Il livello della Logica digitale

Aniello Minutolo

Circuiti digitali e algebra di Boole

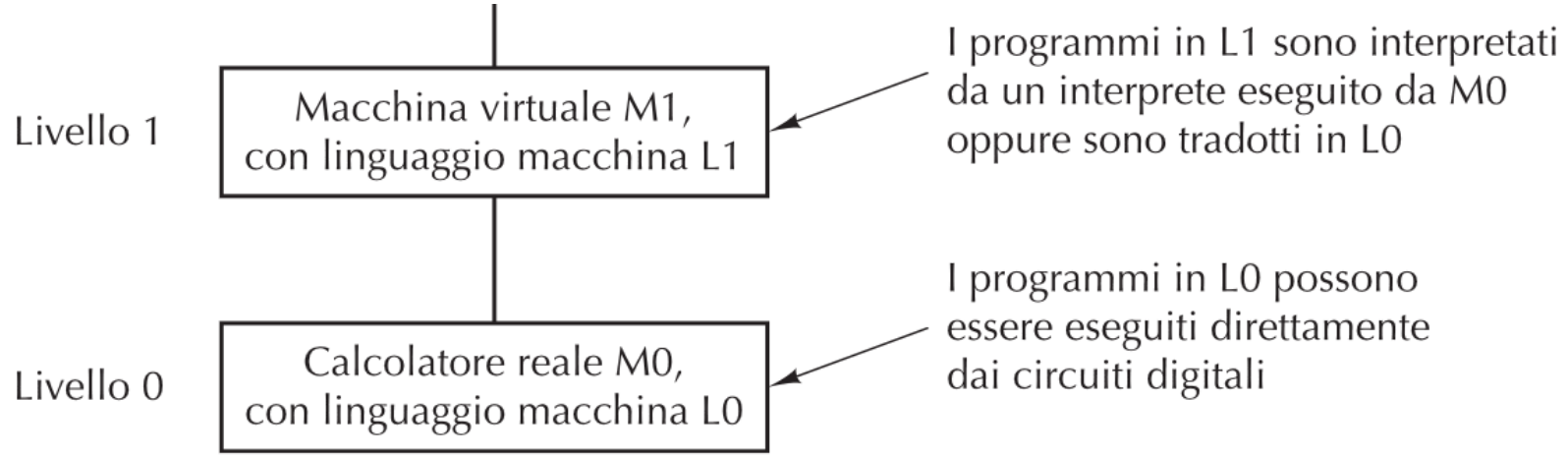
Sommario

1. Circuiti digitali e algebra di Boole
2. Le porte logiche
3. Rappresentazione algebrica di funzioni booleane

Architettura a livelli del calcolatore

Un calcolatore è una macchina a livelli

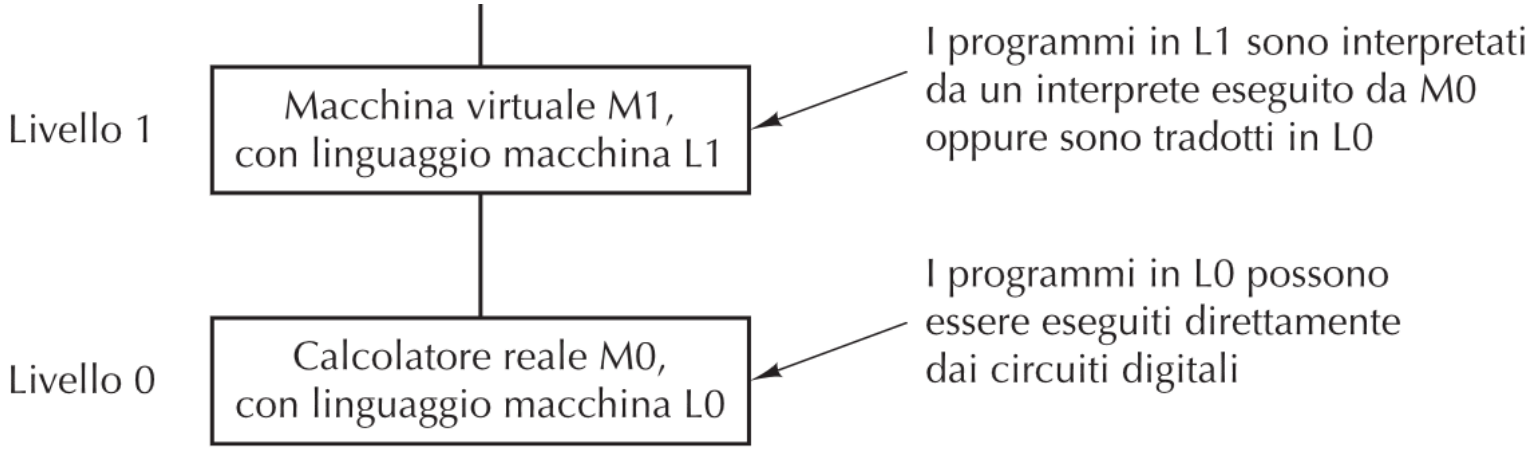
- una macchina virtuale trasforma le istruzioni fornite in un linguaggio di programmazione a un livello inferiore fino al linguaggio macchina;
- l'implementazione progressiva e modulare rende i linguaggi di programmazione indipendenti dall'hardware;
- ogni livello si basa su quello sottostante, creando una gerarchia che semplifica l'interazione tra utente e hardware.



Livello logico digitale (L0)

Il livello L0 rappresenta la logica digitale

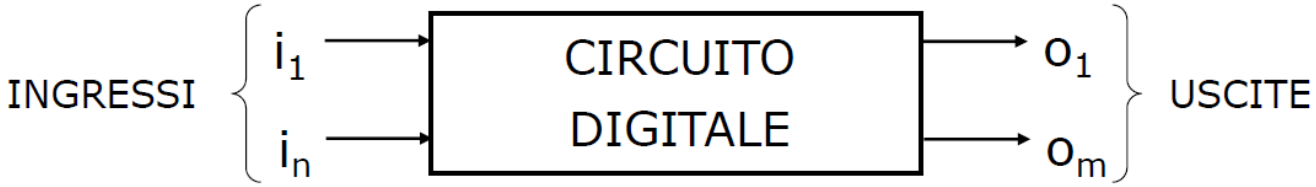
- combinazione dei dispositivi elettronici, chiamati porte logiche (gate), che formano le componenti principali del calcolatore;
- il livello L1 utilizza le componenti di L0 per realizzare microarchitetture come data path, registri, ALU e bus di controllo.



Circuiti digitali

Un circuito digitale è un circuito elettronico molto semplice, in cui gli ingressi e le uscite assumono solo due livelli, rappresentati da tensioni elettriche specifiche

- il valore binario **0** è tipicamente rappresentato da una tensione tra 0 e 0,5 volt, mentre il valore **1** tra 1 e 1,5 volt;
- le tensioni al di fuori di questi intervalli non sono ammesse nei circuiti digitali standard.



Circuiti digitali

Dato un set di input $i_1, i_2, \dots, i_n$, un circuito digitale produce un set di output $o_1, o_2, \dots, o_m$ attraverso *m* funzioni booleane f_j

$$\begin{cases} o_1 = f_1(i_1, i_2, \dots, i_n) \\ \vdots \\ o_m = f_m(i_1, i_2. \dots i_n) \end{cases}$$

Ogni $f_j(i_1, i_2. \dots i_n)$ è una funzione booleana (o logica), avente come dominio l'insieme $\{0, 1\}^n$ e codominio l'insieme $\{0, 1\}$

$$i_k \in \{0, 1\} \; \forall \; k = 1, 2, \dots, n$$
$$o_j \in \{0, 1\} \; \forall \; j = 1, 2, \dots, m$$

$$f_j : \{0, 1\}^n \rightarrow \{0, 1\} \; \forall \; j = 1, 2, \dots, m$$

Algebra di Boole

L’algebra di Boole opera con variabili che assumono solo i valori 0 e 1 ed è fondamentale per **descrivere circuiti digitali**

- **George Boole** sviluppò questa algebra nel XIX secolo come strumento per il ragionamento logico-matematico.

Algebra di Boole

Nell’algebra booleana

- le funzioni hanno una o più variabili di input e generano un valore di output che dipende soltanto dalla specifica combinazione di valori di queste variabili
- una semplice funzione f(x) di una variabile, può essere definita specificando quanto vale f(x) per ogni possibile valore di x
  - f (x) = 1 se x vale 0, e f (x) = 0 se x vale 1 (funzione NOT)

Tabelle di verità

Una funzione booleana $F(v_1, v_2, \dots, v_n)$ può essere definita tramite la sua **tabella della verità**

- elenco di tutte le possibili combinazioni delle variabili di input con i corrispondenti output della funzione
- contiene ha 2^n righe pari alle possibili combinazioni di $v_1, v_2, \dots, v_n$

Esempio

- una funzione $F(v_1, v_2, v_3)$ potrebbe essere definita dalla seguente tabella della verità

v_1	v_2	v_3	F
0	0	0	1
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	1

Funzioni booleane definibili

Il numero di funzioni booleane definibili su n variabili booleane è un insieme finito

- a differenza di quanto accade ad esempio per le funzioni reali di variabili reali.

Poiché sono possibili 2^n combinazioni di valori in input, è possibile definire 2^{2^n} possibili combinazioni in output, e quindi 2^{2^n} funzioni booleane distinte.

Funzioni booleane definibili

Con 1 bit in ingresso, sono possibili 4 funzioni booleane

- le funzioni SET0 e SET1 impostano il valore ad uno o zero, indipendentemente dagli ingressi;
- la funzione identità, restituisce il valore che gli viene passato, viene anche chiamata Buffer;
- la funzione unaria più importante è il NOT che restituisce il valore opposto del valore di ingresso.

x_1	SET0	Buffer	NOT	SET1
0	0	0	1	1
1	0	1	0	1

Funzioni booleane definibili

Con 2 bit in ingresso, sono possibili 16 funzioni booleane

- le più importanti sono AND ed OR, ma esistono altre funzioni di interesse come NAND, NOR, XOR, XNOR.

x_1	x_2	AND	OR	NAND	NOR	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Le porte logiche

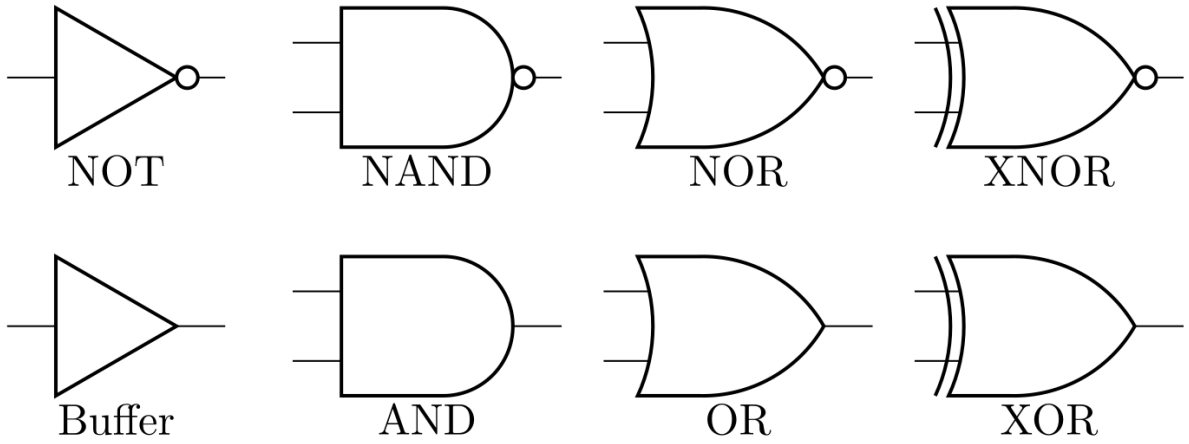
Porte logiche

- Le porte logiche sono dispositivi elettronici che calcolano funzioni di segnali digitali e costituiscono la base hardware dei calcolatori.
- Ogni porta logica implementa una specifica funzione booleana come AND, OR o NOT.
- Il funzionamento interno delle porte logiche appartiene al livello dei dispositivi, situato al di sotto del livello logico digitale.

Simboli delle porte logiche

Le porte logiche sono rappresentate da simboli standardizzati che ne descrivono il comportamento funzionale

- piccoli cerchietti nei simboli rappresentano l'inversione del segnale.

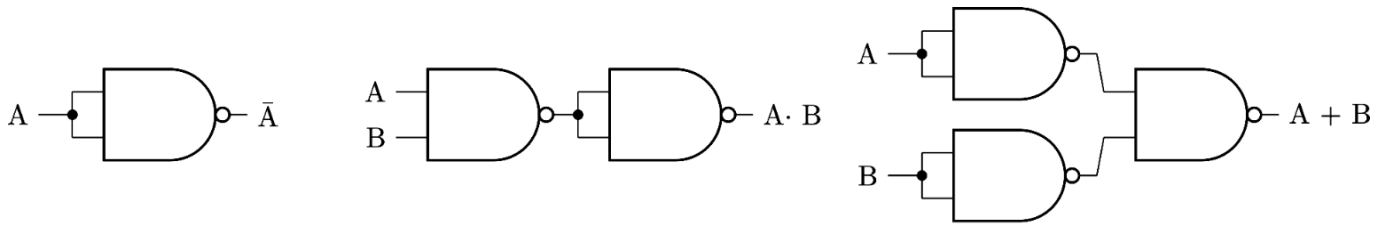


Completezza delle porte NAND

Le porte **NAND** sono logicamente **complete** perché permettono di realizzare qualsiasi tipologia di circuito logico

- qualsiasi funzione booleana può essere realizzata usando un'opportuna combinazione di porte AND, OR e NOT
- AND, OR e NOT possono essere realizzate usando soltanto porte NAND

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



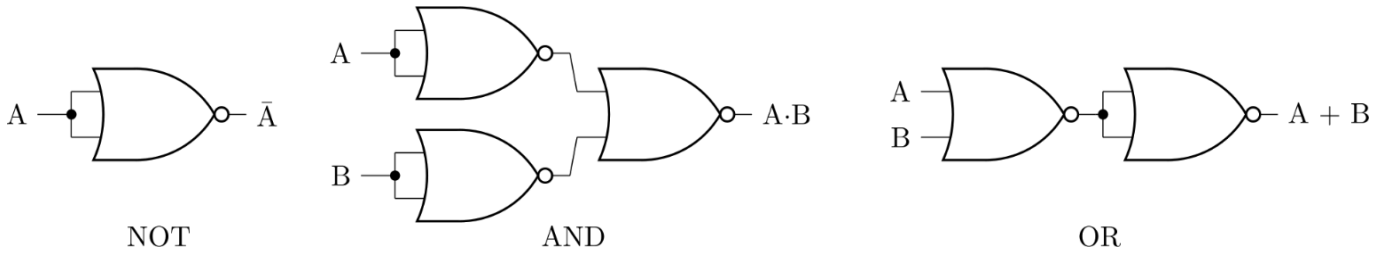
NOT AND OR

Completezza delle porte NOR

Le porte **NOR** sono logicamente **complete** perché permettono di realizzare qualsiasi tipologia di circuito logico

- qualsiasi funzione booleana può essere realizzata usando un'opportuna combinazione di porte AND, OR e NOT
- AND, OR e NOT possono essere realizzate usando soltanto porte NOR

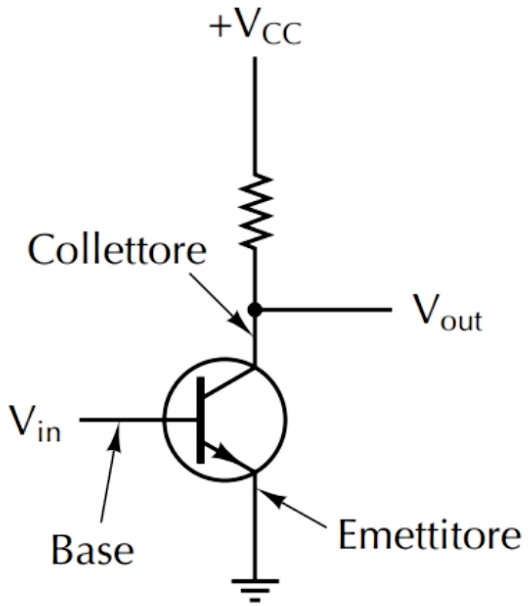
x_1	x_2	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



Esempio di implementazione di una porta NOT

I transistor funzionano come interruttori binari velocissimi con tre connessioni: **collettore**, **base** ed **emettitore**

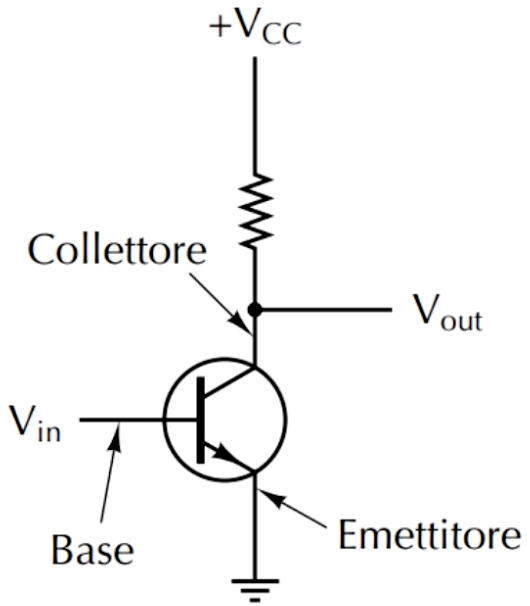
- quando la tensione Vin supera una soglia critica il transistor conduce portando Vout a 0 volt (circuito chiuso);
- quando Vin è sotto soglia il transistor interrompe la conduzione portando Vout a un valore vicino a Vcc (circuito aperto).



Esempio di implementazione di una porta NOT

Il circuito in figura è un circuito invertitore che converte un valore logico 0 in 1 e viceversa, implementando la funzione **NOT**

- infatti, quando l'input è basso, l'output è alto, mentre quando l'input è alto, l'output è basso;
- la resistenza nel circuito è necessaria per limitare la corrente ed evitare danni al transistor.

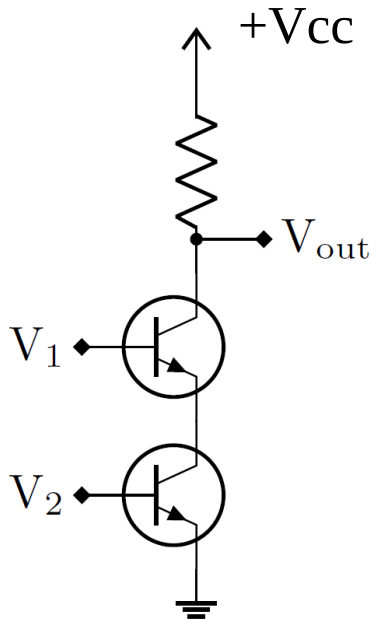


Esempio di implementazione di una porta NAND

Il circuito in figura implementa una porta **NAND**, che restituisce il valore logico invertito rispetto alla AND dei segnali di ingresso

- infatti, quando almeno uno degli input è basso, l'output è alto, mentre quando tutti gli input sono contemporaneamente alti, l'output è basso;
- la porta **AND** si ottiene ponendo un invertitore all'uscita di una NAND.

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

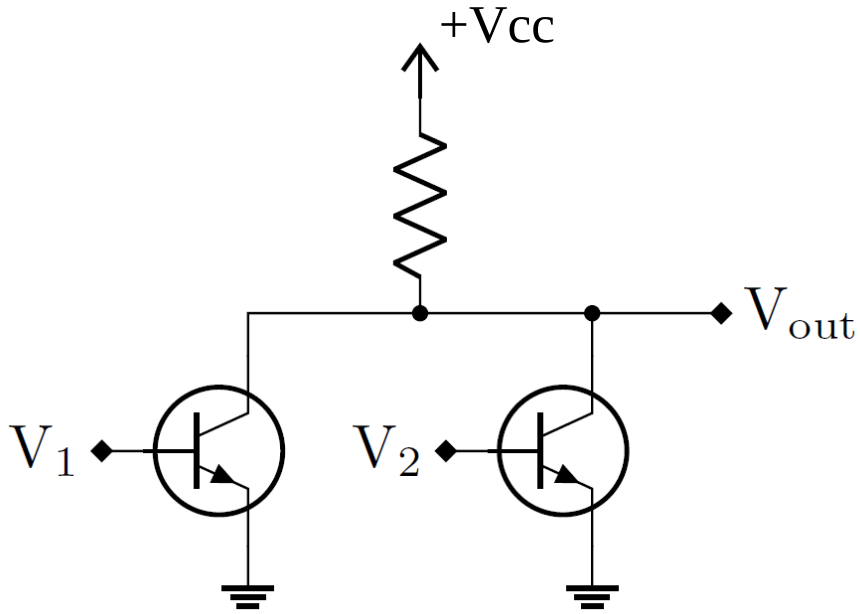


Esempio di implementazione di una porta NOR

Il circuito in figura implementa una porta **NOR**, che restituisce il valore logico invertito rispetto alla OR dei segnali di ingresso

- infatti, quando almeno uno degli input è alto, l'output è basso, mentre quando tutti gli input sono contemporaneamente bassi, l'output è alto;
- la porta **OR** si ottiene ponendo un invertitore all'uscita di una NOR.

x_1	x_2	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



Complessità implementativa

- Le porte NAND e NOR richiedono solo due transistor mentre AND e OR necessitano di tre transistor ciascuna.
- Molti calcolatori preferiscono usare NAND e NOR come elementi base per la loro maggiore semplicità circuitale.
- Le porte logiche possono avere più di due ingressi, anche se raramente superano gli otto ingressi nella pratica.

Tecnologie costruttive

- Le principali tecnologie per costruire porte logiche sono la bipolare (**TTL**, **ECL**) e la **MOS** (PMOS, NMOS, CMOS).
- La tecnologia **CMOS** domina nei moderni calcolatori per il basso consumo e l’alta densità di integrazione.
- Nonostante la minore velocità rispetto alle tecnologie bipolari, la **CMOS** opera a tensioni più basse (circa +1,5 volt).

Rappresentazione algebrica di funzioni booleane

Rappresentazione algebrica di funzioni booleane

- La **tabella di verità** di una funzione booleana specifica completamente gli output per tutti i possibili valori di input.

- All’aumentare delle variabili di input, le tabelle di verità diventano troppo grandi, rendendo necessarie la definizione di **notazioni alternative** e più **sintetiche**.

Rappresentazione algebrica di funzioni booleane

Se x e y sono due variabili booleane

- AND(x, y) si rappresenta come prodotto x · y
- OR(x, y) si rappresenta come somma x + y
- NOT(x) si rappresenta come $\overline{x}$

Teorema

- ogni **funzione booleana** è algebrica, cioè rappresentabile con un'**espressione dell'algebra**
- ogni **funzione booleana** può essere descritta come OR delle combinazioni per cui la funzione è vera (**somma di mintermini**)

Esempi

- *AND*$(x_1 ,x_2 = (x_1 x_2$
- *NAND*$(x_1 ,x_2 = (\overline{x_1}\ \overline{x_2} + x_1\overline{x_2} + \overline{x_1}x_2$

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

Rappresentazione algebrica di funzioni booleane

Forma Canonica

- la forma canonica di una funzione booleana rappresenta l'unione di tutte le combinazioni di input che producono output 1
- data una funzione booleana $f(i_1, i_2, \dots, i_n)$, si definisce forma canonica di f la seguente espressione algebrica

$$f(i_1, i_2, \dots, i_n) \stackrel{\text{def}}{=} \sum_{j=1}^{m} \prod_{i=1}^{n} x_{ij}^*$$

dove $x_{ij}^* \in \{x_{ij}, \overline{x_{ij}}\}$ e viene chiamato **mintermine**

Forma canonica: esempio

Consideriamo la tabella di verità di una funzione booleana a tre variabili: M = f (A, B, C)

- M corrisponde alla funzione logica di maggioranza che vale 0 se la maggioranza dei suoi valori di input vale 0, mentre vale 1 se la maggioranza degli input vale 1.

Righe della tabella di verità che producono in output il bit 1

$$\overline{A}BC, A\overline{B}C, AB\overline{C}, ABC$$

Forma canonica

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

A	B	C	M
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

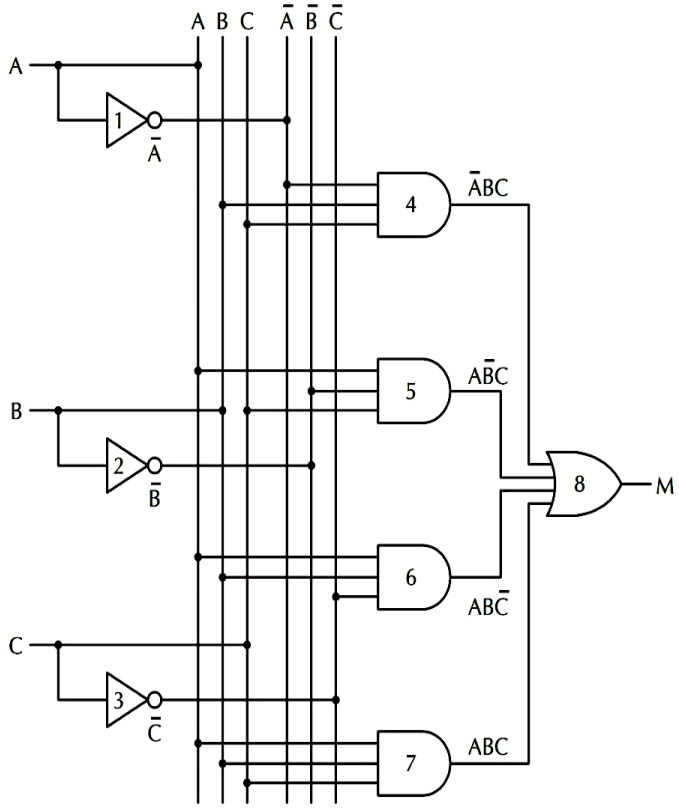
Implementazione di una funzione booleana

Una funzione booleana può essere implementata mediante un circuito elettronico che utilizza segnali per rappresentare le variabili di input e di output oltre ad alcune porte logiche come AND, OR e NOT

- la stessa funzione booleana può avere diverse implementazioni circuitali con caratteristiche diverse

Esempio

$$M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$



Bibliografia

Libro di testo

- Andrew S. Tanenbaum e Todd Austin. Architettura dei calcolatori. Un approccio strutturale. 6/ED. Anno 2013. Pearson Italia spa. (Disponibile nella sezione “Biblioteca”)

Fonte argomenti e immagini

- Capitolo 3, da Paragrafo 3.1 a 3.1.3, pp. 151-158