



I circuiti logici: definizione delle funzioni logiche

Prof. Alberto Borghese
Dipartimento di Informatica
borgnese@di.unimi.it

Università degli Studi di Milano

Riferimenti al testo: Appendice B, sezioni B.1 e B.2



Sommario

Variabili ed operatori logici semplici.

Implementazione circuitale (porte logiche).

Espressioni logiche.

Funzioni logiche e circuiti.



Le appendici del Patterson sono on-line...



<http://online.universita.zanichelli.it/patterson-5e/xstudx-appendici/>

A P P E N D I X

The Basics of Logic Design

- B.1 Introduction B-3
- B.2 Gates, Truth Tables, and Logic Equations B-4
- B.3 Combinational Logic B-9
- B.4 Using a Hardware Description Language B-20
- B.5 Constructing a Basic Arithmetic Logic

I always loved that word, Boolean.

Claude Shannon
IEEE Spectrum, April 1992
(Shannon's master's thesis showed that the algebra invented by George Boole in the 1800s could represent the



Le operazioni logiche fondamentali



All'interno di un elaboratore vengono effettuate **solo** operazioni logiche.

NOT

AND

OR

QUALUNQUE funzione booleana (logica) può essere espressa combinando opportunamente le tre funzioni logiche elementari.

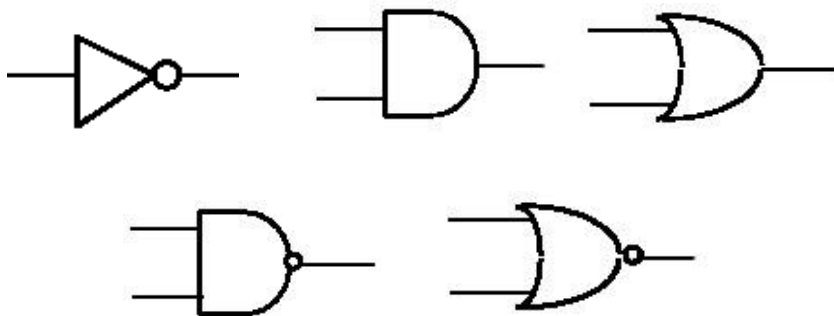
Si dice anche che AND, OR, NOT formano un **insieme completo**.



Porte logiche

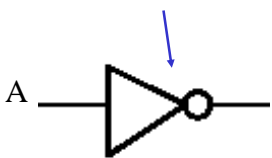


Gli **operatori logici fondamentali (AND, OR, NOT)** vengono implementati da dei circuiti elettronici che vengono chiamati **porte logiche**.



Operatore NOT



Tabella della verità		Simbolo	
A	Y		
F = 0	1 = V		Y = NOT A
V = 1	0 = F		

“Inverter logico”

Se **A** è vero ($A = \text{TRUE} = 1$), **NOT A** è falso ($\text{FALSE} = 0$) e viceversa.

Scrittura algebrica: $\text{NOT } A = \overline{A} = !A$



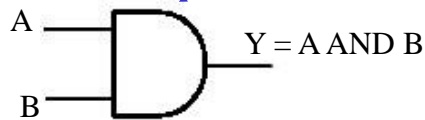
Operatore AND



A	B	Y
FF 0	0	0 F
FV 0	1	0 F
VF 1	0	0 F
VV 1	1	1 V

Tabella della verità

Simbolo



Esempio: “Se c’è il sole e c’è Michela (Michele) vado a sciare”

Scrittura algebrica: $Y = A \text{ AND } B = A \cdot B = AB$



Operatore OR



A	B	Y
FF 0	0	0 F
FV 0	1	1 V
VF 1	0	1 V
VV 1	1	1 V

Tabella della verità

Simbolo



Esempio: “Se c’è il sole o c’è Michela (Michele) vado a sciare”

Scrittura algebrica: $Y = A \text{ OR } B = A + B$



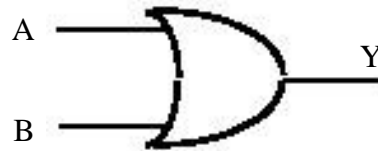
Somma e prodotto logico



Somma logica

=> OR

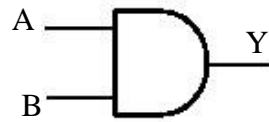
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



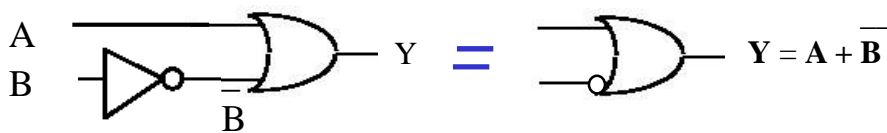
Prodotto logico

=> AND

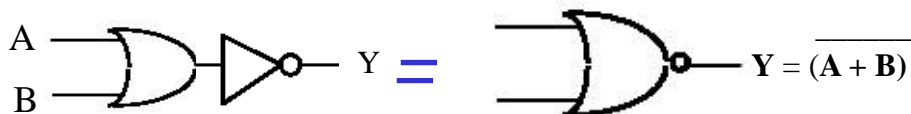
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Concatenazione del NOT



Inserire un cerchietto all'ingresso corrisponde a negare (complementare) l'ingresso.



Inserire un cerchietto all'uscita corrisponde a negare (complementare) l'uscita.

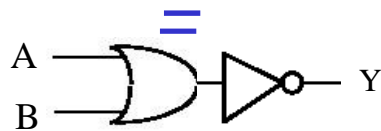
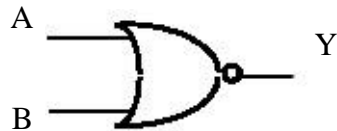


Operatore NOR



A	B	OR(A,B)	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Operatore OR negato



“Not(Or(A,B))”

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

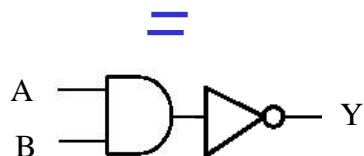
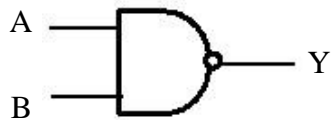


Operatore NAND



A	B	AND(A,B)	Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Operatore AND negato



“Not(And(A,B))”

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



Sommario



Variabili ed operatori logici semplici.

Implementazione circuitale (porte logiche).

Espressioni logiche.

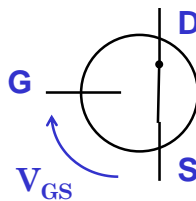
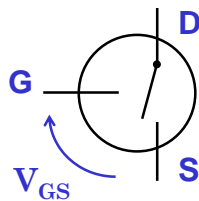
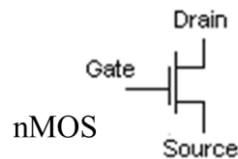
Funzioni logiche e circuiti.



Il Transistor – logica CMOS



- Componente con funzionamento lineare (amplificazione) o **binario** (interruttore).
- Modello: interruttore tra **Source** e **Drain**, **comandato** dalla tensione sul **Gate**.
- Le porte logiche sono costituite da transistor.
- 2 casi “estremi”:
 - Tensione V_{GS} **bassa** \rightarrow **D,S isolati (spento)**
 - Transistor in stato di **INTERDIZIONE**
 - Tensione V_{GS} **alta** \rightarrow **D,S collegati (acceso)**
 - Transistor in stato di **SATURAZIONE** ($V_D = V_S$)





La tecnologia CMOS: dal 1980 a oggi



- **CMOS**: Complementary–MOS (Frank Wanlass, 1967 Fairchild)

- MOS: **M**etal – **O**xide **S**emiconductor
- MOS complementari (**N**-MOS + **P**-MOS) che lavorano “in coppia”: substrati comuni.

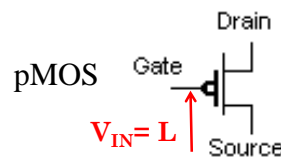
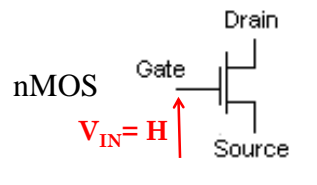
nMOS è acceso se: $V_{GS} > \text{soglia}$ ($V_{DS} < \text{soglia}$)
 $V_G \approx V_{dd}$

pMOS è acceso se: $V_{DG} > \text{soglia}$ ($V_{GS} < \text{soglia}$)
 $V_G \approx V_{ss}$

Un MOS:

- Quando è acceso funziona da corto circuito tra D e S
- Quando è spento funziona da circuito aperto tra D e S

Acceso se: $V_{in} > V_{dd}/2$



Acceso se: $V_{in} < V_{dd}/2$



La tecnologia CMOS: vantaggi



- Vantaggi:

- Tensione di alimentazione, V_{CC} , “flessibile”:

$V_{CC} = 1 \div 15$ Volt (vicina a 1 V quasi sempre)

$V_{LOW} = 0 \approx V_{ss}$ (tensione di source)

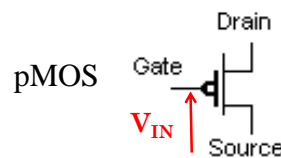
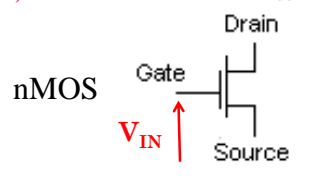
$V_{HIGH} = 1 \approx V_{dd}$ (tensione di drain)

- Consumo bassissimo:

- Consuma solo nella transizione
- In condizioni statiche, consumo praticamente nullo!

Si può vedere come un ponte levatoio che si alza e si abbassa tra Drain e Source ed è pilotato dal gate. Si consuma energia solo quando viene alzato o abbassato.

Acceso se: $V_{in} > V_{dd}/2$



Acceso se: $V_{in} < V_{dd}/2$



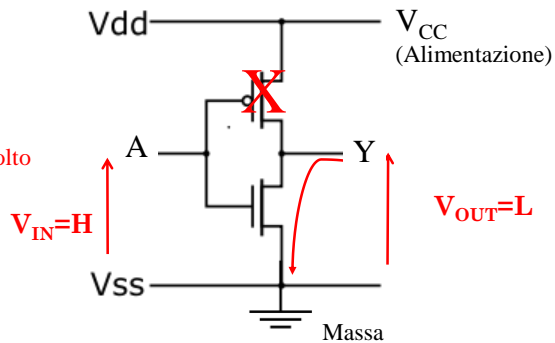
La porta NOT in CMOS



2 MOS complementari

$V_{in} = L = 0V$ il transistor inferiore è spento, $V_{out} = V_{dd} = H$

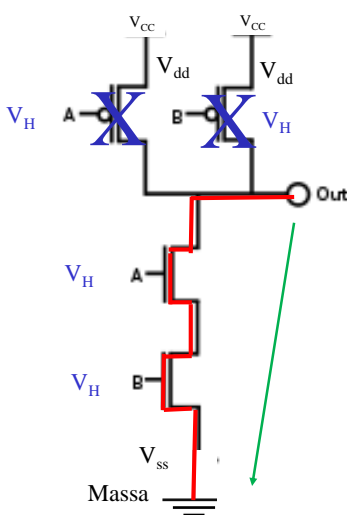
$V_{in} = H = V_{dd}$ passa corrente nel transistor inferiore, la resistenza è molto bassa e $V_{out} \cong 0 = L$. Il transistor superiore è spento.



La porta NOT è chiamata anche Inverter logico.



Porta NAND in C-MOS

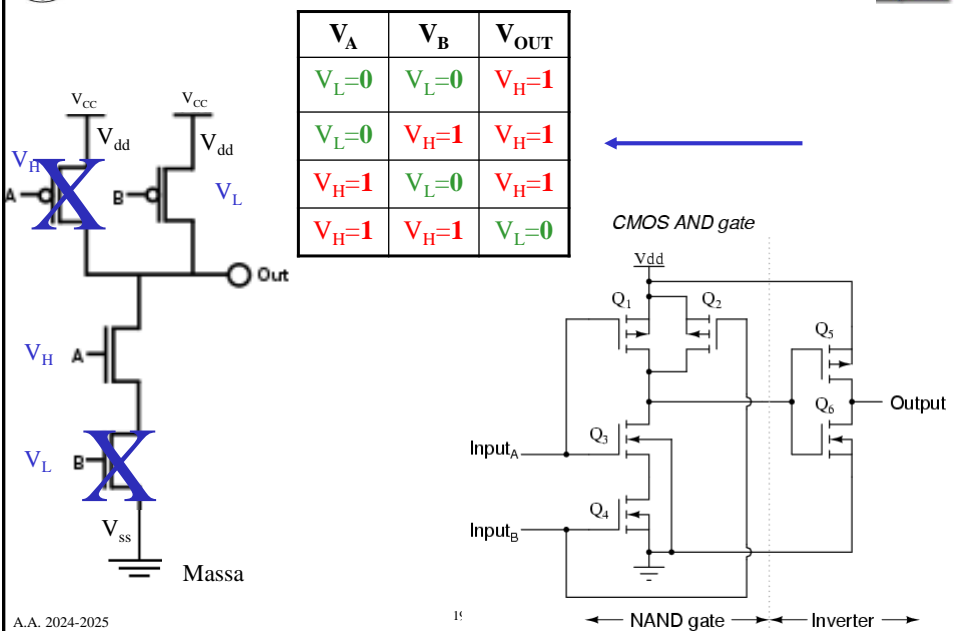


V_A	V_B	V_{OUT}
$V_L=0$	$V_L=0$	$V_H=1$
$V_L=0$	$V_H=1$	$V_H=1$
$V_H=1$	$V_L=0$	$V_H=1$
$V_H=1$	$V_H=1$	$V_L=0$

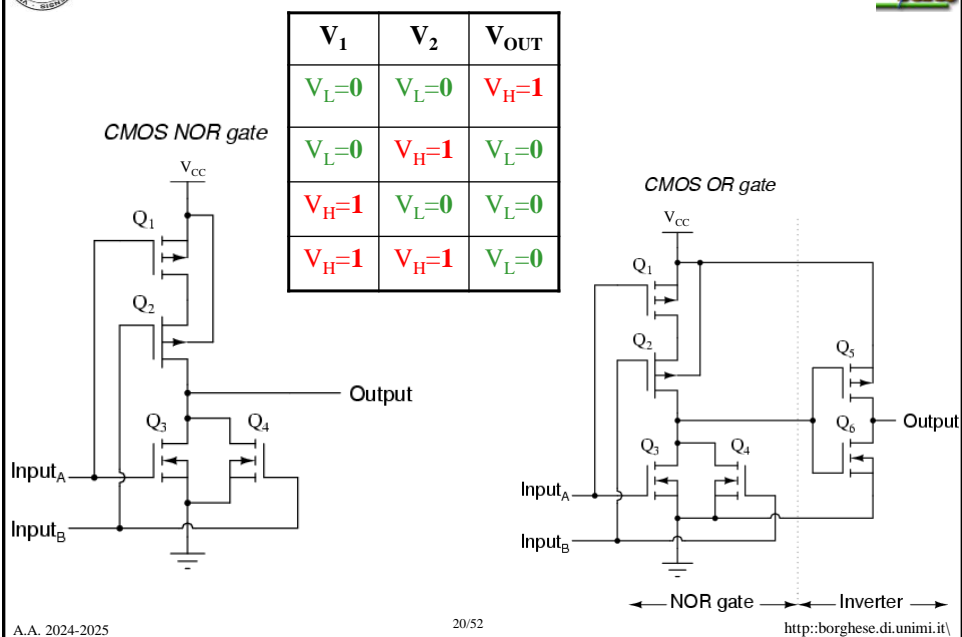
$V_A = 1; V_B = 1 \rightarrow V_{out} = V_{ss} = 0$



Porta AND in C-MOS



PORTA NOR e OR IN CMOS



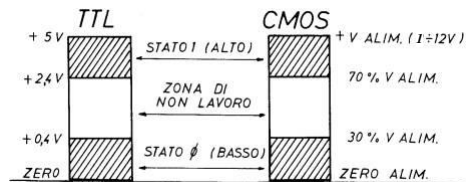


Perchè l'elettronica digitale funziona?



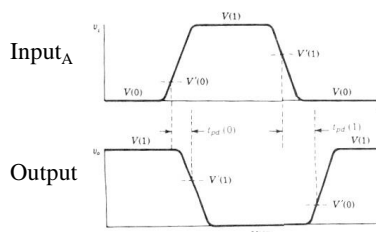
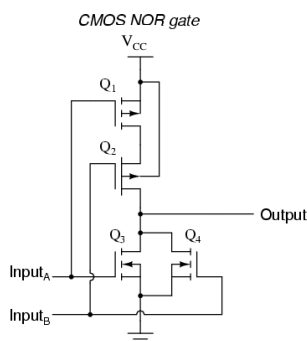
Perchè è progettata per essere resistente al rumore.

Vengono definiti 2 range di tensioni associati ai valori alto e basso, separati da un gap.



Tempo di commutazione

La commutazione non è istantanea



Più porte in cascata generano un ritardo nella commutazione dell'uscita.

Numero limitato di porte logiche che si possono collegare in ingresso e in uscita (fan-in / fan-out)



Sommario



Variabili ed operatori semplici.

Implementazione circuitale (porte logiche).

Espressioni logiche.

Funzioni logiche e circuiti.



Espressioni logiche



Sono espressioni algebriche costruite con variabili binarie e gli operatori logici fondamentali.

Sono comparabili alle operazioni sono monomi e polinomi (non viene utilizzato l'elevamento a potenza)

Queste espressioni si possono manipolare per:

- Semplificarle
- Isolare termini di interesse
- Privilegiare la presenza di operatori AND rispetto a operatori OR o viceversa



Precedenza degli operatori



- 1) NOT
- 2) AND
- 3) OR

$Y = A + B \overline{C}$ va intesa come: $Y = A + (B \cdot \overline{C})$

Ordine di esecuzione:

1) \overline{C} - negazione

2) $B \cdot \overline{C}$ - AND

3) $A + B \cdot \overline{C}$ - OR

Sarebbe sbagliato leggere l'espressione logica come: $Y = (A+B) \cdot \overline{C} \neq \overline{A+B \cdot C}$



Concatenazione degli operatori - I



In assenza di parentesi,

- AND ha la priorità sull'OR
- NOT ha la priorità su entrambi

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

per eseguire prima OR occorre scrivere esplicitamente: $(A+B) \cdot C$

In assenza di parentesi, la negazione ha la priorità sugli altri operatori.

$$\overline{A} \cdot C = (\text{NOT}(A)) \cdot C = !A \cdot C$$



Concatenazione degli operatori - II



In assenza di parentesi,

- AND ha la priorità sull'OR
- NOT ha la priorità su entrambi

Negazione di un'espressione

$\overline{A \ B \ C}$ La negazione riguarda l'AND(A,B). Quindi eseguo prima l'AND di A e B, e nego poi il risultato dell'AND ed infine eseguo l'AND con C:

- 1) \overline{AB}
- 2) $\overline{(AB)}$
- 3) $\overline{(AB) \ C}$

ni.it\



Concatenazione degli operatori - III



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

- 1) Prima eseguo la negazione di A e la negazione di B in parallelo.
- 2) Poi eseguo l'AND.

$$\overline{\overline{A} \ B \ C} = \overline{(\overline{A}) \ (\overline{B}) \ C}$$

- 1) Prima eseguo la negazione di A e la negazione di B in parallelo.
- 2) Eseguo poi il prodotto logico di !A, !B
- 3) Eseguo poi il prodotto logico del risultato con C.

Anche sulle negazioni esiste una gerarchia:

$$\overline{\overline{A} \ \overline{B}} = \overline{(\overline{A}) \ (\overline{B})} = ! \ (\ (!A) \ (!B))$$

- 1) Prima eseguo la negazione di A e la negazione di B in parallelo.
- 2) Eseguo poi l'AND di \overline{A} e $\overline{B} = \text{AND}(!A, !B)$
- 3) Infine nego il risultato dell'AND

A.A



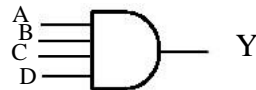
Porte logiche a più ingressi



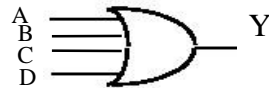
- Rappresentano circuiti che forniscono in uscita il risultato di operazioni logiche elementari applicate a più variabili in ingresso.
- Le variabili in ingresso possono essere n.

Ad esempio:

$$Y = A \text{ AND } B \text{ AND } C \text{ AND } D$$



$$Y = A \text{ OR } B \text{ OR } C \text{ OR } D$$

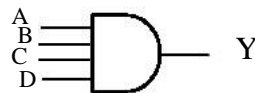


Porte logiche: tabella della verità

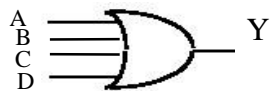


A	B	C	D	AND	OR
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

$$Y = A \text{ AND } B \text{ AND } C \text{ AND } D$$



$$Y = A \text{ OR } B \text{ OR } C \text{ OR } D$$





Manipolazione algebrica - I



	AND	OR
Identità	$1 x = x$	$0 + x = x$
<i>Algebra classica</i>	$1 N = N$	$0 + N = N$

Elemento nullo	$0 x = 0$	$1 + x = 1$
<i>Algebra classica</i>	$0 N = 0$	-----

Uno degli ingressi della porta AND funge da interruttore:
 = 0, uscita è sempre =0, interruttore aperto
 = 1, uscita è uguale all'ingresso, interruttore chiuso.

Inverso	$\bar{x} x = 0$	$\bar{x} + x = 1$
<i>Algebra classica</i>	-----	-----

Idempotenza	$x x = x$	$x + x = x$
<i>Algebra classica</i>	-----	-----



Manipolazione algebrica - II



Doppia Inversione	$\bar{\bar{x}} = x$	-----
<i>Algebra classica</i>	-----	-----

Associativa	$(x y) z = x (y z)$	$(x + y) + z = x + (y + z)$
<i>Algebra classica</i>	$(NM)K = N(MK)$	$(N+M)+K = N+(M+K)$

Commutativa	$x y = y x$	$x + y = y + x$
<i>Algebra classica</i>	$NM = MN$	$N + M = M + N$

AND rispetto ad OR	OR rispetto ad AND	
Distributiva	$x (y + z) = x y + x z$	$x + y z = (x + y) (x + z)$
<i>Algebra classica</i>	$N(M+K) = NM+NK$	-----

Assorbimento	$x (x + y) = x$	$x + x y = x$
<i>Algebra classica</i>	-----	-----



Principio di dualità



Nell'algebra di Boole vale il principio di dualità.

Il duale di una funzione booleana si ottiene sostituendo AND ad OR, OR ad AND, gli 0 agli 1 e gli 1 agli 0.

Esempi:

- | | | |
|------------------|-----------|-------------|
| • Identità | $1 x = x$ | $0 + x = x$ |
| • Elemento nullo | $0 x = 0$ | $1 + x = 1$ |

Proprietà Commutativa:

$$x y = y x$$

$$x + y = y + x$$

• **Proprietà distributiva:**

$$x (y + z) = xy + xz$$

$$x + (y z) = x + yz$$

• **Assorbimento:**

$$x (x + y) = x$$

$$x + x y = x$$



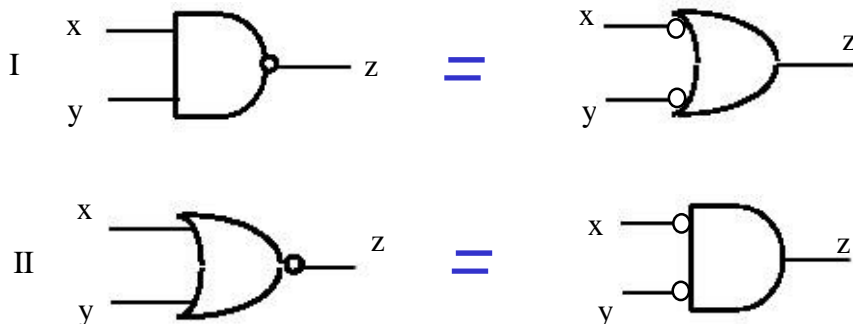
Teoremi di De Morgan



Enunciati:

$$I) \overline{xy} = \overline{x} + \overline{y}$$

$$II) \overline{x + y} = \overline{x} \overline{y}$$





Regole algebriche - riassunto



Doppia Inversione

$$\overline{\overline{x}} = x$$

AND

$$1 \cdot x = x$$

$$0 \cdot x = 0$$

$$x \cdot x = x$$

$$x \cdot \overline{x} = 0$$

$$x \cdot y = y \cdot x$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

AND rispetto ad OR

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

$$x \cdot (x + y) = x$$

OR

$$0 + x = x$$

$$1 + x = 1$$

$$x + x = x$$

$$x + \overline{x} = 1$$

$$x + y = y + x$$

$$(x + y) + z = x + (y + z)$$

OR rispetto ad AND

$$x + y \cdot z = (x + y) \cdot (x + z)$$

$$x + x \cdot y = x$$

Identità

Elemento nullo

Idempotenza

Inverso

Commutativa

Associativa

Distributiva

Assorbimento

De Morgan

$$\overline{xy} = \overline{x} + \overline{y}$$

$$\overline{x + y} = \overline{x} \cdot \overline{y}$$



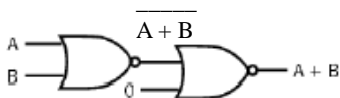
Porta Universale NOR



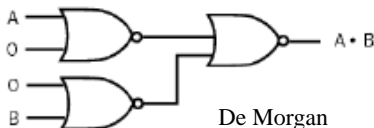
- NOT A = 0 NOR A
- A OR B = (A NOR B) NOR 0
- A AND B = (A NOR 0) NOR (B NOR 0)



NOT



OR



AND

De Morgan

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

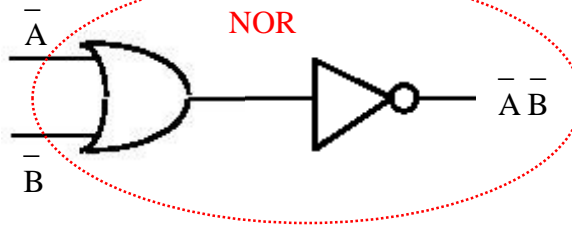
$$AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$$



De Morgan NOR -> AND

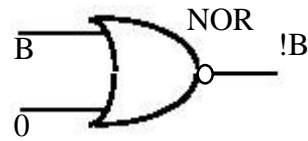
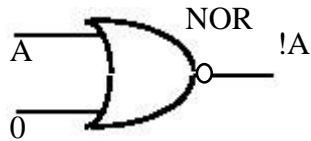


De Morgan: $A+B = \overline{\overline{A} \cdot \overline{B}}$

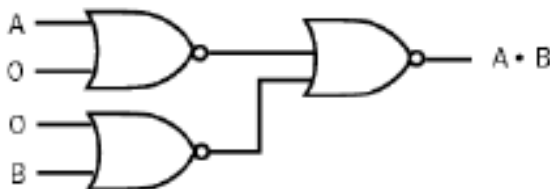
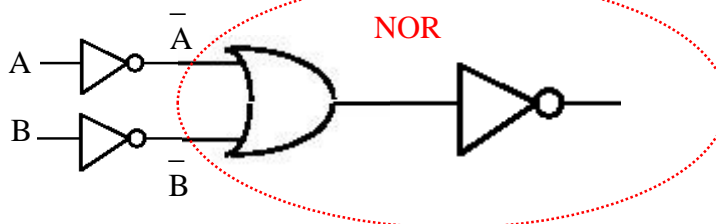


Questo circuito con NOR è equivalente a $AND(\overline{\overline{A}}, \overline{\overline{B}})$. Come faccio con le negazioni?

Applico la negazione agli ingressi:



De Morgan NOR -> AND

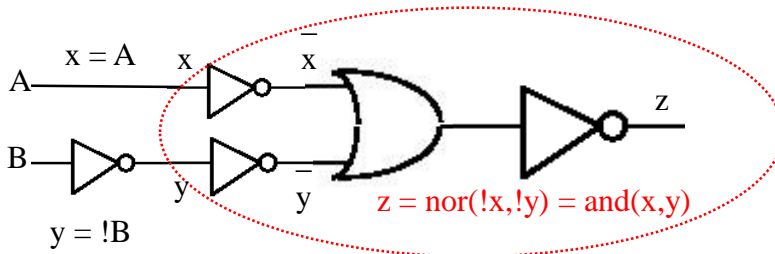




De Morgan in generale



Quale funzione è implementata da questo circuito?



Sostituisco il not e ottengo: $z = \text{and}(A,B) = A \cdot B$



Esercizio



Data la funzione booleana:

$$F = (A \text{ AND } B) \text{ OR } (B \text{ AND NOT}(C))$$

Esprimere la funzione F con il solo connettivo logico NOR e disegnare il circuito.

Esprimere la funzione F con il solo connettivo logico NAND e disegnare il circuito.

Costruire le porte logiche: AND, OR, NOT utilizzando solo la porta NAND.



Sommario



Variabili ed operatori logici semplici.

Implementazione circuitale (porte logiche).

Espressioni logiche.

Funzioni logiche e circuiti.



Funzione



- Una relazione che associa ad ogni elemento dell'insieme $X: \{x\}$, detto dominio, un elemento dell'insieme $Y: \{y\}$, detto codominio, indicandola con $y = f(x)$ (Wikipedia). Sinonimi: mappa, trasformazione.
- Le nostre funzioni saranno tipicamente relazioni tra ingressi e uscite binarie multi-dimensionali (più variabili in ingresso e più variabili in uscita).
- Nulla è detto sulla forma di questa relazione.
 - Funzione analitica (e.g. $Y = \sin(x) \log(\cos(x/2))...$)
 - Funzione a più valori di input e più valori di output (x e y vettori)
 - Tabella di corrispondenza (tabella di verità, caratteristica: descrizione **esplicita** ed **esaustiva** della corrispondenza tra tutti gli elementi del dominio e quelli del codominio).
 -
- **Funzione logica.** Corrispondenza tra un insieme di ingresso (valori possibili di A , B , C) e un insieme di uscita (valori possibili di Y).



Funzioni logiche (Booleane)



“An Investigation of the Laws of Thought on Which to Found the Mathematical Theories of Logic and Probabilities” G. Boole, 1854: approccio alla logica come algebra.

Variabili (binarie, 0 = FALSE; 1 = TRUE).

Operazioni logiche sulle variabili (NOT, AND, OR) come operazioni sui numeri binari.

Equivalenza tra operazioni logiche su proposizioni vere/false e operazioni algebriche su variabili binarie.

Utilizzo dell'algebra Booleana per:

- Progettazione (sintesi) dei circuiti digitali. Data una certa funzione logica, sviluppare il circuito digitale che la implementa (implementeremo anche le operazioni aritmetiche come funzioni logiche!).
- Analisi dei circuiti. Descrizione della funzione logica implementata dai circuiti.
- Semplificazione di espressioni logiche per ottenere implementazioni efficienti.

Ma anche:

- Verifica dei protocolli
- Ottimizzazione con vincoli.



Funzioni logiche



- Funzione a n ingressi e m uscite. Per ciascuna delle 2^n combinazioni degli ingressi, viene definita ciascuna delle m uscite.
- E' una funzione discreta con domini finiti. **Si può calcolare per tutti i punti nei quali la funzione è definita.**
- La funzione logica sarà implementata da un'opportuna combinazione di porte logiche (NOT, AND, OR): prende gli n ingressi e li trasforma nelle m uscite specificate.
- La funzione logica sarà quindi calcolata da un circuito con n ingressi e m uscite, costituito da porte logiche.
- La funzione può essere rappresentata in 3 modi:
 - Circuito
 - Tabella della verità (Truth Table, TT)
 - Espressione simbolica



Rappresentazione delle funzioni logiche



	A	B	C	Y
0	F=0	F=0	F=0	F=0
1	F=0	F=0	V=1	V=1
2	F=0	V=1	F=0	F=0
3	F=0	V=1	V=1	F=0
4	V=1	F=0	F=0	F=0
5	V=1	F=0	V=1	V=1
6	V=1	V=1	F=0	V=1
7	V=1	V=1	V=1	V=1

$(A,B,C) \rightarrow Y$

Input (dominio) Output (codominio)

Tabella della verità (truth table – TT)

Vengono considerate tutte le combinazioni possibili in input, terna A,B,C, dove ciascuna variabile può assumere i valori (V,F). Le combinazioni in ingresso si possono indicizzare con i numeri interi: 0 (000), 1 (001), 2 (010), 3 (011), 4 (100), 5 (101), 6 (110), 7 (111).

Per ogni combinazione in input viene prescritto l’output, Y, desiderato (V,F).
Rappresentazione esaustiva di una funzione.

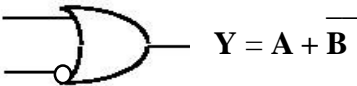


Esempio di tabella della verità



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

$Y = \text{OR}(A, \overline{B})$



Questa funzione è diversa dalla funzione $\text{OR}(A,B)$
La tabella della verità di $Y = \text{OR}(A, \overline{B})$ è diversa dalla TT della funzione OR.



Dall'espressione logica alla TT



Data l'espressione logica: $F = A \text{ AND } B \text{ OR } B \text{ AND NOT } C = AB + B\bar{C}$

Questa espressione si legge: $F = (A \text{ AND } B) \text{ OR } (B \text{ AND } (!C))$

Ricaviamo la tabella delle verità:

A B C	A and B	B and (!C)	F
0 0 0	0	0	0
0 0 1	0	0	0
0 1 0	0	1	1
0 1 1	0	0	0
1 0 0	0	0	0
1 0 1	0	0	0
1 1 0	1	1	1
1 1 1	1	0	1

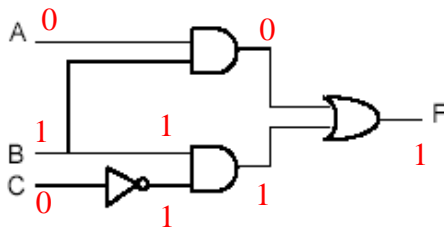


Dal circuito alla TT - 1



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

$$F = (A \text{ AND } B) \text{ OR } (B \text{ AND NOT}(C))$$



Direzione del calcolo

A B C	F
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

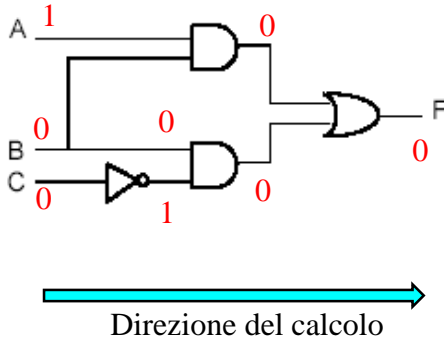


Dal circuito alla TT - 2



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

$$F = (A \text{ AND } B) \text{ OR } (B \text{ AND NOT}(C))$$



A	B	C	F
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

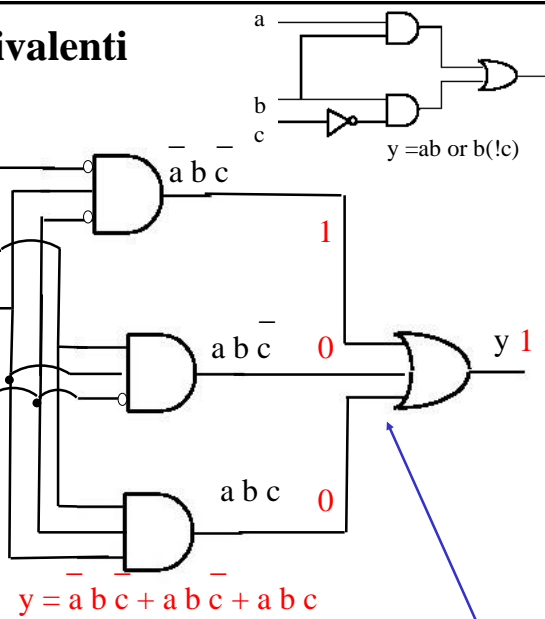


Circuiti equivalenti

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

Espressioni algebriche e tabella danno la stessa uscita y

Espressioni algebriche diverse (e circuiti diversi) possono implementare la stessa funzione logica (stessa TT)



Implementazione circuitale diversa. L'implementazione circuitale di una funzione logica non è unica!

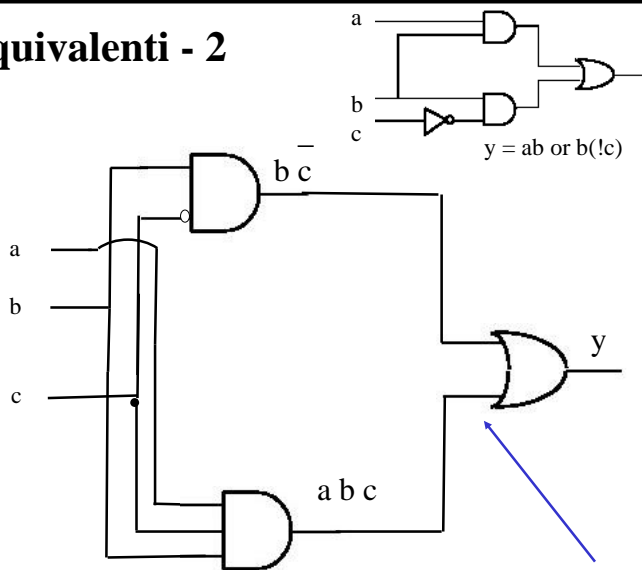


Circuiti equivalenti - 2

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

**Funzione e tabella
danno la stessa uscita y.**

**Espressioni algebriche diverse
e circuiti diversi possono
implementare la stessa
funzione logica (stessa TT).**



Implementazione circuitale diversa. L'implementazione circuitale di una funzione logica (descritta da una TT) non è unica!

51/52

<http://borgnese.di.unimi.it/>



Sommario



Variabili ed operatori logici semplici.

Implementazione circuitale (porte logiche).

Circuiti combinatori.

Funzioni logiche e circuiti.