge dell'idempotenza

Algebra di Boole: Proprietà e Leggi

Proprietà Commutativa

$$x_1 x_2 = x_2 x_1$$
$$x_1 + x_2 = x_2 + x_1$$

Proprietà Distributiva

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

Proprietà Associativa

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

Leggi di Assorbimento

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

Leggi di De Morgan

$$\overline{x_1 + x_2} = \bar{x_1} \cdot \bar{x_2}$$
$$\overline{x_1 \cdot x_2} = \bar{x_1} + \bar{x_2}$$

Altre Note

$$x_1 + \bar{x_1} x_2 = x_1 + x_2$$
$$x_1(\bar{x_1} + x_2) = x_1 x_2$$

Leggi di De Morgan – 1/4

Prima Legge di De Morgan

$$\overline{x_1 + x_2} = \bar{x_1} \times \bar{x_2}$$

- 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 De Morgan – 2/4

Seconda Legge di De Morgan

$$\overline{x_1 \times x_2} = \bar{x_1} + \bar{x_2}$$

- 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 De Morgan – 3/4

- Osservazione: $\overline{\bar{x}} = x$ (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: $\overline{\overline{x_1 + x_2}} = \overline{\bar{x_1} \times \bar{x_2}}$
- Utilizzando ancora (Eq. 1) ottengo: $x_1 + x_2 = \overline{\bar{x_1} \times \bar{x_2}}$

Conclusione

L'**OR** fra x_1 e x_2 può essere espresso in termini delle sole operazioni **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: $\overline{\bar{x}} = x$ (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: $\overline{\overline{x_1 \times x_2}} = \overline{\bar{x_1} + \bar{x_2}}$
- Utilizzando ancora (Eq. 1) ottengo: $x_1 \times x_2 = \overline{\bar{x_1} + \bar{x_2}}$

Conclusione

L'**AND** fra x_1 e x_2 può essere espresso in termini delle sole operazioni **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 $x_1, x_2, \ldots, x_n$, queste possono assumere 2^n configurazioni distinte
- Ad esempio per $n = 3$ si hanno 8 configurazioni
- Ogni riga 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
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

$x_1 x_2 x_3 \rightarrow 010$

x_1	x_2	x_3	F
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

Configurazioni

$000 \rightarrow \bar{x_1}\bar{x_2}\bar{x_3}$

$001 \rightarrow \bar{x_1}\bar{x_2}x_3$

$010 \rightarrow \bar{x_1}x_2\bar{x_3}$

$011 \rightarrow \bar{x_1}x_2x_3$

$100 \rightarrow x_1\bar{x_2}\bar{x_3}$

$101 \rightarrow x_1\bar{x_2}x_3$

$110 \rightarrow x_1x_2\bar{x_3}$

$111 \rightarrow x_1x_2x_3$

- Se in una configurazione una variabile compare con 1, si assume il valore diretto
- Se compare con 0, si assume il valore negato

Definizione di Funzione Booleana

Una variabile y è **funzione** delle n variabili indipendenti $x_1, x_2, \ldots, x_n$, 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)

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

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

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

$$y = x_1 + x_2$$

Funzioni Logiche (o Booleane) – 4/5

x_1	x_2	x_3	F
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

Procedimento:

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

$$F = \bar{x_1}x_2x_3 + x_1\bar{x_2}x_3 + x_1x_2\bar{x_3} + x_1x_2x_3$$

Funzioni Logiche (o Booleane) – 5/5

$$F(x_1, x_2, x_3) = \bar{x_1}x_2x_3 + x_1\bar{x_2}x_3 + x_1x_2\bar{x_3} + x_1x_2x_3$$

Forma Canonica

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)**

Proprietà

Se un minterm assume valore 1, anche la funzione F assume il valore 1

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

Comportamento della funzione XOR

$F =$ "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:

$F =$ "L'output è 1 se (x_1 OR x_2) è 1, AND se (x_1 AND x_2) 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

L'espressione come **somma di prodotti** è:

$$\text{XOR}(x_1, x_2) = \bar{x_1} \times x_2 + x_1 \times \bar{x_2}$$

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

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

x	y	z	F
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

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

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

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

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

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

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:

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

Per ogni combinazione di input:

- Calcolare $\bar{y}$
- Calcolare $\bar{y} + z$
- Calcolare $x \times (\bar{y} + z)$

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

Le mappe di Karnaugh (1/2)

Le **mappe di Karnaugh** sono un mezzo visivo per poter manipolare e semplificare le forme algebriche di una funzione.

Per le funzioni fino a 4 variabili, i rispettivi termini sono disposti su una mappa in forma tabellare, e viene posto:

- **1** in corrispondenza del termine in cui la funzione assume valore 1
- **0** altrimenti

Esempio: funzione XOR

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

	0	1
0	0	1
1	1	0

Le mappe di Karnaugh (2/2)

Esempio con 4 variabili

x_1	x_2	x_3	x_4	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
		...		

Mappa di Karnaugh corrispondente

x_1x_2 \ x_3x_4	00	01	11	10
00	1	1	0	1
01	0	1	1	0
11	0	0	1	1
10	1	0	1	1

Proprietà importante

Nelle mappe di Karnaugh si considerano **adiacenti** i quadratini che hanno un lato in comune e quelli posti alle **estremità opposte** della mappa

Le mappe di Karnaugh – Semplificazione

Le mappe consentono di semplificare la rappresentazione in forma canonica di una funzione booleana.

x_1x_2 \ x_3x_4	00	01	11	10
00	1	1	0	1
01	0	1	1	0
11	0	0	1	1
10	1	0	1	1

Procedimento:

1. Identificare le celle adiacenti con valore 1
2. Rappresentare i termini caratterizzanti le celle così identificate

Espressione semplificata:

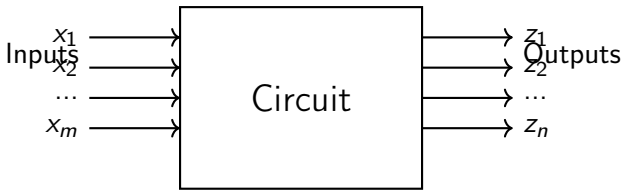
$$\begin{aligned} f(x_1, x_2, x_3, x_4) = & \bar{x_1}\bar{x_2}\bar{x_3} + \\ & x_2x_3x_4 + \\ & x_3x_4 + \\ & \bar{x_2}x_4 \end{aligned}$$

Circuito Logico

Definizione

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

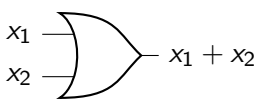
- **Building block** utilizzati per creare circuiti digitali
- Qualsiasi circuito può essere implementato usando solo porte logiche elementari (AND, OR e NOT)
  - Le cose si fanno complicate quando si hanno numerosi input ed output
- Dispositivi elettronici che implementano semplici funzioni booleane
- 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 **tavola di verità** o utilizzando la **notazione booleana**

Funzione OR: Tavola di Verità e Porta Logica

Tavola di verità

x_1	x_2	x_1 OR x_2
0	0	0
0	1	1
1	0	1
1	1	1

Porta logica

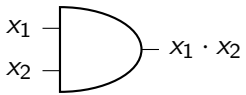


Funzione AND: Tavola di Verità e Porta Logica

Tavola di verità

x_1	x_2	x_1 AND x_2
0	0	0
0	1	0
1	0	0
1	1	1

Porta logica

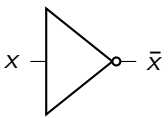


Funzione NOT: Tavola di Verità e Porta Logica

Tavola di verità

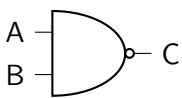
x	NOT x
0	1
1	0

Porta logica



Porta NAND

(a) Circuit symbol



(b) Truth table

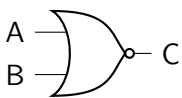
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

(c) Boolean expression

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

Porta NOR

(a) Circuit symbol



(b) Truth table

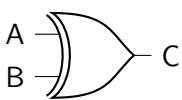
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

(c) Boolean expression

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

Porta XOR

(a) Circuit symbol



(b) Truth table

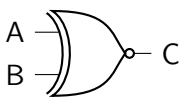
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

(c) Boolean expression

$$C = A \oplus B$$

Porta Exclusive NOR

(a) Circuit symbol



(b) Truth table

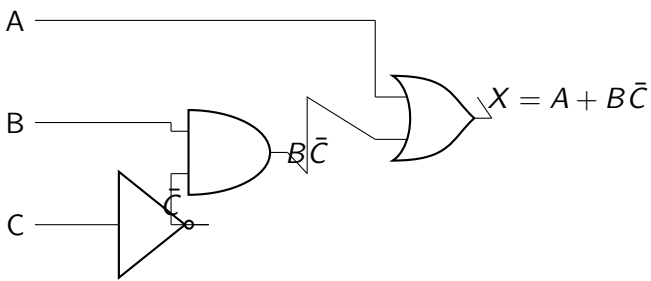
A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

(c) Boolean expression

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

Esempio 5: dalla Funzione al Circuito

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

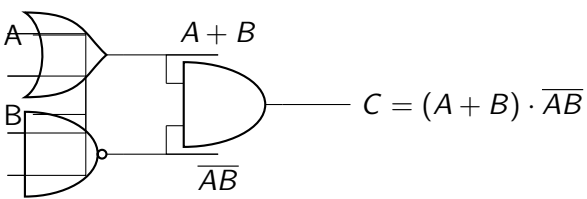


Esempio 6: dalla Funzione al Circuito

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

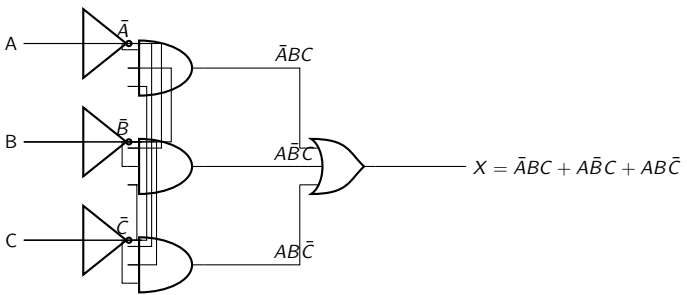
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

Porta NAND



Esempio 7: dalla Funzione al Circuito

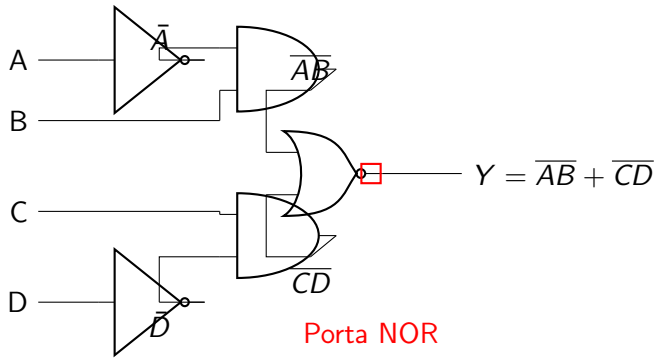
$$X = \bar{A}BC + A\bar{B}C + AB\bar{C}$$



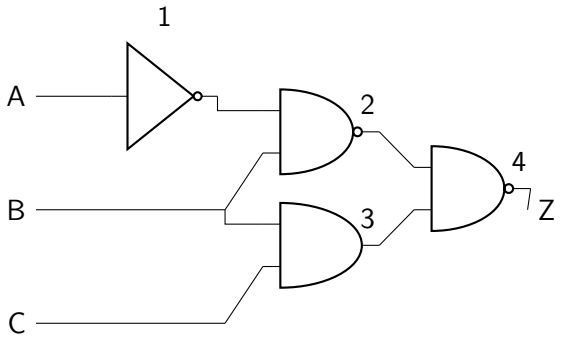
Esempio 8: dalla Funzione al Circuito

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

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

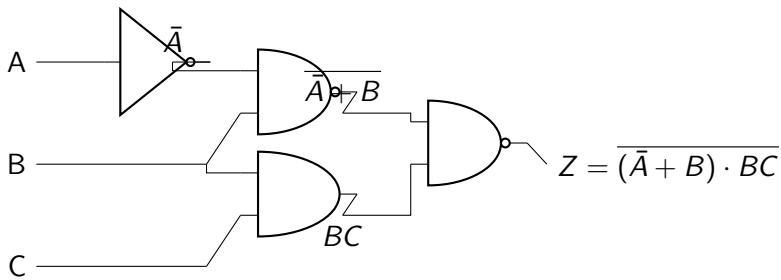


Esempio 9: dal Circuito alla Funzione – 1/2



Procedere progressivamente dagli input verso l'output

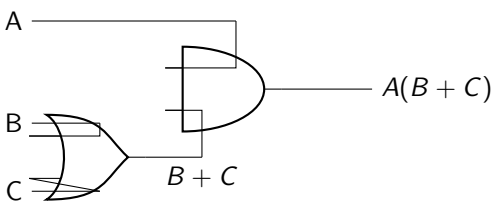
Esempio 9: dal Circuito alla Funzione – 2/2



Esempio 10: Funzione ⇒ Tavola di Verità ⇒ 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** e **Mappe di Karnaugh**
  - **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 semplificata)
  - Si trasforma l'espressione booleana in un **circuito**

Esercizio 1: determinare la funzione espressa dalla 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

Soluzione:

Identificare le righe con output 1:

- 001: $\bar{A}\bar{B}C$
- 101: $A\bar{B}C$
- 110: $AB\bar{C}$

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

Esercizio 2: minimizzare la funzione con mappa di Karnaugh

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

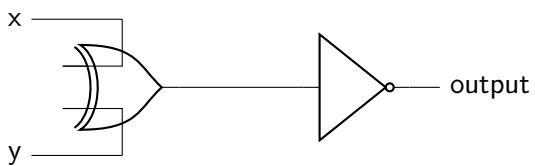
Mappa di Karnaugh:

A \ BC	00	01	11	10
0		1		
1		1		1

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

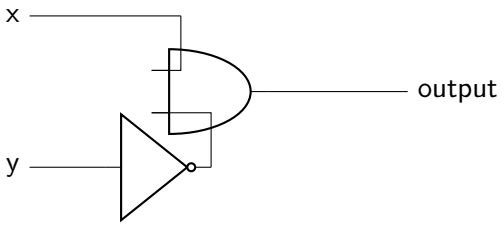
Esercizio 3: trovare l'output del circuito

Circuito da analizzare



Determinare: Tavola di verità e funzione booleana

Esercizio 4: trovare l'output del circuito

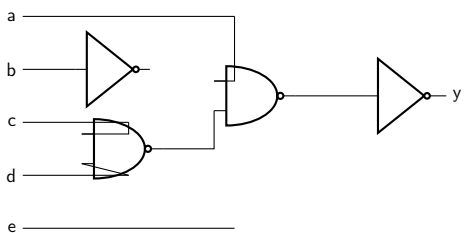


Soluzione:

$$\text{output} = x \cdot \bar{y}$$

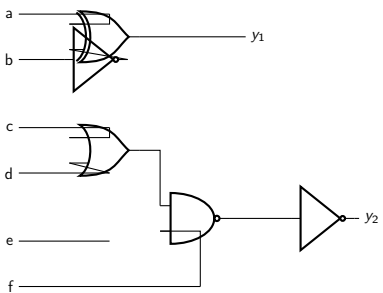
Esercizio 5: circuito complesso

Trovare l'output del seguente circuito (tavola di verità e funzione)



Esercizio 6: circuito con due output

Trovare gli output del seguente circuito



Esercizio 7: progettare circuiti per espressioni

Progettare il circuito per ciascuna delle seguenti espressioni:

1. $\bar{x} + y$
2. $(x + y) \cdot z$
3. Scrivere la funzione XOR usando AND, OR e NOT

Suggerimento per il punto 3:

$$\text{XOR}(x, y) = \bar{x}y + x\bar{y}$$

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