PROGRAMMAZIONE INFORMATICA

ALGEBRA BOOLEANA

STORIA

George Boole era un matematico e logico inglese, nato nel 1815. Egli sviluppò l'algebra booleana nel suo libro "An Investigation of the Laws of Thought" (1854). L'obiettivo principale di Boole era sviluppare un sistema che modellasse i processi del ragionamento umano attraverso un formalismo matematico.

L'algebra booleana nacque in un'epoca in cui i matematici cercavano di formalizzare e strutturare la logica per renderla più rigorosa. Boole fu uno dei pionieri nel tentativo di descrivere la logica non solo con la filosofia, ma con equazioni matematiche. Questo sistema si rivelò essenziale per la nascita dell'informatica moderna e della progettazione dei circuiti digitali.

Sebbene l'algebra booleana abbia radici nel XIX secolo, il suo utilizzo è cresciuto notevolmente con lo sviluppo dei computer e della logica digitale. Viene usata per progettare e ottimizzare circuiti logici, basi dati e sistemi informatici. Ogni componente di un computer, dal processore alla memoria, utilizza operazioni booleane per funzionare.

VARIABILI BOOLEANE

Nell'algebra booleana, una variabile può assumere solo **due valori**:

- **0** (che rappresenta Falso),
- **1** (che rappresenta Vero).

A differenza dell'algebra tradizionale, dove le variabili possono assumere una gamma infinita di valori, nell'algebra booleana tutto è binario, limitato a **due possibili stati**.

Per rappresentare le variabili in modo più sistematico, utilizziamo le **tabella di verità**. Per una singola variabile sarà:

Variabile (x)	Valore
0	Falso
1	Vero

OPERATORI

L'algebra booleana si basa su tre operatori fondamentali, che permettono di combinare e manipolare i valori delle variabili booleane. Questi sono **NOT** (negazione), **AND** (coniugazione) e **OR** (disgiunzione).

NOT:

x	~x
0	1
1	0

OR:

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

AND:

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

OPERATORI AGGIUNTIVI

Questi operatori sono utili nella progettazione di circuiti logici più complessi.

NAND:

x	y	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0

XOR:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

NOR:

x	y	$\overline{x + y}$
0	0	1
0	1	0
1	0	0
1	1	0

XNOR:

x	y	$\overline{x \oplus y}$
0	0	1
0	1	0
1	0	0
1	1	1

COSTRUZIONE DI TABELLE DI VERITA’

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

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

COSTRUZIONE DI TABELLE DI VERITA’

$$\bar{x} + y$$

x	y	$\bar{x}$	$\bar{x} + y$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

ESERCIZI

$$\overline{(\bar{x} \cdot y) + (z \cdot \bar{y} \cdot x)}$$

$$((x\ XOR\ y)\ AND\ z)\ XNOR\ (NOT(x)\ AND\ \ y)$$

COSTRUZIONE DI TABELLE DI VERITA'

$$\overline{\underbrace{(\bar{x} \cdot y)}_{A} + \underbrace{(z \cdot \bar{y} \cdot x)}_{B}}$$

x	y	z	A	B	A + B	$\overline{A + B}$
0	0	0	0	0	0	1
0	0	1	0	0	0	1
0	1	0	1	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	1	0	1	1	0
1	1	0	0	0	0	1
1	1	1	0	0	0	1

COSTRUZIONE DI TABELLE DI VERITA'

$$((x\ XOR\ y)\ AND\ z)\ XNOR\ (NOT(x)\ AND\ \ y)$$

A B

x	y	z	A	B	A XNOR B
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	0	1

PROPRIETA' DELL'ALGEBRA BOOLEANA

L'algebra booleana, come altre branche della matematica, segue una serie di proprietà e leggi che ci permettono di manipolare e semplificare espressioni logiche. Queste proprietà sono essenziali quando lavoriamo con circuiti logici o nella logica computazionale.

- **Proprietà commutativa**:

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

- **Proprietà associativa**:

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

- **Proprietà distributiva**:

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

PROPRIETA' DELL'ALGEBRA BOOLEANA

- **Proprietà di identità:**

$$x + 0 = x$$
$$x \cdot 1 = x$$

- **Proprietà complementare:**

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

- **Legge di assorbimento:**

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

- **Leggi di de Morgan:**

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

TEOREMI DELL'ALGEBRA BOOLEANA

I teoremi dell'algebra booleana ci aiutano a semplificare espressioni logiche. Alcuni di questi derivano direttamente dalle proprietà che abbiamo appena visto.

- **Doppia negazione:**

$$\bar{\bar{x}} = x$$

- **Idempotenza:**

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

- **Annichilimento:**

$$x \cdot 0 = 0$$
$$x + 1 = 1$$

SEMPLIFICAZIONE DI ESPRESSIONI BOOLEANE

L'obiettivo è trasformare un'espressione booleana complessa in una più semplice equivalente, mantenendo lo stesso risultato. Le espressioni semplificate richiedono meno risorse computazionali o hardware, facilitando così la progettazione e la realizzazione di sistemi più veloci e meno costosi.

Esistono diversi metodi per semplificare espressioni booleane, che includono: uso delle **proprietà e leggi** dell'algebra booleana, metodo della **tabella di verità**, mappe di Karnaugh (**K-map**) e l'**algoritmo di Quine-McCluskey.**

SEMPLIFICAZIONE CON LEGGI E TEOREMI

Applichiamo le varie proprietà e teoremi che abbiamo già discusso.

Esempio 1:

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

Legge di assorbimento

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

Esempio 2:

$$(x + y) \cdot (x + \bar{y})$$

Distributività dell'AND sull'OR

$$x + (y \cdot \bar{y})$$

Proprietà complementare

$$x + 0$$

Identità

$$x + 0 = x$$

METODO DELLA TABELLA DI VERITA’

Con questo metodo, costruiamo una tabella che descrive il comportamento di un'espressione booleana per tutte le combinazioni possibili delle variabili.

Consiste in 4 passaggi:

1- **Identificare le righe con output pari a "1":** Le righe in cui l'uscita è "1" sono quelle che ci interessano, poiché rappresentano le condizioni in cui la funzione booleana è vera.

2- **Scrivere i mintermini**: Per ogni combinazione che dà "1", scrivi il minterm corrispondente, ovvero il prodotto logico (AND) delle variabili.

3- **Formare l'espressione canonica**: L'espressione completa sarà la somma (OR) di tutti i mintermini.

4- **Semplificare l'espressione**: Una volta ottenuta l'espressione canonica, la semplifichiamo usando le leggi dell'algebra booleana o con altri metodi.

METODO DELLA TABELLA DI VERITA'

Esempio 1:

x	y	f(x,y)
0	0	0
0	1	1
1	0	1
1	1	1

Righe con uscita 1:

- x = 0; y = 1 $\bar{x}y$
- x = 1; y = 0 $x\bar{y}$
- x = 1; y = 1 xy

$\bar{x}y + x\bar{y} + xy$

$\bar{x}y + x(\bar{y} + y) = \bar{x}y + x$

$\bar{x}y + x = (x + \bar{x})(x + y)$

$(x + \bar{x})(x + y) = 1(x + y) = \boldsymbol{x + y}$

METODO DELLA TABELLA DI VERITA'

Esempio 2:

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Righe con uscita 1:

- x = 0; y = 1; z =1 $\bar{x}yz$
- x = 1; y =0; z = 0 $x\bar{y}\bar{z}$
- x = 1; y = 0; z = 1 $x\bar{y}z$
- x = 1; y = 1; z = 0 $xy\bar{z}$
- x = 1; y = 1; z = 1 xyz

METODO DELLA TABELLA DI VERITA’

Esempio 2:

$$\bar{x}yz + x\bar{y}\bar{z} + x\bar{y}z + xy\bar{z} + xyz$$
$$yz(x + \bar{x}) + x(\bar{y}\bar{z} + \bar{y}z + y\bar{z})$$
$$yz + x(\bar{y}(z + \bar{z}) + y\bar{z})$$
$$yz + x(\bar{y} + y\bar{z})$$
$$yz + x\bar{y} + xy\bar{z}$$

MAPPE DI KARNAUGH (K-MAP)

Sono uno strumento grafico che permette di semplificare espressioni booleane in modo visivo. Questo metodo aiuta ad eliminare i termini ridondanti tramite un approccio basato su blocchi o raggruppamenti.

Una K-map è essenzialmente una **rappresentazione bidimensionale di una tabella di verità**. Ogni cella della mappa corrisponde a una combinazione specifica delle variabili di ingresso e contiene il valore di uscita della funzione booleana per quella combinazione.

Le variabili sono disposte in modo tale da minimizzare il numero di cambiamenti tra celle adiacenti. Questo è conosciuto come il **codice Gray**. Una K-map per n variabili avrà 2^n celle.

MAPPE DI KARNAUGH (K-MAP)

1- Comincia con la tabella di verità della funzione booleana e riporta i valori di uscita (0 o 1) nelle celle della K-map corrispondenti alle combinazioni di variabili.

2- Cerca le celle che contengono il valore 1 o **celle attive**, poiché rappresentano le condizioni in cui la funzione è vera.

3- **Raggruppa** le celle che contengono 1 in **quadrati e rettangoli**. Ogni gruppo deve contenere un numero di celle che è una potenza di 2. I gruppi possono sovrapporsi, e devono essere il più grandi possibile. I bordi della K-map si "connettono", permettendo raggruppamenti anche su lati opposti.

4 – Derivazione dell'espressione: Ogni gruppo di celle adiacenti corrisponde a un termine dell'espressione booleana semplificata. Quando un gruppo è formato, osserva quali variabili rimangono costanti in tutte le celle del gruppo: Se una variabile è 1 in tutte le celle, la scrivi così com'è, Se è 0 in tutte le celle, la scrivi come il suo complemento. I termini di ciascun gruppo vengono uniti con l'operatore OR.

MAPPE DI KARNAUGH (K-MAP)

Esempio 1:

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

MAPPE DI KARNAUGH (K-MAP)

Esempio 1:

z\xy	00	01	11	10
0	1	1	1	1
1	0	1	1	0

MAPPE DI KARNAUGH (K-MAP)

Esempio 1:

z\xy	00	01	11	10
0	1	1	1	1
1	0	1	1	0

Gruppo 1:

x	y	z
0	0	**0**
0	1	**0**
1	1	**0**
1	0	**0**

Gruppo 2:

x	y	z
0	**1**	0
0	**1**	1
1	**1**	0
1	**1**	1

$$y + \bar{z}$$

MAPPE DI KARNAUGH (K-MAP)

Esempio 2:

a	b	c	d	f(a,b,c,d)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

MAPPE DI KARNAUGH (K-MAP)

Esempio 2:

ab\cd	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	0
10	1	0	0	0

MAPPE DI KARNAUGH (K-MAP)

Esempio 2:

ab\cd	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	1	1	1	0
10	1	0	0	0

Gruppo 1:

a	b	c	d
0	**1**	0	**1**
0	**1**	0	**1**
1	**1**	1	**1**
1	**1**	1	**1**

Gruppo 2:

a	b	c	d
0	**1**	**1**	1
0	**1**	**1**	0

Gruppo 3:

a	b	c	d
1	1	**0**	**0**
1	0	**0**	**0**

$$bd + \bar{a}bc + a\bar{c}\bar{d}$$

ALGORITMO DI QUINE-MCCLUSKEY

Metodo sistematico per minimizzare le espressioni booleane. Funziona in modo simile alle K-map, ma è più adatto per essere implementato come algoritmo in un computer, in particolare per problemi con molte variabili.

L'algoritmo prevede i seguenti passi:

1- **Generazione dei Mintermini**: Creazione di una lista di mintermini dalla tabella di verità o dalla funzione booleana.

2- **Semplificazione**: Raggruppamento e semplificazione dei mintermini per ottenere l'espressione finale.

Ogni mintermino rappresenta una combinazione di variabili che produce un output di 1. Vengono scritti come numeri decimali che rappresentano le righe della tabella dove l'output è 1.

ALGORITMO DI QUINE-MCCLUSKEY

Nella semplificazione, I mintermini vengono raggruppati in base al numero di 1. Ogni gruppo è composto da mintermini che differiscono per un solo bit. Questo viene fatto per identificare i mintermini che possono essere semplificati.

I mintermini vengono poi comparati tra gruppi adiacenti per trovare quelli che differiscono di un solo bit. Se due mintermini possono essere uniti, il risultato sarà un'espressione più semplice.

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 1:

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 1:

Gruppo	Mintermine	xyz
1	m1	001
	m2	010
2	m3	011
	m5	101
	m6	110

Gruppo	Coppie	xyz	
1	m1-m3	0_1	$\bar{x}z$
	m1-m5	_01	$\bar{y}z$
	m2-m3	01_	$\bar{x}y$
	m2-m6	_10	$y\bar{z}$

$$\bar{x}z + \bar{y}z + \bar{x}y + y\bar{z}$$

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 2:

a	b	c	d	f(a,b,c,d)
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 2:

Gruppo	Mintermine	abcd
0	m0	0000
1	m1	0001
	m8	1000
2	m3	0011
	m9	1001
3	m7	0111
	m11	1011
4	m15	1111

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 2:

Gruppo	Coppia	abcd
0	m0-m1	000_
	m0-m8	_000
1	m1-m3	00_1
	m1-m9	_001
	m8-m9	100_
2	m3-m7	0_11
	m3-m11	_011
	m9-m11	10_1
3	m9-m15	_111
	m11-m15	1_11

ALGORITMO DI QUINE-MCCLUSKEY

Esempio 2:

Gruppo	Coppia	abcd	
0	m0-m1-m8-m9	_00_	$\bar{b}\bar{c}$
	m0-m8-m1-m9	_00_	
1	m1-m3-m9-m11	_0_1	$\bar{b}d$
	m1-m9-m3-m11	_0_1	
2	m3-m7-m11-m15	__11	cd
	m3-m11-m9-m15	__11	

$$\bar{b}\bar{c} + \bar{b}d + cd$$

CIRCUITI LOGICI

La progettazione di circuiti logici è una delle applicazioni più fondamentali dell'Algebra Booleana. I circuiti logici sono **dispositivi elettronici che eseguono operazioni logiche su segnali binari** (0 e 1), che rappresentano il falso e il vero. Questi circuiti sono alla base di tutti i sistemi digitali, come i computer, i sistemi di comunicazione e i dispositivi elettronici moderni.

Esistono principalmente due tipi di circuiti logici:

- **Circuiti logici combinatori**: L'output dipende solo dagli input correnti.

- **Circuiti logici sequenziali**: L'output dipende sia dagli input correnti sia dallo stato precedente (questo implica che i circuiti abbiano una memoria).

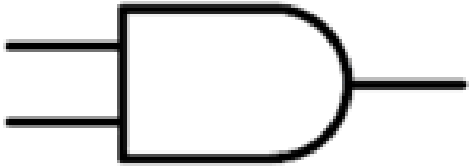
CIRCUITI COMBINATORI

Un circuito logico combinatorio è un circuito digitale in cui l'output dipende esclusivamente dagli input presenti in un dato istante, senza dipendenza da stati precedenti (ossia **non esistono stati interni o memoria**).

Gli input e gli output sono rappresentati da **valori binari**: 0 (falso) e 1 (vero) e sono descritti da **espressioni booleane**, che possono essere ricavate dalle tabelle di verità.

PORTE LOGICHE FONDAMENTALI

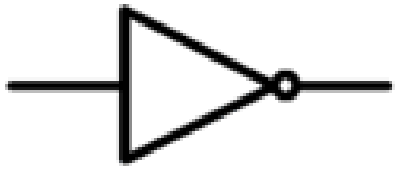
OR



AND

NOR

NAND



NOT

XOR

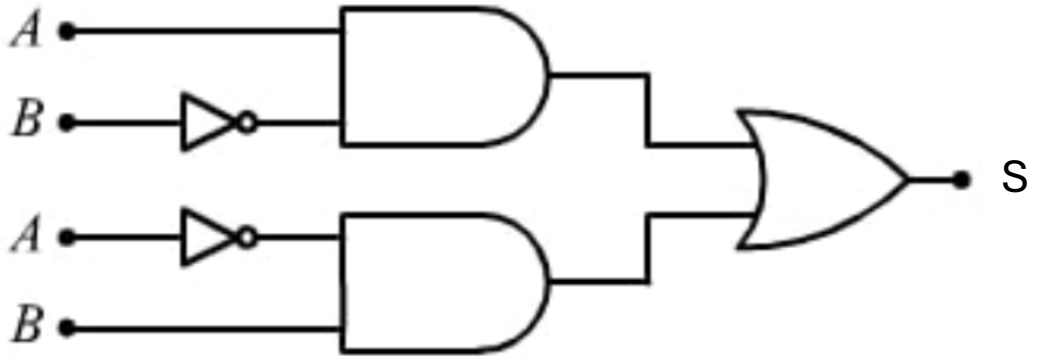
NXOR

SOMMATORE BINARIO

HALF ADDER

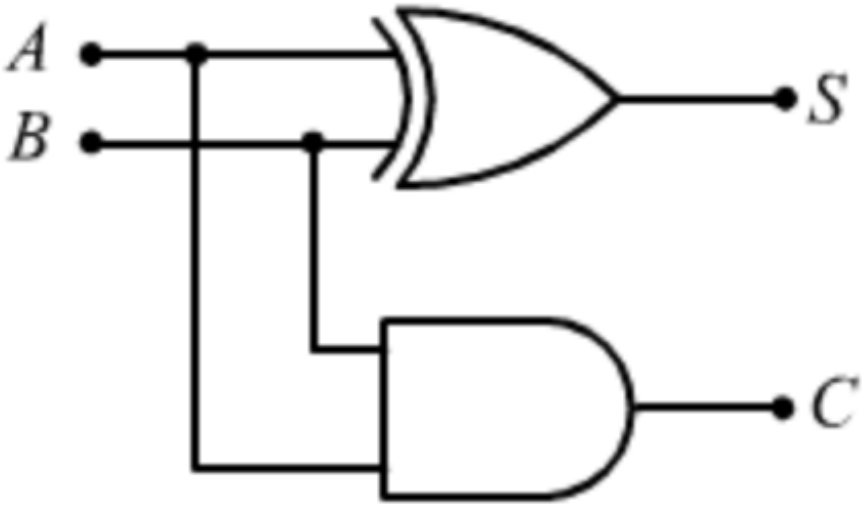
A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

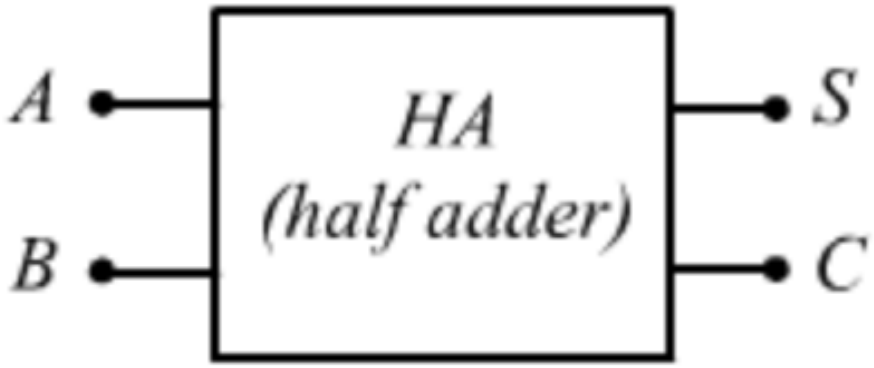
$S = A \oplus B$



$R = AB$

SOMMATORE BINARIO





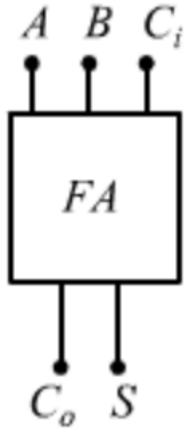
SOMMATORE BINARIO

FULL ADDER

A	B	C_i	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \overline{A}\,\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\,\overline{C}_i + ABC_i$$
$$S = C_i(\overline{A}\,\overline{B} + AB) + \overline{C}_i(\overline{A}B + A\overline{B})$$
$$S = C_i(\overline{A \oplus B}) + \overline{C}_i(A \oplus B)$$
$$S = (A \oplus B) \oplus C_i$$

$$C_o = \overline{A}BC_i + A\overline{B}C_i + AB\overline{C}_i + ABC_i$$
$$C_o = C_i(\overline{A}B + A\overline{B}) + AB(\overline{C}_i + C_i)$$
$$C_o = C_i(A \oplus B) + AB$$



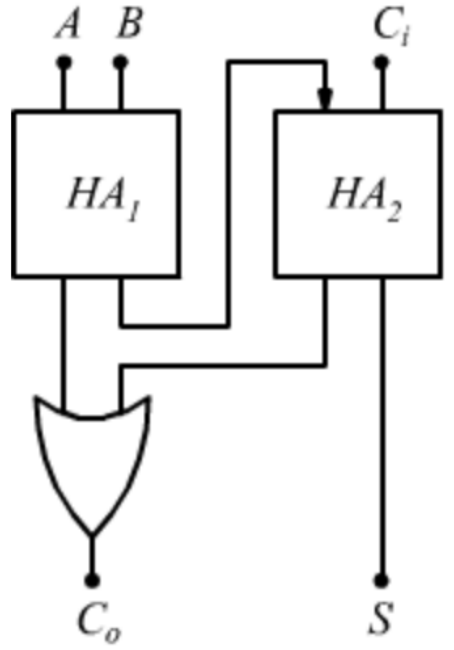
SOMMATORE BINARIO

FULL ADDER

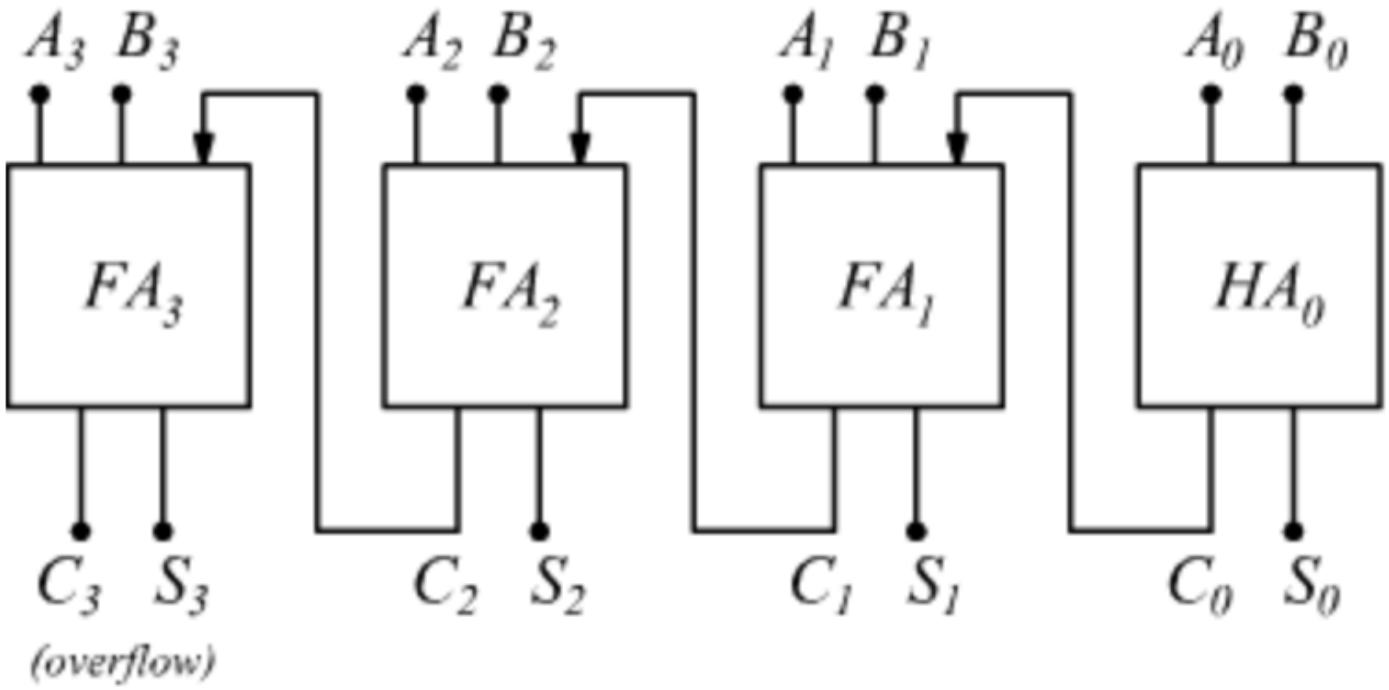
A	B	C_i	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S = \overline{A}\,\overline{B}C_i + \overline{A}B\overline{C}_i + A\overline{B}\,\overline{C}_i + ABC_i$$
$$S = C_i(\overline{A}\,\overline{B} + AB) + \overline{C}_i(\overline{A}B + A\overline{B})$$
$$S = C_i(\overline{A \oplus B}) + \overline{C}_i(A \oplus B)$$
$$S = (A \oplus B) \oplus C_i$$

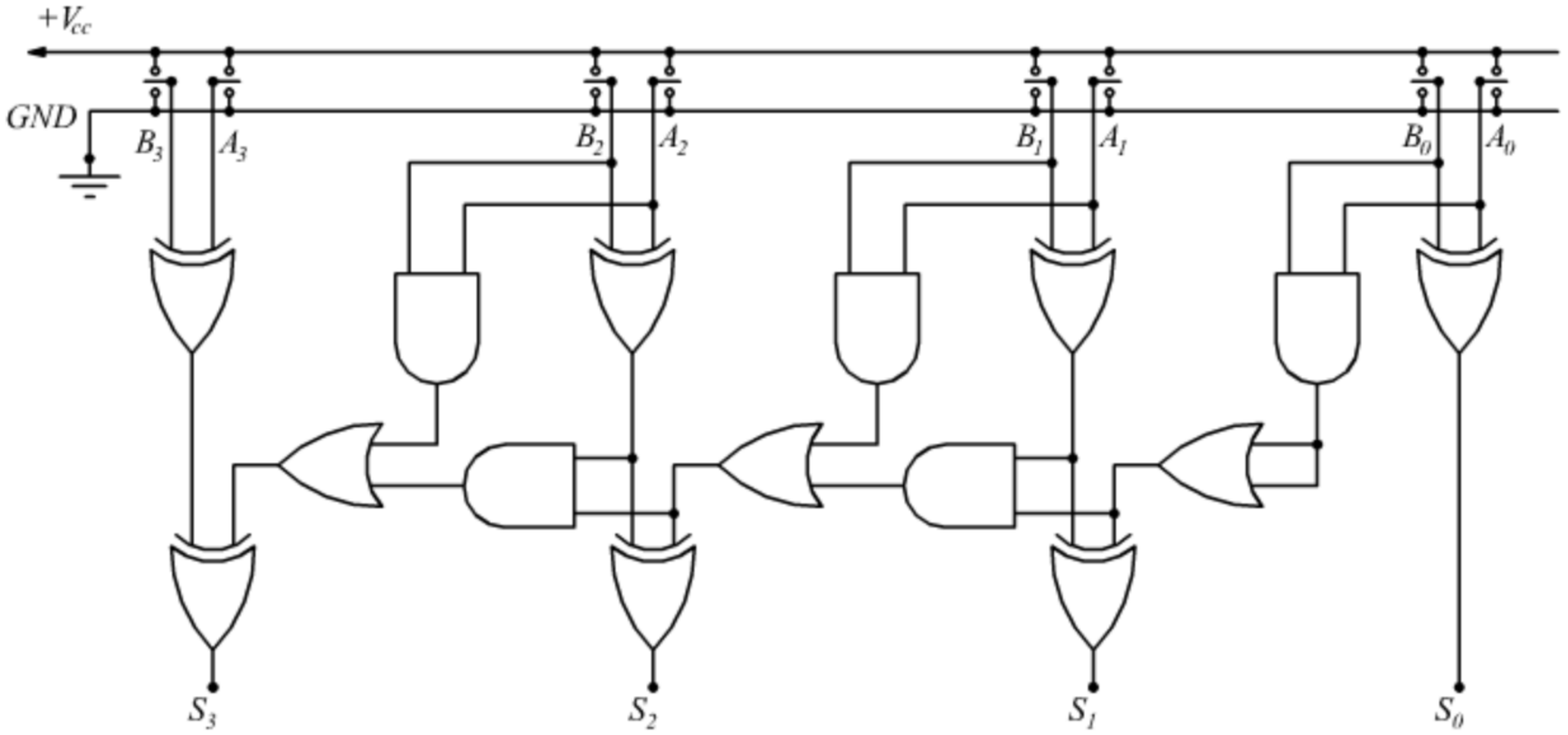
$$C_o = \overline{A}BC_i + A\overline{B}C_i + AB\overline{C}_i + ABC_i$$
$$C_o = C_i(\overline{A}B + A\overline{B}) + AB(\overline{C}_i + C_i)$$
$$C_o = C_i(A \oplus B) + AB$$



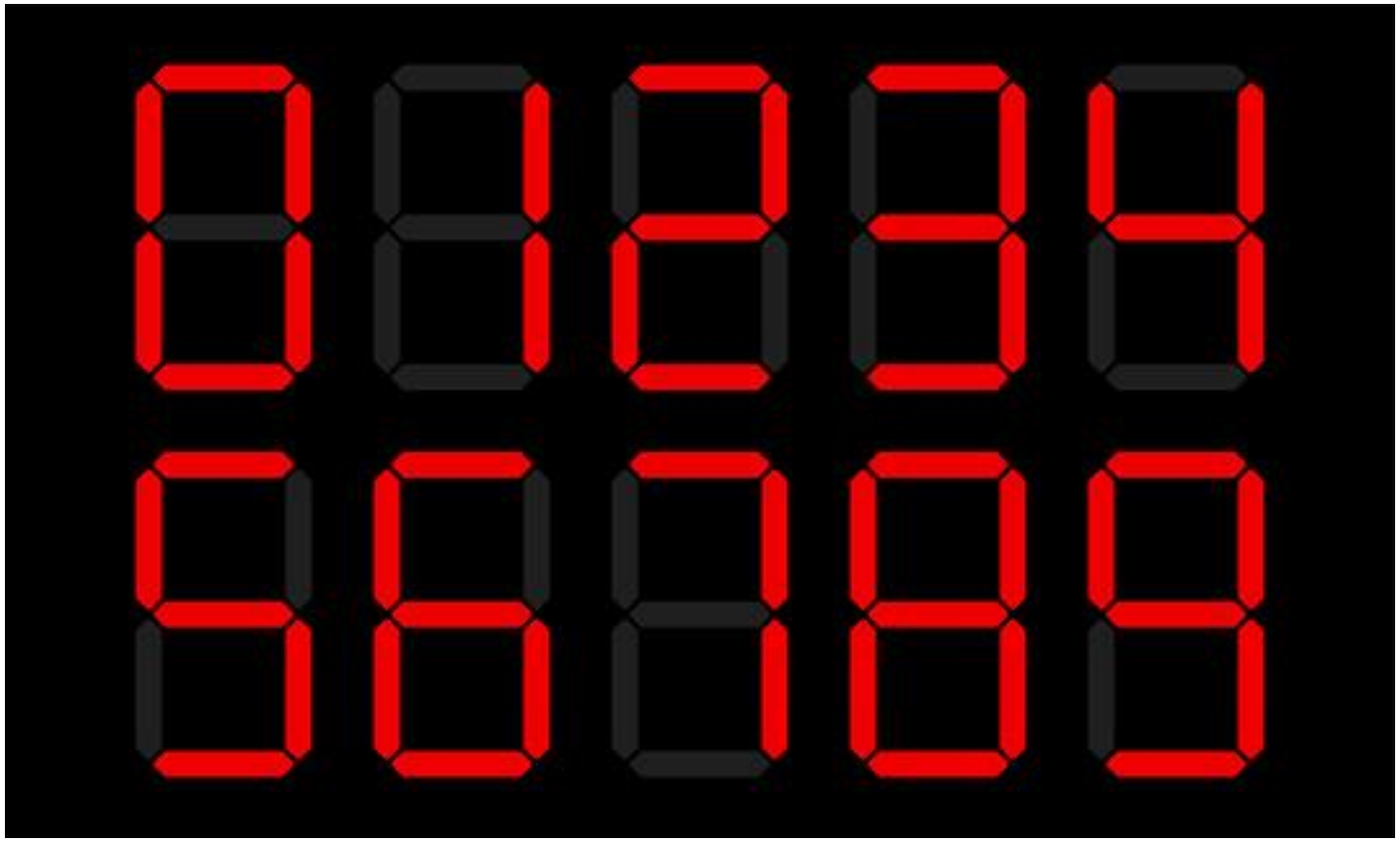
SOMMATORE BINARIO

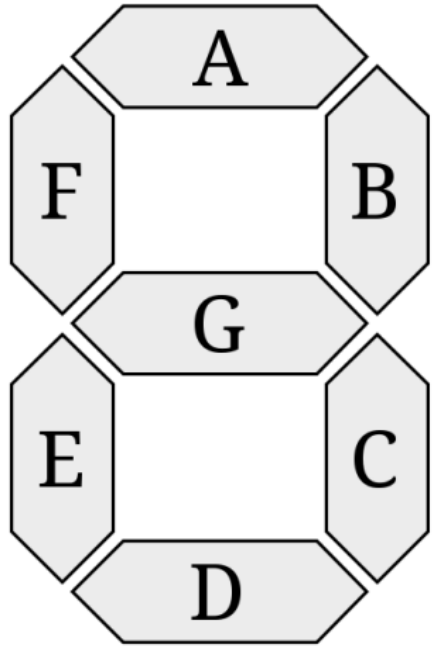


SOMMATORE BINARIO



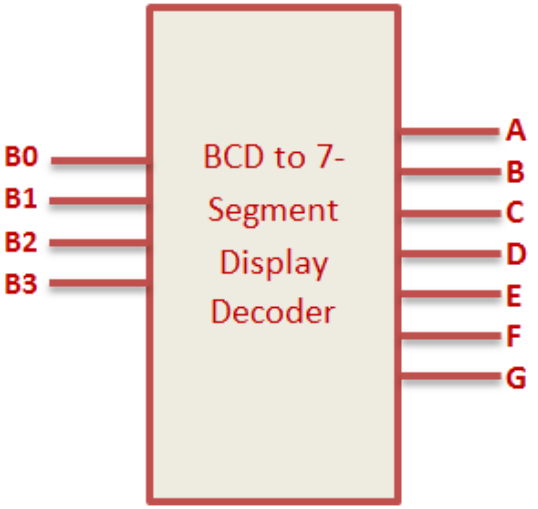
DECODIFICATORE A 7 SEGMANTI

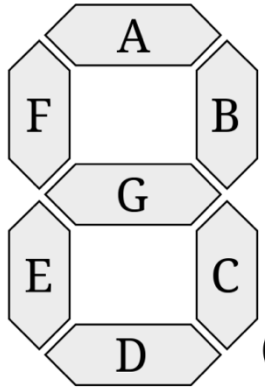




DECODIFICATORE A 7 SEGMANTI

x_3	x_2	x_1	x_0	N	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	1	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1





DECODIFICATORE A 7 SEGMANTI