

# ARCHITETTURE DI CALCOLO LEZIONE 5

## Algebra di Boole e reti combinatorie

### ALGEBRA DI BOOLE

Per passare dalla rappresentazione mediante tabella di verità alla notazione tramite espressione booleana è necessario:

- Identificare tutte le righe della tabella di verità che danno 1 in output;
- Per ogni riga con un 1 in output scrivere la configurazione delle variabili che la definiscono (tutte le variabili della configurazione saranno in AND tra loro);
- Collegare tramite OR tutte le configurazioni ottenute.

La rappresentazione così ottenuta è detta in **prima forma canonica** o somma di prodotti. Nell'esempio qui riportato abbiamo 3 variabili (X Y Z), 4 moltiplicazioni e 3 somme. Ricorda che per somma s'intende OR e per prodotto AND; la somma di prodotti è quindi equivalente ad OR di AND.

Ciascuna variabile si può presentare in due formati:

-**Negato**, con valore pari a 0 e con simbolo un trattino apicalmente.

-**Non negato**, con valore pari ad 1.

Prendiamo le uscite in corrispondenza delle quali c'è il valore 1 e poi andiamo a scrivere un'espressione di somma di prodotti.

Ciascun termine di questa somma è costituito da 3 variabili, che possono essere in formato NEGATO se la variabile vale 0 nella linea ed in formato NON NEGATO se la variabile vale 1 nella linea.

Qui abbiamo che per X=0 Y=0 Z=1 il risultato è uguale a 1, quindi il termine è corrispondente per x negato, y negato e z non negato; il segno sopra le lettere va a significare che quel termine è negato.

Ricordatevi che quando andiamo a leggere questa espressione di somma e di prodotti, per somma si intende la OR mentre per prodotti si intende la AND, quindi in realtà questa espressione ci indica NOT X AND NOT Y AND Z, i risultati non negati verranno poi addizionati tra di loro tramite la OR; per cui questa espressione può essere chiamata somma di prodotti ma anche or di and.

Questi sono and a 3 termini, noi avevamo visto la forma più semplice a due termini dove il risultato vale uno se entrambi gli input valgono 1, qui vale lo stesso principio per cui il valore risultante nella and a 3 termini vale 1 solo se tutti e tre gli input valgono 1.

Una volta che abbiamo l'espressione in cui vengono considerati i termini che valgono 1 possiamo implementare il circuito. Adesso vi faccio vedere un circuito, non questo poiché viene un po' grandicello ma su un altro.

**Esempio:** Traduzione della funzione rappresentata dalla tabella di verità:

X	Y	Z	V	
0	0	0	0	
0	0	1	1	← $\overline{X}\overline{Y}Z$
0	1	0	1	← $\overline{X}Y\overline{Z}$
0	1	1	0	
1	0	0	1	← $X\overline{Y}\overline{Z}$
1	0	1	0	
1	1	0	0	
1	1	1	1	← $XYZ$

→

$$\overline{X}\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\overline{Z} + XYZ$$

Le dimensioni delle tabelle di verità crescono al crescere delle variabili in input, sono quindi necessarie delle rappresentazioni alternative:

- Si può specificare l'output di ogni funzione booleana esprimendo, tramite una espressione booleana, **quali combinazioni delle variabili di input determinano l'output 1**. Per l'esempio precedente: (011-101-110-111). Questa rappresentazione è importante perché può essere direttamente tradotta in un circuito di porte digitali.

### Convenzioni notazionali

- Una variabile con valore 1 è indicata dal suo nome
- Una variabile con valore 0 è indicata dal suo nome con sopra una barra
- L'operazione di AND booleano è indicata da un  $\cdot$  moltiplicativo oppure viene considerato implicitamente presente
- L'operazione di OR booleano è indicata da un  $+$

$$(011-101-110-111) \Rightarrow \bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ$$

### PORTA XOR

XOR o esclusiva or: restituisce 1 se e solo un solo valore è 1.

Questa porta restituisce valore non nullo (1) se e solo se tra gli input c'è un solo 1 in ingresso, ciò vuol dire che se in input gli 1 sono due allora la porta ci restituirà valore nullo.

Abbiamo poi oltre alle porte di cui abbiamo già parlato (AND, NAND, OR, NOR) la porta XOR ovvero esclusiva or; funziona così: la xor restituisce 1 se e solo se uno solo degli input è pari a 1.

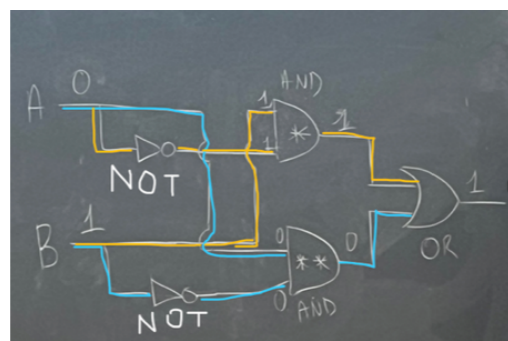
Con le conoscenze che abbiamo fin ora siamo in grado di scrivere una espressione booleana equivalente quindi in prima forma canonica basata su una tabella di verità basata su and, or, xor e not.



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$\Rightarrow \begin{matrix} A\bar{B} \\ AB\bar{} \end{matrix}$$

$$A\bar{B} + AB\bar{} = A \oplus B \text{ (XOR)}$$



## CIRCUITO EQUIVALENTE

È possibile creare dei circuiti equivalenti, ossia circuiti con struttura diversa (per tipo, numero e/o disposizione delle porte logiche) che conducono allo stesso output.

Prendendo in esame il circuito XOR, vediamo che risulta equivalente al seguente circuito: (A negato AND B) OR (A AND B negato).

La cosa interessante è che quando abbiamo in espressione logica come una tra queste possiamo anche progettare un circuito logico equivalente, cioè prendere delle porte logiche differenti ma a formare un circuito per ottenere quel medesimo risultato.

Noi abbiamo due input: A e B; l'input ci serve sia in forma normale che negata, noi possiamo quindi generare dall'input queste due versioni, dopo di che dobbiamo fare due and e una or, di fatto questa è una rete a due livelli in cui le due and faranno da input ad una or; la prima and è con A negato e B, la seconda and è con A diretto e B negato, questo è il circuitino corrispondente alla XOR e realizzato con la and, la or e la not.

Per verificare se è corretto dobbiamo dargli in input tutte e quattro le combinazioni.

Cominciando dalla parte destra questo è un or, mentre questi due sono delle and, ora come mai io ho messo in input nella or il valore di output delle due and?

Per emulare il comportamento della tavola di verità sopra riportata.

Il modo per capire se un circuito è corretto è provarlo con dei test esaustivi in cui vengono immessi tutti i valori possibili di input e vedere poi l'output.

Il prof prova quindi il circuito con tutti i valori della tabella per dimostrare la validità del circuito.

Le funzioni booleane possono essere descritte da più espressioni equivalenti

$$X Y \bar{Z} + X \bar{Y} Z + X Y Z \equiv X(Y + Z)$$

X	Y	Z	V	Y+Z	X(Y+Z)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

**Def.** Due funzioni booleane si dicono *equivalenti* se presentano lo stesso output per qualsiasi configurazione dell'input.

Determinare **la più semplice** espressione booleana equivalente alla funzione data facilita l'interpretazione della funzione stessa e permette di semplificare anche i circuiti logici corrispondenti.

*“Cosa si intende per più semplice?”*

Una espressione  $f'$  è più semplice di una espressione  $f$  ( $f \equiv f'$ ) secondo il **criterio del minor numero di operatori** se il suo calcolo richiede meno operazioni di AND e OR rispetto al calcolo di  $f$ .

N.B. Sebbene questo sia il criterio più utilizzato ne esistono altri

## SECONDA FORMA CANONICA

Esiste anche una **seconda forma canonica** detta prodotti di somme; in questo caso si prendono gli 0 anziché 1, ottenendo i rispettivi circuiti. È importante notare che le possibili forme di

un'espressione sono infinite.

Le funzioni booleane possono essere descritte da ***più espressioni equivalenti***.

Le espressioni sono equivalenti se lo si può dimostrare tramite due metodi:

1. Usando le proprietà dell'algebra.
2. Si scrivono le tavole di verità di entrambe le funzioni e si mettono a confronto i valori di output; se sono uguali allora sono equivalenti.

Due funzioni booleane si dicono equivalenti se presentano lo stesso output per qualsiasi configurazione dell'input.

Dato il risultato dell'espressione posso costruire per ragionamento inverso la tabella di verità.

Esiste la seconda forma canonica che si chiama prodotti di somme, in cui si fa un ragionamento equivalente ma inverso, esistono queste forme perché in realtà ciò che abbiamo precedentemente rappresentato (ovvero un circuito XOR ma ripartito) può essere rappresentato anche mediante questa seconda forma, a dire il vero per ogni tabella di verità esistono infinite espressioni logiche e quindi anche circuiti che possono rappresentarla.

$$f(A, B, C) = (A + B + C) \cdot (A + B + \bar{C}) \cdot (A + \bar{B} + C) \cdot (A + \bar{B} + \bar{C})$$

Usando la tabella della verità:

numero	ABC	Mintermini	Maxtermini
0	000	$\bar{A} \cdot \bar{B} \cdot \bar{C}$	$A + B + C$
1	001	$\bar{A} \cdot \bar{B} \cdot C$	$A + B + \bar{C}$
2	010	$\bar{A} \cdot B \cdot \bar{C}$	$A + \bar{B} + C$
3	011	$\bar{A} \cdot B \cdot C$	$A + \bar{B} + \bar{C}$
4	100	$A \cdot \bar{B} \cdot \bar{C}$	$\bar{A} + B + C$
5	101	$A \cdot \bar{B} \cdot C$	$\bar{A} + B + \bar{C}$
6	110	$A \cdot B \cdot \bar{C}$	$\bar{A} + \bar{B} + C$
7	111	$A \cdot B \cdot C$	$\bar{A} + \bar{B} + \bar{C}$

A	B	C	U	
0	0	0	0	→ (A+B+C)
0	0	1	0	→ (A+B+ $\bar{C}$ )
0	1	0	1	
0	1	1	0	→ (A+ $\bar{B}$ +C)
1	0	0	1	
1	0	1	0	→ ( $\bar{A}$ +B+ $\bar{C}$ )
1	1	0	1	
1	1	1	0	→ ( $\bar{A}$ + $\bar{B}$ + $\bar{C}$ )

Si cerca l'equazione più semplice per rendere il circuito fisico più semplice nella realizzazione ed ottimizzare il computer.

Potrebbe essere più semplice un circuito con solo tipo di porte oppure con meno porte oppure più veloce. In generale vale il **criterio del minor numero di operatori**: *una espressione  $f'$  è più semplice di un'espressione  $f$  se il suo calcolo richiede meno operazioni.*

È chiaro che tra tutte le possibili forme noi dobbiamo cercare di fare quella più efficiente, veloce e che usa il minor numero di copie.

Esiste in realtà anche un modo automatico per generare la forma migliore e più efficiente di un dato circuito.

È importante ricordare che le funzioni booleane possono essere descritte mediante l'uso di più espressioni booleane stesse. Qui c'è un esempio, questa espressione  $x, y$  e  $z + x, y$  e  $z$  ecc ... è equivalente a quest'altra forma.

Esistono due approcci per ricercare il circuito più semplice:

- **Metodo analitico**, basato sull'uso di alcune proprietà;
- **Metodo meccanico**, basato su degli algoritmi tra cui il più importante è quello di Karnaugh.

Noi useremo soprattutto il metodo meccanico.

Come facciamo a capire se una forma è equivalente ad un'altra? Esistono due modi, il primo è sfruttare l'algebra per dimostrare l'equivalenza, il secondo approccio è scrivere la tavola di verità della prima funzione e la confronto con la tavola di verità della seconda, il secondo approccio è più diretto e più semplice soprattutto quando abbiamo a che fare con espressioni con poche variabili.

Ad esempio scriviamo la tabella di verità di questa espressione, ricordate che bisogna formare tutte le combinazioni, valutando a partire dal risultato anche se eventualmente una o più variabili sono negate (il che implicherà un risultato inverso dal quale partire [not y {con  $y=0$ }  $= 1$  nel circuito]).

A questo punto andiamo a calcolare la tavola di verità della seconda espressione per confrontarla mediante il criterio di equivalenza, una volta valutate tutte le casistiche viene calcolata quindi l'espressione finale (nell'esempio abbiamo y or z, e il risultato and x) e si verifica se le colonne delle risultanti sono identiche e quindi si dimostra l'equivalenza tra le due espressioni.

Perché è importante trovare le espressioni più semplici? Perché una volta che io ho un'espressione in realtà io dopo dovrò andarla a realizzare fisicamente su un circuito che fa il medesimo calcolo, se io volessi realizzarlo così come è (in forma complessa) necessiterò di più porte (2 or e 3 and ciascuna con 3 ingressi) se invece andrò a semplificarla avrò bisogno di meno porte ma anche di meno ingressi (1 or e 1 and a 2 ingressi)(quindi da 5 porte a 3 ingressi siamo passati a 2 porte a 2 ingressi), è chiaro che così andrò a ottimizzare il circuito in quanto sarà più veloce, risparmierò componenti, spazio e anche energia.

Immaginando questo concetto a una dimensione maggiore e quindi non su un singolo circuito ma sui molteplici presenti in un computer, si riesce a valutare l'enorme differenza tra l'usare dei circuiti più complessi o più semplificati.

Quello che bisogna ricordare non sono le formule ma solo che per verificare l'equivalenza di due espressioni bisogna confrontare le rispettive tabelle di verità e vedere se ci portano allo stesso risultato.

All'atto pratico è possibile verificare l'equivalenza anche tramite i circuiti ovvero si formano i circuiti corrispondenti alle espressioni e si testa con l'ingresso di corrente se la risposta è la medesima ( $\text{ingresso} \geq 5V \Rightarrow 1$ ,  $\text{ingresso} < 5V \Rightarrow 0$ ).

Le reti si suddividono in reti combinatorie e reti sequenziali.

Possiamo affermare che lo scopo di coloro che formano le reti logiche è quello di rendere le rete stessa più semplice, più semplice però non ha la sempre la stessa accezione, in quanto un circuito o una rete è più semplice se:

- ha lo stesso tipo di porte (tutte and, or o xor ad esempio)
- ha un minor numero di componenti
- è più veloce rispetto alla rete di partenza.

Esistono quindi diversi modi per definire la semplicità di una rete, ma in generale possiamo dire che minore è il numero di porte e quindi di operatori utilizzati meglio sarà la rete.

Esistono due approcci fondamentali per semplificare le reti booleane,

- Il primo è l'approccio analitico, basato sull'uso di alcune regole
- Il secondo è l'approccio meccanico, basato su degli algoritmi, due sono i fondamentali e quello di cui parleremo è l'algoritmo di Karnaugh.

L'algebra di Boole presenta le seguenti **proprietà**:

Legge	Forma AND	Forma OR
Identità	$1A = A$	$0 + A = A$
Annullamento	$0A = 0$	$1 + A = 1$
Idempotenza	$AA = A$	$A + A = A$
Inverso	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutativa	$AB = BA$	$A + B = B + A$
Associativa	$(AB)C = A(BC)$	$(A+B)C = A + (B+C)$
Distributiva	$A + BC = (A+B)(A+C)$	$A(B+C) = AB + AC$
Assorbimento	$A(A+B) = A$	$A + AB = A$
Legge di DeMorgan	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A+B} = \bar{A}\bar{B}$

È possibile verificare le seguenti proprietà tramite il calcolo delle tavole di verità per ciascuna funzione dell'equivalenza; se i risultati sono coincidenti le funzioni si equivalgono.

Tramite questa procedura è possibile verificare tutte le proprietà dell'algebra di Boole. Come nell'algebra binaria anche in quella booleana ci sono delle proprietà come quella distributiva e commutativa.

Ci sono però anche altre proprietà come la proprietà di IDENTITÀ valida sia in porta or che in porta and (1 and a == a oppure 0 or a == a [elemento neutro PORTA valore == valore]), la proprietà di ANNULLAMENTO (0 AND a == 0 oppure 1 or a == 1), la proprietà di IDEMPOTENZA (a and a == a oppure a or a == a), la proprietà di INVERSO (a and !a == 0 oppure a or !a == 1), la proprietà COMMUTATIVA (possiamo fare in ordine diverso entrambe le porte), la proprietà DISTRIBUTIVA (si possono scindere o raggruppare le and), la proprietà di ASSORBIMENTO (x and [x or y] == x oppure x or [x and y] == x) e la legge di De Morgan (il negato di a più il b negato è uguale al negato di a per b negato !x or !y == !x and !y. Ciò ci consente quindi inter-cambiare le porte and e or mediante delle not [CIO CI FA CAPIRE CHE QUALSIASI CIRCUITO PUÒ ESSERE COMPOSTO DI SOLE NAND E SOLE NOR]).

**Legge di DeMorgan:** Tale legge permette di trasformare un'espressione di AND e OR con una OR e NOT oppure AND e NOT. Questo implica che bastano solo NAND o NOR per fare un circuito (con manipolazioni); nella AND l'1 è l'elemento neutro e lo 0 l'elemento nullo mentre per la OR invece lo 0 è l'elemento neutro.

Legge di DeMorgan					$\overline{AB} = \overline{A} + \overline{B}$		$\overline{A + B} = \overline{A}\overline{B}$		
A	B	$\overline{A}$	$\overline{B}$	A+B	$\overline{A} + \overline{B}$	AB	$\overline{AB}$	$\overline{A}\overline{B}$	$\overline{A + B}$
0	0	1	1	0	1	0	1	1	1
0	1	1	0	1	1	0	1	0	0
1	0	0	1	1	1	0	1	0	0
1	1	0	0	1	0	1	0	0	0

Volendo verificare se  $(a \cdot b) \cdot c$  è uguale a  $a \cdot (b \cdot c)$  e quindi verificare la proprietà associativa per la porta and andiamo a effettuare le annesse tabelle di verità e confrontare successivamente i risultati, di fatto per come vediamo in figura le due danno gli stessi risultati e sono dunque equivalenti.

In maniera analoga a come abbiamo fatto per verificare la proprietà associativa possiamo anche verificare la proprietà commutativa per la porta or e quindi dimostrare che  $(a + b) + c$  è equivalente di  $a + (b + c)$ ; infatti in figura possiamo notare che le precedenti espressioni portano allo stesso risultato.

Lo stesso discorso è applicabile alla proprietà distributiva, valida sia per la porta or che per la porta and; non ci andremo a fare i calcoli anche per questa ma la procedura resta la stessa, quindi creiamo tutte le possibili combinazioni e verifichiamo che diano lo stesso risultato.

Per quanto concerne la legge di De Morgan, la quale enuncia che  $(!a) \cdot (!b) == (!a) + (!b)$ , ciò è bivalente cioè può essere letto in ambedue i lati, quindi ciò ci concede di passare dalle porta NOR alla NAND e viceversa; la suddetta legge non vale solo per due variabili ma anche per più variabili (il prof dimostra a due e tre variabili).

#### Proprietà associativa:

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

#### Proprietà distributiva:

A	B	C	BC	A+BC	A+B	A+C	(A+B)(A+C)	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0
0	1	1	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	0	0	0	0	0
1	0	1	0	1	1	1	1	1	1	0	1	1
1	1	0	0	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1



Quando noi disegniamo una porta logica stiamo in realtà rappresentando in maniera grafica un oggetto che è un componente elettronico reale.

Queste porte logiche come sono realizzate a livello hardware ?

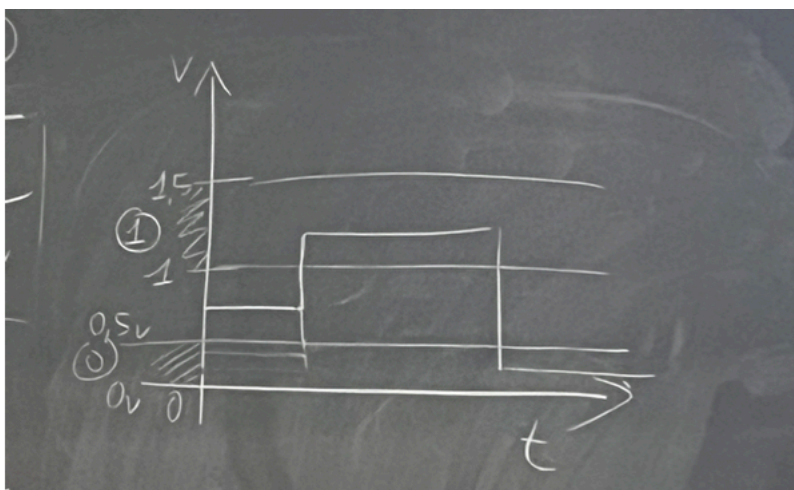
### PORTE LOGICHE: HARDWARE

Quando usiamo una porta logica stiamo facendo riferimento a delle componenti fisiche elettroniche (hardware). In particolare, in un circuito digitale sono presenti solo due valori logici, uno basso ed uno alto, per esempio 0 e 1:

- Il valore 0 potrebbe essere rappresentato da un segnale compreso tra 0 e 0.5 volt;
- Il valore 1 potrebbe essere rappresentato da un segnale compreso tra 1 e 1.5 volt;
- Le tensioni al di fuori di questi intervalli non sono ammesse.

Questi intervalli servono a *gestire la grande variabilità dei valori*, così da poter rappresentarli senza incorrere nei disturbi del segnale.

I valori di alto a basso sono variabili; ad esempio, esistono circuiti con valori da 0 a 5 v.



La base hardware di tutti i calcolatori digitali è costituita da alcuni piccoli dispositivi elettronici, chiamati **porte logiche** (gate), ciascuna delle quali calcola una diversa funzione di questi segnali.

Le porte logiche sono dei componenti che ovviamente sono capaci di processare i contenuti logici ovvero 0 ed 1, ma in realtà cosa vuol dire 0 e cosa vuol dire 1 ?

Se noi nel tempo andiamo ad osservare il livello di tensione presente in un circuito noi possiamo notare che può assumere un valore alto oppure un valore basso, e quindi si potrebbe dire che i valori di tensione tra 0 e 0,5 sono interpretati come 0 mentre valori di tensione alta ( in questo caso 1 ed 1,5) sono interpretati come 1.

Se un valore per errore si ritrova ad un valore di tensione tra 0,5 e 1 deve essere interpretato come 0 o come 1 ?

Questo in realtà è un valore non ammesso, un errore, ciò vuol dire che valori presenti in questo intervallo devono essere eliminati perché non ammessi.

Questi intervalli servono a gestire l'enorme e inevitabile variabilità di impulsi immessi nei circuiti elettronici, i quali sono anche soggetti a disturbi e quindi interferenze dell'impulso stesso, questi range ci permettono quindi di essere meno influenzati da questo tipo di errori.

Ci sono circuiti attraversati da una tensione che oscilla tra 0 e 5v e quindi in questo tipo di circuiti si può decidere che lo 0 sia rappresentato da una tensione tra 0 ed 1 mentre l'1 da una tensione da 4 a 5v ; l'importante è definire un valore minimo e un valore massimo con i rispettivi range in cui si opera.



## TRANSISTOR

Quale è il componente elettronico che si usa per implementare le porte logiche ?

Il transistor è il componente alla base delle porte logiche, esso viene rappresentato con questo disegno tripartito nella rappresentazione dei circuiti logici, a rappresentare le sue componenti: una base, un collettore e un emettitore.

Esso si realizza mediante dei materiali semiconduttori, spesso il silicio il quale mediante una operazione chiamata drogaggio ci permette di realizzare le caratteristiche di questo oggetto.

Il transistor funziona mediante una tensione in ingresso ( $V_{in}$ ) emessa dalla sorgente all'interno della base, viene collegato a un punto con potenziale 0 attraverso l'emettitore (la messa a terra), abbiamo poi una tensione di uscita ( $V_{out}$ ) la quale passa attraverso il collettore.

## PORTA NOT

Il transistor riportato in immagine è effettivamente una porta not che appunto inverte l'uscita e l'ingresso, ovvero quando entra bassa tensione (0) esce alta tensione (1) e viceversa; per questo viene chiamato invertitore o porta not, e come fa a funzionare ? Funzionamento:

- Se la tensione in ingresso ( $V_{in}$ ) è inferiore ad un valore critico, il transistor viene disabilitato e si comporta come una resistenza infinita (circuito aperto, la corrente non passa). La conseguenza è che la tensione in output ( $V_{out}$ ) assume un valore vicino alla tensione applicata esternamente ( $V_{cc}$ ).
- Se  $V_{in}$  supera il valore critico, il transistor si attiva e si comporta come un conduttore ideale, facendo scaricare  $V_{out}$  a terra (per convenzione 0 volt).

Per convenzione  $V_{cc}$  è sempre 1, mentre la corrente a terra è sempre 0.

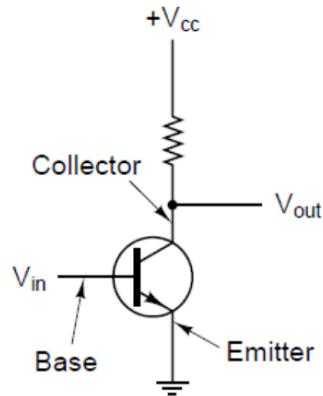
Quando la tensione di ingresso è inferiore ad un certo valore critico allora il transistor viene disattivato e quindi si comporta come un circuito aperto o come una resistenza infinita per cui la  $V_{out}$  avrà un valore molto vicino alla  $V_{cc}$  (corrente generatore) quindi alta tensione.

Quando invece la tensione in ingresso è superiore al valore soglia il transistor si comporta come un conduttore ideale e chiude il circuito per cui la corrente verrà scaricata a terra e la  $V_{out}$  avrà valore 0 volt per cui bassa tensione.

Una volta i computer e i loro circuiti erano basati su i relè, degli interruttori a distanza molto rumorosi che funzionavano mediante onde elettromagnetiche, oggi giorno vengono utilizzati per la domotica (per comandare a distanza le luci).

Con l'avvento dei transistor sono stati sostituiti i relè per via della velocità e la minore rumorosità, ma un transistor non è altro che un interruttore che reagisce a un determinato comando, ovvero la tensione entrante, e in base a come è impostato determinerà una particolare risposta.

Quello precedentemente descritto corrisponde a una porta NOT, che grazie a un fattore soglia è capace di scaricare o meno a terra, e quindi di invertire l'impulso uscente rispetto a quello entrante.

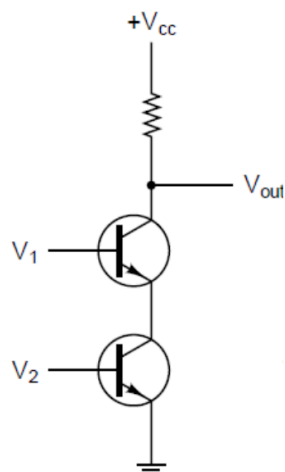


### PORTA NAND (ovvero NOT AND)

A livello circuitale una porta NAND non è nient'altro che due transistor collegati in serie, ciò è possibile innanzitutto perché abbiamo due input nei due transistor, mentre l'output sarà uno solo.

-Se  $V_1$  e  $V_2$  sono alte, allora entrambi i transistor saranno in conduzione e allora  $V_{out}$  sarà in conduzione con la messa a terra, ci si traduce in un valore uscente basso.

-In tutti gli altri casi, quindi se uno degli ingressi è basso, il transistor corrispondente alla tensione bassa si spegnerà, si aprirà quindi l'interruttore, non sarà più possibile scaricare a terra e quindi l'uscita sarà alta.



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

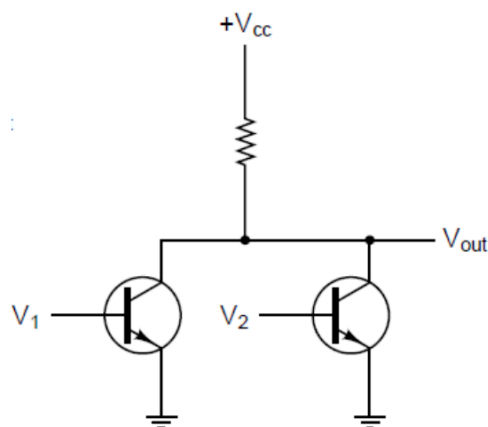
## PORTA NOR (ovvero NOT OR)

Si realizza sempre con due transistor ma questa volta collegati in parallelo.

-Se uno degli ingressi è alto, il transistor corrispondente attiverà la conduzione verso la messa a terra e di conseguenza l'uscita sarà a bassa tensione.

-Se entrambi gli ingressi sono bassi, i transistor rimarranno aperti, per cui sarà impossibile scaricare tensione a terra e di conseguenza l'uscita sarà alta.

L'alta tensione uscente vale solamente se a e b valgono 0, ed infatti solamente quando  $v_1$  e  $v_2$  valgono 0 allora  $V_{out}$  vale 1, in tutti gli altri casi basta che uno dei transistor sia attivo, conduca a terra e noi avremo bassa tensione uscente.



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

## CHIP

Le porte logiche non vengono prodotte o vendute individualmente, ma in unità chiamate circuiti integrati (IC) o chip. Il più piccolo chip è il **SSI** (Small Scale Integrated) che contiene da 1 a 10 porte.

Oltre ai transistor e alle porte di input

ed output (possono essere più di una)

abbiamo la porta di alimentazione ( $V_{cc}$ , comune a tutte le porte logiche) e la porta di messa terra (GND – ground).

Altri tipi di chip sono:

- **LSI** (Large Scale Integrated): da 100 a 10.000 porte;
- **VLSI** (Very Large Scale Integrated): da 10.000 a 100.000 porte;
- **ULSI** (Ultra Large Scale Integrated): maggiore di 100.000 porte.

Dal punto di vista estetico le porte possono presentarsi con due contenitori o packages:

- **DIP** (Dual Inline Packages): i contatti in uscita sono realizzati a due file parallele poste ai lati.

- **BGA (Ball Grid Array):** più complessi con molti contatti in cui i piedini sono su tutta la parte posteriore del supporto.

Da un punto di vista fisico voi non andate in un negozio e comprate lo stesso articolo quello che si fa è che si comprano i cosiddetti circuiti integrati, questi sono dei chip contenenti all'interno un certo numero di porte logiche, il caso più piccolo di circuito integrato è quello dei circuiti SSI (Small Scale Integrated) cioè integrati su piccola scala che contengono da 1 a 10 porte logiche (fa vedere una foto di circuito a 4 porte and).

Il circuito non avrà solamente due ingressi e un'uscita ma avrà una tensione di alimentazione che sarebbe la VCC che abbiamo detto prima e avrà anche la messa a terra (fa vedere un'altra foto), quindi se andiamo a guardare c'è un piedino, la vcc dove si collega l'alimentazione e da lì l'alimentazione a tutti, e che è collegata di fatto ai circuiti.

Quindi una porta logica con più porte in ingresso e una in uscita ha comunque anche una porta di alimentazione e una porta di messa a terra ma la porta di alimentazione vcc e la messa a terra sono in comune fra tutte le porte logiche che si trovano nello stesso circuito integrato, ecco perché in questo circuito integrato vcc ce ne sta una non 4 come le porte presenti (sempre riferito al caso in foto).

Quindi in un circuito integrato ci possono essere più porte di ingresso e uscita ma una sola di alimentazione e una di messa a terra perché valgono per tutte.

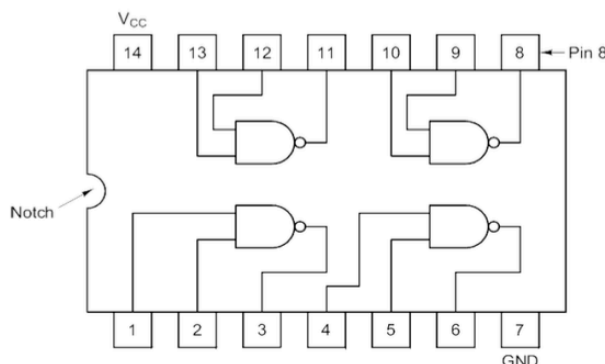
I circuiti di questo tipo ssi ma ce ne sono altri fino ad arrivare all'ultimo livello vlsi in cui si riescono anche più di 100000 porte in un unico circuito.

Dal punto di vista estetico i circuiti integrati possono presentarsi in due forme dette anche package, quello dei cosiddetti DIP (Dual Inline Package) in cui ci sono linee di piedini una a sinistra e una a destra (fa vedere una foto) bisogna stare attenti quando si inserisce nella sua casella, l'inserimento corretto si capisce attraverso il notch.

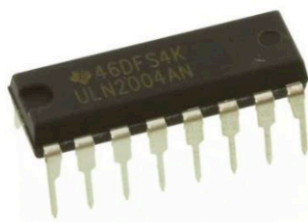
Questo è il formato semplice ma esiste anche il formato a matrice, in cui i piedini non sono su due lati, ma sono su tutta la superficie dove si va ad incastrare il chip; questo si chiama Ball Grid Array e si usa soprattutto per i microchip che si trova nel computer che hanno un livello di integrazione molto elevato.

Se voi avete solamente questo piccolo numero di porte, il numero di gate è più piccolo; qui il numero di porte è maggiore e quindi è maggiore il numero di chip che ci sono all'interno.

- **SSI (Small Scale Integrated):** da 1 a 10 porte
- **MSI (Medium Scale Integrated):** da 10 a 100 porte
- **LSI (Large Scale Integrated):** da 100 a 10.000 porte
- **VLSI (Very Large Scale Integrated):** 10.000 a 100.000 porte
- **ULSI (Ultra Large Scale Integrated):** > 100.000 porte



*Un semplice circuito SSI con 4 porte NAND*



*Chip con package:  
DIP (sinistra) e  
BGA (destra)*



# RETI COMBINATORIE

In quelli di base l'output dipende sostanzialmente dal tipo di input.

Le reti si dividono in **combinatorie semplici** (dipendono solo dai valori in input) e **sequenziali** (si formano a partire dalle combinatorie semplici); una rete combinatoria è un sistema che ha una serie (n) di variabili in ingresso e in uscita.

Queste variabili di uscita (variabili dipendenti) *dipendono esclusivamente dal valore istantaneo delle variabili d'ingresso* (indipendenti). Nelle reti combinatorie non può quindi avvenire che gli stessi valori in input diano valori di output diversi.

Una rete combinatoria è quindi un oggetto che ha in ingresso una serie di variabili  $x_1, x_2 \dots x_n$  e una serie di variabili in uscita  $z_1, z_2, \dots z_m$ .

La variabile di uscita non è necessariamente una ma possono essere m, quindi in generale ci saranno n variabili in entrata ed m variabili in uscita, queste variabili in uscita assumono un valore che dipende solo ed esclusivamente dal valore attuale delle variabili d'ingresso, quindi noi parliamo di rete logica combinatoria quando il valore di uscita dipende da valori istantanei delle variabili di ingresso, in altri termini non può avvenire che in una rete combinatoria che gli stessi ingressi in momenti diversi producano output diversi.

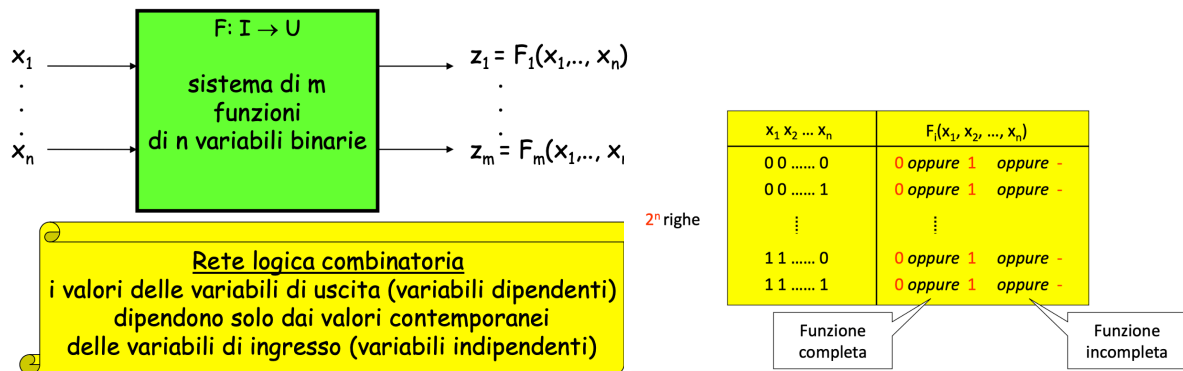
I circuiti logici combinatori si descrivono molto semplicemente con le tavole di verità.

Quindi, dato un circuito combinatorio si passa alla numerazione di tutte le possibili combinazioni o configurazioni ( $2^n$ ). Per ciascuna uscita si avrà un solo valore tra 1, 0 e - (si legge "indifferente").

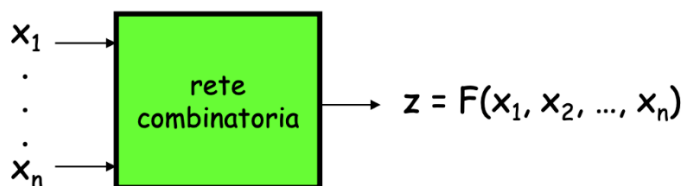
- oppure x = valore indeterminato, generalmente non di interesse (don't care).

Basta un solo valore di don't care per rendere la funzione incompleta.

I circuiti logici combinatori si descrivono semplicemente con la tecnica delle tabelle di verità, quindi data una funzione/circuito combinatorio elenchiamo tutti i possibili valori di input sulle colonne di sinistra, che come al solito il numero corrisponde a  $2^n$ , quindi se abbiamo x variabili le possibili combinazioni o configurazioni sono  $2^x$  quindi 2 variabili 4 righe, 3 variabili 8 righe, 4 variabili 16 righe ecc; a destra ci sono una serie di colonne, una colonna per ogni variabile in uscita, quindi c'è scritto  $f(i) (x_1, x_2, \dots, x_n)$ , quindi per ogni variabile di uscita abbiamo una colonna, avremo la colonna della variabile x se ne abbiamo una sola, se abbiamo x ed y avremo due colonne ecc ..., per ogni riga della colonna abbiamo 0 o 1 oppure questo simboletto - avvolto al posto del trattino c'è la x, che vuol dire **Valore indeterminato (DON'T CARE)**, cioè data una funzione e dato un ingresso di valore 0 oppure 1 oppure qualsiasi valore, questo concetto ci interessa perché ci sono funzioni che potrebbero definire l'output in modo determinato solamente per un sotto-insieme degli input, mentre per quegli altri input potrebbe non dire qual è l'output semplicemente non gli interessa e in quel caso si usa il simbolo che è il trattino o la x. Quando sono completamente specificati i valori di output cioè 0 1 la funzione è completa basta che sia presente un unico valore di don't care (- o x) che la funzione non è completata.

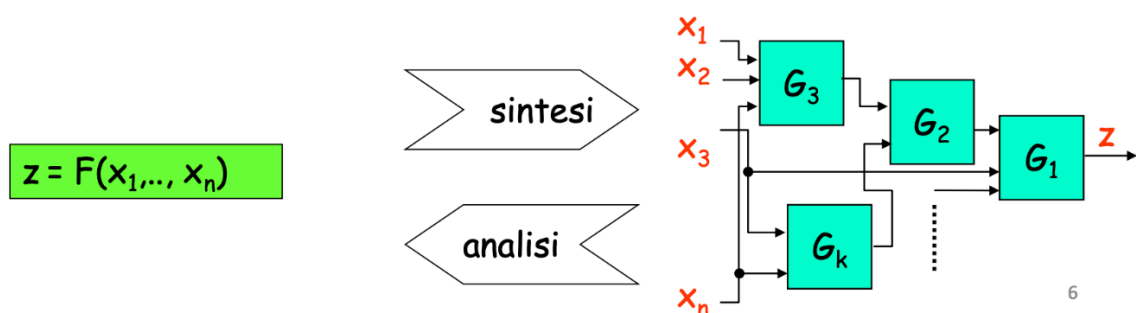


reti combinatorie non può avvenire che gli stessi ingressi forniti in istanti i diano luogo ad uscite diverse



Comportamento

Struttura



Nel caso di reti combinatorie, è possibile passare dalla descrizione del comportamento alla realizzazione della struttura(e viceversa) mediante *analisi e sintesi*.

Dato il circuito si può analizzare la funzione e viceversa, data la funzione possiamo sintetizzare la struttura.

Il problema più importante è la **sintesi**; si può definire come il problema che si occupa di definire, a partire da una funzione assegnata (tabella di verità), una serie di equazioni equivalenti alla ricerca del circuito "migliore" e, infine, passa allo schema logico circuitale.

Quando abbiamo un circuito composto da una serie di input e con una funzione di output, data questa rete combinatoria noi possiamo sostanzialmente valutare da un lato il comportamento della funzione stessa, tramite tabella di verità e dire dato l'input quello è l'output, dall'altro definire quella che è la struttura fisica della rete stessa cioè definire l'insieme delle porte logiche che ci sono.

Dato il comportamento cioè data la funzione o tabelle di verità si giunge alla struttura con un'operazione che è quella di sintesi cioè noi sintetizziamo il circuito a partire dal comportamento quindi dalla tabella di verità, ma esiste anche il percorso inverso data la struttura del circuito noi possiamo dedurre qual è la funzione, cioè possiamo fare un'analisi.

Qual è il modo con il quale data la tabella ricostruiamo la funzione?

Applichiamo ad es qui sui 4 input tutti i possibili valori di 0 e 1 andiamo a misurare in uscita l'output che sarà o 0 o 1 e costruiamo una tavola di verità, quindi si può fare sia una sintesi (dalla tabella alla struttura) sia un'analisi (dalla struttura alla tabella).

Il problema più importante è quello della sintesi perché andare a misurare l'output di un circuito è un'operazione banale perché il circuito c'è già.

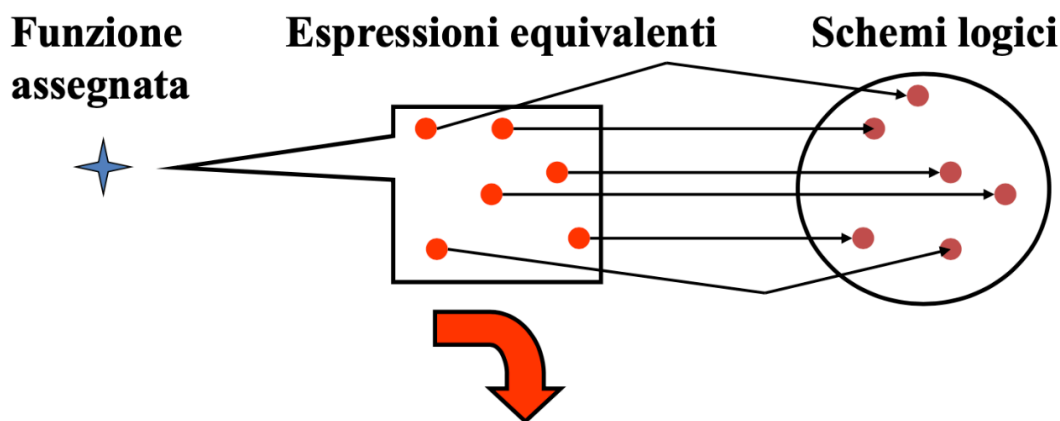
Il problema della sintesi si può definire come il problema che si occupa di prendere una funzione assegnata descritta ad esempio con una tavola di verità, da questa funzione assegnata andiamo a definire delle eventuali espressioni equivalenti che siano più semplici/efficienti, ed infine andiamo a realizzare uno schema logico cioè l'insieme delle porte che implementano una funzione assegnata.

La sintesi quindi consiste nel passare da una funzione agli schemi equivalenti per poi arrivare allo schema logico finale.

L'aspetto importante della sintesi è che ci sono infiniti schemi logici equivalenti della funzione logica, quindi dobbiamo cercare di scegliere l'espressione logica migliore in termini di velocità o che minimizzi la complessità.

I due temi complessità e velocità sono quindi fondamentali e dipendono in grande misura dalla forma del circuito che dobbiamo progettare; gli indicatori fondamentali di questi due parametri sono:

- il numero di gate cioè il numero di componenti che usiamo,
  - il numero di connessioni cioè i fili che collegano questi gate fra loro
  - il numero maggiore di gate in cascata che sono collegati tra loro
- (cioè sostanzialmente possiamo collegare i gate in una sequenza potremmo avere and seguito da un or seguito da un and e poi da un or e fare un percorso di lunghezza 4, da un'altra parte c'è un percorso di lunghezza 2, chiaramente l'output viene determinato dal percorso più lungo perché comunque i circuiti devono passare dai diversi percorsi per arrivare alla fine, quindi il numero di elementi in cascata determina il percorso più lungo che attraversiamo quindi in qualche modo indica il tempo che ci vuole ad attraversare l'intero circuito minore è questo valore maggiore è la velocità).



**SINTESI: abbiamo infiniti schemi che risolvono il nostro problema, vogliamo trovare quelli che sono ottimali secondo i parametri di**

**Massima velocità**

**Minima complessità**

La complessità è una funzione crescente rispetto al numero di gate e al numero di connessioni, cioè maggiore è il numero di gate e maggiore è il numero di fili maggiore è la complessità, d'altra parte la velocità è una funzione decrescente cioè è una funzione inversamente proporzionale al percorso più lungo.

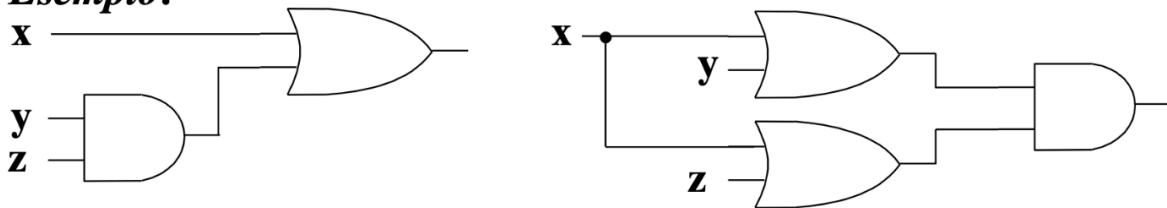


**Indicatori :**

- $N_{gate}$  = numero di gate
- $N_{conn}$  = numero di connessioni
- $N_{casc}$  = numero di gate disposti in cascata sul più lungo percorso di elaborazione

**Complessità**  $\Rightarrow$  funzione crescente di  $N_{gate}$ ,  $N_{conn}$   
**Velocità di elaborazione**  $\Rightarrow$  funzione decrescente di  $N_{case}$

***Esempio:***



- **Le due reti sono equivalenti.**
- **Hanno la stessa velocità di elaborazione.**
- **La rete di sinistra è meno complessa.**

Es. due circuiti equivalenti in entrambi i casi ricevono tre variabili d'ingresso  $x$  e  $y$  e  $z$  dando un'unica variabile d'uscita ma sono strutturate in modo diverso, la prima usa solamente due porte una and e una or, l'altra usa due or e una and; in termini di numero di porte una è meno complessa dell'altra in termini di velocità sono equivalenti perché il percorso più lungo è uguale ma è preferibile quella con un numero minore di porte.

Per vedere che i due circuiti sono equivalenti si deve ovviamente fare la tabella di verità.

Esistono circuiti non combinatori sono tutti quelli che non seguono la regola sopra riportata, ovvero i circuiti in cui lo stesso input produce nel tempo output diversi, quindi tutti quei circuiti sequenziali in cui il circuito conserva all'interno uno stato per cui l'output non dipende solo dall'input ma anche dallo stato attuale del circuito.

# APPROFONDIMENTO

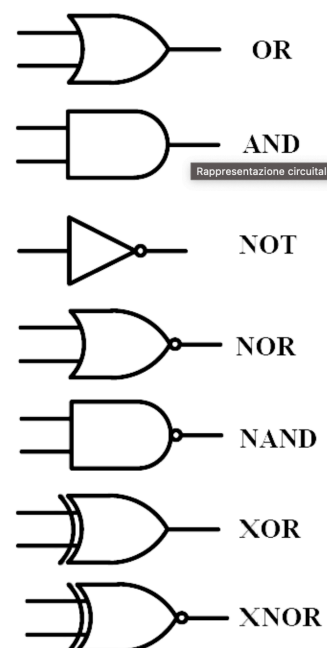
## CIRCUITO PORTA

Si chiama circuito porta il circuito elementare che possiede  $n$  ingressi ed un'unica uscita, il cui valore logico di uscita è 1 oppure 0 a seconda della descrizione logica degli operatori logici: AND, OR, NOT, NOR, NAND, e così via.

### Differenze chiave tra porta AND e OR

1. La porta AND fornisce come output il prodotto di due valori binari. Tuttavia, la porta OR fornisce la somma dei due ingressi binari applicati come uscita.
2. La porta AND segue la regola della congiunzione logica. Rispetto alla porta OR segue la disgiunzione logica.
3. L'espressione booleana della porta AND è rappresentata come  $AB$ . Mentre l'espressione booleana della porta OR è data come  $A+B$ , dove  $A$  e  $B$  sono considerati i due ingressi applicati.
4. Nella porta AND, l'uscita logica alta si ottiene solo quando entrambi gli ingressi applicati sono alti. Mentre ci si trova nella porta OR, si ottiene un livello logico alto se qualcuno o entrambi gli ingressi sono alti.

Quindi, possiamo concludere che in campo elettronico ed [informatico](#) sia AND che OR sono le due porte logiche di base progettate per eseguire operazioni booleane. Tuttavia, la differenza cruciale sta nell'operazione da loro eseguita.



## LE OPERAZIONI CON LE VARIABILI BOOLEANE

SI DIVIDONO IN OPERAZIONI A:

### 1 operando

L'unica operazione possibile con i valori booleani è la **negazione** (detto anche **complemento**) che molto spesso è indicata con *NOT*,  $!$  o  $\sim$ .

Si definisce:

- **not falso** = vero
- **not vero** = falso

### 2 operandi

Esistono varie operazioni, le principali sono: **EQV**, **AND**, **OR**, **XOR**. Tutte queste funzioni hanno 2 operandi (booleani) in ingresso e producono un output booleano.

Nella maggior parte dei linguaggi vengono indicati anche con i simboli:

EQV	"=="
-----	------

AND	“&&”
OR	“  ”
XOR	“^”

## SEMPLIFICAZIONE DEI CIRCUITI

Un'espressione booleana in forma normale disgiuntiva può essere semplificata in forma minimale. Per farlo basta applicare le [leggi dell'algebra booleana](#).

Date tre variabili booleane x,y,z valgono le seguenti leggi dell'[algebra booleana](#).

$x = \overline{\overline{x}}$	doppio complemento
$x + y = y + x$ $x \cdot y = y \cdot x$	commutatività
$(x + y) + z = x + (y + z)$ $(xy)z = x(yz)$	associatività
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	distributività
$x + 0 = x$ $x \cdot 1 = x$	elemento neutro
$\overline{x + y} = \overline{x} \overline{y}$ $\overline{xy} = \overline{x} + \overline{y}$	leggi di De Morgan
$x + x = x$ $x \cdot x = x$	idempotenza
$x + \overline{x} = 1$ $x \cdot \overline{x} = 0$	inversi
$x + 1 = 1$ $x \cdot 0 = 0$	dominanza
$x + xy = x$ $x(x + y) = x$	assorbimento

WWW.ANDREAMININI.ORG

Molte leggi precedenti sono leggi duali.

Un'espressione in forma minimale esprime la stessa funzione booleana, ha la stessa tavola di verità, ma è composta da un minor numero di clausole e letterali.

### Un esempio pratico

Queste due espressioni rappresentano la stessa funzione booleana f.

$xyz + xy$ ,

$xy$

In entrambi i casi la tavola di verità della funzione è

x	y	f
0	0	0
0	1	0
1	0	0
1	1	1

**Nota.** La prima espressione ha un letterale in più (z) ma è ininfluente. Se x e y sono uguali a 1, la funzione è uguale a 1 indipendentemente dal valore del letterale z.

x	y	z	xyz + xy	x	y	xy
0	0	0	0	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	0	1	1	1
1	0	0	0			
1	0	1	0			
1	1	0	1			
1	1	1	1			

WWW.ANDREAMININI.ORG

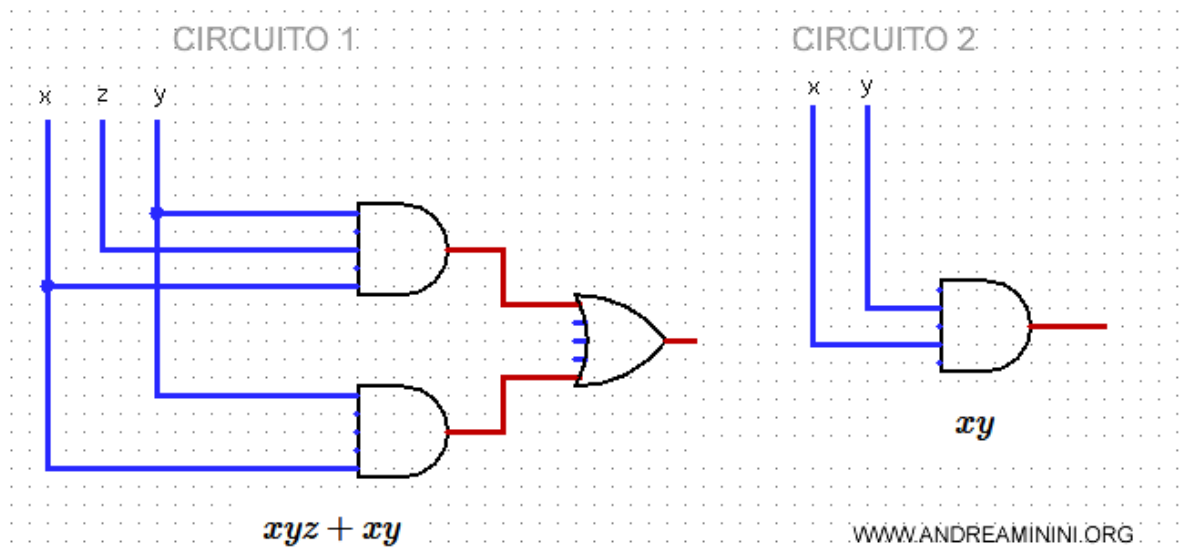
Quindi, dovendo scegliere un'espressione booleana per rappresentare la funzione è meglio prendere la seconda.

~~$xyz + xy$~~   
 $xy$

E' più corta, ci sono meno operazioni e meno letterali da considerare.

**La seconda espressione è una forma minimale della prima.**

**Nota.** La forma minimale è molto utile per semplificare un [circuito logico combinatorio](#) perché riduce al minimo il numero delle porte ( operazioni ) e dei collegamenti ( letterali ). Nell'esempio precedente il secondo circuito è una semplice porta AND.



Per semplificare l'espressione devo prima trasformarla in [forma normale disgiuntiva \(DNF\)](#).

### Esempio

$Xy(zv + v),$

$xyzv + xyz$

**Nota.** Un'espressione in forma normale disgiuntiva è composta da n prodotti di letterali sommati tra loro. I prodotti sono le clausole dell'espressione booleana.

**Le principali regole di semplificazione sono due.**

- **Regola 1.** Se due o più clausole contengono lo stesso prodotto al loro interno, elimino le clausole più lunghe. Ad esempio.
- **Regola 2.** Se due prodotti diversi differiscono soltanto per un letterale, elimino il letterale differente e prendo soltanto i letterali in comune.

Queste regole mi permettono di semplificare l'espressione booleana.

Ci sono comunque degli algoritmi di riduzione che standardizzano il processo di semplificazione dell'espressione booleana.

Uno dei più noti è l'algoritmo o [mappe di Karnaugh](#).

## LA FORMA NORMALE CONGIUNTIVA E DISGIUNTIVA

La forma normale disgiuntiva (DNF) e congiuntiva (CNF) sono due modi diversi per scrivere un'espressione logica.

Qual è la differenza? Nella forma normale disgiuntiva (DNF) c'è una disgiunzione di clausole e ogni clausola è composta da una congiunzione di letterali. Nella forma normale congiuntiva (CNF) accade esattamente l'inverso.

$xy + xyz + yz$	DNF
$(x+y) (x+y+z) (y+z)$	CNF

WWW.ANDREAMININI.ORG

Una stessa espressione può essere espressa in forma DNF o CNF.

### **La forma normale disgiuntiva (DNF)**

La forma normale disgiuntiva è anche detta **DNF** dal termine inglese **Disjunctive Normal Form**.

Nei libri di testo italiani è anche detta forma normale disgiunta e indicata con la sigla **FND**

Un'espressione logica è composta da una o più clausole.

$$C1 + C2 + \dots + Cn$$

Ogni **clausola** è composta da una congiunzione di letterali.

$$Ci = xyz$$

**Cos'è una congiunzione logica?** Una congiunzione è un prodotto di letterali.

Infine, le clausole sono sommate tra loro formando l'espressione logica in forma normale disgiuntiva.

### **La forma normale congiuntiva (CNF)**

La forma normale congiuntiva è anche detta **CNF** dal termine inglese **Conjunctive Normal Form**.

Spesso nei libri italiani la trovo anche con il nome di forma normale congiunta e indicata con la sigla **FNC**.

Un'espressione logica è composta da una più clausole.

$$C1 + C2 + \dots + Cn$$

Ogni clausola è composta da una disgiunzione di letterali.

$$C_i = x + y + z$$

**Cos'è una disgiunzione logica?** Una disgiunzione è una somma di letterali. Infine, le clausole sono moltiplicate tra loro formando l'espressione logica in forma normale congiuntiva.

### **Il caso di uguaglianza tra DNF e CNF**

In un caso particolare la forma CNF eguaglia la forma DNF.

E' il caso di n clausole composte da un solo letterale.

#### **Esempio**

Questa espressione è composta tra letterali.

$$X \vee Y \vee Z$$

Posso interpretarla come una singola clausola in forma normale congiuntiva.

D'altra parte, posso vederla anche come tre clausole, ognuna composta da un singolo letterale, in forma normale disgiuntiva.

**Nota.** Lo stesso discorso posso farlo con la seguente clausola.

$$X \wedge Y \wedge Z$$

## **LA MAPPA DI KARNAUGH**

### **A cosa serve la mappa di Karnaugh**

Le mappe di Karnaugh sono un metodo per semplificare un'espressione booleana o, è la stessa cosa, per ridurre il numero delle porte logiche in un circuito combinatorio.

- Come trovare la mappa di Karnaugh
- Come migliorare la mappa di Karnaugh

### **Come trovare la mappa di Karnaugh**

Il metodo funziona soltanto se l'espressione è in forma normale disgiuntiva ( DNF ).

**Nota.** Per ogni funzione logica esiste sempre un'espressione booleana in forma normale disgiuntiva ed è unica.

Inoltre, le clausole devono essere composte dagli stessi letterali.

#### **Esempio**

Questa espressione booleana è composta da tre clausole.

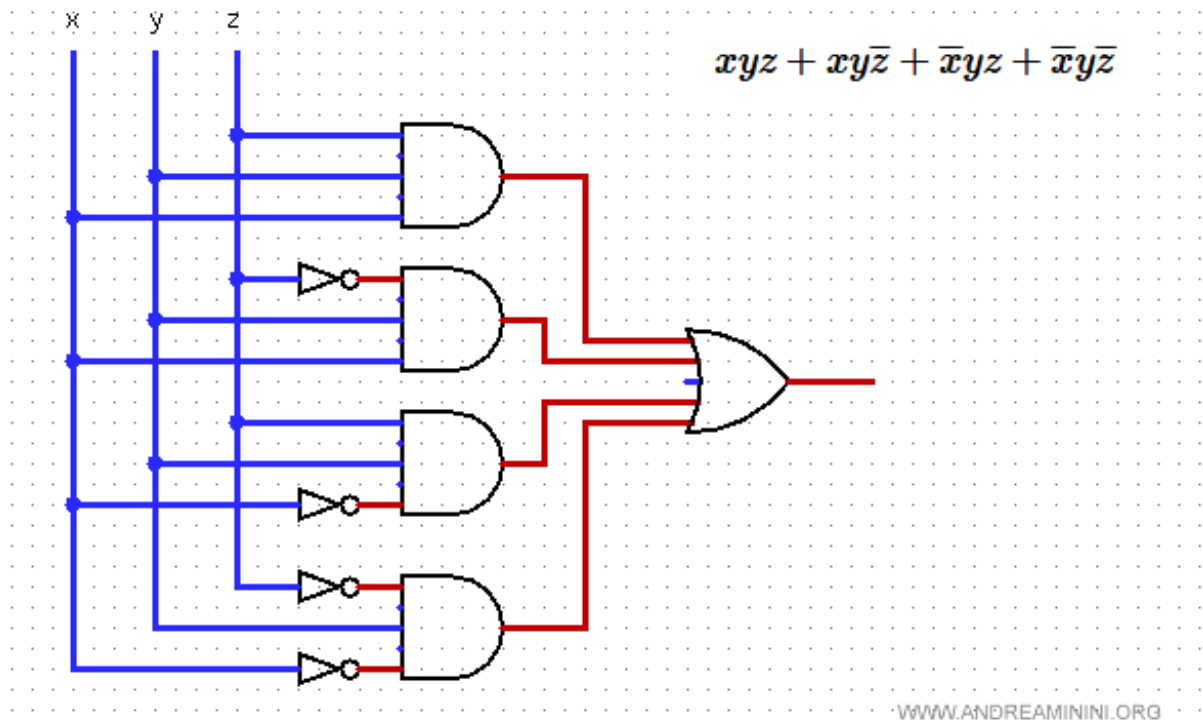
$$xyz + xy(!z) + (!x)y$$

La terza clausola non ha il letterale z.

Per risolvere il problema, la trasformo in una somma di clausole con la z attiva e negata.



**Nota.** Questa espressione booleana equivale a un circuito logico con quattro porte AND, una porta OR e quattro porte NOT. Complessivamente, il circuito è composto da 9 porte.



La tavola di verità dell'espressione è

$x$	$y$	$z$	$xyz + xy\bar{z} + \bar{x}yz + \bar{x}\bar{y}\bar{z}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

WWW.ANDREAMININI.ORG

### 1) Costruire la mappa

Scelgo due gruppi di letterali dell'espressione. Ad esempio  $xy$  e  $z$ .

Poi costruisco una tabella mettendo le xy sulle colonne e la z sulle righe.

z \ xy				
	00	01	10	11
0				
1				

WWW.ANDREAMININI.ORG

Nelle righe e nelle colonne indico le combinazioni di valori (0,1) che i letterali x e xy possono assumere.

## **2] Individuare le celle della mappa a cui corrispondono le clausole dell'espressione**

A questo punto, verifico quali celle si collocano le clausole dell'espressione booleana.

### **Prima clausola**


La prima clausola dell'espressione è

**xyz**

Equivale a  $xy=11 * z=1$ .

Questa clausola corrisponde alla cella in cui si incrociano l'ultima colonna ( $xy=11$ ) e la seconda riga ( $z=1$ ) della mappa.

z \ xy				
	00	01	10	11
0				
1				+



WWW.ANDREAMININI.ORG

### **Seconda clausola**

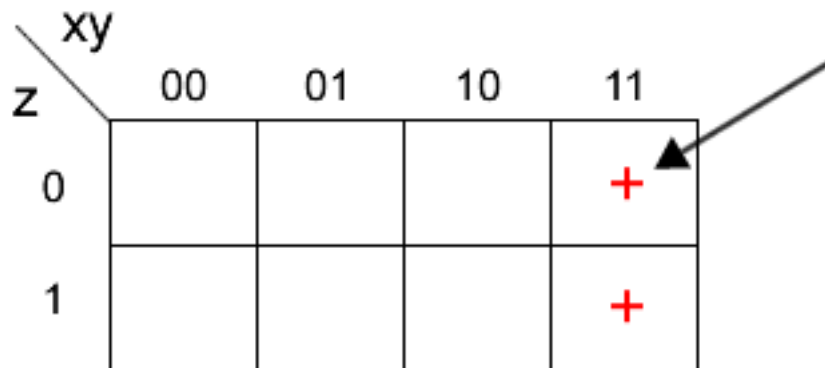
La seconda clausola dell'espressione è

**xy(!z)**

Equivale a  $xy = 11$  e  $z=0$ .

Questa clausola corrisponde alla cella in cui si incrociano l'ultima colonna ( $xy=11$ ) e la prima riga ( $z=0$ ) della mappa.

xy		00	01	10	11
z	0				+
	1				+



WWW.ANDREAMININI.ORG

### Terza clausola

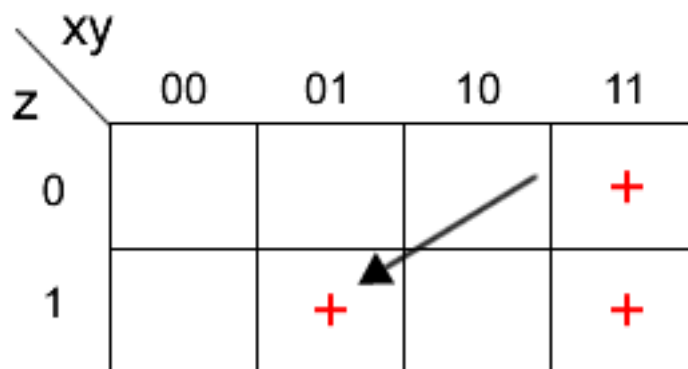
La terza clausola dell'espressione è

$(!x)yz$

Equivale a  $xy = 01$  e  $z=1$ .

Questa clausola corrisponde alla cella in cui si incrociano la seconda colonna ( $xy=01$ ) e la seconda riga ( $z=1$ ) della mappa.

xy		00	01	10	11
z	0				+
	1		+		+



WWW.ANDREAMININI.ORG

### Quarta clausola

La quarta clausola dell'espressione è

$(!x)y(!z)$

Equivale a  $xy = 01$  e  $z=0$ .

Questa clausola corrisponde alla cella in cui si incrociano la seconda colonna ( $xy=01$ ) e la prima riga ( $z=0$ ) della mappa.

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

WWW.ANDREAMININI.ORG

### **3) Raggruppare le celle confinanti in gruppi di 2, 4 o 8 elementi**

Una volta costruita la mappa di Karnaugh, posso raggruppare i simboli per multipli di 2.

Ogni gruppo deve essere composto da 2,4,8 simboli vicini tra loro.

I gruppi possono anche sovrapporsi parzialmente.

Nota. Le pareti della mappa di Karnaugh confinano con la parete opposta.

In questo caso, nella mappa ci sono due gruppi da 2.

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

WWW.ANDREAMININI.ORG

### **3) Trasformare i gruppi in congiunzioni di letterali**

Prendo ciascun gruppo e lo trasformo in una congiunzione, eliminando i letterali che assumono contemporaneamente i valori 0 e 1.

Primo gruppo

Nel primo gruppo xy assume il valore 01.

Elimino, invece, il letterale z perché ha sia il valore 0 che 1.

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

$\bar{x}y$

WWW.ANDREAMININI.ORG

Il risultato finale è

$(\bar{x})y$

Secondo gruppo

Nel primo gruppo  $xy$  assume il valore 11.

Elimino, invece, il letterale  $z$  perché ha sia il valore 0 che 1.

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

$xy$

WWW.ANDREAMININI.ORG

Il risultato finale è

$xy$

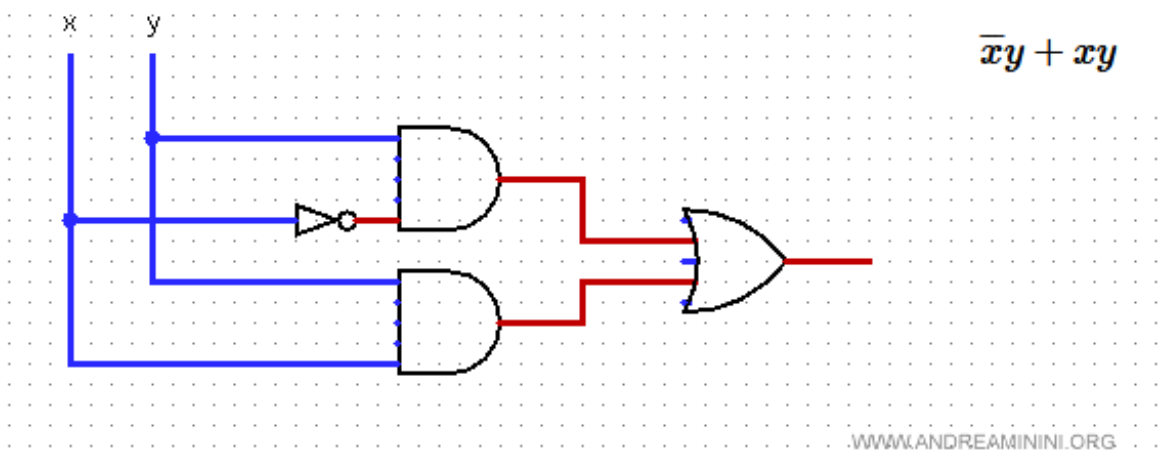
#### 4] Sommare le congiunzioni di letterali

Sommo tra loro le congiunzioni appena trovate e ottengo l'espressione booleana in forma ridotta della funzione booleana.

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

Il nuovo circuito logico è composto soltanto da due porte AND, una porta OR e una porta NOT.

Complessivamente, il circuito ha 4 porte.



La tavola di verità dell'espressione è sempre la stessa.

Anche se non c'è più la lettera z, il valore 1 è abbinato alle stesse combinazioni x, y della precedente.

$x$	$y$	$z$	$xyz + xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z}$		$x$	$y$	$\overline{xy} + xy$
0	0	0	0		0	0	0
0	0	1	0		0	1	1
0	1	0	1		1	0	0
0	1	1	1		1	1	1
1	0	0	0				
1	0	1	0				
1	1	0	1				
1	1	1	1				

WWW.ANDREAMININI.ORG

Nota. L'espressione è sicuramente migliore rispetto all'espressione iniziale. Tuttavia, non è ancora ottimale. Come si può vedere, nelle due clausole il letterale x appare sia in forma diretta che negata.

**(!x)y + xy**

Una semplice regola di semplificazione mi consente di eliminare la x in entrambe le clausole. L'espressione si riduce a

**y + y**

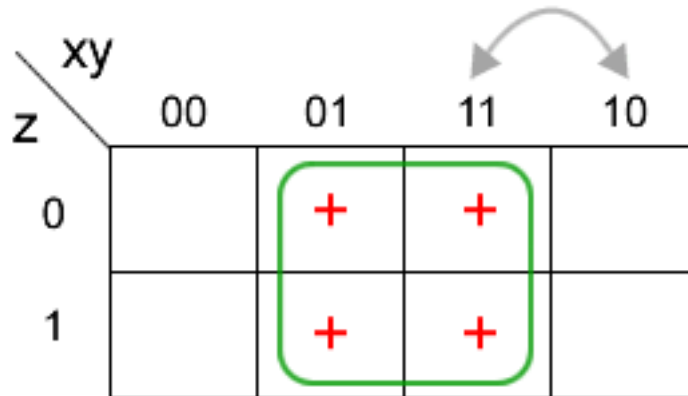
Un'altra legge dell'algebra booleana mi permette di ridurre ulteriormente l'espressione in: **y**

Perché la mappa di Karnaugh non l'ha trovata? Perché il risultato dipende anche dalla scelta dei gruppi di letterali e dall'ordine delle colonne nella mappa.

Come migliorare la mappa di Karnaugh

La riduzione di un'espressione booleana tramite la mappa di Karnaugh dipende dalla scelta dei gruppi e dall'ordine delle colonne.

Ad esempio, riprendo la precedente mappa scambiando la posizione della quarta e della terza colonna.

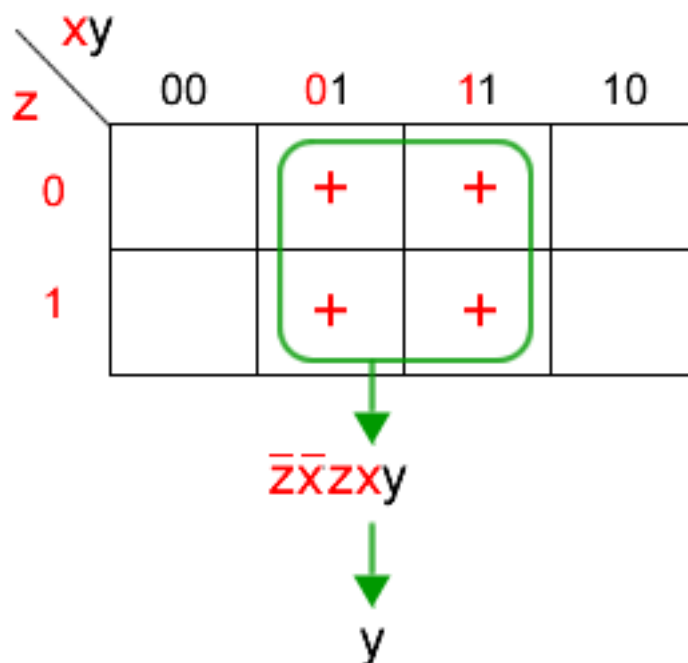


WWW.ANDREAMININI.ORG

In questo modo creare un gruppo di quattro celle.

Il nuovo raggruppamento mi permette di semplificare ulteriormente il risultato finale.

Sia la z che la x appaiono con il segno zero e con il segno meno. Quindi, posso eliminarle.



WWW.ANDREAMININI.ORG

L'espressione finale è.

**y**

Ovviamente l'espressione non può essere ulteriormente ridotta.



## **TRANSISTOR**

Il transistor è composto da un materiale semiconduttore al quale sono applicati tre terminali che lo collegano al circuito esterno.

Il funzionamento del transistor è basato sulla giunzione p-n, scoperta casualmente da Russell Ohl il 23 febbraio 1939.

Le principali funzioni che gli vengono affidate all'interno di un circuito elettronico sono:

- l'amplificazione di un segnale in entrata;
- il funzionamento da interruttore (switcher).

I transistor sono impiegati principalmente come amplificatori di segnali elettrici o come interruttori elettronici comandati e hanno quasi ovunque sostituito i tubi termoionici.

I transistor sono l'elemento base dei chip (o circuiti integrati), al punto che la loro presenza numerica all'interno di questi ultimi è dichiarata ufficialmente come ordine di grandezza: ad esempio, i circuiti integrati semplici ne hanno da qualche decina a qualche centinaia, quelli complessi ne hanno qualche migliaio. L'Apple A13 conta 8,5 miliardi di transistor.

Esistono principalmente due diversi tipi di transistor: il transistor a giunzione bipolare ed il transistor ad effetto di campo, ed è possibile miniaturizzare i dispositivi di entrambe le categorie all'interno di circuiti integrati, il che li rende componenti fondamentali nell'ambito della microelettronica.

Il transistor a giunzione bipolare, anche chiamato con l'acronimo BJT, è un tipo di transistor largamente usato nel campo dell'elettronica analogica principalmente come amplificatore ed interruttore.

Si tratta di tre strati di materiale semiconduttore drogato, solitamente il silicio, in cui lo strato centrale ha drogaggio opposto agli altri due, in modo da formare una doppia giunzione p-n, cioè o una giunzione p-n-p o una n-p-n. Ad ogni strato è associato un terminale: quello centrale prende il nome di base, quelli esterni sono detti collettore ed emettitore. Il principio di funzionamento del BJT si fonda sulla possibilità di controllare la conduttività elettrica del dispositivo, e quindi la corrente elettrica che lo attraversa, mediante l'applicazione di una tensione tra i suoi terminali. Tale dispositivo coinvolge sia i portatori di carica maggioritari sia quelli minoritari, e pertanto questo tipo di transistor è detto bipolare.

Un sistema costituito da un singolo transistor può essere rappresentato come un generico quadripolo avente due terminali di ingresso e due di uscita. I tre terminali del transistor saranno uno il terminale di ingresso, un altro quello di uscita ed il terzo in comune, connesso cioè sia all'ingresso sia all'uscita. A seconda di quale sia il terminale comune il transistor può assumere le seguenti configurazioni: a base comune, a collettore comune o a emettitore comune.

Insieme al transistor ad effetto di campo, il BJT è il transistor più diffuso in elettronica, è in grado di offrire una maggiore corrente in uscita rispetto al FET, mentre ha lo svantaggio di non avere il terminale di controllo isolato.

Il transistor ad effetto di campo, anche chiamato con l'acronimo FET, è un tipo di transistor largamente usato nel campo dell'elettronica digitale e diffusa, in maniera minore, anche nell'elettronica analogica.

Si tratta di un substrato di materiale semiconduttore, solitamente il silicio, al quale sono applicati quattro terminali: gate (porta), source (sorgente), drain (pozzo) e bulk (substrato); quest'ultimo, se presente, è generalmente connesso al source. Il principio di funzionamento del transistor a effetto di campo si fonda sulla possibilità di controllare la conduttività elettrica del dispositivo, e quindi la corrente elettrica che lo attraversa, mediante la formazione di un campo elettrico al suo interno. Il processo di conduzione coinvolge solo i portatori di carica maggioritari, pertanto questo tipo di transistor è detto unipolare.

Insieme al transistor a giunzione bipolare, il FET è il transistor più diffuso in elettronica: a differenza del BJT esso presenta il vantaggio di avere il terminale gate di controllo isolato dal resto del circuito, nel quale non passa alcuna corrente; mentre ha lo svantaggio di non essere in grado di offrire molta corrente in uscita. In genere i circuiti con transistor FET hanno infatti un'alta impedenza di uscita, erogando quindi correnti molto deboli.

I transistor elettrochimici organici (OECTs) sono una delle applicazioni più proficue del PEDOT:PSS nella sua forma di film sottile e la loro accessibilità è dovuta alla facilità di produzione degli stessi e alla bassa tensione che è necessario applicare per il loro funzionamento.

La struttura dei transistor elettrochimici è composta da due parti fondamentali: canale e gate. Il canale, un film sottile di polimero semiconduttore (in questo caso PEDOT:PSS) depositato su un substrato isolante, presenta ai capi i due contatti di source e drain, esattamente come nei transistor a effetto di campo, con i quali condividono anche le curve caratteristiche. Il gate invece può presentarsi in due forme. La prima è l'elettrodo metallico che, immerso nella soluzione elettrolitica in cui si trova il canale, le fornisce tensione di gate; l'elettrodo e il canale in PEDOT non entrano in contatto tra loro. La seconda forma del gate è il film sottile di PEDOT:PSS che, depositato sullo stesso substrato del canale, è isolato rispetto ad esso in assenza di soluzione elettrolitica.

Con l'evolversi della tecnologia sono stati creati diversi altri tipi di transistor, adatti a usi particolari. Tra i più diffusi vi sono il transistor unigiunzione, un generatore di impulsi che non può amplificare né commutare, e gli Insulated Gate Bipolar Transistor, dispositivi ibridi fra i transistor bipolari e i MOSFET, adatti a gestire correnti elevate. Vi sono inoltre transistor sviluppati per applicazioni di ricerca, capaci di ottenere prestazioni quali sopportare elevate correnti o elevate frequenze di funzionamento: nel 2006 un transistor al silicio-germanio ha raggiunto in laboratorio la frequenza di commutazione di 500 GHz.

## **RETI COMBINATORIE**

Un circuito combinatorio (o anche rete combinatoria) è un circuito il cui funzionamento riguarda solo la relazione ingresso-uscita. Tale relazione è descritta da una funzione logica.

I circuiti combinatori in particolare sono quelli in cui gli ingressi e le uscite possono assumere solo due stati corrispondenti ai livelli alto o basso, e le uscite sono funzione unicamente degli ingressi. Per tale motivo sono anche chiamati circuiti senza memoria: le uscite in ogni istante sono funzione esclusivamente dei valori degli ingressi in quello stesso istante. Nei circuiti combinatori una funzione logica degli ingressi si realizza attraverso componenti capaci di assumere uno dei due stati di tensione: il livello alto e il livello basso, indicati rispettivamente con h e l.

In un circuito combinatorio si può lavorare in logica positiva, cioè la convenzione secondo la quale il valore logico 1 viene associato al livello alto e di conseguenza il valore logico 0 viene associato al livello basso. In alternativa si può lavorare in logica negativa, secondo la convenzione che al valore logico 1 si associa al livello di tensione basso e di conseguenza il valore logico 0 si associa al livello di tensione alto. Le reti logiche combinatorie sono quelle reti in cui lo stato d'uscita viene a dipendere solo dallo stato degli ingressi propri presenti in quell'istante.

Si pone il problema di ottimizzare i circuiti combinatori. I criteri di ottimizzazione possono essere diversi a seconda dei problemi: per esempio, ottimizzazione dei costi, della funzionalità o della velocità; nella maggior parte dei casi si cerca una via ponderata per tutte queste esigenze. In particolare il costo di un circuito logico è caratterizzato dal numero di porte logiche utilizzate e dalla profondità, cioè dalla lunghezza del percorso tra l'input e l'output, che tiene conto dei ritardi temporali e dal numero di ingressi.

In questo contesto la sintesi di un circuito combinatorio o rete combinatoria è l'individuazione, una volta assegnata la specifica funzionale del circuito, del sistema digitale e delle interconnessioni che realizzano tale specifica. Per analisi di un circuito combinatorio si intende l'individuazione delle relazioni causa-effetto tra i segnali di ingresso, cioè le variabili booleane in ingresso, e le uscite del circuito. Per svolgere questi compiti si utilizza l'algebra di Boole.

In generale una rete logica combinatoria viene progettata partendo da una descrizione funzionale della rete logica, cioè di quello che deve fare. A partire dall'algebra di Boole, si rappresenta la rete logica con una funzione logica, una tabella della verità o con una mappa di Karnaugh. Poi entra in gioco la riduzione della funzione in forma minima, per ottimizzare la rete logica e infine si passa alla realizzazione effettiva circuitale.

In realtà, la progettazione è praticamente un problema di sintesi, ma nel quale, poi si deve verificare la correttezza del segnale, in quanto corrispondente alle specifiche.