Algebra booleana e sistemi di numerazione

Algebra di Bool

I circuiti logici sono componenti hardware che gestiscono informazioni binarie. Un circuito logico di base è detto porta logica. Potremmo immaginare un calcolatore, allora, come una complessa rete logica, un insieme di porte logiche opportunamente connesse e, che può essere descritto per mezzo di una particolare notazione matematica che specifica lo stato di ogni porta, *l'algebra di Bool* o *algebra booleana*.
Le funzioni dell'algebra booleana sono isomorfe ai circuiti digitali, ovvero, un circuito può essere descritto tramite un'espressione booleana e viceversa.
Tale algebra, contempla dolo due costanti: 0 e 1, falso e vero ed è basata su tre operatori fondamentali: AND, OR e NOT:

- **AND**: si definisce operatore di prodotto logico. Il valore del prodotto logico è 1 se il valore di tutti gli operandi è il simbolo 1

A	B		X
0	0	$\rightarrow$	0
0	1	$\rightarrow$	0
1	0	$\rightarrow$	0
1	1	$\rightarrow$	1

- **OR**: si definisce operatore di somma logica. Il valore della somma logica è 1 se almeno uno degli operandi è il simbolo 1

A	B		X
0	0	$\rightarrow$	0
0	1	$\rightarrow$	1
1	0	$\rightarrow$	1
1	1	$\rightarrow$	1

- **NOT**: si definisce operatore di negazione. Inverte il valore della costante su cui opera

A		X
0	$\rightarrow$	1
1	$\rightarrow$	0

Funzioni booleane

Si dicono funzioni booleane le funzioni con variabili booleane sia nel dominio che nel codominio.

$$f : B^n \to B$$

Una funzione booleana può avere una o più variabili nel codominio.
Il valore di una funzione booleana è una variabile booleana (Vero o Falso).

Proprietà dell'algebra booleana

Commutativa	$a + b = b + a$	$a \cdot b = b \cdot a$
Associativa	$(a + b) + c = a + (b + c)$	$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
Idempotenza	$a + a = a$	$a \cdot a = a$
Assorbimento	$a + (a \cdot b) = a$	$a \cdot (a + b) = a$
Distributiva	$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$	$a + (b \cdot c) = (a + b) \cdot (a + c)$
Min e Max	$a \cdot 0 = 0$	$a + 1 = 1$
Elemento neutro	$a + 0 = a$	$a \cdot 1 = a$
Complemento	$a \cdot !a = 0$	$a + !a = 1$
De Morgan	$!(a + b) = !a \cdot !b$	$!(a \cdot b) = !a + !b$

Sistemi di numerazione in base B

La base di un sistema di numerazione posizionale è il numero di cifre, inclusa quella per lo zero, che lo stesso sistema usa per rappresentare i numeri. La cifra di minor valore è sempre lo zero, mentre le altre sono nell'ordine $1, 2, 3, ..., B - 1$.
Un numero intero x lo si rappresenta come $(c_n c_{n-1} ... c_2 c_1 c_0)_B$.

$$x = c_n B^n + c_{n-1} B^{n-1} + ... + c_2 B^2 + c_1 B^1 + c_0 B^0$$

Dunque, da destra verso sinistra, la cifra meno significativa è c_0, quella più significativa c_n.
Allo stesso modo, un numero frazionario y $(0 c_1 c_2 ... c_n)_B$

$$y = c_1 B^{-1} + c_2 B^{-2} + ... + c_n B^{-n}$$

Nota che: con n cifre in base B si rappresentano tutti numeri interi positivi da 0 a $B^n - 1$. Ad esempio, in base $B = 10$ ed $n = 2$

$$0 \text{ a } 10^2 - 1 = 99$$

Sistema binario, bit e byte

La base 2 è la più piccola base per un sistema di numerazione, con sole due cifre: 0 e 1 (*bit*-binary digit). Consideriamo la sequenza binaria $(010111)_2$ e calcoliamo il corrispondente in base 10:

$$(010111)_2 \rightarrow (?)_{10} = 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$
$$= 0 + 16 + 0 + 4 + 2 + 1$$
$$= (23)_{10}$$

L'elemento alla base della rappresentazione delle informazioni nei calcolatori è il *byte*, un numero binario ad otto cifre (8 bit).

Se per passare da un valore in base 2 ad un valore in base 10 è necessario scrivere i valori dati in forma polinomiale moltiplicando per potenze di 2 in base alla posizione occupata dalle singole cifre, il procedimento per passare, invece, da un numero in base 10 al rispettivo in base 2 implica dover dividere ripetutamente il valore di partenza e i quozienti che man mano si ottengono per 2, fino ad ottenere zero.
Facciamo un esempio. Consideriamo il numero $(20)_{10} \rightarrow (?)_2$

$20 : 2 = 10$ resto 0
$10 : 2 = 5$ resto 0
$5 : 2 = 2$ resto 1
$2 : 2 = 1$ resto 0
$1 : 2 = 0$ resto 1

a questo punto possiamo, dal baso verso l'alto: $(20)_{10} = (10100)_2$

Aritmetica binaria

Addizione

1. incolonnare i numeri partendo dalla cifra più a destra
2. addizionare le cifre corrispondenti tenendo conto dei quattro possibili casi:

   $0 + 0 = 0$
   $1 + 0 = 1$
   $0 + 1 = 1$
   $1 + 1 = 0$ con riporto 1 nella colonna di sinistra

Sommiamo $(10100)_2 + (111)_2$

01000	riporti
10100	+
111	=
11011	

Sottrazione

1. incolonnare i numeri come per l'addizione

2. sottrarre le cifre corrispondenti tenendo conto dei quattro possibili casi:

   $0 - 0 = 0$
   $1 - 0 = 1$
   $0 - 1 = 1$ con prestito di 1 della colonna di sinistra
   $1 - 1 = 0$

Nota che quando facciamo un prestito da una colonna a un'altra è come se ricevessimo in prestito due 1!

Sottraiamo $(11110)_2 - (1101)_2$

000(0)1	prestiti
111(0)0	-
01101	=
10001	

Moltiplicazione

1. incolonnare i numeri
2. moltiplicare ciascuna cifra del secondo fattore per il primo, ricordando che:
   $0 \cdot 0 = 0$
   $1 \cdot 0 = 0$
   $0 \cdot 1 = 0$
   $1 \cdot 1 = 1$

3. incolonniamo i prodotti parziali e sommiamo

Moltiplichiamo $(110)_2 \cdot (10)_2$

1	1	0	x
	1	0	=
	0	0	0	+
1	1	0		=
1	1	0	0	

Divisione

1. incolonnare divisore e dividendo:

$$10010 : 10$$

2. prendere un numero di cifre nel dividendo di valore pari o maggiore del divisore

$$(10)010 : 10$$

3. Dividere il gruppo di cifre per il divisore riportando il quoziente parziale a destra

10010	:	10
		1

4. moltiplicare il divisore per 1 e riportare il risultato sotto il dividendo contrassegnato

10010	:	10
10		1

5. eseguire la sottrazione per ottenere il resto parziale

10010	:	10
10		1
—		

6. abbassiamo l'altra cifra

10010	:	10
10		1
—		
—0		

7. se il numero ottenuto è maggiore o uguale al divisore riportiamo 1 nel quoziente, altrimenti 0

10010	:	10
10		10
—		
—0		

8. moltiplichiamo il divisore per il nuovo quoziente parziale e riportiamo il prodotto a sinistra

10010	:	10
10		10
—		
—0		
—0		

9. proseguire come nei punti 5 - 8 fino a quando non terminano le cifre del dividendo

Sistema binario e complemeento a 2

La tecnica più utilizzata in informatica per rappresentare un numero con segno è il *complemento a due*: ogni numero binario che ha il primo bit uguale a 1 è considerato negativo, viceversa, se inizia con 0 è considerato positivo (rappresentazione unica dello zero).
In questo modo, un numero binario di binario di N cifre potrà rappresentare numeri nell'intervallo -2^{n-1} e $2^{n-1} - 1$.

Nella pratica, il complemento a due di un numero binario si calcola invertendo tutti i bit del numero e poi sommando 1. Se consideriamo il numero 5 in binario 0101, la cifra più significativa è 0 (il numero è positivo). A questo punto invertiamo tutti i bit: $0101 \rightarrow 1010$ (complemento a 1) e sommiamo 1 ottenendo il complemento a 2: 1011 è il complemento a 2 del numero 5.

Ricapitolando, supponiamo di avere un numero binario di 8 bit:

$27_{10} =$	00011011_2
complemento a 1 :	00011011_2
	$+1$
	————
$-27_{10} =$	11100101_2

Viceversa, se abbiamo una sequenza di 8 bit e sappiamo che rappresenta un numero intero con segno, con i numeri negativi rappresentati in complemento a 2, per ottenere il numero rappresentato, la prima cosa da fare è a analizzare il segno: se 0, il numero è non negativo, basterà usare la normale conversione binario-decimale. Se invece il bit di segno è 1, allora sarà sicuramente un numero negativo. Quello che si fa è ottenere il modulo, complementando tutti i bit e sommando 1:
Ad esempio, se il numero 11100101 è la rappresentazione in complemento a 2 di un numero, il valore assoluto lo si ottiene in questo modo

complemento a 1 :	00011010
	$+1$

$27_{10} =$	00011011_2

Conversione di un numero reale in binario

Si convertono separatamente la parte intera e la parte frazionaria. La parte intera nel modo visto in precedenza, la parte frazionaria, invece si moltiplica per 2 e si toglie la parte intera del risultato, che diventa la prima cifra dopo il punto. Si procede allo stesso modo per le successive cifre, finchè la parte frazionaria non si annulla o finchè non abbiamo ottenuto un numero sufficiente di cifre binarie.

Consideriamo il numero 5.375_{10}. La parte intera è 101_2. Per convertire la parte frazionaria 0.375_{10}:

$0.375 \cdot 2 = 0.750$ la parte intera 0 diventa la prima cifra binaria dopo il punto
$0.750 \cdot 2 = 1.5$ la parte intera \$01 diventa la seconda cifra binaria dopo il punto
$0.5 \cdot 2 = 1.0$ la parte intera \$01 diventa la terza cifra binaria dopo il punto

il risultato è: 101.011_2

Sistema esadecimale

Altra base largamente usata in informatica e $B = 16$.

$$0123456789ABCDEF$$

oltre il 9, la corrispondenza in base 10 è:

$A = (10)_{10}$ $B = (11)_{10}$
$C = (12)_{10}$ $D = (13)_{10}$
$E = (14)_{10}$ $F = (15)_{10}$

Se condiseriamo il valore 0x9F1C, il corrispondente in base 10 sarà:

$$\begin{aligned}(9F1C)_{16} \rightarrow (?)_{10} &= 9 \cdot 16^3 + 15 \cdot 16^2 + 1 \cdot 16^1 + 12 \cdot 16^0 \\ &= 36864 + 3840 + 16 + 12 \\ &= (40732)_{10}\end{aligned}$$

Una cifra esadecimale corrisponde esattamente a 4 bit.

0x	0b
0	0000
1	0001
2	0010

0x	0b
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

È facile intuire che in questo modo è possibile rappresentare numeri binari molto grandi usando poche cifre. Ad esempio: 32 bit con solo 8 cifre esadecimali.
Il procedimento per convertire dal sistema esadecimale a quello binario è estremamente semplice: è sufficiente espandere ciascuna cifra esadecimnale con i quattro bit corrispondenti. Proviamo con il numero 0x0F7A:

0	F	7	A
0000	1111	0110	1010

$$(0F7A)_{16} = (0000111101101010)_2$$