

Algebra di Boole e porte logiche

L'algebra di Boole

L'algebra booleana è un particolare tipo di algebra in cui le variabili e le funzioni possono solo avere valori 0 e 1. Deriva il suo nome dal matematico inglese George Boole che la ideò.



George Boole

Si studia l'algebra booleana poiché le funzioni dell'algebra booleana sono **isomorfe** ai circuiti digitali. In altre parole, un circuito digitale può essere espresso tramite un'espressione booleana e viceversa.

Una funzione booleana ha una o più variabili in input e fornisce risultati che dipendono solo da queste variabili.

L'algebra di Boole

Poiché le variabili possono assumere solo i valori 0 o 1 una funzione booleana con n variabili di input ha solo 2^n combinazioni possibili e può essere descritta dando una tabella, detta **tabella di verità**, con 2^n righe.

Input

X	Y	Z	V
0	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	1

Output

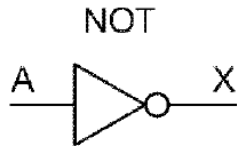
$$V = f(X, Y, Z)$$

Che funzione codifica f ?

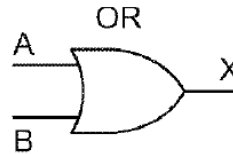
Ogni riga mostra il valore restituito a partire da una particolare configurazione dell'input.

L'algebra di Boole

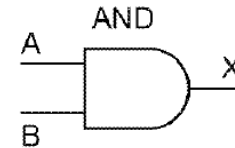
L'algebra di Boole si basa su tre operatori di base: **AND, OR, NOT**



A	X
0	1
1	0



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1



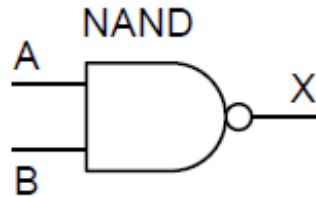
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Tutte le funzioni booleane possono essere espresse come combinazione di questi tre operatori.

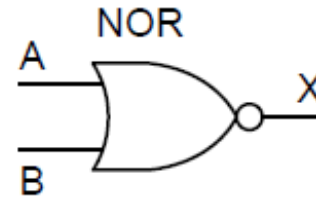
Ad ogni funzione di base corrisponde una porta logica e quindi ogni espressione booleana può essere tradotta in un circuito.

L'algebra di Boole

Molto utili sono gli operatori booleani **NAND** e **NOR**



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0



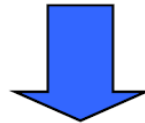
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Usando solo porte NOR o solo porte NAND è possibile realizzare qualsiasi circuito digitale in quanto tutte le porte fondamentali (AND, OR e NOT) possono essere realizzate a partire da queste.

L'algebra di Boole

Tramite le proprietà dell'algebra booleana è possibile semplificare espressioni booleane complesse

Anche i circuiti corrispondenti saranno più semplici e richiederanno un minor numero di porte logiche



Minori costi di realizzazione dei circuiti
&
Minore occupazione di spazio

Le espressioni booleane vengono utilizzate nei linguaggi di programmazione per la definizione dei criteri decisionali

L'algebra di Boole

Le dimensioni delle tabelle di verità crescono al crescere delle variabili in input, sono quindi necessarie delle rappresentazioni alternative:

- Ordinando le righe dell'input come numeri binari è possibile codificare le tabelle di verità memorizzando solo la colonna dell'output.

X	Y	Z	V
0	1	0	0
1	0	1	1
0	0	0	0
1	0	0	0
0	1	1	1
0	0	1	0
1	1	0	1
1	1	1	1

X	Y	Z	V
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

V
0
0
0
1
0
1
1
1

L'algebra di Boole

Le dimensioni delle tabelle di verità crescono al crescere delle variabili in input, sono quindi necessarie delle rappresentazioni alternative:

- Si può specificare l'output di ogni funzione booleana esprimendo, tramite una espressione booleana, **quali combinazioni delle variabili di input determinano l'output 1**. Per l'esempio precedente: (011-101-110-111). **Questa rappresentazione è importante perché può essere direttamente tradotta in un circuito di porte digitali.**

Convenzioni notazionali

- Una variabile con valore 1 è indicata dal suo nome
- Una variabile con valore 0 è indicata dal suo nome con sopra una barra
- L'operazione di AND booleano è indicata da un \cdot moltiplicativo oppure viene considerato implicitamente presente
- L'operazione di OR booleano è indicata da un $+$

$$(011-101-110-111) \Rightarrow \bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ$$

L'algebra di Boole

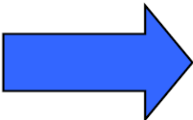
Per passare dalla rappresentazione mediante tabella di verità alla notazione tramite espressione booleana è necessario:

1. Identificare tutte le righe della tabella di verità che danno 1 in output;
2. Per ogni riga con un 1 in output scrivere la configurazione delle variabili che la definiscono (tutte le variabili della configurazione saranno in **AND** tra loro)
3. Collegare tramite **OR** tutte le configurazioni ottenute.

La rappresentazione così ottenuta è detta in **prima forma canonica**.

Esempio: Traduzione della funzione rappresentata dalla tabella di verità:

X	Y	Z	V	
0	0	0	0	
0	0	1	1	← $\overline{X}\overline{Y}Z$
0	1	0	1	← $\overline{X}Y\overline{Z}$
0	1	1	0	
1	0	0	1	← $X\overline{Y}\overline{Z}$
1	0	1	0	
1	1	0	0	
1	1	1	1	← XYZ


$$\overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$$

Semplificazione di funzioni booleane

Le funzioni booleane possono essere descritte da più espressioni equivalenti

$$X Y \bar{Z} + X \bar{Y} Z + X Y Z \equiv X(Y + Z)$$

X	Y	Z	V	Y+Z	X(Y+Z)
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	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

Def. Due funzioni booleane si dicono *equivalenti* se presentano lo stesso output per qualsiasi configurazione dell'input.

Semplificazione di funzioni booleane

Determinare **la più semplice** espressione booleana equivalente alla funzione data facilita l'interpretazione della funzione stessa e permette di semplificare anche i circuiti logici corrispondenti.

“Cosa si intende per più semplice?”

Una espressione f' è più semplice di una espressione f ($f \equiv f'$) secondo il **criterio del minor numero di operatori** se il suo calcolo richiede meno operazioni di AND e OR rispetto al calcolo di f .

N.B. Sebbene questo sia il criterio più utilizzato ne esistono altri

Semplificazione di funzioni booleane

Le funzioni booleane possono essere semplificate applicando ripetutamente le seguenti proprietà:

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Semplificazione di funzioni booleane

Proprietà associativa:

A	B	C	AB	BC	A(BC)	(AB)C	A+B	B+C	A+(B+C)	(A+B)+C
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	1	1
0	1	1	0	1	0	0	1	1	1	1
1	0	0	0	0	0	0	1	0	1	1
1	0	1	0	0	0	0	1	1	1	1
1	1	0	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Semplificazione di funzioni booleane

Proprietà distributiva:

A	B	C	BC	A+BC	A+B	A+C	(A+B)(A+C)	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0
0	1	1	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	0	0	0	0	0
1	0	1	0	1	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1

Semplificazione di funzioni booleane

Teorema di DeMorgan:

A	B	\overline{A}	\overline{B}	A+B	$\overline{A} + \overline{B}$	AB	\overline{AB}	$\overline{A} \overline{B}$	$\overline{A + B}$
0	0	1	1	0	1	0	1	1	1
0	1	1	0	1	1	0	1	0	0
1	0	0	1	1	1	0	1	0	0
1	1	0	0	1	0	1	0	0	0

Il teorema di DeMorgan si può estendere a più variabili: $\overline{ABC} = \overline{A} + \overline{B} + \overline{C}$

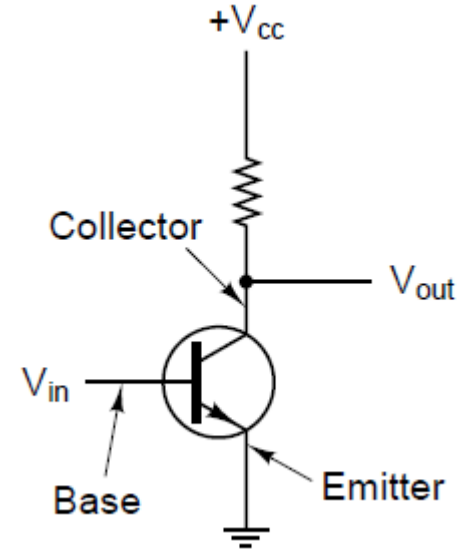
Il teorema di DeMorgan permette di esprimere l'operatore AND/OR in funzione dell'operatore OR/AND.

Porte logiche

- In un circuito digitale sono presenti solo due valori logici, per esempio 0 e 1:
 - il valore 0 potrebbe essere rappresentato da un segnale compreso tra 0 e 0.5 volt
 - il valore 1 potrebbe essere rappresentato da un segnale compreso tra 1 e 1.5 volt
 - le tensioni al di fuori di questi intervalli non sono ammesse
- La base hardware di tutti i calcolatori digitali è costituita da alcuni piccoli dispositivi elettronici, chiamati **porte logiche (gate)**, ciascuna delle quali calcola una diversa funzione di questi segnali.
- La logica digitale si fonda sul fatto che un **transistor** può essere costruito in modo da funzionare come un velocissimo interruttore binario.

Porta NOT

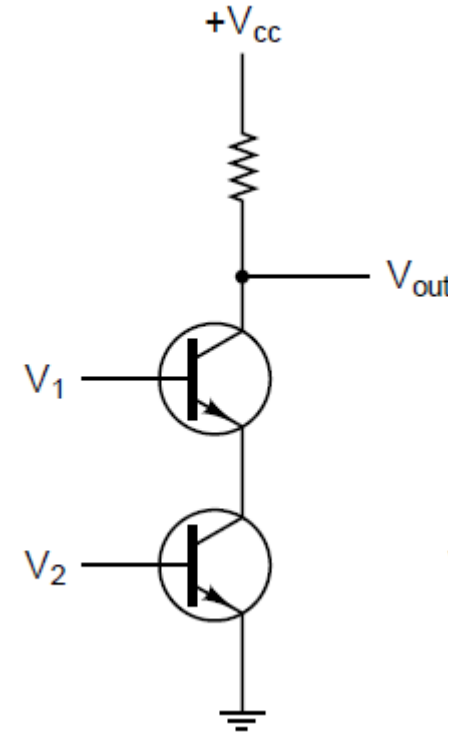
- Nella figura accanto è illustrato un transistor integrato in un semplice circuito.
- In questo circuito, quando V_{in} è basso, V_{out} è alto e viceversa.
- Si tratta quindi un **invertitore (NOT)** che converte un valore logico 0 in un valore logico 1 e un valore logico 1 in un valore logico 0.
- Funzionamento:
 - Se la tensione in ingresso (V_{in}) è inferiore ad un valore critico, il transistor viene disabilitato e si comporta come una resistenza infinita. La conseguenza è che la tensione in output (V_{out}) assume un valore vicino alla tensione applicata esternamente (V_{cc}).
 - Se V_{in} supera il valore critico, il transistor si attiva e si comporta come un conduttore ideale, facendo scaricare V_{out} a terra (per convenzione, 0 volt).



Porta NAND

- La figura accanto mostra due transistor collegati in serie.
- Se V_1 e V_2 sono alte, allora entrambi i transistor saranno in conduzione e V_{out} sarà portato al valore basso.
- Se uno degli ingressi è basso, il transistor corrispondente si spegnerà e l'uscita sarà alta.
- Il comportamento di questo circuito è quello di una porta NAND:

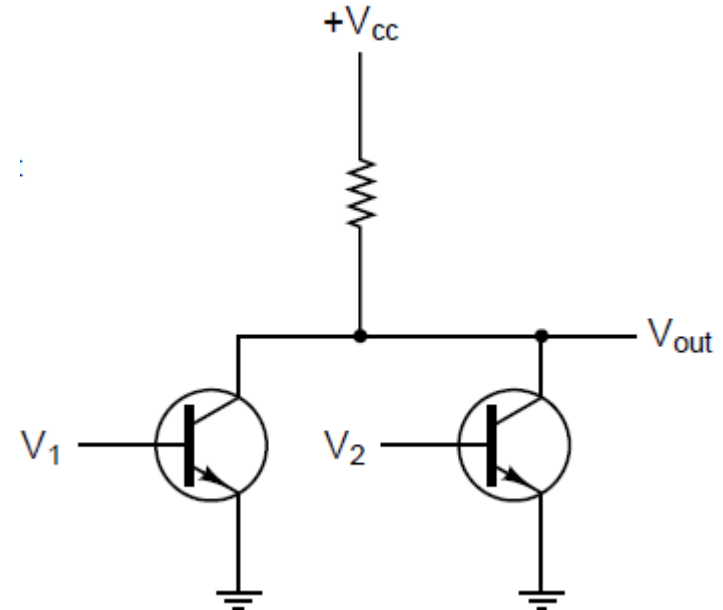
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0



Porta NOR

- La figura accanto mostra due transistor collegati in parallelo.
- Se uno degli ingressi è alto, il transistor corrispondente si accenderà e porterà l'uscita a terra.
- Se entrambi gli ingressi sono bassi, l'uscita rimarrà alta.
- Il comportamento di questo circuito è quello di una porta NOR:

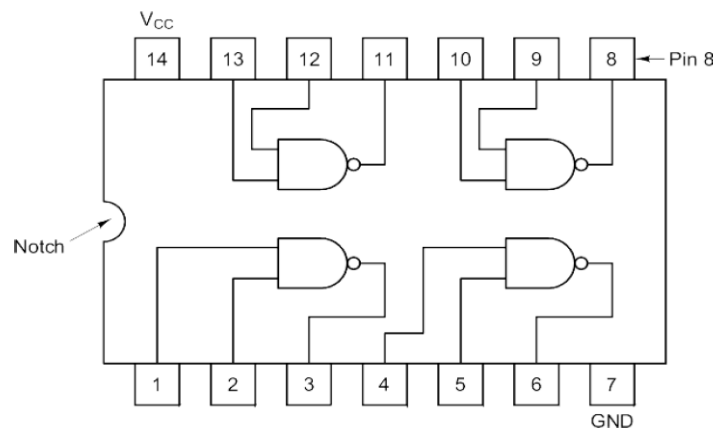
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0



Circuiti integrati

Le porte logiche non vengono prodotte o vendute individualmente, ma in unità chiamate **circuiti integrati (IC)** o più genericamente **chip**. Un circuito integrato è un pezzetto rettangolare di silicio con dimensioni che variano da pochi mm^2 ad alcuni cm^2 su cui vengono realizzate le porte (gates). I chip possono essere classificati grossolanamente sulla base del numero di porte:

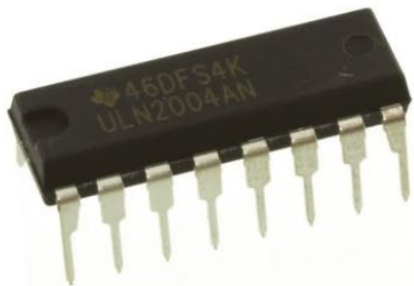
- **SSI (Small Scale Integrated)**: da 1 a 10 porte
- **MSI (Medium Scale Integrated)**: da 10 a 100 porte
- **LSI (Large Scale Integrated)**: da 100 a 10.000 porte
- **VLSI (Very Large Scale Integrated)**: 10.000 a 100.000 porte
- **ULSI (Ultra Large Scale Integrated)**: > 100.000 porte



*Un semplice
circuito SSI
con 4 porte
NAND*

Circuiti integrati

I circuiti integrati vengono poi montati in contenitori (**package**) plastici o ceramici. Nella maggior parte dei circuiti integrati (semplici) i contatti di uscita sono realizzati in due file parallele (**DIP** = Dual Inline Packages) poste ai lati del package. Circuiti più complessi con molti contatti adottano package in cui i piedini sono su tutta la parte posteriore del supporto (es. **BGA** = Ball Grid Array).



*Chip con
package:
DIP (sinistra) e
BGA (destra)*

