

MOMENTI IMPORTANTI

Aristotele V/F categorie
Fibonacci 0 s.decimale
Leibniz 0,1 s.binario

$$A \oplus A = A$$

Babbage HW Analytical Engine
Lovelace SW Numeri di Bernoulli

Boole AND OR NOT
Logica proposizionale
 $X \cdot X = X$

PERSONAGGI

Shannon
Boolean algebra
Switching circuits

Turing
Macchina di Turing



	nascita	morte	età	
Aristotele	383	322	61	a.c.
Fibonacci	1170	1242	72	
Leibniz	1646	1716	70	
Babbage	1791	1871	80	
Lovelace	1815	1852	37	
Boole	1815	1864	49	
Shannon	1916	2001	85	
Turing	1912	1954	42	

2.276	Aristotele-Turing
1.553	Aristotele-Fibonacci
476	Leibniz-Fibonacci
145	Babbage-Leibniz
169	Lovelace-Leibniz
169	Boole-Leibniz
101	Shannon-Boole
97	Turing-Boole

Scansione
temporale

Età dei principali
protagonisti
dell'era informatica

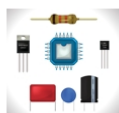
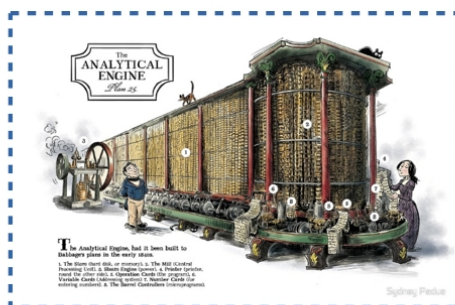
George Boole
Logic + algebra → Boolean algebra
AND OR NOT
Il passaggio dal decimale al binario

$$X \cdot X = X$$



ADA

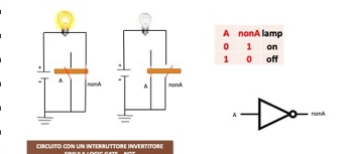
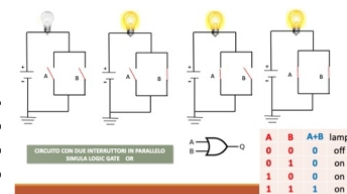
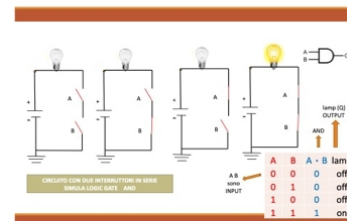
HW & SW
Charles Babbage & Ada Lovelace



Momento Tecnologico
I transistor

Key word

Algoritmo
Programma
Calcolabilità
Bit & BYTE
AND OR NOT
HW e SW
Logic Gates
Algebra Boolean
Logica
combinatoria



Claude Elwood Shannon
Logic Gates
Boolean algebra & Switching circuits

INTRODUZIONE	3
I BIT & BYTE	4
LE LOGIC GATES	6
LE MAPPE DI KARNAUGH	14
FUNZIONE BOOLEANA	14
FLIP-FLOP	19

Figura 1 La pascalina	3
Figura 2 La macchina di Leibniz	4
Figura 3 bit & BYTE	5
Figura 4 Le tre logic gates fondamentali	6
Figura 5 Le Logic Gates	7
Figura 6 Sommatore completo	8
Figura 7 Half-Adder porta XOR e porta AND per il carry	8
Figura 8 porta XOR	9
Figura 9 half-adder dattiloscritto con carry	10
Figura 10 full adder	11
Figura 11 Più sommatore	12
Figura 12 Nativi americani	13
Figura 13 f iniziale	17
Figura 14 f ridotta secondo metodo di Robinson	18
Figura 15 Circuito flip-flop	19
Figura 16 Archiviazione dati da Wikipedia	19
Figura 17 Pierce	19
Figura 18 Charles Sanders Peirce	20
Figura 19 CHIP 7400	20
Figura 20 Porte logiche universali tratto da Wikipedia	20

LOGIC

I bit & Byte

Introduzione

Questa è una appendice dove viene trattato l'argomento bit e BYTE e la loro matematica o meglio aritmetica. Il problema è sempre stato creare un meccanismo, una macchina che potesse eseguire automaticamente i calcoli. Il passaggio dalle macchine calcolatrici ai moderni computer è una storia affascinante con sviluppi davvero sorprendenti.

Storicamente la prima calcolatrice fu la pascalina, macchina inventata da Blaise Pascal, effettuava somme e sottrazioni in base a rotismi vari.

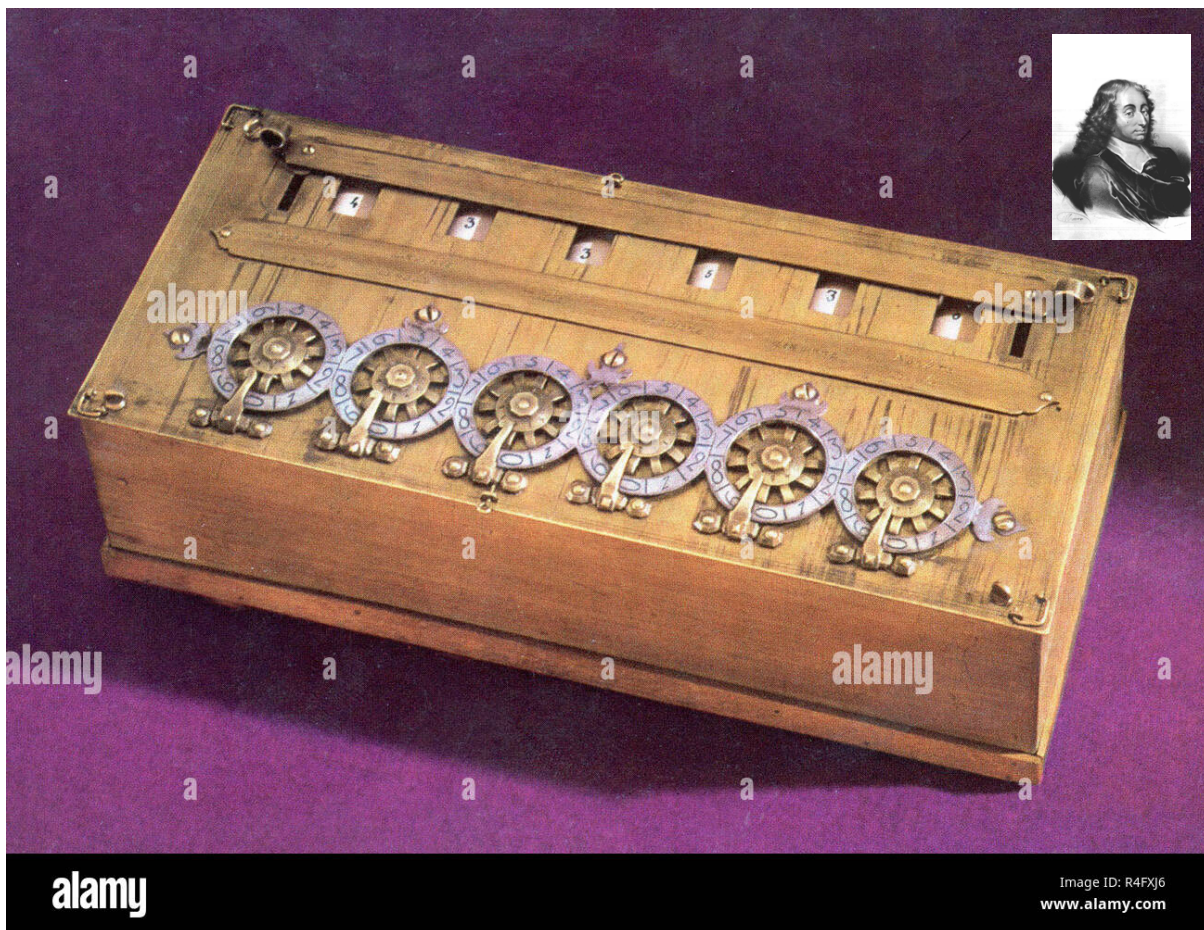


Figura 1 La pascalina

La pascalina è lo strumento di calcolo precursore della moderna calcolatrice. Fu inventata nel 1642 dal matematico e filosofo francese Blaise Pascal, da cui prese il nome. Lo strumento consente di aggiungere e sottrarre numeri composti da un massimo di dodici cifre, operando automaticamente i riporti.

La macchina calcolatrice di Leibniz, altro grande filosofo e matematico, invece eseguiva tutte e quattro le operazioni.



Figura 2 La macchina di Leibniz

La calcolatrice di Leibniz è stata la prima calcolatrice meccanica della storia in grado di eseguire le quattro operazioni matematiche. Fu ideata intorno al 1672 dal matematico e filosofo tedesco Gottfried Wilhelm von Leibniz, ma il progetto venne ultimato solo nel 1694.

La calcolatrice *di Leibniz* (in inglese Stepped Reckoner) è stata la prima calcolatrice meccanica della storia in grado di eseguire le quattro operazioni...

I bit & BYTE

La materia prima dei computer sono i bit, contrazione delle parole Binar Digit (cifra binaria):

0 1

Il sistema numerico binario è utilizzato perché l'elettricità ha due stati: ON (acceso) OFF (spento). Un singolo segnale elettrico può essere attivo ON o non attivo OFF.

Le "parole" che il computer adopera sono i Byte che sono stringhe di otto bit. La prima domanda che possiamo porci è:

i bit quante disposizioni con ripetizioni possiamo avere?

$$D_n^k = D_2^8 = 2^8 = 256$$

Dove **n** è il numero degli oggetti e **k** il numero dei posti.

Ricapitolando:

1. Adoperiamo un sistema duale, binario, 0 e 1
2. Con questo sistema numerico gestiamo i segnali elettrici i cui stati sono due ON OFF
3. Le parole o WORD del computer sono i BYTE, una serie di otto bit con ripetizioni che danno la possibilità di 256 caratteri.

Questa è la base di calcolo di ogni PC! Vediamo con quali metodi e artefici possiamo manipolare i bit! Possono eseguire le quattro operazioni? E se del caso, la manipolazione dei BYTE¹ come possiamo far interpretare al computer le WORD in un linguaggio da noi comprensibile.

¹ Le parole **bit** e **BYTE** sono scritte rispettivamente in minuscolo e maiuscolo. Il BYTE è anche chiamato la WORD (parola) con la quale il computer comunica.

0	00000000	128	10000000
1	00000001	129	10000001
2	00000010	130	10000010
3	00000011	131	10000011
4	00000100	132	10000100
5	00000101	133	10000101
6	00000110	134	10000110
7	00000111	135	10000111
8	00001000	136	10001000
9	00001001	137	10001001
10	00001010	138	10001010
11	00001011	139	10001011
12	00001100	140	10001100
13	00001101	141	10001101
14	00001110	142	10001110
15	00001111	143	10001111
16	00010000	144	10010000
17	00010001	145	10010001
18	00010010	146	10010010
19	00010011	147	10010011
20	00010100	148	10010100
21	00010101	149	10010101
22	00010110	150	10010110
23	00010111	151	10010111
24	00011000	152	10011000
25	00011001	153	10011001
26	00011010	154	10011010
27	00011011	155	10011011
28	00011100	156	10011100
29	00011101	157	10011101
30	00011110	158	10011110
31	00011111	159	10011111
32	00100000	160	10100000
33	00100001	161	10100001
34	00100010	162	10100010
35	00100011	163	10100011
36	00100100	164	10100100
37	00100101	165	10100101
38	00100110	166	10100110
39	00100111	167	10100111
40	00101000	168	10101000
41	00101001	169	10101001
42	00101010	170	10101010
43	00101011	171	10101011
44	00101100	172	10101100
45	00101101	173	10101101
46	00101110	174	10101110
47	00101111	175	10101111
48	00110000	176	10110000
49	00110001	177	10110001
50	00110010	178	10110010
51	00110011	179	10110011
52	00110100	180	10110100
53	00110101	181	10110101
54	00110110	182	10110110
55	00110111	183	10110111
56	00111000	184	10111000
57	00111001	185	10111001
58	00111010	186	10111010
59	00111011	187	10111011
60	00111100	188	10111100
61	00111101	189	10111101
62	00111110	190	10111110
63	00111111	191	10111111
64	01000000	192	11000000
65	01000001	193	11000001
66	01000010	194	11000010
67	01000011	195	11000011
68	01000100	196	11000100
69	01000101	197	11000101
70	01000110	198	11000110
71	01000111	199	11000111
72	01001000	200	11001000
73	01001001	201	11001001
74	01001010	202	11001010
75	01001011	203	11001011
76	01001100	204	11001100
77	01001101	205	11001101
78	01001110	206	11001110
79	01001111	207	11001111
80	01010000	208	11010000
81	01010001	209	11010001
82	01010010	210	11010010
83	01010011	211	11010011
84	01010100	212	11010100
85	01010101	213	11010101
86	01010110	214	11010110
87	01010111	215	11010111
88	01011000	216	11011000
89	01011001	217	11011001
90	01011010	218	11011010
91	01011011	219	11011011
92	01011100	220	11011100
93	01011101	221	11011101
94	01011110	222	11011110
95	01011111	223	11011111
96	01100000	224	11100000
97	01100001	225	11100001
98	01100010	226	11100010
99	01100011	227	11100011
100	01100100	228	11100100
101	01100101	229	11100101
102	01100110	230	11100110
103	01100111	231	11100111
104	01101000	232	11101000
105	01101001	233	11101001
106	01101010	234	11101010
107	01101011	235	11101011
108	01101100	236	11101100
109	01101101	237	11101101
110	01101110	238	11101110
111	01101111	239	11101111
112	01110000	240	11110000
113	01110001	241	11110001
114	01110010	242	11110010
115	01110011	243	11110011
116	01110100	244	11110100
117	01110101	245	11110101
118	01110110	246	11110110
119	01110111	247	11110111
120	01111000	248	11111000
121	01111001	249	11111001
122	01111010	250	11111010
123	01111011	251	11111011
124	01111100	252	11111100
125	01111101	253	11111101
126	01111110	254	11111110
127	01111111	255	11111111

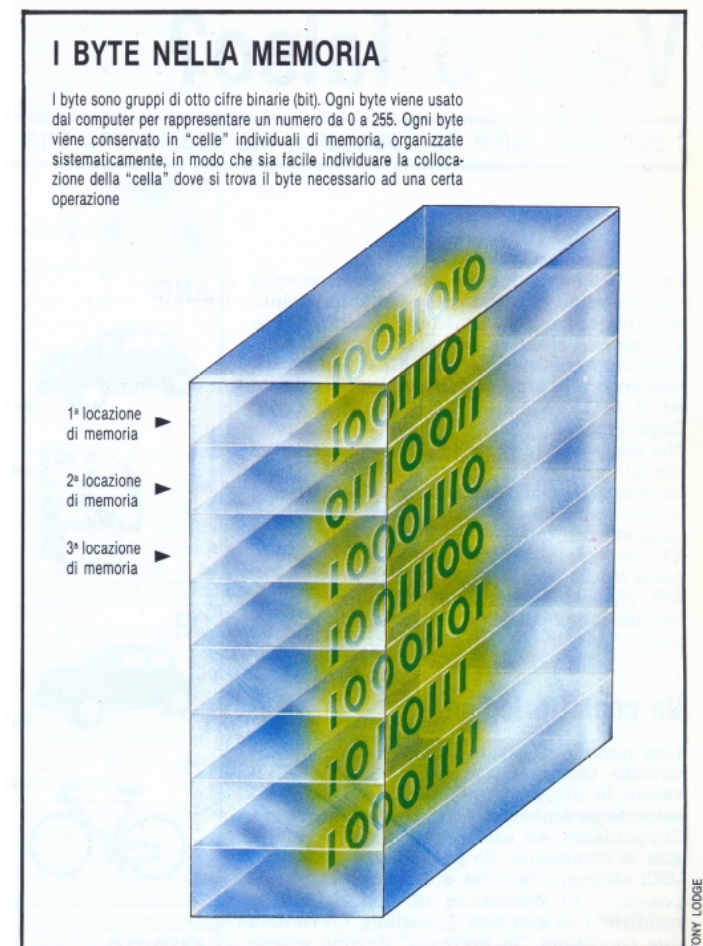
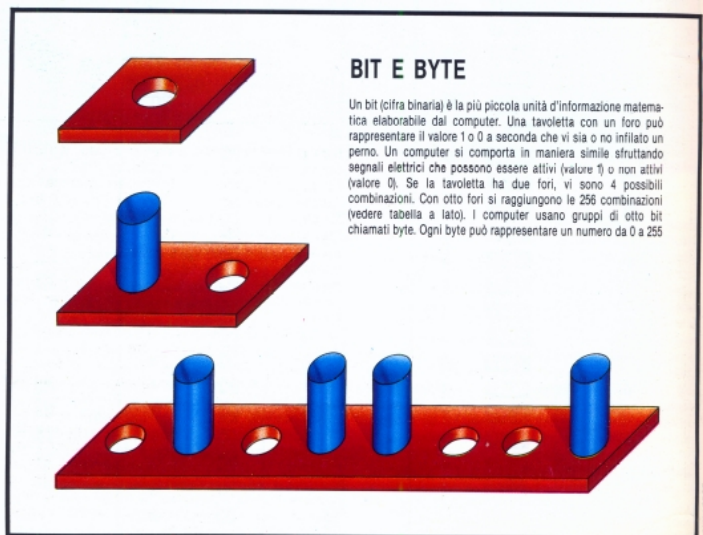


Figura 3 bit & BYTE

Le Logic Gates

Prima di vedere quali sono gli strumenti matematici di un computer, facciamo un riepilogo sulle Logic Gates. Il padre teorico fu George Boole che formalizzò matematicamente la logica aristotelica, creando la “Boolean algebra” che prevedeva l’introduzione dei connettivi logici:

AND (congiunzione) OR (disgiunzione) NOT (negazione)

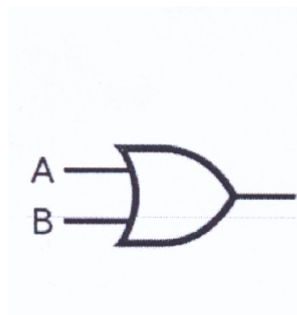
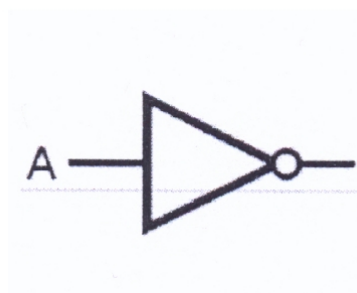
AND; OR sono due connettivi biargomentali, mentre NOT è mono argomentale.

Dopo l’introduzione dell’algebra Booleana fu Claude Elwood Shannon, ottanta anni dopo, che la applicò ai circuiti elettrici costruendo le prime Logic Gates, diventando il padre dell’informatica moderna.

Una porta (gate) logica è un dispositivo che svolge una funzione booleana, un’operazione logica eseguita su uno o più ingressi binari che produce un singolo output binario.

Porta NOT
Entra un segnale, esce il suo contrario.

NOT		
	A	nonA
	0	1
	1	0



Porta OR
È una disgiunzione
Entrano due segnali

OR			
	A	B	S
	0	0	0
	1	0	1
	0	1	1
	1	1	1

AND			
	A	B	S
	0	0	0
	1	0	0
	0	1	0
	1	1	1

Porta AND
È una congiunzione
Entrano due segnali

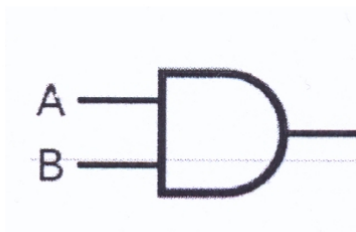
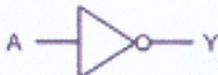


Figura 4 Le tre logic gates fondamentali

Con le combinazioni di queste tre porte si possono costruire altre porte logiche, di seguito:

Logic function	Logic symbol	Truth table	Boolean expression															
Buffer		<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	Y	0	0	1	1	$Y = A$									
A	Y																	
0	0																	
1	1																	
Inverter (NOT gate)		<table><tr><th>A</th><th>Y</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Y	0	1	1	0	$Y = \bar{A}$									
A	Y																	
0	1																	
1	0																	
2-input AND gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$
A	B	Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
2-input NAND gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = \overline{A \cdot B}$
A	B	Y																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
2-input OR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
2-input NOR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = \overline{A + B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
2-input EX-OR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
2-input EX-NOR gate		<table><tr><th>A</th><th>B</th><th>Y</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = \overline{A \oplus B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Questo è un elenco delle porte logiche usate principalmente nei circuiti elettronici. Vi sono riportate anche le notazioni booleane.

La porta NAND, detta anche negazione congiunta, è una porta AND e alla sua uscita è collegata una porta NOT. Il risultato è che la tavola delle verità è quella riportata, nella quale solo con due segnali positivi si ottiene lo 0.

La porta, disgiunzione inclusiva detta oppure dal **vel** latino, permette la costruzione di altre due porte: NOR (OR con negazione NOT) e XOR.

La porta XOR detta disgiunzione esclusiva **aut aut** (o fai questo o fai altro) è una porta importante per la nostra trattazione: osservate la tavola di verità.

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

TAVOLA delle VERITA' di XOR

Figura 5 Le Logic Gates

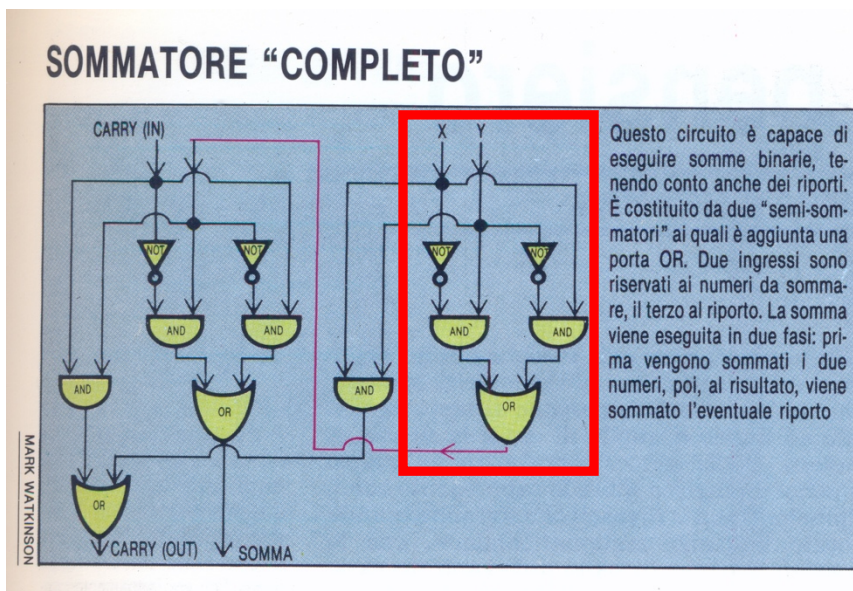
Proviamo a sommare due numeri binari, 0 e 1, e osservare i casi che si possono avere.

$0 + 0 = 0$
$1 + 0 = 1$
$0 + 1 = 1$
$1 + 1 = 0 \text{ carry } 1$

Tabella delle somme aritmetiche possibili con 0 1

Le regole dell'aritmetica sono sempre le stesse. I primi tre passaggi sono intuitivi, il problema arriva quando effettuiamo la somma 1+1 che offre come risultato 10, quindi scrivo 0 e riporto (carry) 1.

Come possiamo fare eseguire queste somme al computer e come possiamo gestire il carry? La figura sottostante mi offre un sommatore completo con l'utilizzo delle tre porte logiche.



Nel riquadro rosso ho evidenziato una porta XOR, che è la porta che simula i risultati delle somme di 0 1. X e Y sono i due numeri che introduciamo esempio:

X=1 Y=1

SOMMA = 0

CARRY (out) 1

Risultato finale 1+1=10

Figura 6 Sommatore completo

Lo schema semplificato di un half-adder o semi sommatore è visualizzato nella figura nella quale compare una porta XOR che esegue tutte le somme e per il carry utilizziamo una porta AND che fornisce il famoso riporto quando i due segnali sono positivi.

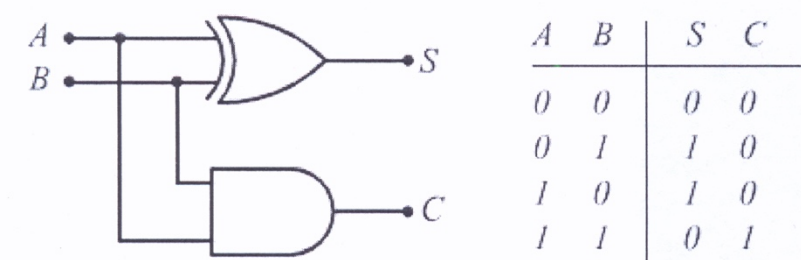


Figura 7 Half-Adder porta XOR e porta AND per il carry

Di seguito presento alcune pagine dattiloscritte.

Inizio con la spiegazione della porta XOR.

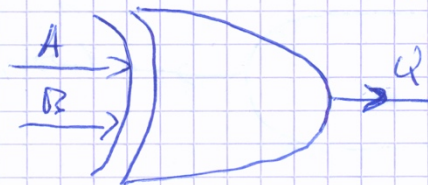
la porta XOR

disgiuntiva esclusiva
(out out)

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

$$A \oplus B$$

$$A \vee B$$



Half adder o semisommatore

Operazioni di somma con i numeri binari:

$$0 + 0 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$1 + 1 = 0 \quad \text{carry } 1$$

Come si vede corrisponde ad una porta XOR

$$Q = A \oplus B = A \bar{B} + \bar{A} B$$

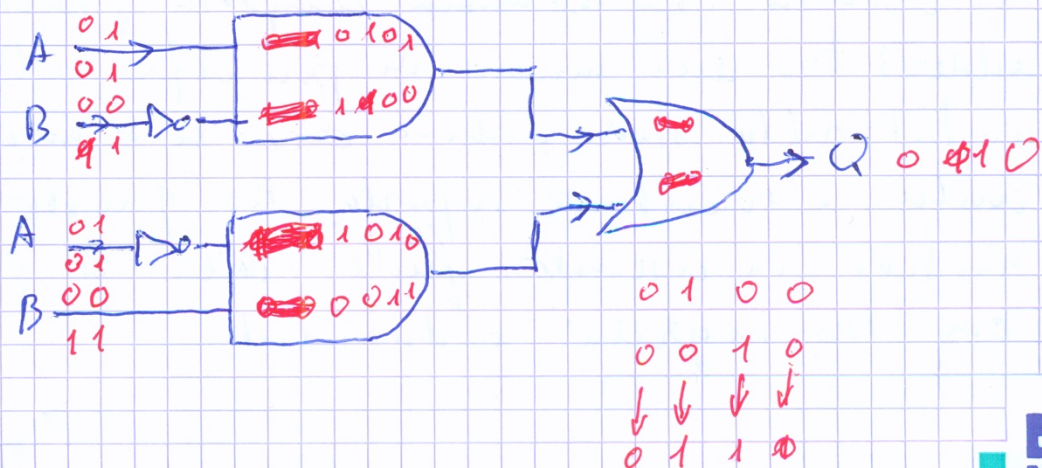


Figura 8 porta XOR

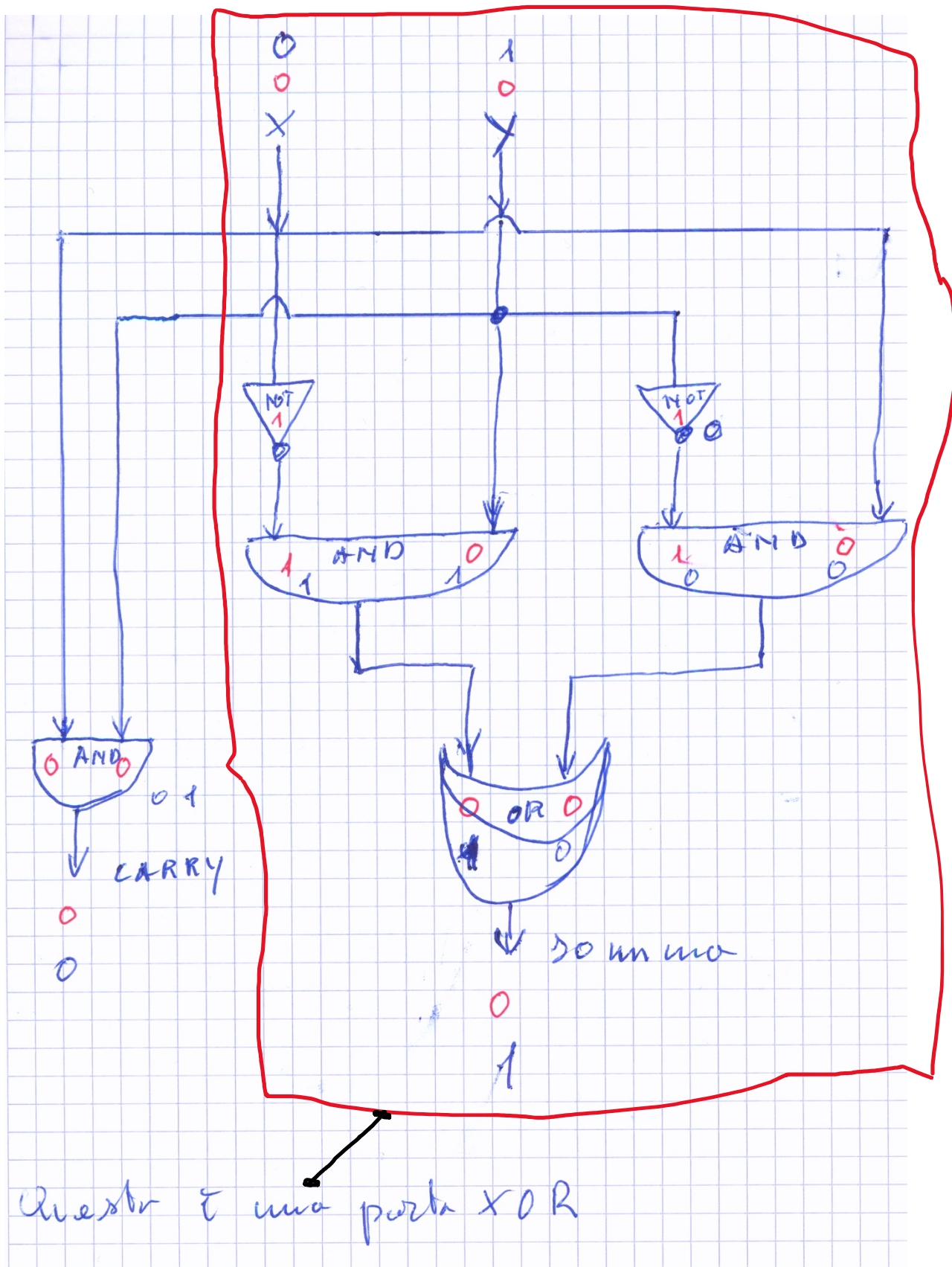
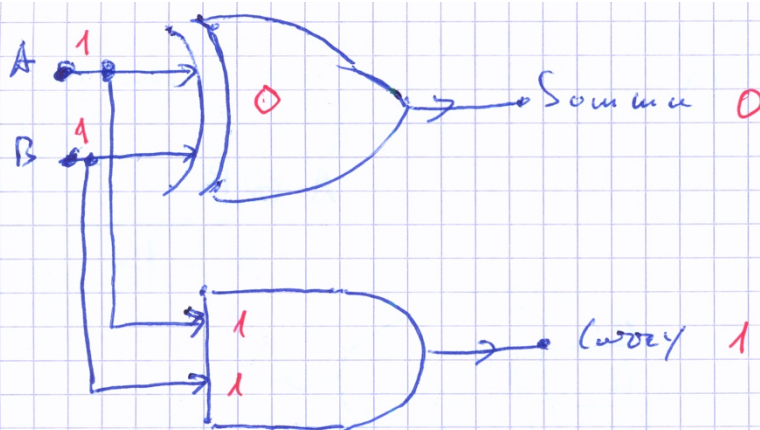
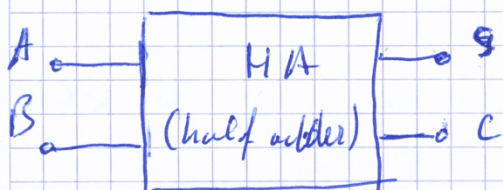


Figura 9 half-adder dattiloscritto con carry



A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Questo è il Half Adder



$$S = A \oplus B$$

$$C = A \cdot B$$

Per ottenere la somma completa (full adder) tra due numeri di più cifre, oltre ai bit dello stesso ordine occorre sommare anche l'eventuale riporto. Per questo motivo il circuito full-adder si presenta con tre ingressi e due uscite.

Figura 10 full adder

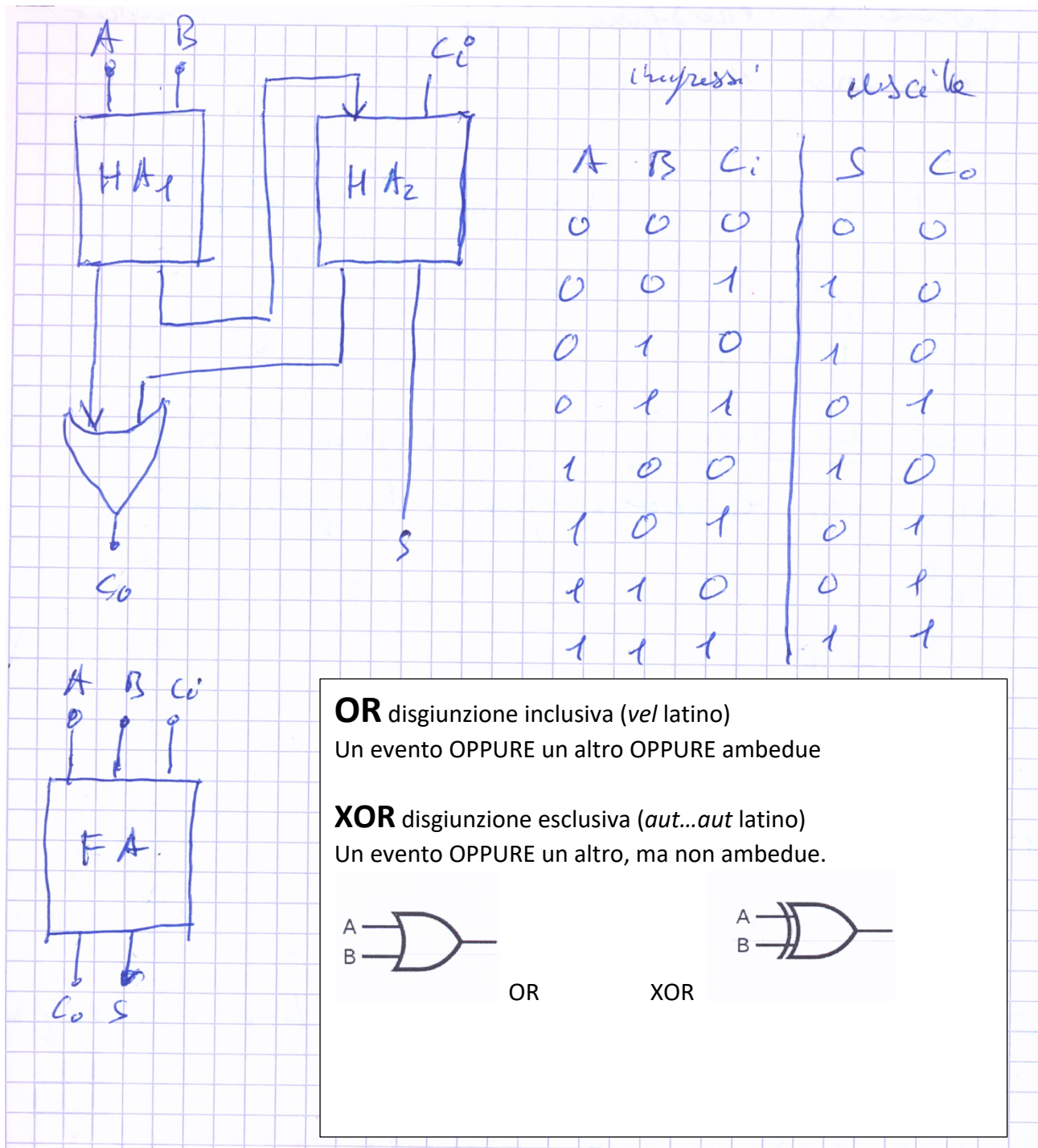


Figura 11 Più sommatore

Abbiamo visto che utilizzando le Logic Gates è possibile eseguire l'operazione di somma con i numeri binari: li possiamo manipolare!

Benché i computer siano in grado tutte le funzioni aritmetiche, la circuiteria interna è in realtà capace di fare solo le somme. Tutte le altre operazioni si ottengono mediante ingegnose combinazioni di porte logiche (i sommatore) e di funzioni controllate da SW.

Come avete potuto notare basta poco per complicare i circuiti, ma il pericolo più grande che può condurre o errore è la RIDONDANZA dei numeri binari e avere circuiti molto complicati da implementare nei CHIP.

L'ingegnere Informatico Maurice Karnaugh è colui che ha introdotto le cosiddette mappe di Karnaugh per aiutarci a risolvere questo problema.

Laureato all'università Yale nel 1952, fu governatore emerito dell'ICCC (International Council for Computer Communication). Lavorò come ricercatore ai Laboratori Bell dal 1952 al 1966 e al centro di ricerca IBM dal 1966 al 1993. Insegnò inoltre informatica al Politecnico della New York University dal 1980 al 1999, e dal 1975 alla morte è stato membro dell'IEEE per i suoi lavori sull'utilizzo di tecniche numeriche nelle telecomunicazioni.

Le mappe di Karnaugh essendo complicate sia come argomento, si può fare ricorso anche a pagine dattiloscritte.



Figura 12 Nativi americani

Le mappe di Karnaugh

Le mappe di Karnaugh sono di una importanza cruciale per la progettazione e la sintesi delle reti logiche. In sostanza sono uno strumento grafico che ci aiuta nella semplificazione delle espressioni booleane, senza addentrarci nell'algebra booleana che è di per sé abbastanza ostica e complicata.

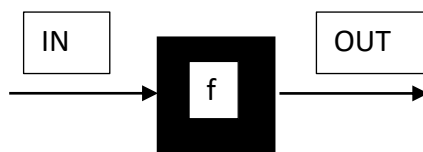
Funzione booleana

Iniziamo con il definire una funzione booleana.

È una funzione che associa bit a bit.

*In matematica e in informatica, una funzione booleana a n variabili è una funzione $f=f(x,y,z,...)$ con $x,y,z,...$ variabili indipendenti, f variabile dipendente, si dice **booleana** di variabili **booleane** se e solo se ciascuna delle variabili indipendenti può avere solo valore 0 o 1, quindi ha un dominio $\{0;1\}$ e la variabile dipendente può anch'essa avere solo valore 0 o 1.*

La funzione può essere vista come una black-box dove entra un INPUT che viene elaborato e sforna un OUTPUT, le mappe semplificano il lavoro all'interno della black-box.



Supponiamo di avere una funzione f a 3 variabili x, y, z .

La rappresentazione in forma tabulare, cioè tutti i casi possibili che sono otto, i risultati sono stati ipotizzati a caso.

$$2^3 = 8$$

IN				OUT	
	x	y	z	f(xyz)	
1	0	0	0	1	1
2	0	0	1	0	
3	0	1	0	1	3
4	0	1	1	1	4
5	1	0	0	1	5
6	1	0	1	0	
7	1	1	0	0	
8	1	1	1	1	8

In questa rappresentazione tabulare sono rappresentati tutti i casi possibili di combinazione delle variabili booleane con a fianco i risultati della funzione $f(xyz)$. Questa può essere rappresentata in forma SP, dove S e P stanno per Somme e Prodotti. Si procede considerando tutti i casi nei quali la funzione assume il valore 1. Questi casi riguardano le righe:

1 3 4 5 8

1	$\bar{x} \bar{y} \bar{z}$	f vale 1, xyz sono segnate negare perché IN (0,0,0)
3	$\bar{x} y \bar{z}$	f vale 1, x e z negare IN (0,1,0)
4	$\bar{x} y z$	f vale 1, con x negata IN (0,1,1)
5	$x \bar{y} z$	f vale 1, con y negata IN (1,0,1)
8	$x y z$	f vale 1 con IN (1,1,1)

Le variabili IN hanno come valore 0 si indicano segnate, mentre le variabili IN con 1 si scrivono normalmente. Alla fine la funzione diventa questa espressione analitica:

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

Il segno (sottinteso) è una moltiplicazione tra le variabili è identificato come un prodotto (porta AND), il segno + è quello della disgiunzione inclusiva (porta OR). Il valore, come sappiamo, della **f** scritta nella forma **SP** (Somme Prodotti): è **1!**

Per il teorema della dualità, se si vuole analizzare anche il caso per il quale $f=0$, basta realizzare lo stesso processo visto sopra, sostituire gli IN nei quali $f=0$. La funzione espressa nella forma SP è analoga a quella scritta per il caso di $f=1$.

Per il momento non interessa e restiamo nel caso nel quale $f=1$.

L'espressione analitica è chiaramente rindondante e siamo solo a una funzione a otto bit. È necessario semplificare: lo si può fare con le mappe di Karnaugh.

Lavorando con una funzione a tre variabili otteniamo otto possibili combinazioni dette K3.

$$(KN) \rightarrow 2^N \text{ nella forma generale}$$

Nel nostro caso:

$$2^3 = 8 \text{ che si definisce come (K3)}$$

Si disegna un rettangolo con otto riquadri dove inserire gli **IMPLICANTI**, cioè i risultati della f.

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

1 riquadro xyz 000→1

2 riquadro xyz 010→1

3 riquadro xyz 110→0

4 riquadro xyz 100→1

5 riquadro xyz 001→0

6 riquadro xyz 011→1

7 riquadro xyz 111→1

8 riquadro xyz 101→0

L'ordine delle caselle non è lo stesso della forma tabulare, si chiama codifica di Gray.

Alla base delle mappe di Karnaugh ci sono tre concetti, tutti basati sul raggruppamento di 1.

1. I raggruppamenti devono essere potenze di due, si possono fare raggruppamenti 2 a 2, 4 a 4 ecc.
2. Copertura della f, devo coprire tutti gli uno.
3. GL concetto di adiacenza (Geometrico Logico) due caselle si dicono adiacenti se hanno un lato in comune

Nel nostro caso abbiamo due raggruppamenti di due coppie di 1 segnati con un rettangolo rosso. Per l'1 dell'ultima casella si adopera il principio dell'adiacenza come se il rettangolo fosse pieghevole e quindi adiacente all'1 della prima casella. L'adiacenza è segnalata con dei lazos verdi. In questo caso varia solo la y.

1° Riquadro rosso

$$\begin{array}{cc} \bar{x} \bar{y} \bar{z} & + & \bar{x} y \bar{z} & = & \bar{x} \bar{z} (\bar{y} + y) & = & \bar{x} \bar{z} \\ 000 & & 010 & \end{array}$$

Questo è il risultato del primo IMPLICANTE. Da due *minitermini* composti da tre valori di bit ne otteniamo solo uno composto da due valori di bit.

2° Riquadro rosso

$$\begin{array}{cc} \bar{x} y z & + & x y z & = & yz (\bar{x} + x) & = & yz \\ 011 & & 111 & \end{array}$$

Questo è il risultato del secondo IMPLICANTE. Come sopra, in questo caso varia solo la x.

3° Adiacenza tra l'ultimo 1 e il primo 1 segnati dai lazos.

$$\begin{array}{cc} \bar{x} \bar{y} \bar{z} & + & x \bar{y} \bar{z} & = & \bar{y} \bar{z} (\bar{x} + x) & = & \bar{y} \bar{z} \\ 000 & & 100 & \end{array}$$

Questo è il risultato dell'adiacenza. Come sopra, varia solo la x.

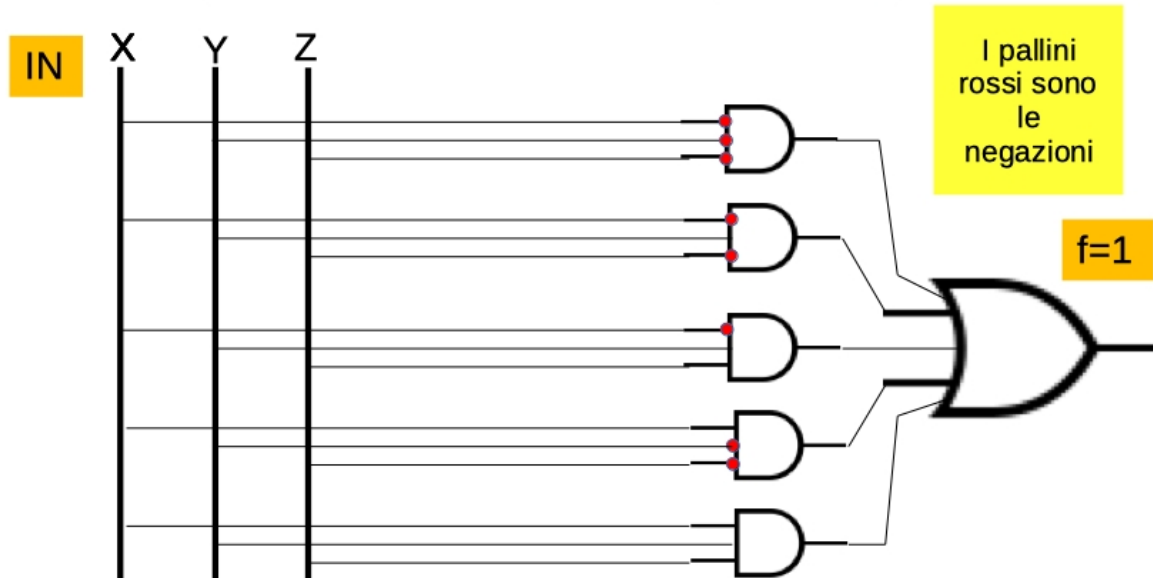
Questo metodo si chiama riduzione alla Robinson e la f che otteniamo è la seguente:

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

Costruiamo i circuiti relativi alle due forme di f realizzate: si nota subito la semplificazione ottenuta nel realizzare i circuiti.

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

	IN			OUT	
	x	y	z	f(xyz)	
1	0	0	0	1	1
2	0	0	1	0	
3	0	1	0	1	3
4	0	1	1	1	4
5	1	0	0	1	5
6	1	0	1	0	
7	1	1	0	0	
8	1	1	1	1	8



La funzione è rappresentata da 5 minitermini che danno come risultato 1. Questa rappresentazione è la forma SP, Somme Prodotti. I prodotti sotto intesi, sono porte AND le somme sono porte OR.

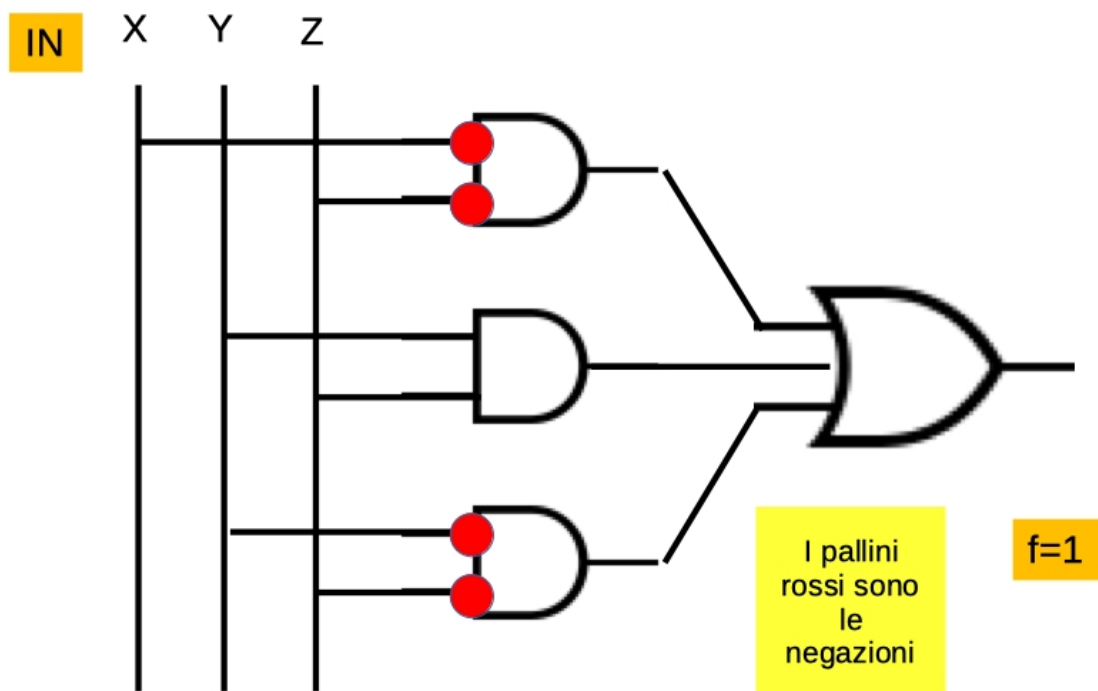
Figura 13 f iniziale

Funzione f in funzione del OUT 1

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

Funzione f ridotta con il metodo di Robinson

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



La funzione è rappresentata da 3 minitermini che danno come risultato 1. Questa rappresentazione è la forma SP, Somme Prodotti. I prodotti sotto intesi, sono porte AND le somme sono porte OR. Nella f iniziale i minitermini erano prodotti di 3 fattori, nella forma ridotta vi sono prodotti di 2 fattori.

Figura 14 f ridotta secondo metodo di Robinson

Flip-Flop

Questa strana parola sta ad indicare un particolare circuito che è in grado di memorizzare un bit in modo stabile. Usando sempre e le Logic Gates siamo in grado di creare la memoria dei risultati ottenuti.

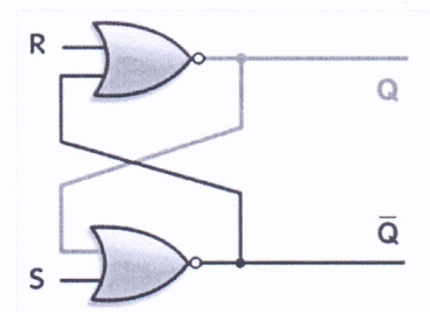
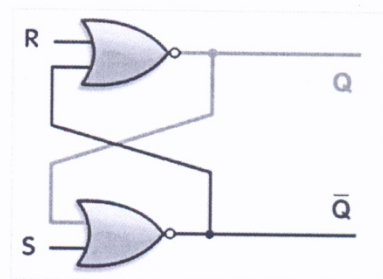


Figura 15 Circuito flip-flop

Archiviazione dei dati e logica sequenziale

I cancelli logici possono anche essere utilizzati per contenere uno stato, consentendo l'archiviazione dei dati. Un elemento di stoccaggio può essere costruito collegando diversi cancelli in un circuito "a chicca". I circuiti di bloccaggio sono utilizzati nella memoria statica ad accesso casuale. Disegni più complicati che utilizzano i segnali dell'orologio e che cambiano solo su un bordo ascendere o in caduta dell'orologio sono chiamati flip-flops ". Formalmente, un flip-flop è chiamato circuito bistabile, perché ha due stati stabili che può mantenere indefinitamente. La combinazione di più flip-flop in parallelo, per memorizzare un valore a più bit, è nota come registro. Quando si utilizza una di queste impostazioni di gate, il sistema complessivo ha memoria; viene quindi chiamato un sistema logico sequenziale poiché la sua uscita può essere influenzata dai suoi stati precedenti, cioè dalla *sequenza* di stati di input. Al contrario, l'output dalla logica combinata è puramente una combinazione dei suoi attuali input, non influenzato dagli stati di input e output precedenti.



Animazione di come funziona un latch SR NOR gate.

Questi circuiti logici sono utilizzati nella memoria del computer. Variano nelle prestazioni, in base a fattori di velocità, complessità e affidabilità dello storage e molti tipi diversi di design vengono utilizzati in base all'applicazione.

Figura 16 Archiviazione dati da Wikipedia

Per chiudere un omaggio ad uno dei più grandi filosofi e matematici degli USA: Charles Sanders Peirce. Il suo contributo nella moderna era informatica è stato fondamentale.



Figura 17 Pierce

"It is not knowing,
but the love of
learning, that
characterizes the
scientific man."

- Charles Sanders Peirce

SPRINGER NATURE
On This Day



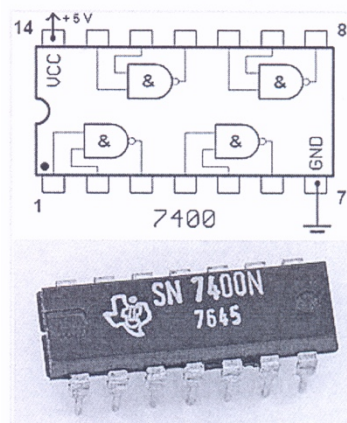
Figura 18 Charles Sanders Peirce

“ Vi sono tre cose che non possiamo mai sperare di raggiungere con il ragionamento: la certezza assoluta, l'esattezza assoluta, l'universalità assoluta. ”

Charles Sanders Peirce



Esempi congruenti con quanto trattato fin ad ora



Il chip 7400, contenente quattro NAND. I due pin aggiuntivi forniscono energia (+5 V) e collegano il terreno.

Figura 19 CHIP 7400

Porte logiche universali

Charles Sanders Peirce (durante il 1880-1881) ha mostrato che le porte NOR da sole (o in alternativa solo le porte NAND) possono essere utilizzate per riprodurre le funzioni di tutte le altre porte logiche, ma il suo lavoro su di esso è stato inedito fino al 1933.^[18] La prima prova pubblicata è stata di Henry M. Sheffer nel 1913, quindi l'operazione logica NAND è talvolta chiamata *tratto di Sheffer*; il NOR logico è talvolta chiamato *freccia di Peirce*.^[19] Di conseguenza, queste porte sono talvolta chiamate *porte logiche universali*.^[20]

Figura 20 Porte logiche universali tratto da Wikipedia