Breve introduzione storica
Nel 1854, il prof. Boole pubblica un trattato ormai famosissimo: "Le leggi del pensiero".
Obiettivo finale del trattato è di far nascere la matematica dell'intelletto umano, un'algebra in grado di interpretare le proposizioni logiche fondamentali. Egli affermava che "Il calcolo logico deve avere un posto fra le forme d'analisi matematica ormai generalmente riconosciute". La logica in sostanza, secondo Boole, doveva essere vista come un capitolo della matematica.
Il lavoro di Boole fu dimenticato per quasi 100 anni. Solo nel 1950, quando nasce il calcolatore elettronico, l'algebra di Boole diventa lo strumento principale per la progettazione dei suoi circuiti, i cosiddetti circuiti logici.

Variabili binarie e linguaggio naturale
Oggetto fondamentale dell'algebra di Boole sono le *variabili binarie*, ossia variabili che possono assumere soltanto due valori distinti: 1 e 0. Per non generare confusione con i valori matematici è più esatto parlare di *VERO* e *FALSO*. Che cosa è una variabile?
Si immagini una scatola alla quale attribuiamo il nome Pippo. Tale nome serve ad individuare la scatola ma è completamente distinto dal suo contenuto. Supponiamo che abbiamo a disposizione due foglietti di carta: uno su cui è scritto SI e l'altro su cui è scritto NO. Possiamo introdurre dentro la scatola solo uno dei due foglietti alla volta. Il nome Pippo associato alla variabile rimane sempre quello, per tutta la storia della variabile, mentre il suo contenuto può variare: in qualche momento può essere SI ed in qualche altro può essere NO. All'interno del calcolatore è chiaro che queste variabili utilizzeranno un solo bit: il valore 1 indica VERO o SI, mentre il valore 0 rappresenta FALSO o NO.
Punto di partenza di Boole è la stretta correlazione che può stabilirsi fra una certa famiglia di proposizioni del linguaggio naturale e le variabili binarie. Vi sono proposizioni che hanno il valore di una *dichiarazione*: ad esempio "sta piovendo" oppure "fa caldo". Per esse si può dire se sono vere o false. Tali proposizioni, se vengono combinate insieme, definiscono una frase il cui risultato dipende dalla loro combinazione. Allo stesso modo vi sono *variabili* che indicheremo come *indipendenti*, con dei nomi stabiliti, che definiscono attraverso le operazioni fondamentali se un *variabile* che chiamiamo *dipendente* è vera o no. Per convenzione il contenuto di una scatola cioè il valore di una variabile è uguale a 1 (cioè SI) se quella proposizione è vera mentre il contenuto della stessa scatola è uguale a NO o a 0 se quella proposizione è falsa.
Ad esempio, immaginiamo una scatola a cui diamo il nome P in cui se inseriamo un biglietto su cui è scritto 1 ciò vuol dire per convenzione che in quel momento piove, ossia P=1 vuol dire che sta piovendo, mentre P=0 vuol dire che non sta piovendo.

Operazioni fondamentali
Sulle variabili binarie si possono introdurre tre *operazioni fondamentali*: prodotto logico, somma logica e negazione.

Prodotto logico (*AND*)
Supponiamo che le variabili indipendenti P e S indichino rispettivamente le proposizioni "Sta piovendo" e "Ho tanti soldi", e che la variabile dipendente T rappresenti la proposizione "Prendo il taxi". Possono accadere i seguenti casi:

"non piove", "non ho tanti soldi" allora "non prendo il taxi"
"non piove", "ho tanti soldi" allora "non prendo il taxi"
"piove", "non ho tanti soldi" allora "non prendo il taxi"
"piove", "ho tanti soldi" allora "prendo il taxi"

Dunque *prendo il taxi* solo **se piove ed ho tanti soldi**.
Allora l'operazione prodotto logico, o AND o "×", interpreta il significato di quella operazione logica detta *congiunzione* e che usualmente viene indicata con "e" oppure "e inoltre". Come si vede l'AND è un'operazione binaria, ossia necessita di due variabili. É possibile visualizzare tale operazione tramite quella che viene chiamata *tabella di verità*, ossia una tabella che riassume per ogni combinazione delle variabili indipendenti il valore assunto da quella dipendente.

Variabile A	Variabile B	Prodotto logico (A × B)
0	0	0
0	1	0
1	0	0
1	1	1

Come si vede la tabella mostra tutte le possibili combinazioni delle due variabili binarie e il relativo risultato (A × B), detto prodotto logico delle due variabili. Il termine prodotto proviene dal comportamento dell'operazione logica che assume 1 solo nel caso in cui tutte e due le variabili assumono valore VERO.

Somma logica (*OR*)
Supponiamo che P = "Sta piovendo", F = "Fa freddo", I = "Metto l'impermeabile".
La variabile dipendente I è legata alle due variabili P e F esattamente in questo modo:

"non piove", "non fa freddo" allora "non metto l'impermeabile"
"non piove", "fa freddo" allora "metto l'impermeabile"
"piove", "non fa freddo" allora "metto l'impermeabile"
"piove", "fa freddo" allora "metto l'impermeabile"

Quindi *metto l'impermeabile* quando fa freddo, quando piove o quando piove e fa freddo.
In sostanza l'operazione somma logica, o OR o "+" ha il significato che i logici chiamano di *disgiunzione* e che nel linguaggio ordinario rappresentiamo con la congiunzione *oppure*. Anche l'OR è un'operazione binaria. La tabella di verità è:

Variabile A	Variabile B	Prodotto logico (A + B)
0	0	0
0	1	1
1	0	1
1	1	1

La quarta riga prova appunto che i simboli 0 ed 1, in questo caso, non sono dei valori numerici (ci aspetteremmo in una somma 1 + 1 = 10 ossia 0 guardando il bit meno significativo).

Complementazione o negazione
Il significato logico dell'operazione di complementazione booleana è quello di dire che se la variabile indipendente è 0 la variabile dipendente è 1 e viceversa.
In termini di proposizioni: se la proposizione associata a F è "fa caldo" e la proposizione associata a C è "fa freddo", allora può accadere:

"Non fa caldo" allora "fa freddo"
"fa caldo" allora "non fa freddo".

Il NOT è un'operazione unaria, ossia viene applicata ad una sola variabile e di solito viene indicata con una lineetta sopra la variabile.

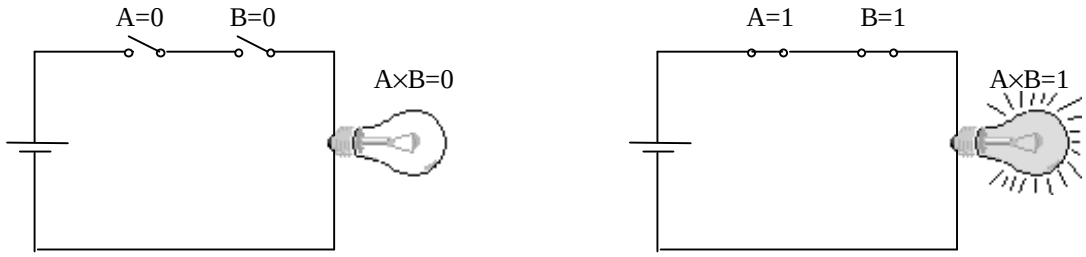
Variabile A	Negazione ($\overline{A}$)
0	1
1	0

Lavorando su variabili binarie complementare vuol dire cambiare l in 0 e viceversa.

Rappresentazione circuitale delle operazioni logiche

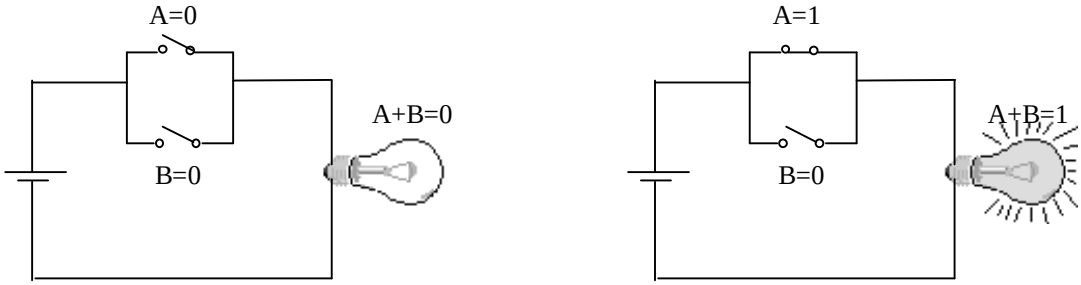
È possibile rappresentare le operazioni logiche mediante i circuiti elettrici. In un circuito elettrico il simbolo rappresenta un generatore di tensione (come ad esempio la pila) e il simbolo rappresenta un interruttore.

Supponiamo di avere un circuito con due interruttori collegati in **serie**:



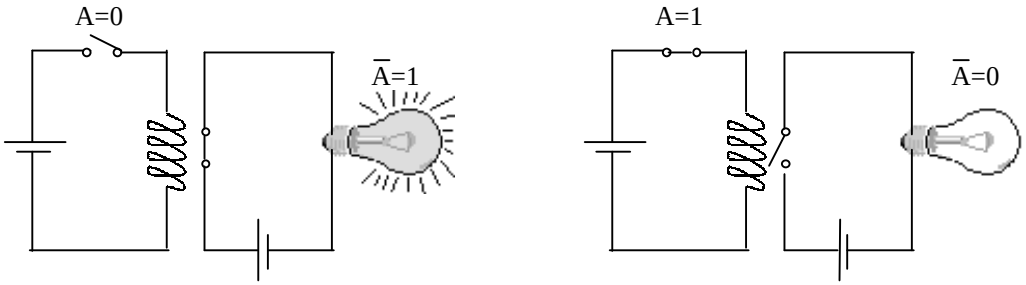
È evidente che per accendere la lampadina tutti e due gli interruttori devono essere chiusi. Basta che uno dei due interruttori sia aperto (o entrambe) che la lampadina rimane spenta. Dando all'interruttore aperto valore 0 e a quello chiuso valore 1 è evidente che la serie rappresenta l'operatore AND.

Supponiamo adesso di avere un circuito con due interruttori collegati in **parallelo**:



In questo caso basta che uno dei due interruttori sia chiuso per accendere la lampadina. La lampadina sarà spenta solo nel caso in cui tutti e due gli interruttori sono aperti. Il parallelo quindi rappresenta l'operatore OR.

Per rappresentare il NOT utilizziamo due circuiti. Supponiamo di avere un interruttore elettromagnetico ossia una *bobina* che agisce su un interruttore: quando l'interruttore A è aperto (A = 0) nella bobina non circola corrente e l'interruttore nel secondo circuito rimane chiuso; quando passa la corrente (interruttore A = 1) nella bobina circola corrente che fa aprire l'interruttore nel secondo circuito. In tal modo se A = 0, NOT A = 1 e viceversa.



Esempio di calcolo logico
Si considerino le seguenti 4 variabili indipendenti:

LUIAMALEI, LEIAMALUI, LUISOLDI, LEISOLDI

che possiamo pensare materializzate con 4 scatole che hanno i nomi indicati. I contenuti delle 4 scatole in un certo istante mi indicano la situazione globale delle variabili indipendenti. Vogliamo arrivare alla determinazione della variabile dipendente: SISPOSANO.
Perché si sposino è necessario che entrambe si amino e che almeno uno dei 2 abbia i soldi (per il matrimonio). In termini algebrici possiamo scrivere:

SISPOSANO = LUIAMALEI $\times$ LEIAMALUI $\times$ (LUISOLDI + LEISOLDI)

prima faremo il prodotto LUIAMALEI $\times$ LEIAMALUI secondo le regole definite, poi effettuiamo la somma logica LUISOLDI + LEISOLDI e ne moltiplichiamo il risultato al precedente valore.
Si può verificare facilmente che la variabile dipendente SISPOSANO assumerà valore 1 solo nel caso in cui LUIAMALEI = 1, LEIAMALUI = 1 e LUISOLDI + LEISOLDI = 1.
Volendo possiamo realizzare la tabella di verità per la funzione descritta:

LUIAMALEI	LEIAMALUI	LUISOLDI	LEISOLDI	SISPOSANO
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	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Proprietà delle operazioni logiche
Le operazioni logiche fondamentali viste hanno certe proprietà, vediamo le più importanti di esse.

1) *Associatività della somma logica*

$(A + B) + C = A + (B + C)$

Per la dimostrazione è sufficiente costruire una tabellina avente tutte le combinazioni di valori delle 3 variabili indipendenti A, B e C, quindi (A+B), (B+C) e infine (A+B)+C e A+(B+C):

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

Regola generale: la somma logica di *n* variabili assume il valore 0 solo se tutte le variabili hanno valore 0 ed assume valore 1 in tutti gli altri casi.

2) *Associatività del prodotto logico*

$(A \times B) \times C = A \times (B \times C)$

A	B	C	(A×B)	(B×C)	(A×B)×C	A×(B×C)
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Regola generale: il prodotto logico di *n* variabili assume il valore 1 solo se tutte le variabili hanno valore 1 ed assume valore 0 in tutti gli altri casi.

3) *Doppia negazione*

$\overline{(\overline{A})} = A$

Se considero la variabile A e la complemento e poi complemento il risultato trovo ancora il valore della variabile A.

4) *Distributività del prodotto rispetto la somma*

$A \times (B + C) = (A \times B) + (A \times C)$

(Si lascia la dimostrazione come esercizio)

5) *Distributività della somma rispetto il prodotto*

$A + (B \times C) = (A + B) \times (A + C)$

(Si lascia la dimostrazione come esercizio)

6) *Assorbimento*

$A + A = A$
$A \times A = A$

La dimostrazione è abbastanza facile: se A=0 allora 0 + 0 = 0, se A = 1 allora 1 + 1 = 1 ed analoga dimostrazione nel caso del prodotto.

7) *Leggi di De Morgan*

$$\overline{A + B} = \overline{A} \times \overline{B}$$

$$\overline{A \times B} = \overline{A} + \overline{B}$$

(Si lascia la dimostrazione come esercizio)

La legge di De Morgan può essere estesa. Se in un'espressione booleana sostituisco:

1. ogni variabile V con il suo complemento $\overline{V}$
2. ogni occorrenza del simbolo + con il simbolo $\times$
3. ogni occorrenza del simbolo $\times$ con il simbolo +

trovo il complemento dell'espressione data (tutte le volte che l'espressione data assume valore 1 l'espressione nuova assume il valore 0 e viceversa).

Esempio

SISPOSANO = LUIAMALEI $\times$ LEIAMALUI $\times$ (LUIHASOLDI + LEIHASOLDI)

applichiamo le regole indicate:

$$\overline{\text{NONSISPOSANO}} = \text{SISPOSANO} = \overline{\text{LUIAMALEI}} + \overline{\text{LEIAMALUI}} + (\overline{\text{LUIHASOLDI}} \times \overline{\text{LEIHASOLDI}})$$

dunque basta che lui non ami lei o che lei non ami lui oppure lui non ha soldi e inoltre lei non ha soldi ed allora non si sposano.

Funzioni booleane e tabelle di verità

Fino a adesso abbiamo visto alcune funzioni booleane elementari. La generica funzione booleana è un'estensione del concetto di funzione ordinaria con la sola differenza che le variabili in gioco prendono i loro valori in un insieme costituito dai due soli elementi "0" e "1".

Definizione di tipo operativo:
Una **funzione booleana** di *n* variabili specifica, per ogni combinazione dei valori delle *n* variabili indipendenti, il valore della variabile dipendente.

Ad esempio nelle due tabelle di seguito sono definite due differenti funzioni booleane.

A	B	C	VALORE DELLA FUNZIONE U_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

A	B	C	VALORE DELLA FUNZIONE U_2
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ogni volta che si definiscono diverse combinazioni di 0 e 1 in corrispondenza alla variabile dipendente viene dato un diverso significato logico all'espressione, viene definita una nuova funzione!

Quante sono le funzioni booleane di N variabili
Conviene chiedersi: quante sono le righe di una tabella di *N* variabili?
Evidentemente queste sono 2^N. Infatti *N* variabili booleane corrispondo a *N* bit, giacché una variabile booleana ha due stati.
Per esempio nel caso $N = 3$ ogni riga sarà una combinazione distinta di 0 e 1 su 3 bit e dunque avrò $2^3 = 8$ righe.

Per individuare il numero di funzioni booleane diverse di *N* variabili, basta riflettere sul fatto che, per descrivere una funzione diversa devo scrivere in corrispondenza alla variabile dipendente una combinazione di valori diversi. Quante combinazioni distinte di 0 e 1 posso scrivere su $M = 2^N$ posizioni (ossia bit)?

Esattamente 2^M ossia date N variabili logiche è possibile creare 2 elevato a 2^N funzioni booleane distinte.

Se fosse $N = 4$ avremmo $2^4 = 16$ righe della tabella di verità che corrispondono a $2^{16} = 65536$ funzioni booleane differenti!
Nel caso $N = 3$, poiché ogni riga sarà una combinazione distinta di 0 e 1 su 3 bit avrò $2^3 = 8$ righe, e quindi $2^8 = 256$ differenti funzioni booleane.

Costruzione della funzione booleana associata ad una tabella di verità
Sia data una certa funzione logica descritta tramite la tavola della verità e si voglia scrivere la funzione booleana corrispondente. Supponiamo sia assegnata la seguente tabella di verità, con 4 variabili indipendenti:

A	B	C	D	U
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	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Vogliamo costruire U = f (A,B,C,D) ossia l'espressione algebrica contenente le variabili A, B, C e D e gli operatori booleani che legano le variabili.
La funzione U vale 1 in corrispondenza delle ultime 3 righe della tabella. Allora possiamo scrivere:

U = ad una 1° espressione algebrica che corrisponde alla terzultima riga della tabellina +
una 2° espressione algebrica che corrisponde alla penultima riga della tabellina +
una 3° espressione algebrica che corrisponde all'ultima riga della tabellina.

Poiché il risultato dell'operazione di somma vale 1 quando è VERO uno qualsiasi dei 3 addendi, se scrivessimo un'espressione che è VERA solo nel caso della terzultima riga, una che è VERA solo nel caso della penultima riga ed una che è VERA solo in relazione all'ultima riga, avremmo risolto il nostro problema. Nel caso in questione:

$A \times B \times \overline{C} \times D \rightarrow$ Terzultima riga

$A \times B \times C \times \overline{D} \rightarrow$ Penultima riga

$A \times B \times C \times D \rightarrow$ Nel caso dell'ultima riga:

e dunque $U = A \times B \times \overline{C} \times D + A \times B \times C \times \overline{D} + A \times B \times C \times D$

È facile verificare che la funzione U così scritta vale 1 in tutte e tre le condizioni elementari degli ultimi 3 casi della tabellina e vale 0 in tutti gli altri casi.

Se per esempio consideriamo la 4° riga (A = 0 B = 0 C = 1 D = 1) vediamo subito che tutti e tre gli addendi valgono 0 (si ricordi che $A \times B \times C \times D = 0$ anche se una sola delle 4 variabili è nulla):

$A \times B \times \overline{C} \times D = 0 \times 0 \times 0 \times 1 = 0$

$A \times B \times C \times \overline{D} = 0 \times 0 \times 1 \times 0 = 0$

$A \times B \times C \times D = 0 \times 0 \times 1 \times 1 = 0$

Ne consegue che la funzione booleana è appunto:

$U = A\,B\,\overline{C}\,D + A\,B\,C\,\overline{D} + A\,B\,C\,D$

(il segno del "×" si può omettere (come nel calcolo algebrico ordinario)).

Riassumendo la funzione booleana data la relativa tabella di verità con *N* variabili:

1) avrà tanti termini quanto sono le righe con valore 1
2) ogni termine è rappresentato dal prodotto logico delle *N* variabili mettendo la variabile negate se in quella riga tale variabile vale 0
3) i termini ottenuti vengono sommati.

Costruzione della tabella di verità associata a una funzione booleana
Il procedimento è inverso a quello visto prima. Supponiamo sia data la funzione booleana:

$U = A\,B\,\overline{C} + A\,\overline{B}\,C + \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C}$

Si vede subito che la funzione dipende da tre variabili (A, B e C) e che la relativa tabella di verità avrà 4 righe il cui valore di uscita sarà 1. Per individuare tali righe basta sostituire 1 alla variabile non negata e zero a quella negata. Ad esempio:

$A\,B\,\overline{C}$ corrisponde ad A = 1, B = 1, C = 0

$A\,\overline{B}\,C$ corrisponde ad A = 1, B = 0, C = 1

$\overline{A}\,\overline{B}\,C$ corrisponde ad A = 0, B = 0, C = 1

$\overline{A}\,\overline{B}\,\overline{C}$ corrisponde ad A = 0, B = 0, C = 0

Quindi la tabella è:

A	B	C	U
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	0

Riassumendo la tabella di verità associata alla funzione booleana di *N* variabili:

1) avrà tante righe di valore 1 quanti sono i termini della funzione booleana
2) ogni termine corrisponde univocamente ad una riga della tabella secondo la trasformazione: 1 se la variabile è non negata, 0 se la variabile è negata

Semplificazioni delle funzioni booleane
La funzione booleana ottenuta dalla tabella di verità potrebbe essere semplificata utilizzando le proprietà degli operatori booleani. Supponiamo che la funzione booleana sia:

$U = A\,B\,\overline{C}\,D + A\,B\,C\,\overline{D} + A\,B\,C\,D$

Si osservi che fra il 1° e 3° termine posso mettere in evidenza ABD (sfrutto la proprietà distributiva) ottenendo:

$ABD\,(\overline{C}+C) = ABD \times 1 = ABD$ → infatti $\overline{C}+C = 1$ (0 + 1 = 1, 1 + 0 = 1) e la moltiplicazione per uno è ininfluente e dunque ottengo ABD.

Posso anche mettere in evidenza fra il 2° e 3° termine ottenendo:

$ABC\,(\overline{D}+D) = ABC \times 1 = ABC$

Dovrei utilizzare il 3° termine della U due volte. Ciò è ammissibile per la proprietà dell'assorbimento:

$A = A + A$

E nel nostro caso $A\,B\,C\,D = A\,B\,C\,D + A\,B\,C\,D$

Abbiamo dunque ottenuto:

$U = A\,B\,\overline{C}\,D + A\,B\,C\,\overline{D} + A\,B\,C\,D + A\,B\,C\,D = ABD\,(\overline{C}+C) + ABC\,(\overline{D}+D) =$

$= ABD + ABC$

Funzioni booleane e circuiti logici
Le tabelle di verità e le funzioni booleane servono per costruire i circuiti logici che consentono al calcolatore di effettuare le operazioni richieste. Le funzioni booleane dovrebbero essere semplificate per ridurre il numero di operatori da utilizzare, dato che ogni operatore corrisponde ad un dispositivo fisico. La riduzione consente un risparmio in termini di costo e di risorse utilizzate. Nel caso precedente siamo partiti da una funzione:

$U = A\,B\,\overline{C}\,D + A\,B\,C\,\overline{D} + A\,B\,C\,D$

che necessitava di nove AND, due OR e due NOT e abbiamo ottenuto una versione più semplice

U = ABD + ABC

che utilizza quattro AND e un OR con un evidente risparmi in termini di numero di dispositivi utilizzati, pur con lo stesso comportamento della precedente.

Fra le attività che i circuiti logici devono effettuare rientrano anche le operazioni matematiche.
Supponiamo di dover sommare due variabili binarie:

A = 011
B = 111

Possiamo costruire le funzioni logiche che rappresentano la somma di bit e il relativo riporto.
Per la cifra meno significativa della variabile A e della B devo effettuare la somma su due bit, mentre per le cifre successive devo effettuare la somma fra tre bit (quello di A, quello di B e il riporto del precedente bit).

Definiamo A = $a_{n-1} \ldots a_2\ a_1\ a_0$ e B = $b_{n-1} \ldots b_2\ b_1\ b_0$ dove a_i e b_i (i = 0, …, n-1) possono assumere valori 0 o 1. Supponiamo di voler sommare i due bit meno significativi a_0 e b_0.
Abbiamo due ingressi e due uscite. Dobbiamo quindi definire due funzioni booleane: R_0 per il risultato e C_0 per il riporto (detto *carriage* in inglese).

Vediamo le 2 funzioni booleane:

a_0	b_0	R_0	C_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Le espressioni algebriche sono:

$R_0 = \overline{a_0}\,b_0 + a_0\,\overline{b_0}$
$C_0 = a_0\,b_0$

Entrambe le funzioni non sono semplificabili.
Per effettuare la somma degli altri bit avremo tre differenti variabili booleane: a_i, b_i e C_{i-1} (bit i-esimo di A e di B e riporto del precedente bit) con i = 1, …, n-1. Avremo quindi 3 ingressi e 2 uscite e dunque la tabella di verità avrà 8 righe:

a_i	b_i	C_{i-1}	R_i	C_i
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

Le due funzioni sono:

$$R_i = \overline{a}_i \overline{b}_1 C_{i-1} + \overline{a}_i b_i \overline{C}_{i-1} + a_i \overline{b}_i \overline{C}_{i-1} + a_i b_i C_{i-1}$$

$$C_i = \overline{a}_i b_1 C_{i-1} + a_i \overline{b}_i C_{i-1} + a_i b_i \overline{C}_{i-1} + a_i b_i C_{i-1}$$