I circuiti logici:
definizione delle funzioni logiche

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

Università degli Studi di Milano

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

Sommario

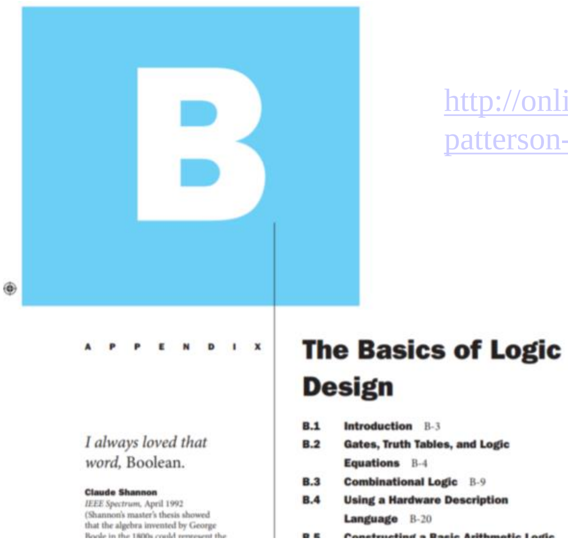
Variabili ed operatori semplici.

Implementazione circuitale (porte logiche).

Dal circuito alla funzione.

Algebra Booleana.

Le appendici del Patterson sono on-line…



B

A P P E N D I X

The Basics of Logic Design

I always loved that word, Boolean.

Claude Shannon
IEEE Spectrum, April 1992
(Shannon's master's thesis showed
that the algebra invented by George

http://online.universita.zanichelli.it/
patterson-4e/xstudx-appendici/

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 tre funzioni booleane elementari.

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

Circuiti Booleani

"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 sulle variabili (NOT, AND, OR).

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.

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) → Y

Input (dominio) Output (codominio)

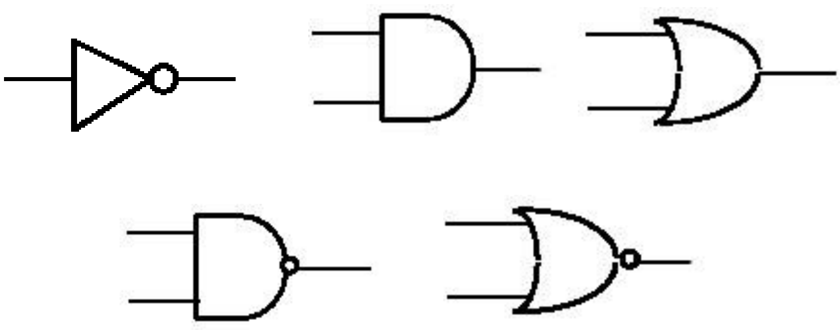
Tabella della verità (truth table – TT)

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.

Porte logiche

Gli **operatori logici** vengono implementati da dei circuiti elettronici che vengono chiamati **porte logiche**

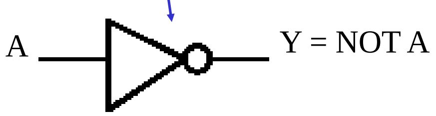


Operatore NOT

Tabella della verità

A	Y
F = 0	1 = V
V = 1	0 = F

Simbolo



"Inverter logico"

Se **A** *è vero (A =* ***TRUE****=1),* ***NOT*** **A** *è falso (****FALSE****=0) e viceversa.*

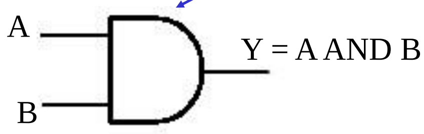
Scrittura algebrica: ***NOT*** **A** = $\overline{\mathbf{A}}$ = **!A**

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"*

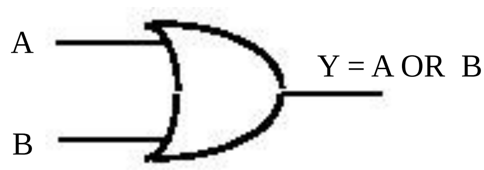
Scrittura algebrica: Y = A ***AND*** B = A · B = AB

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 ***OR*** B = A + B

Somma e prodotto logico

Somma logica

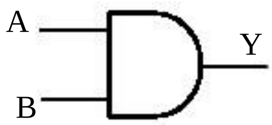
=> OR

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

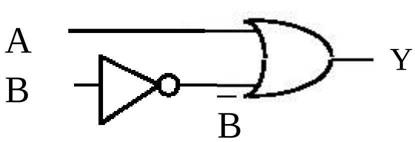
Prodotto logico

=> AND

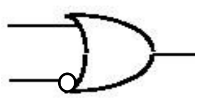
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Concatenazione del NOT

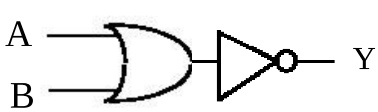


=

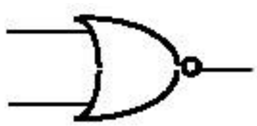


$\mathbf{Y} = \mathbf{A} + \overline{\mathbf{B}}$

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



=



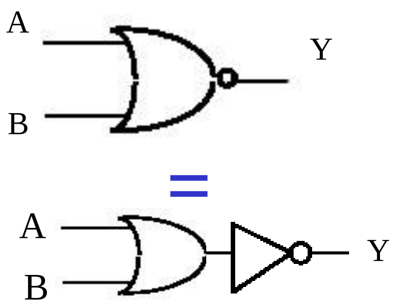
$\mathbf{Y} = \overline{(\mathbf{A} + \mathbf{B})}$

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



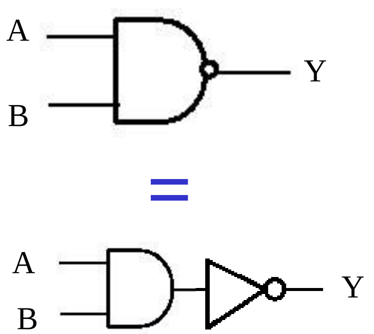
"Not(Or(A,B))"

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

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



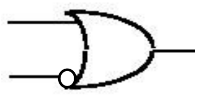
"Not(And(A,B))"

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

Tabella della verità

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

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



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

Questa funzione è diversa dalla funzione OR(A,B)
La tabella della verità è diversa.

Precedenza degli operatori

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

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

Ordine di esecuzione:

1) $\overline{C}$ - negazione

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

3) $A + B\overline{C}$

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

Concatenazione degli operatori - I

In assenza di parentesi, AND ha la priorità sull'OR ed il NOT 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.

NOT $A \cdot C$ = (NOT(A)) $\cdot$ C = $\bar{A}C$

$\overline{AB}\,C$ Eseguo prima la negazione. Quindi eseguo prima l'AND di A e B, nego il risultato dell'AND ed infine eseguo l'AND con C:

1) AB
2) !(AB)
3) (!AB) C)

Concatenazione degli operatori - II

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

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

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

1) Prima eseguo la negazione di A e la negazione di B in parallelo.
2) Eseguo il prodotto logico di !A, !B e C.

Anche sulle negazioni esiste una gerarchia:

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

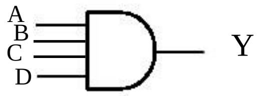
1) Prima eseguo la negazione di A e la negazione di B in parallelo.
2) Eseguo poi l'AND di !A e !B = AND(!A, !B)
3) Infine nego il risultato dell'AND

Porte logiche a più ingressi

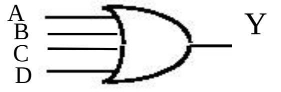
- Rappresentano circuiti che forniscono in uscita il risultato di operazioni logiche elementari applicate a più variabili in ingresso.
- Le variabili in ingesso possono essere n.

Ad esempio:

Y = A AND B AND C AND D



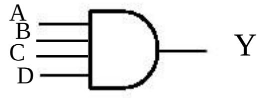
Y = A OR B OR C OR D



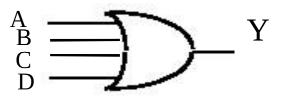
Porte logiche: tabella della verità

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 AND B AND C AND D



Y = A OR B OR C OR D



Sommario

Variabili ed operatori semplici.

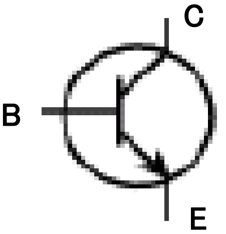
Implementazione circuitale (porte logiche).

Dal circuito alla funzione.

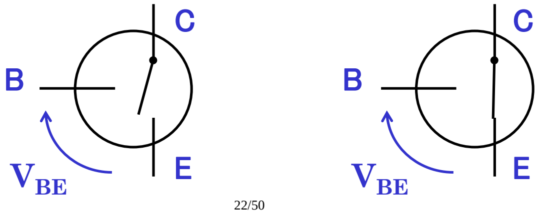
Algebra Booleana.

Il Transistor

- Componente con funzionamento lineare (amplificazione) o **binario** (interruttore).
- Modello: interruttore tra **Emettitore** e **Collettore**, **comandato** dalla tensione sulla **Base.**
- Le porte logiche sono costituite da transistor.



- 2 casi "estremi":
  - Tensione **V_{BE} bassa → C,E isolati (spento)**
    - Transistor in stato di **INTERDIZIONE**
  - Tensione **V_{BE} alta → C,E collegati (acceso)**
    - Transistor in stato di **SATURAZIONE (V_C = V_E)**

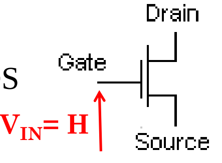


La tecnologia CMOS: dal 1980 a oggi

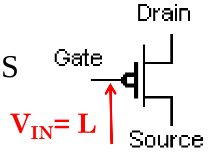
- CMOS: Complementary–MOS (Frank Wanlass, 1967 Fairchild)
  - MOS: Metal – Oxide Semiconductor
  - MOS complementari (N-MOS + P-MOS) che lavorano "in coppia": substrati comuni.

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

nMOS



pMOS



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

nMOS è acceso se: V_{GS} > soglia
$V_G \approx V_{dd}$

pMOS è acceso se: V_{DG} > soglia (V_{GS} < 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

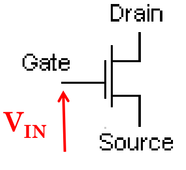
La tecnologia CMOS: vantaggi

- Vantaggi:
  - Tensione di alimentazione, V_{CC}, "flessibile":
    V_{CC} = 1 ÷ 15 Volt (vicina a 1 V quasi sempre)

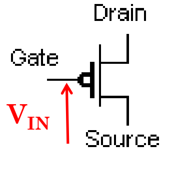
    V_{LOW} = 0 ≈ V_{ss} (tensione di source)
    V_{HIGH} = 1 ≈ V_{dd} (tensione di drain)
  - Consumo bassissimo:
    - Consuma solo nella transizione
    - In condizioni statiche, consumo praticamente nullo!

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

nMOS



pMOS



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

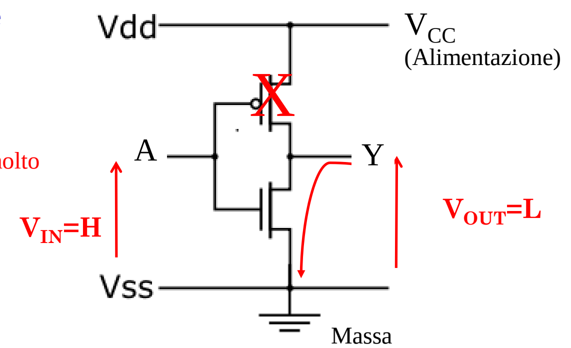
Si può vedere come un ponte levatoio che si alza e si abbassa tra Drain e Source ed è pilotato dal gate. Si consuma energia solo quando viene alzato o abbassato.

La porta NOT in CMOS

2 MOS complementari

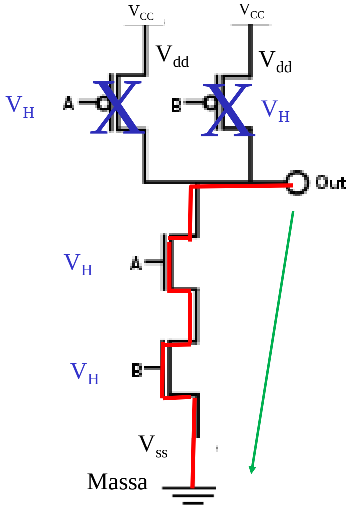
V_{in} = L = 0V il transistor inferiore è spento, $V_{out} = V_{dd}$ = H

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

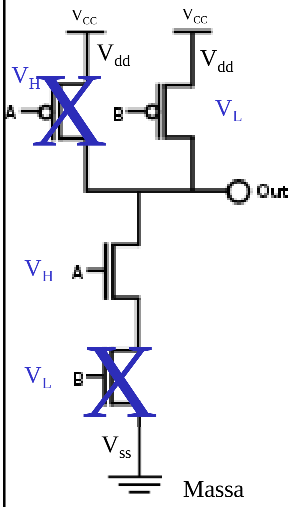


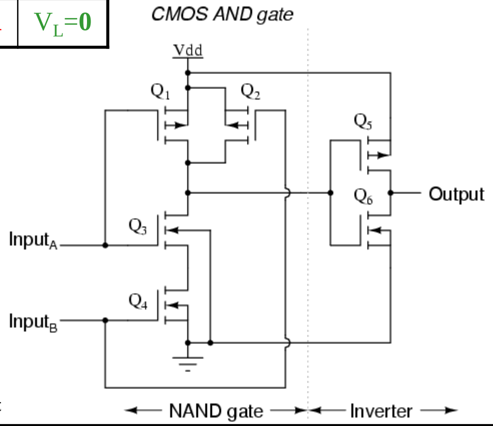
V_A	V_B	V_{OUT}
V_L=0	V_L=0	V_H=1
V_L=0	V_H=1	V_H=1
V_H=1	V_L=0	V_H=1
V_H=1	V_H=1	V_L=0

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

Porta AND in C-MOS

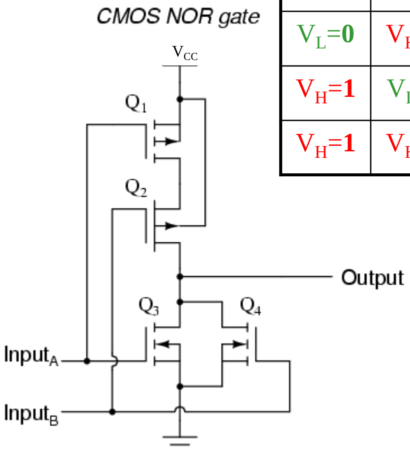
V_A	V_B	V_{OUT}
V_L=0	V_L=0	V_H=1
V_L=0	V_H=1	V_H=1
V_H=1	V_L=0	V_H=1
V_H=1	V_H=1	V_L=0

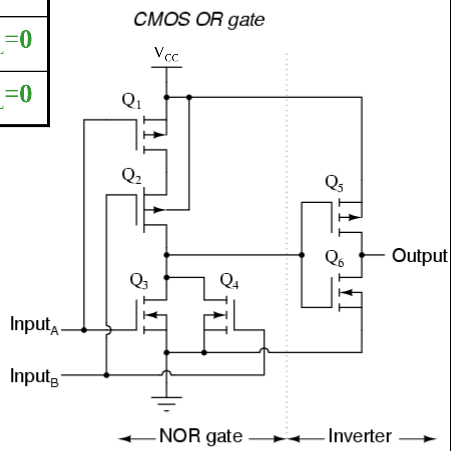




PORTA NOR e OR IN CMOS

V_1	V_2	V_{OUT}
V_L=0	V_L=0	V_H=1
V_L=0	V_H=1	V_L=0
V_H=1	V_L=0	V_L=0
V_H=1	V_H=1	V_L=0

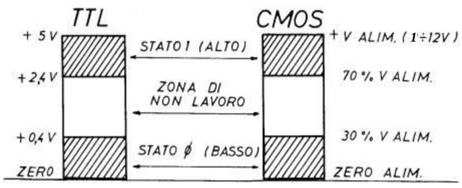




Perchè l'elettronica digitale funziona?

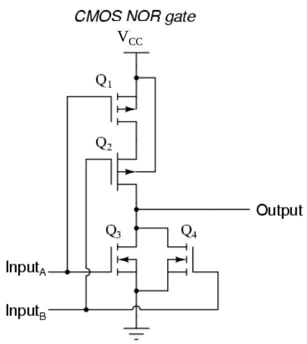
Perchè è progettata per essere resistente al rumore.

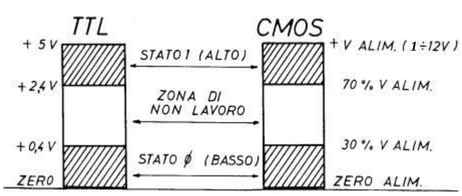
Vengono definiti 2 range di tensioni associati ai valori alto e basso, separati da un gap.



Tempo di commutazione

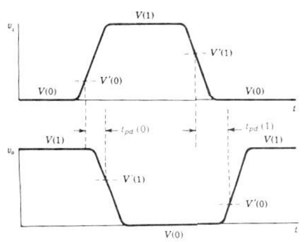
La commutazione non è istantanea





$Input_A$

Output



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).

Dal circuito alla funzione.

Algebra Booleana.

Funzione

- Una relazione che associa ad ogni elemento dell'insieme X:{x}, detto dominio, associa 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 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 ed y vettori)
  - Tabella di corrispondenza (tabella di verità, caratteristica: descrizione esplicita ed esaustiva della corrispondenza).
  - ....

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) che prende gli n ingressi e li trasforma nell'uscita specificata.
- 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

Dall'espressione logica alla TT

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

Questa espressione si legge: F = (A AND B) OR (B AND (!C))

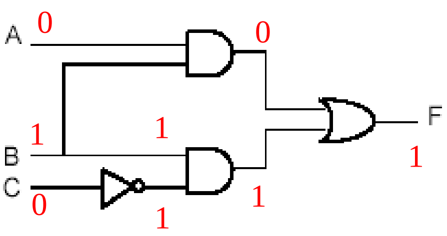
Ricaviamo la tabella delle verità:

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

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

F = (A AND B) OR (B AND NOT(C))



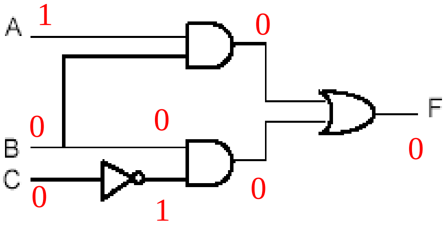
Direzione del calcolo

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

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

F = (A AND B) OR (B AND NOT(C))

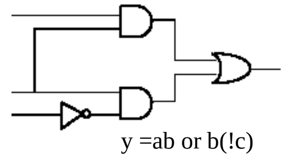


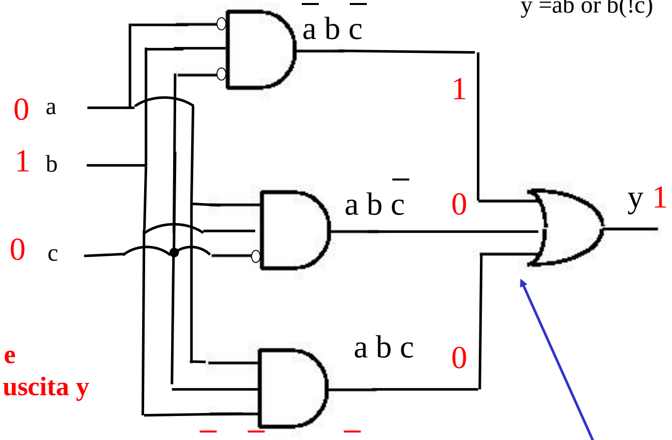
Direzione del calcolo

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

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





$y = \bar{a}\, b\, \bar{c} + a\, b\, \bar{c} + a\, b\, c$

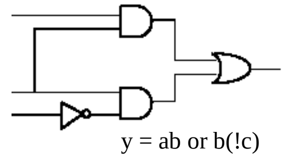
Espressioni algebriche e tabella danno la stessa uscita y

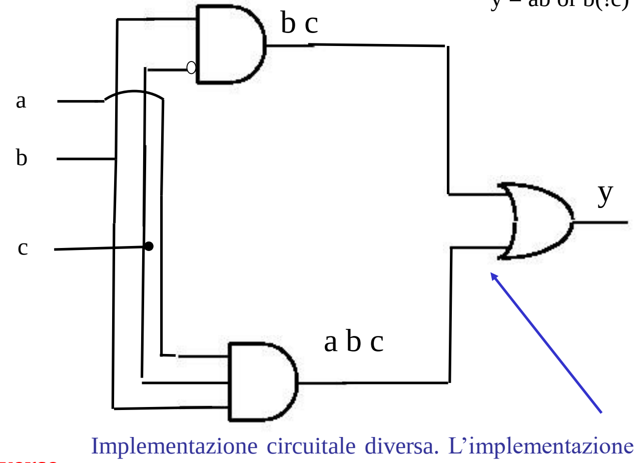
Espressioni algebriche diverse (e circuiti diversi) possono implementare la stessa funzione logica (stessa TT)

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!

$y = a\, b\, c + b\, \bar{c}$

Sommario

Variabili ed operatori semplici.

Implementazione circuitale (porte logiche).

Dal circuito alla funzione.

Algebra Booleana.

Manipolazione algebrica - I

	AND	OR
Identità	$1\ x = x$	$0 + x = x$
Algebra classica	$1\ N = N$	$0 + N = N$
Elemento nullo	$0\ x = 0$	$1 + x = 1$
Algebra classica	$0\ N = 0$	-----------

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

	AND	OR
Inverso	$\overline{x}\ x = 0$	$\overline{x} + x = 1$
Algebra classica	---------	-----------
Idempotenza	$x\ x = x$	$x + x = x$
Algebra classica	-----------	-----------

Manipolazione algebrica - II

Doppia Inversione	$\overline{\overline{x}} = x$	
Algebra classica	-----------	-----------
Associativa	$(x\,y)\,z = x\,(y\,z)$	$(x + y) + z = x + (y + z)$
Algebra classica	$(NM)K = N(MK)$	$(N+M)+K = N+(M+K)$
Commutativa	$x\,y = y\,x$	$x + y = y + x$
Algebra classica	$NM = MN$	$N + M = M + N$
AND rispetto ad OR	*OR rispetto ad AND*	
Distributiva	$x\,(y + z) = x\,y + x\,z$	$x + y\,z = (x + y)\,(x + z)$
Algebra classica	$N(M+K) = NM+NK$	---------------------------
Assorbimento	$x\,(x + y) = x$	$x + x\,y = x$
Algebra classica	--------------	----------------------------

Principio di dualità

Nell'algebra di Boole vale il principio di dualità.

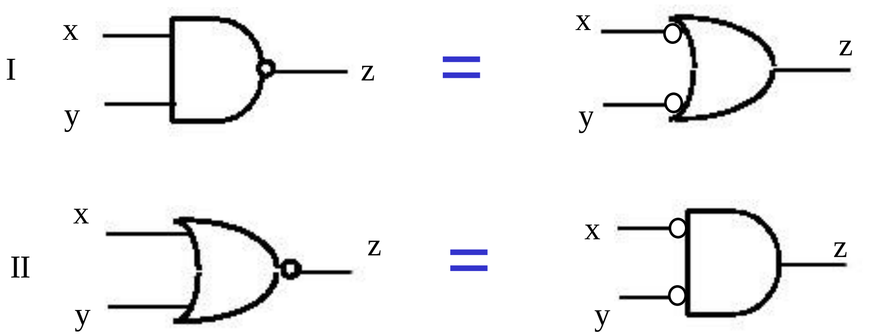
Il duale di una funzione booleana si ottiene sostituendo AND ad OR, OR ad AND, gli 0 agli 1 e gli 1 agli 0.

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)* $\overline{xy} = \bar{x} + \bar{y}$ *II)* $\overline{x + y} = \bar{x}\,\bar{y}$



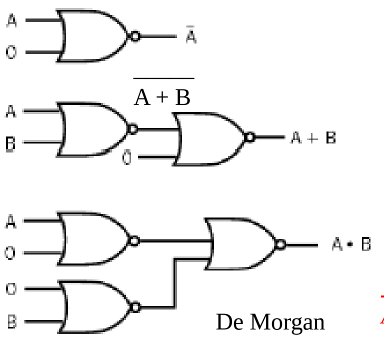
Regole algebriche - riassunto

Doppia Inversione $\bar{\bar{x}} = x$

	AND	OR
Identità	$1\,x = x$	$0 + x = x$
Elemento nullo	$0\,x = 0$	$1 + x = 1$
Idempotenza	$x\,x = x$	$x + x = x$
Inverso	$x\,\bar{x} = 0$	$x + \bar{x} = 1$
Commutativa	$x\,y = y\,x$	$x + y = y + x$
Associativa	$(x\,y)\,z = x\,(y\,z)$	$(x + y) + z = x + (y + z)$
	AND rispetto ad OR	**OR rispetto ad AND**
Distributiva	$x\,(y + z) = x\,y + x\,z$	$x + y\,z = (x + y)\,(x + z)$
Assorbimento	$x\,(x + y) = x$	$x + x\,y = x$
De Morgan	$\overline{xy} = \bar{x} + \bar{y}$	$\overline{x + y} = \bar{x}\,\bar{y}$

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

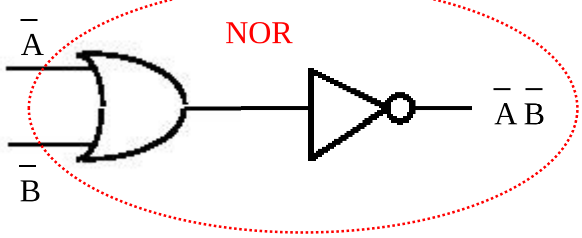
De Morgan

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

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

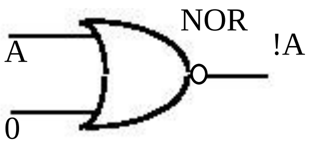
De Morgan NOR -> AND

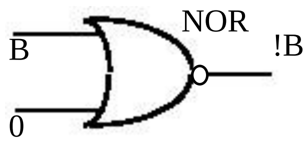
De Morgan: $\overline{A+B} = \overline{A}\ \overline{B}$



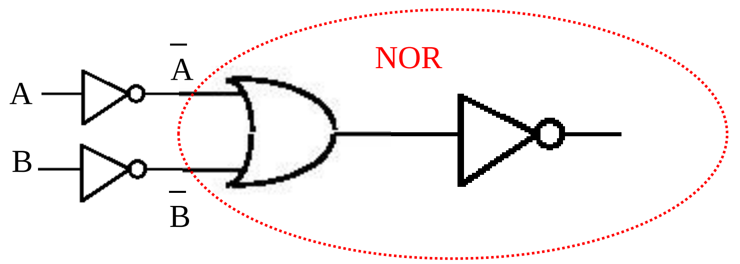
Questo circuito con NOR è equivalente a AND($\overline{A}$,$\overline{B}$). Come faccio con le negazioni?

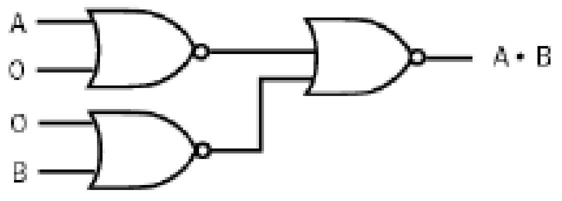
Applico la negazione agli ingressi:





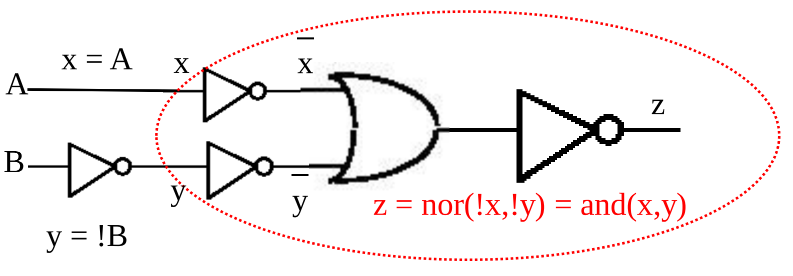
De Morgan NOR -> AND





De Morgan in generale

Quale funzione è implementata da questo circuito?



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

Esercizio

Data la funzione booleana:

F = (A AND B) OR (B 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 semplici.

Implementazione circuitale (porte logiche).

Dal circuito alla funzione.

Algebra Booleana.