Architettura dei Calcolatori
Algebra delle reti Logiche

Ing. dell'Automazione
A.A. 2011/12
Gabriele Cecchetti

Algebra delle reti logiche

- Sommario:
  - Segnali e informazione
  - Algebra di commutazione
  - Porta logica e transitor
  - Analisi di reti combinatorie
  - Sintesi di reti combinatorie
  - Minimizzazione di reti combinatorie

- Riferimenti:
  - Hamacher "Introduzione all'architettura del Calcolatore", cap. 2, sezioni 2.1, 2.2, 2.3, 2.4, 2.5
  - G. Bucci "Architetture e organizzazione dei Calcolatori Elettronici – Fondamenti", Cap. 3, sezioni 3.1, 3.2, 3.3, 3.4, 3.5 e 3.6.

Nozioni fondamentali di tecnologia microelettronica

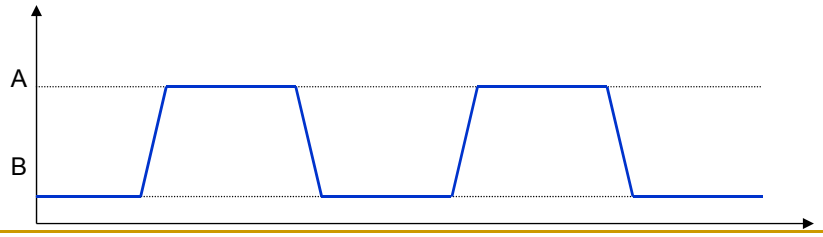
SEGNALE E INFORMAZIONE

Segnale e informazione

- Per elaborare informazioni, occorre rappresentarle (o codificarle) mediante una tecnica di rappresentazione.
- Per rappresentare le informazioni si usano segnali.
- I segnali devono essere elaborati, nei modi opportuni, tramite dispositivi di elaborazione.
- In un sistema digitale le informazioni sono rappresentate, elaborate e trasmesse mediante grandezze fisiche che assumono solo valori discreti.
- Ogni valore è associato a una cifra (*digit*) della rappresentazione.

Segnali digitali vs. segnali analogici

- Un segnale analogico porta più informazione di uno digitale:
  - infinità di valori nell'intervallo di definizione,
  - infinità di andamenti della forma d'onda.
- Nella pratica disturbi e limiti fisici alla velocità di variazione rendono inaccurato sia il processo di generazione sia quello di riconoscimento e misura dei segnali analogici.
- I segnali digitali sono meno sensibili al rumore e ai fenomeni transitori.
- **I sistemi digitali più semplici ed affidabili sono quelli *binari***, in cui i segnai possono assumere ***due*** valori e ai cui componenti è richiesta solo la capacità di discriminarli.



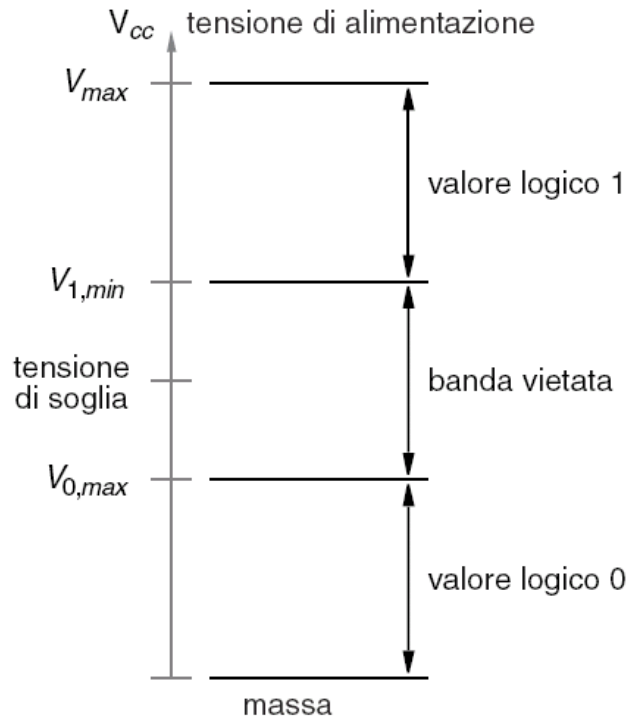
Segnale binario (1/2)

- Segnale binario: grandezza fisica che assume due valori distinti, indicati per convenzione con le cifre 0 e 1:

  $s \in \{0, 1\}$ (low, high - false, true - falso, vero)

- Grandezze fisiche usate per rappresentare l'informazione nel sistema digitale:
  - *elettrica* (tensione o corrente)
  - *magnetica* (intensità magnetica)
  - *ottica* (potenza ottica)

Segnale binario nei circuiti elettronici



Valori tipici per la tecnologia TTL:

- $V_{cc} = 5$ Volt
- $V_{massa} = 0$ Volt
- $V_{1,\,min} \approx 2$ Volt
- $V_{0,\,max} \approx 0{,}8$ Volt
- soglia $\approx 1{,}5$ Volt

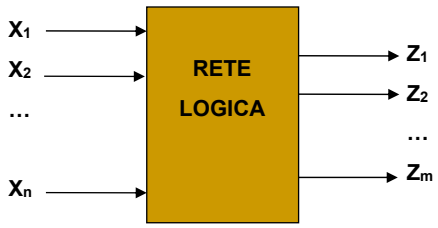
Valori tipici per la tecnologia MOS:

- $V_{cc} = 5$ Volt
- $V_{massa} = 0$ Volt
- $V_{1,\,min} \approx 3{,}8$ Volt
- $V_{0,\,max} \approx 1{,}3$ Volt
- soglia $\approx 2{,}5$ Volt

- La grandezza fisica assume solo due valori discreti

Classificazione dei sistemi digitali

- Per elaborare il segnale binario, si usano due classi di dispositivi (*digitali*) di elaborazione:



- **Reti combinatorie** *(reti senza memoria)*: l'uscita all'istante di tempo *t* dipende *esclusivamente* dall'ingresso:

$$O = f(I)$$

- **Reti sequenziali** *(reti con memoria)*: quando l'uscita è all'istante di tempo *t* è funzione, oltre che dell'ingresso allo stesso istante di tempo, anche dello *stato della rete*.

$$O = f(I,S)$$

$$S_{futuro} = g(I,S)$$

Concetti, operazioni, proprietà dell'Algebra di Boole

Funzione combinatoria

ALGEBRA DI COMMUTAZIONE (O *ALGEBRA DI BOOLE*)

Algebra di commutazione

- Deriva dall'Algebra di Boole e consente di descrivere matematicamente i circuiti digitali (o circuiti logici).
- Definisce le espressioni logiche che descrivono il comportamento del circuito da realizzare, nella forma seguente (I ingressi, U uscite):

$$\mathbf{U} = \boldsymbol{f}\,(\mathbf{I})$$

- A partire dalle equazioni logiche si può derivare la realizzazione circuitale (rete logica).
- I componenti dell'algebra di Boole sono: le variabili di commutazione, gli operatori fondamentali e le proprietà degli operatori logici, tramite le quali è possibile trasformare le espressioni logiche.

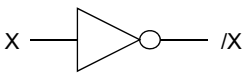
Variabile e operatore

- Una variabile di commutazione (o variabile logica) corrisponde al singolo bit di informazione rappresentata e elaborata.
- Gli operatori fondamentali sono i seguenti:

negazione (o inversione o complemento)	!A, /A, $\overline{A}$	= 1 se A = 0 = 0 se A = 1
prodotto logico	$A \cdot B$	= 1 se A = B = 1 = 0 altrimenti
somma logica	$A + B$	= 0 se A = B = 0 = 1 altrimenti

Negazione: NOT

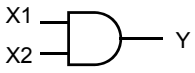
- ***Il risultato è 1 se il termine a cui si applica è 0 ed è 0 se il termine a cui si applica è 1.***
  - Operatore logico: $\overline{}$
  - Simbolo circuitale:



  - Tabella di verità

A	/A
0	1
1	0

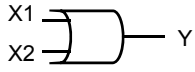
Prodotto logico: AND

- ***Il risultato è 1 se e solo se sono 1 ambedue i termini del prodotto.***
  - Operatore logico: •
  - Simbolo circuitale: 

  - Tabella di verità

A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

Somma logica: OR

- ***Il risultato è 1 se almeno uno dei due termini è 1.***
  - Operatore logico: +
  - Simbolo circuitale: 

  - Tabella di verità

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Proprietà dell'algebra (1/3)

1. Idempotenza

$$x + x = x$$
$$x * x = x$$

2. Distributiva

$$x * (y + z) = (x * y) + (x * z)$$
$$x + (y * z) = (x + y) * (x + z)$$

3. Associativa

$$x + (y + z) = (x + y) + z$$
$$x * (y * z) = (x * y) * z$$

4. Commutativa

$$x + y = y + x$$
$$x * y = y * x$$

Proprietà dell'algebra (2/3)

5. Assorbimento

$$x + x * y = x$$
$$(x + y) * x = x$$

6. Operatore neutro

$$0 + x = x$$
$$1 * x = x$$

7. Operatore nullo (o *di complemento*)

$$x * /x = 0$$
$$x + /x = 1$$

8. Doppia negazione (o *involuzione*)

$$//x = x$$

Proprietà dell'algebra (3/3)

9. Elementi delle operazioni forzanti AND e OR

$$x + 1 = 1$$

$$x * 0 = 0$$

10. Teorema di De Morgan

$$/(x + y) = /x * /y$$

$$/(x * y) = /x + /y$$

- Principio di dualità
  - Per ciascuna coppia di regole si passa all'altra se si scambiano tra loro le operazioni di somma e prodotto e gli 1 con gli 0.

Costante logica e variabile logica

- **Costante logica** → *un simbolo cui è permanentemente assegnato uno dei possibili valori presi dall'insieme {0,1}*

- ***Variabile logica*** → *un simbolo che può assumere indifferentemente uno dei due valori presi dall'insieme {0,1}*

Espressione logica

- *Def.* **Espressione logica** → *una qualunque combinazione di variabili o costanti legate tra loro da operatori logici fondamentali.*

- Combinando costanti logiche (0 e 1), variabili logiche (*a*, *b*, ecc, *x*, *y*, ecc) e operatori logici si ottengono espressioni logiche (o booleane).
- Un'espressione logica assume diversi valori secondo come vangano assegnate le variabili, a 0 o 1.
- Ogni proprietà degli operatori logici è formulata come uguaglianza tra espressioni, la quale vale per qualunque assegnamento alle variabili (si dice essere un'*identità logica*).

Trasformazione delle espressioni logiche

- L'espressione logica può assumere forme equivalenti, le quali assumono valore identico a fronte dello stesso assegnamento.

- Espressioni logiche equivalenti sono ottenibili l'una dall'altra tramite l'applicazione ripetuta delle proprietà degli operatori logici.

Esempio di trasformazione

- Trasformare $/a\ /b\ c + /a\ b\ c + /a\ b\ /c$

espressione trasformata	proprietà utilizzata
/a /b c + /a b c + /a b /c	idempotenza x + x = x
/a /b c + /a b c + /a b c + /a b /c	distributiva x y + x z = x (y + z)
/a c (/b + b) + /a b (c + /c)	complemento x + /x = 1
/a c 1 + /a b 1	prop. dello 1 x 1 = x
/a c + /a b	distributiva
/a (c + b)	

Funzione logica

- *Def.* **Funzione logica delle *n* variabili booleane $x_1, \ldots, x_n$** → *la relazione $\{0,1\}^n \rightarrow \{0,1\}$ che associa un valore booleano a ciascuna delle 2^n configurazioni possibili delle n variabili:*

$$y = f(x_1, \ldots, x_n)$$

- **Una funzione logica** (o combinatoria, o funzione booleana) corrisponde a un'espressione booleana, contenente una o più variabili booleane e gli operatori booleani AND, OR e NOT:
  - dando dei valori alle variabili booleane della funzione combinatoria, si calcola il corrispondente valore della funzione

Esempio di funzione logica

- Esempio:
  funzioni logiche *S* e *C* a 2 ingressi *a* e *b*, tali che
  - *S* = 1 se e solo se solo uno degli ingressi vale 1
  - *C* = 1 se e solo se entrambi gli ingressi valgono 1

→ le espressioni booleane per le due funzioni *S* e *C* sono:

- $S = a/b + /a\ b$
- $C = a\ b$

Tabella di verità

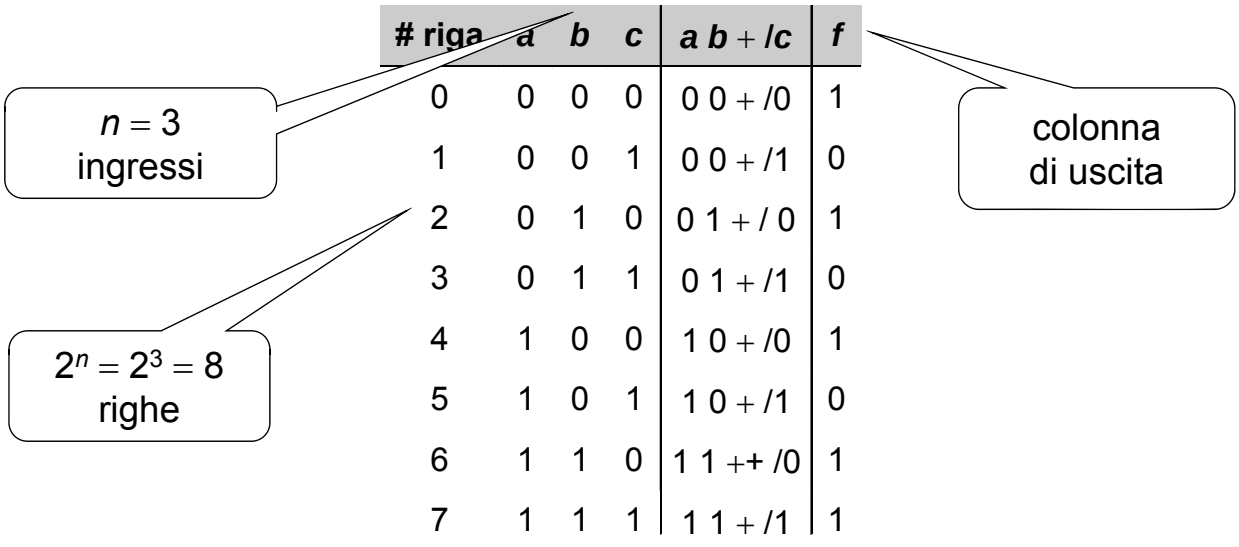
- *Per specificare il comportamento di una funzione logica basta specificare, per ogni possibile configurazione degli ingressi, il valore dell'uscita*.
- **La tabella di verità di una funzione a *n* ingressi ha 2^n righe, che corrispondono a tutte le possibili configurazioni di ingresso.**
- La tabella di verità ha due "gruppi" di colonne:
  - *colonne degli ingressi*, le cui righe contengono tutte le combinazioni di valori (assegnamenti) delle variabili della funzione
  - *colonna di uscita*, che riporta i corrispondenti valori assunti dalla funzione

Esempio di tabella di verità (1/2)

- $f(a, b, c) = a\ b + /c$ è una funzione logica a 3 variabili a, b e c:
  - $f(0, 0, 0) = 0\ 0 + /0 = 0 + 1 = 1$
  - $f(0, 0, 1) = 0\ 0 + /1 = 0 + 0 = 0$
  - $f(0, 1, 0) = 0\ 1 + /0 = 0 + 1 = 1$
  - … (e così via)
  - $f(1, 1, 1) = 1\ 1 + /1 = 1 + 0 = 1$

- In totale ci sono $2^3 = 8$ assegnamenti, pertanto la tabella di verità ha 8 righe.

Esempio di tabella di verità (2/2)



# riga	a	b	c	a b + /c	f
0	0	0	0	0 0 + /0	1
1	0	0	1	0 0 + /1	0
2	0	1	0	0 1 + / 0	1
3	0	1	1	0 1 + /1	0
4	1	0	0	1 0 + /0	1
5	1	0	1	1 0 + /1	0
6	1	1	0	1 1 ++ /0	1
7	1	1	1	1 1 + /1	1

Altro esempio di tabella di verità

#	x_1	x_2	x_3	$\overline{x}_1\overline{x}_2$	x_2x_3	$\overline{x}_1\overline{x}_2 + x_2x_3 = f_1$
0	0	0	0	1	0	1
1	0	0	1	1	0	1
2	0	1	0	0	0	0
3	0	1	1	0	1	1
4	1	0	0	0	0	0
5	1	0	1	0	0	0
6	1	1	0	0	0	0
7	1	1	1	0	1	1

Equivalenza di funzioni logiche

- *Una funzione logica può ammettere più espressioni booleane differenti che la definiscono*.
- Esse hanno tutte lo stesso comportamento (cioè la medesima tabella di verità), ma possono avere struttura (e costo) differente.

- Per vedere se due *reti combinatorie* siano equivalenti, basta confrontarne le tabelle di verità.
- Alternativamente, si può cercare di trasformare un'espressione booleana nell'altra (ciò in genere è più difficile del ricavarne le tabelle di verità).

Esempio di funzioni logiche equivalenti

- le funzioni $x\,(y + z)$ e $x\,y + x\,z$ sono equivalenti

#	x	y	z	$y + z$	$x\,(y + z)$	xy	xz	$xy + xz$
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0
2	0	1	0	1	0	0	0	0
3	0	1	1	1	0	0	0	0
4	1	0	0	0	0	0	0	0
5	1	0	1	1	1	0	1	1
6	1	1	0	1	1	1	0	1
7	1	1	1	1	1	1	1	1

Porta Logica e Transistore

Tecnologia del transistore
porte logiche fondamentali

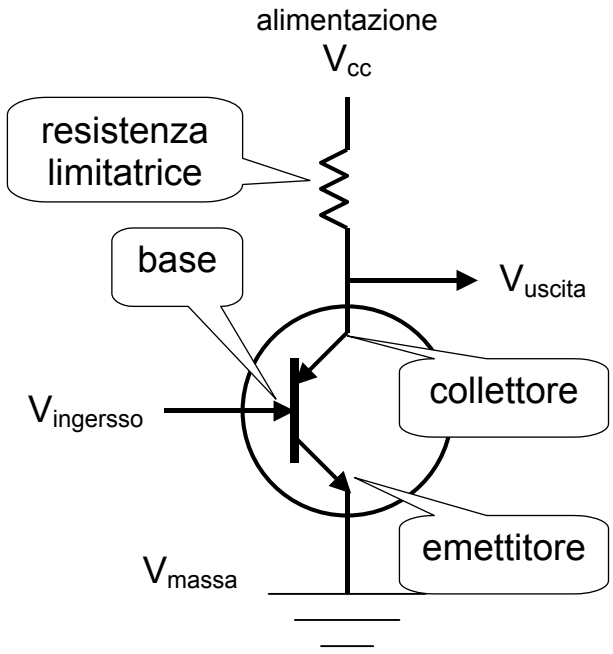
Porta logica

- Il circuito digitale è formato da componenti digitali elementari, chiamati porte logiche.
- La porta logica è il circuito minimo per l'elaborazione di segnali binari e corrisponde agli operatori elementari dell'algebra di commutazione.
- Le porte logiche sono classificate in base al modo di funzionamento: porta **NOT**, **AND** od **OR**.
- Queste sono le porte logiche di base e costituiscono un insieme di operatori funzionalmente completo:
  - classificazione per numero di ingressi: porte a 1 ingresso, a 2 ingressi, 3 ingressi, e così via ...

Transistore

- L'elemento funzionale fondamentale per la costruzione della porta logica è il transistore.
- Il transistore è un dispositivo elettronico.
- Opera su grandezze elettriche:
  - tensione e corrente
- Funziona come un interruttore.
- Ha due stati di funzionamento:
  - interruttore aperto
  - interruttore chiuso

Struttura del Transistore Bipolare



transistore bipolare
(*bipolar junction transistor, BJT*)

Funzionamento di BJT (1/2)

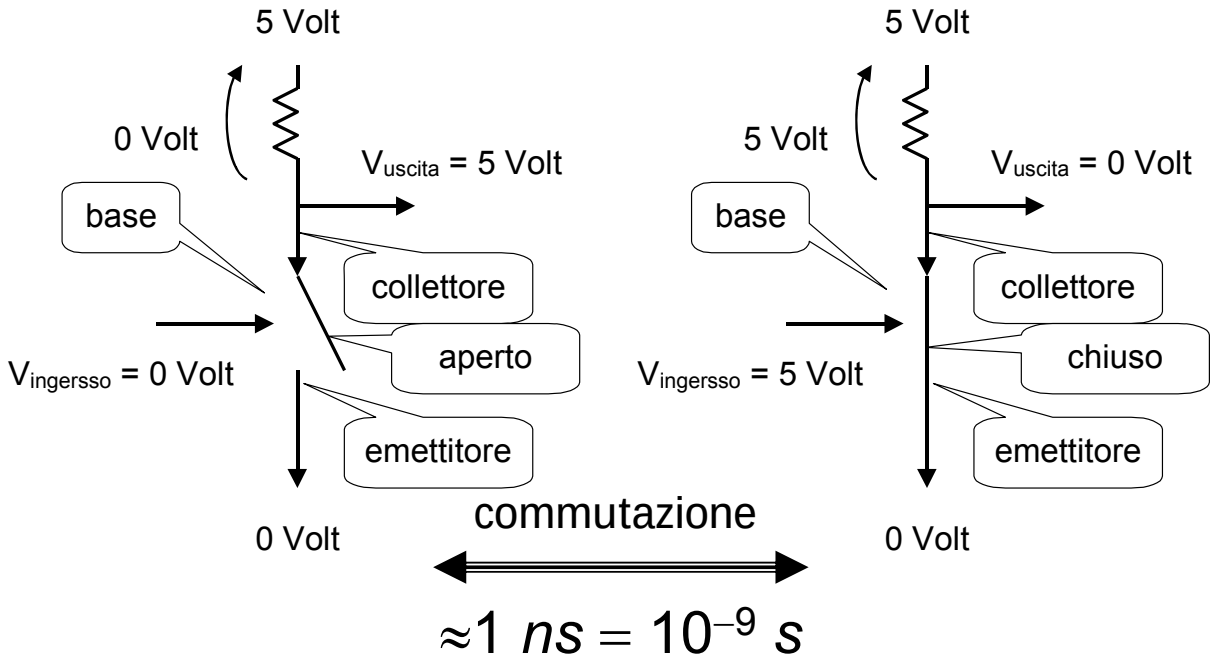
- Se la tensione di base $V_{ingresso}$ è inferiore a una data soglia critica, il transistore si comporta come un interruttore aperto, cioè tra emettitore e collettore non passa corrente, e pertanto la tensione di uscita diventa uguale a quella di alimentazione:

$$V_{uscita} = V_{cc} = 5 \text{ Volt} \qquad \text{(in tecnologia TTL)}$$

- Se la tensione di base $V_{ingresso}$ è superiore a una data soglia critica, il transistore si comporta come un interruttore chiuso, cioè tra emettitore e collettore passa corrente, e pertanto la tensione di uscita diventa uguale a quella di massa:

$$V_{uscita} = V_{massa} = 0 \text{ Volt} \qquad \text{(in tecnologia TTL)}$$

Funzionamento di BJT (2/2)



Porta NOT con BJT

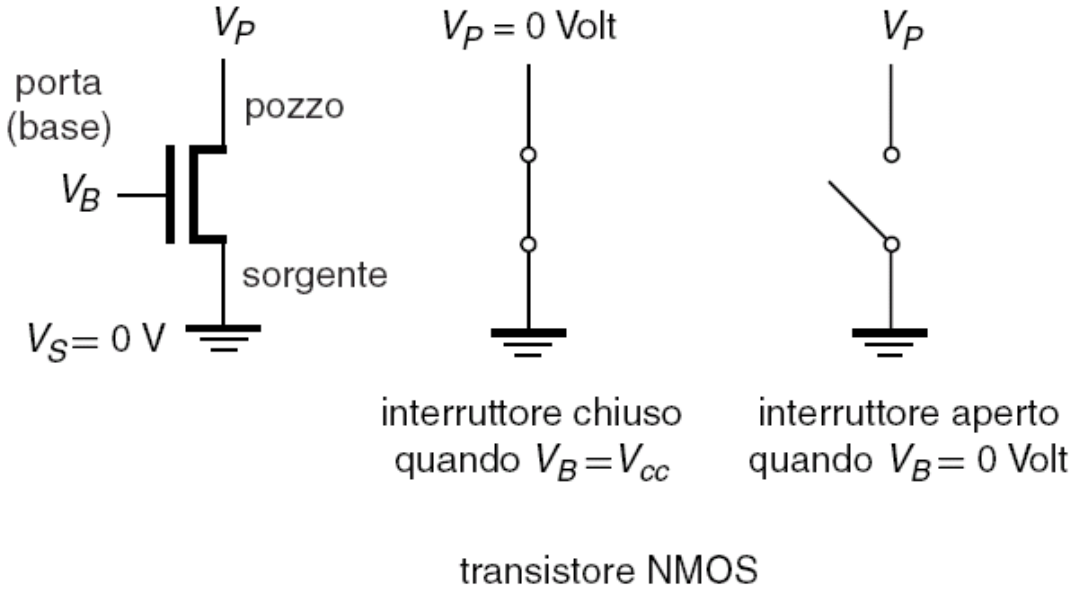
- Il singolo transistore BJT della figura è una porta NOT.
- Se l'ingresso vale 0 Volt, l'uscita vale 5 Volt.
- Se l'ingresso vale 5 Volt, l'uscita vale 0 Volt.
- La tabella seguente rappresenta il funzionamento della porta NOT, e si può interpretare come tabella di verità:

valore logico di ingresso	$V_{ingresso}$	V_{uscita}	valore logico di uscita
0	0 Volt	5 Volt	1
1	5 Volt	0 Volt	0

Tecnologia CMOS complementare

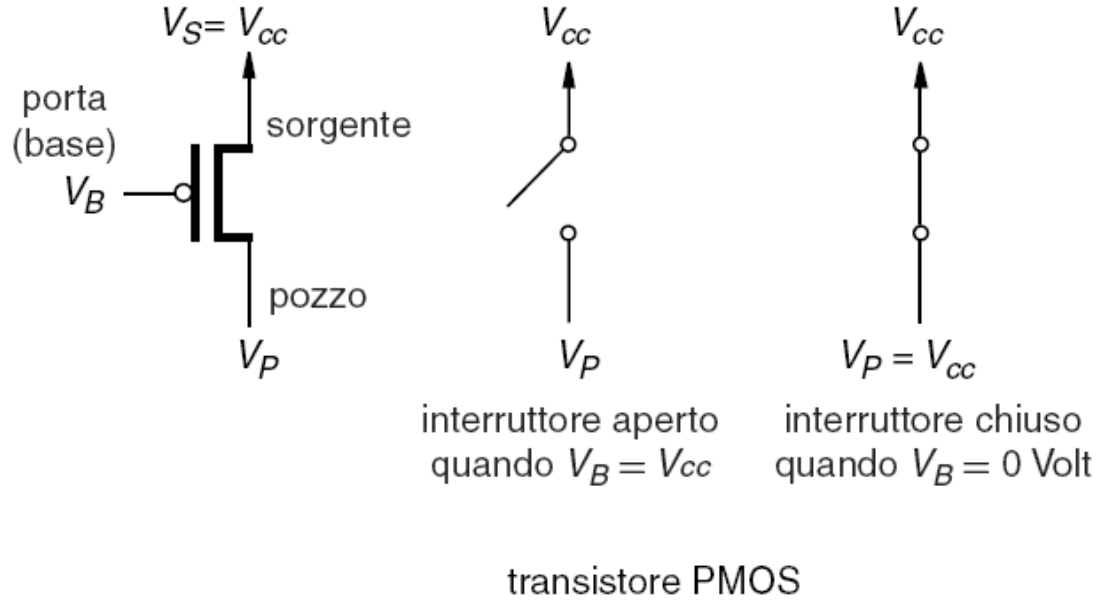
- Il transistore BJT non è oggi il più usato nei circuiti integrati costituenti il calcolatore.
- Si usa il transistore MOS (*Metal Oxide Silicon*), più compatto e a consumo di energia minore.
- Ci sono due tipi di tale transistore, NMOS e PMOS (*negative* e *positive MOS*), che funzionano in modo complementare:
  - lo stato di chiuso di NMOS corrisponde a quello di aperto di PMOS (e viceversa).
- Si usano in combinazione per realizzare circuiti logici particolarmente compatti e a basso consumo:
  - tecnologia CMOS (*complementary MOS*).

Struttura del transistore NMOS



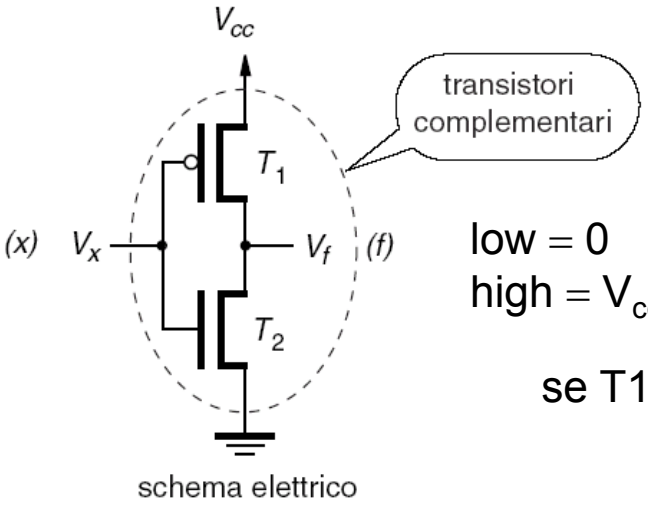
transistore NMOS

Struttura del transistore PMOS



transistore PMOS

Porta NOT in Tecnologia CMOS



schema elettrico

low = 0
high = V_{cc}

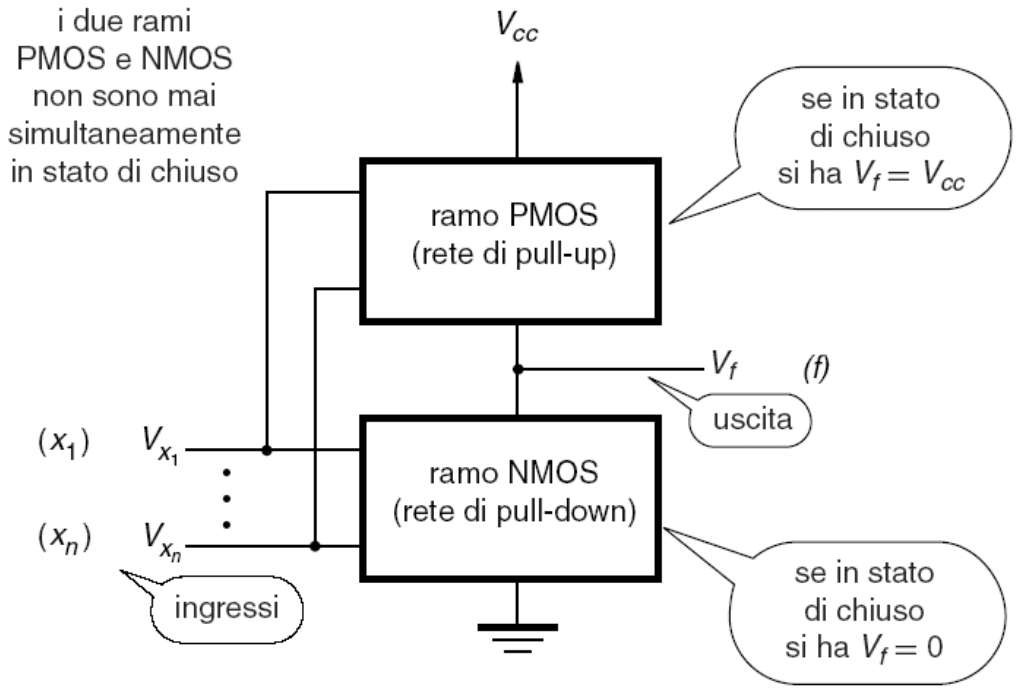
costo = 2 transistori

x	V_x	T_1	T_2	V_f	f
0	low	on	off	high	1
1	high	off	on	low	0

tabella di verità

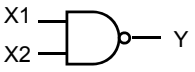
se T1 è chiuso T2 è aperto,
e viceversa

Circuito CMOS generale



L'operatore logico NAND

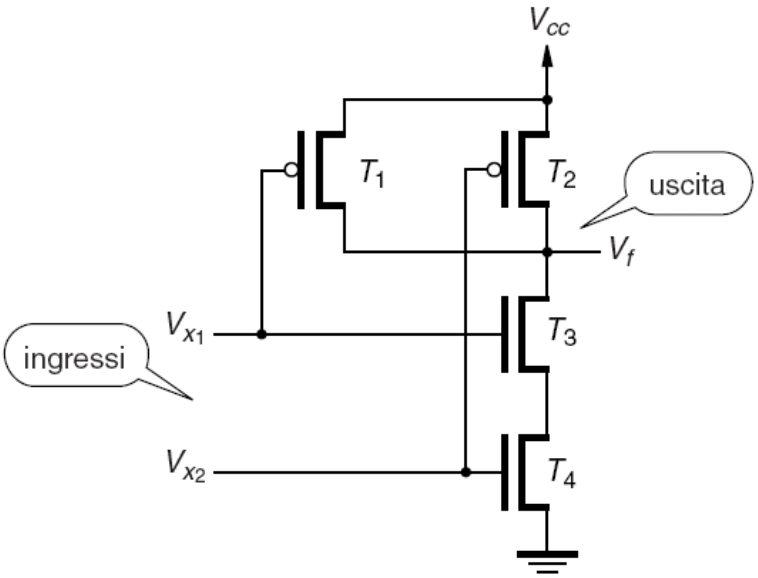
- ***Il risultato è 0 se e solo se sono 1 ambedue i termini del prodotto.***
  - Operatore logico: |
  - Simbolo circuitale:



  - Tabella di verità

A	B	A|B
0	0	1
0	1	1
1	0	1
1	1	0

Sintesi di NAND in CMOS



schema elettrico

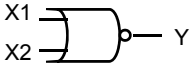
costo = 4 transistori

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

tabella di verità

L'operatore logico NOR

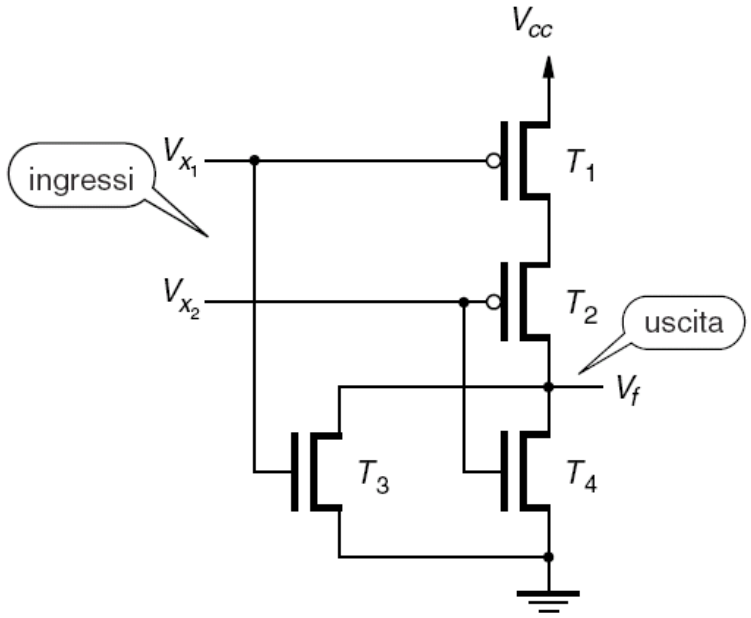
- ***Il risultato è 0 se almeno uno dei due termini è 1.***
  - Operatore logico: ↓
  - Simbolo circuitale:



  - Tabella di verità

A	B	A↓B
0	0	1
0	1	0
1	0	0
1	1	0

Sintesi di NOR in CMOS



schema elettrico

costo = 4 transistori

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

tabella di verità

Insiemi di operatori algebrici funzionalmente ridondanti e completi

- **{ + , • , ¯ }** è funzionalmente *ridondante*
- **{ • , ¯ }** è funzionalmente *completo*
- **{ + , ¯ }** è funzionalmente *completo*

- Dimostrazione tramite il teorema di De Morgan:

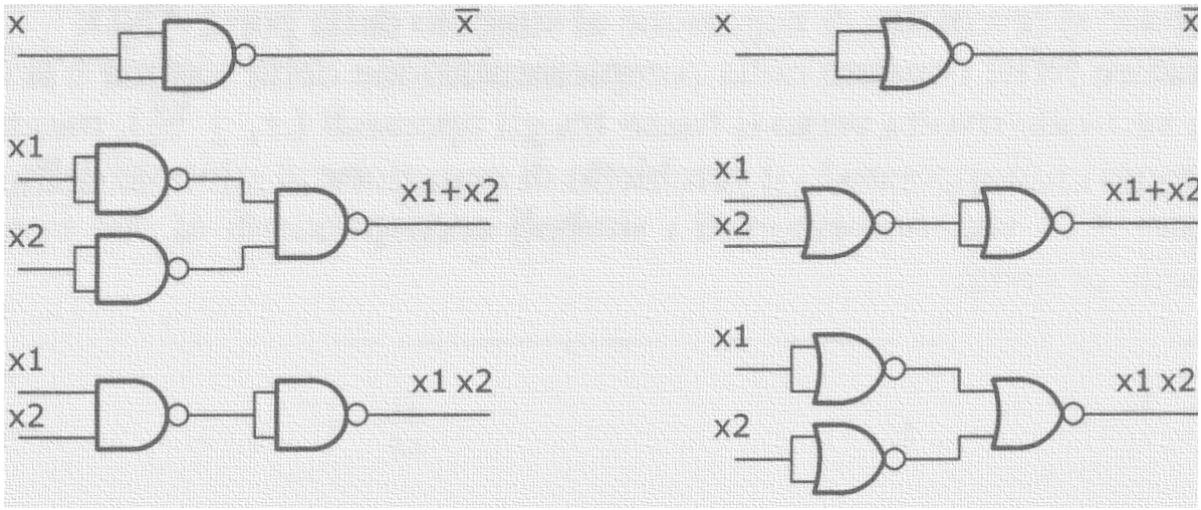
x + y = //(x + y) = /(/x * /y)

x * y = //(x * y) = /(/x + /y)

Porte funzionalmente complete

- La porta NAND è funzionalmente completa:
  - da sola basta per esprimere tutte le altre porte
- Trasformare una porta qualunque in una combinazione di NAND si può fare sempre, ma in generale in più di un modo.
- Se non si bada particolarmente al costo, la trasformazione è molto semplice.
- Le stesse considerazioni valgono anche per la porta NOR (è funzionalmente completa).

Costruzione delle operazioni di NOT, OR e AND dalle porte NAND e NOR



L'operatore logico XOR (OR esclusivo)

- *Il risultato è 1 se e solo se vale 1 solo un termine del prodotto.*
  - Tabella di verità XOR a due ingressi (diseguaglianza)

A	B	y
0	0	0
0	1	1
1	0	1
1	1	0

  - generalizzato a *n* variabili di ingresso: l'uscita vale 1 se e solo se il numero di 1 è dispari

L'operatore logico NXOR (o X-NOR)

- *Il risultato è 1 se entrambi i termini hanno lo stesso valore.*
  - Tabella di verità NXOR a due ingressi (eguaglianza)

A	B	y
0	0	1
0	1	0
1	0	0
1	1	1

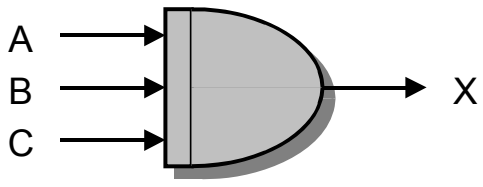
  - generalizzato a *n* variabili di ingresso: l'uscita vale 1 se e solo se il numero di 1 è pari

Porte a tre o più ingressi

- Alcuni tipi di porte a 2 ingressi si possono generalizzare a 3, 4, ecc ingressi.
- Le due porte a più ingressi più usate sono la porta AND e la porta OR.
- Tipicamente si usano AND (o OR) a 2, 4 o 8 ingressi (raramente più di 8).

Porta AND a tre ingressi

simbolo funzionale



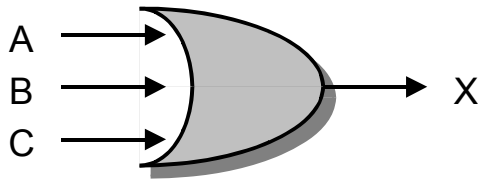
l'uscita vale 1 se e solo se tutti e 3 gli ingressi valgono 1

tabella di verità

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

Porta OR a tre ingressi

simbolo funzionale



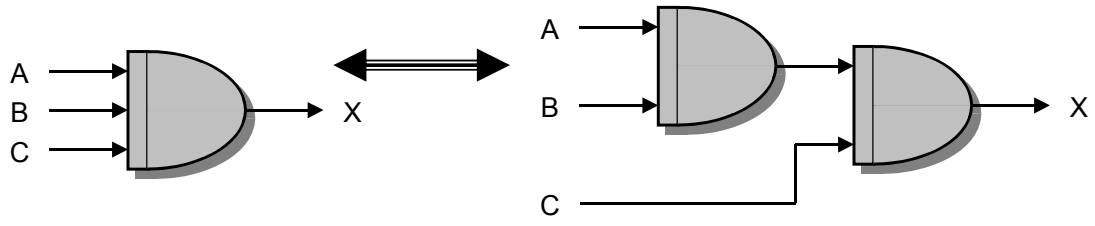
l'uscita vale 0 se e solo se tutti e 3 gli ingressi valgono 0

tabella di verità

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

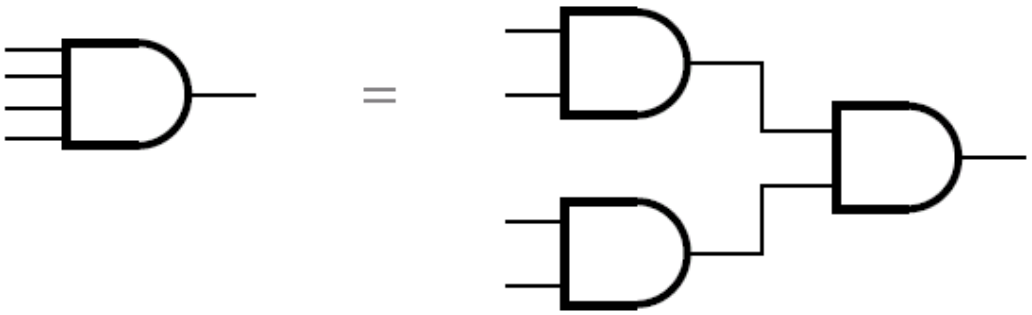
Realizzazione ad albero

- La porta AND a 3 ingressi si realizza spesso come albero di porte AND a 2 ingressi (ma non è l'unico modo).



- Nota bene: non tutti i tipi di porte a più di 2 ingressi si possono realizzare come alberi di porte a 2 ingressi (funziona sempre con AND, OR, X-OR e X-NOR).

Porta AND a 4 ingressi



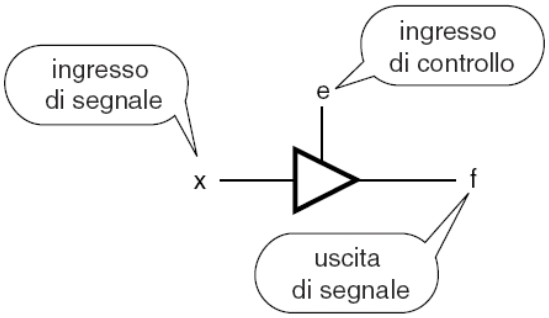
costruzione ad albero della porta AND a quattro ingressi
usando solo porte AND a due ingressi (idem per porta OR)

- si generalizza a *n* ingressi nel modo ovvio

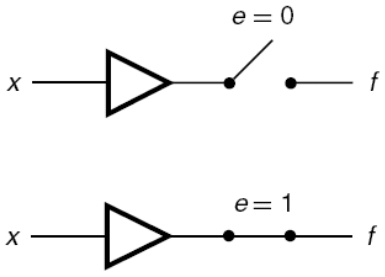
Porta 3-State (1/3)

- **La porta *3-state*** (3-state buffer) **lascia passare in uscita il segnale di ingresso, così come è.**
- **Essa dispone però anche di un ingresso di controllo, che permette di porre l'uscita in stato di isolamento elettrico** (o di alta impedenza, Z).
- Quando la porta è isolata, è come se l'uscita non fosse presente.
- La porta 3-state è fondamentale per collegare più dispositivi alla medesima linea di uscita.
- Tipicamente si usa per collegare più dispositivi allo stesso bus (fascio di linee).

Porta 3-state (2/3)



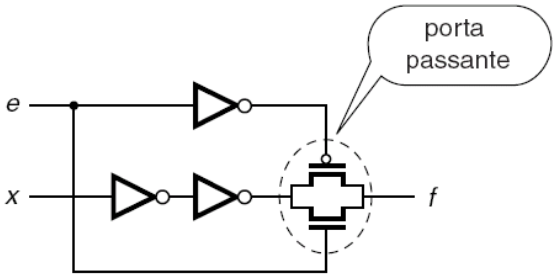
rappresentazione grafica



schema logico

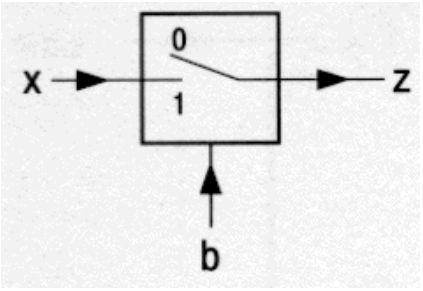
e	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

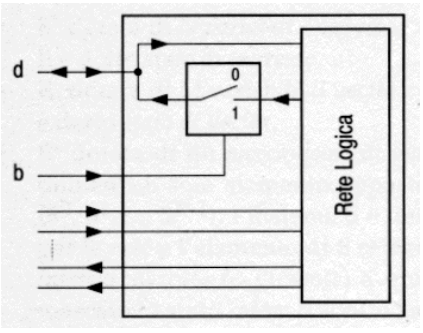
tabella di verità



schema elettrico

Porte 3-state (3/3)





- Se b = 1 ⇒ z = x
- Se b = 0 ⇒ z è in alta impedenza (indefinito)
- In questo esempio la var. d diventa *bidirezionale* grazie alla presenza di una porta 3-state

Costo di realizzazione

- Costo di realizzazione:
  - il numero di transistori per realizzare una porta dipende dalla tecnologia, dalla funzione e dal numero di ingressi
- In tecnologia CMOS si ha:
  - porta NOT: 2 transistori
  - porte NAND e NOR: 4 transistori
  - porte AND e OR: 6 transistori
  - altre porte: $\geq$ 4 transistori

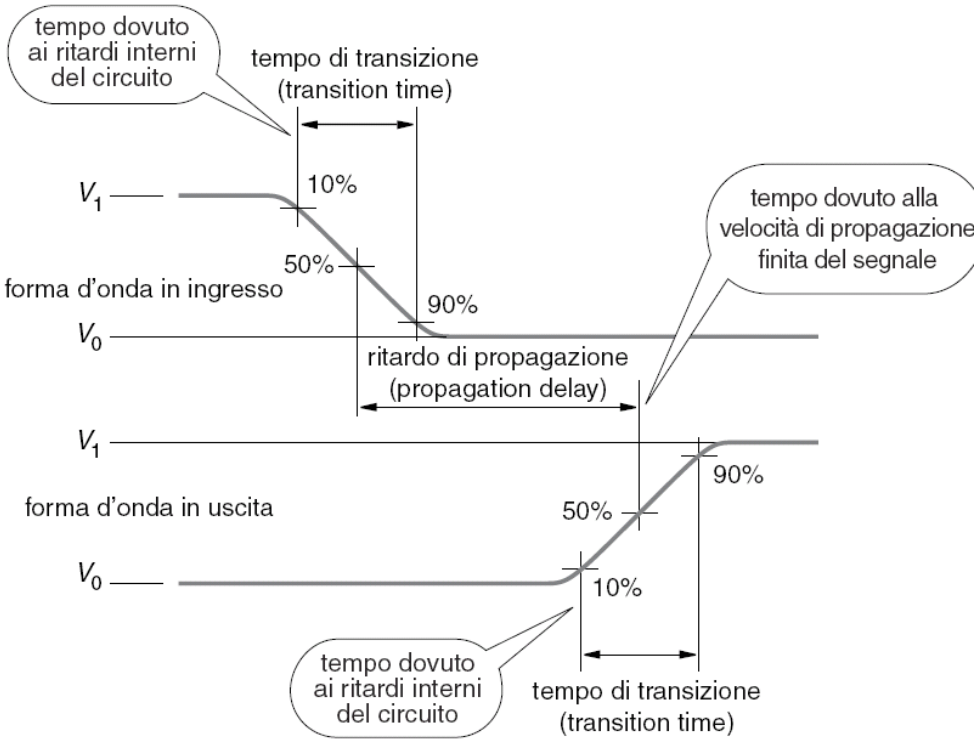
Velocità di commutazione

- Velocità di commutazione:
  - la velocità di commutazione di una porta dipende dalla tecnologia, dalla funzione e dal numero di ingressi
  - le porte più veloci (oltre che più piccole) sono porte NAND e NOR a 2 ingressi: commutano in meno di 1 nanosecondo (10^{-9} s, un miliardesimo di secondo)

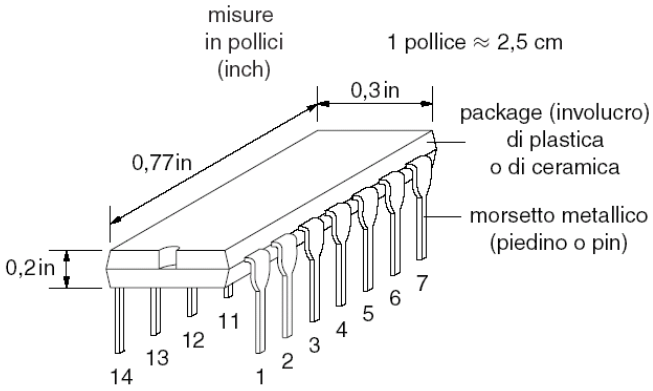
Costo e velocità di una rete logica

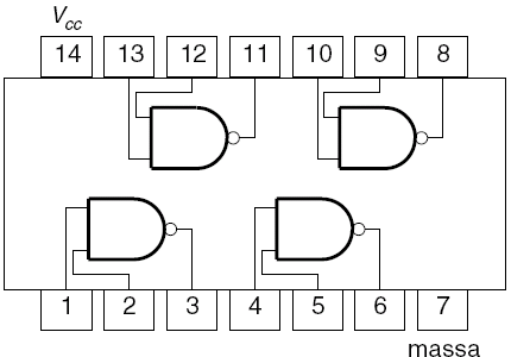
- Il costo e la velocità di commutazione delle porte logiche danno un'indicazione del costo della rete logica e del ritardo di propagazione associato alla rete stessa, rispettivamente.

Cause fisiche del ritardo di una porta



Esempio di circuito integrato





- Circuito a scala di integrazione piccola (SSI)

Dalla rete combinatoria alla tabella di verità
e all'espressione booleana

ANALISI DI RETI COMBINATORIE

Rete Combinatoria (1/2)

- A ogni funzione combinatoria, data come espressione booleana, si può sempre associare un circuito digitale, formato da porte logiche di vario tipo, chiamato rete combinatoria.

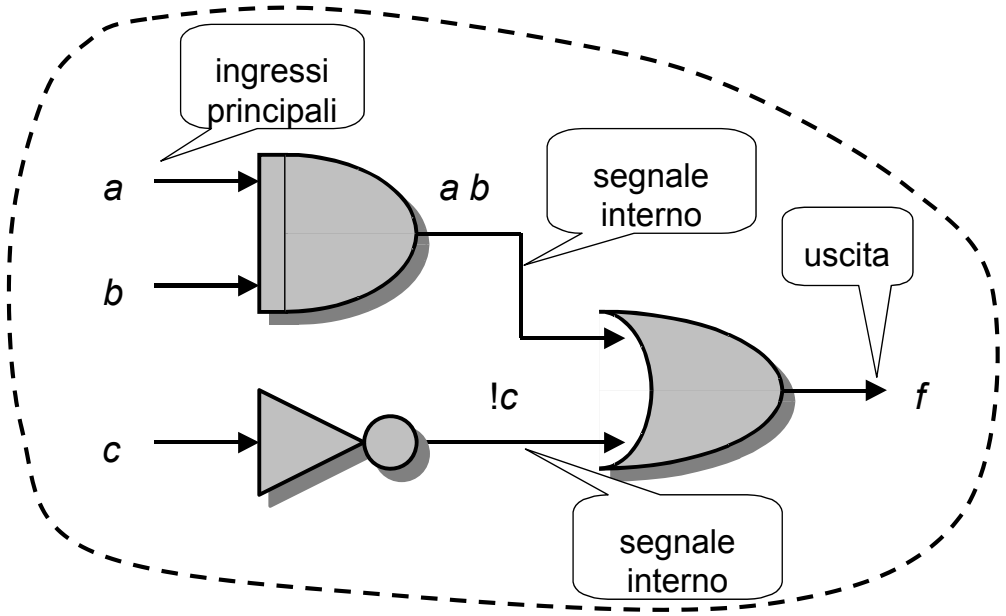
- Gli *ingressi* della rete combinatoria sono le variabili della funzione.
- L'*uscita* della rete combinatoria emette il valore assunto dalla funzione.

Rete Combinatoria (2/2)

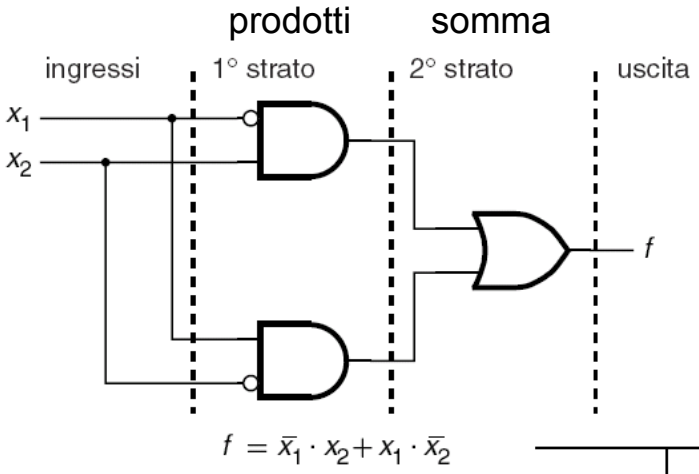
- Una rete combinatoria è un circuito digitale:
  - dotato di $n \geq 1$ ingressi principali e di un'uscita
  - formato da porte logiche AND, OR e NOT (ed eventualmente anche da altri tipi di porte)
  - e privo di retroazioni
- Costruzione della rete combinatoria a partire dalla funzione logica data come espressione booleana:
  - variabili dirette e negate
  - ogni termine dell'espressione è sostituito dalla corrispondente rete di porte fondamentali
  - le uscite corrispondenti a ogni termine si compongono come indicato dagli operatori

Esempio

$$f(a, b, c) = a\,b + /c$$



Altro Esempio



rete combinatoria a 2 livelli (o strati), di tipo somma di prodotti

tabella di verità della funzione combinatoria corrispondente alla rete combinatoria

#	x_1	x_2	$\bar{x}_1 \cdot x_2$	$x_1 \cdot \bar{x}_2$	$f = \bar{x}_1 \cdot x_2 + x_1 \cdot \bar{x}_2 = x_1 \oplus x_2$
0	0	0	0	0	0
1	0	1	1	0	1
2	1	0	0	1	1
3	1	1	0	0	0

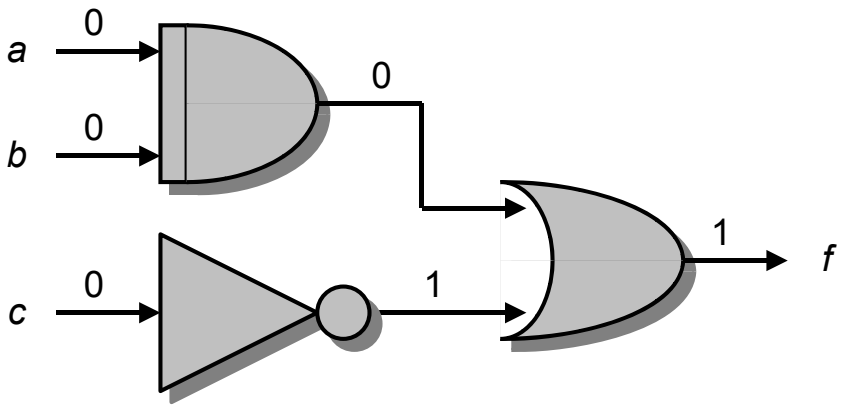
Simulazione Circuitale

- Analizzare una rete combinatoria significa passare dalla rete stessa data come circuito formato da porte logiche, alla tabella di verità.

- La tabella di verità della rete combinatoria può anche essere ricavata per simulazione del funzionamento circuitale della rete combinatoria stessa.
- Per simulare il funzionamento circuitale di una rete combinatoria, si applicano valori agli ingressi e li si propaga lungo la rete fino all'uscita, determinandone il valore.

Rete ⇒ Tabella di Verità (1/4)

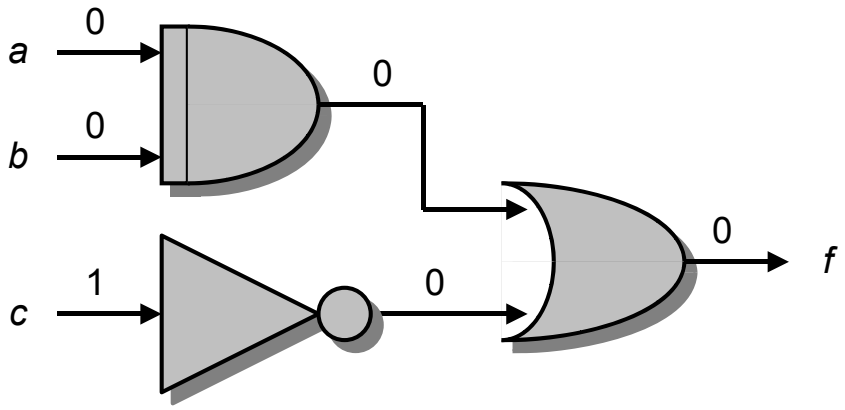
(corrisponde alla riga 0 della tabella)



risultato della simulazione: $f(0, 0, 0) = 1$

Rete ⇒ Tabella di Verità (2/4)

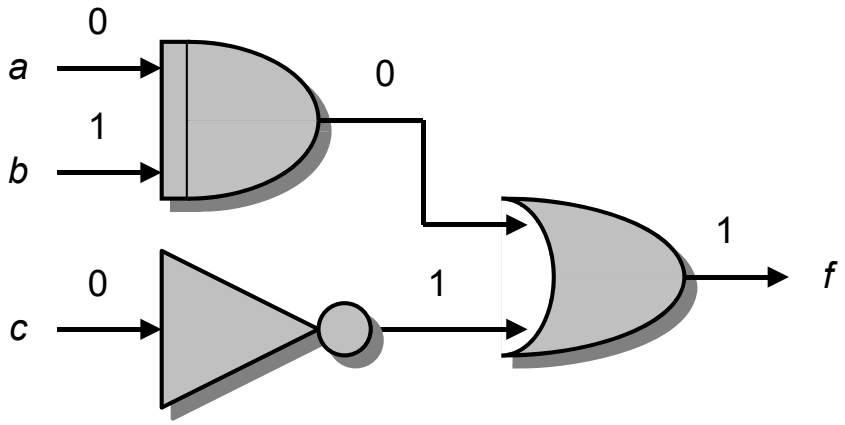
(corrisponde alla riga 1 della tabella)



risultato della simulazione: $f(0, 0, 1) = 0$

Rete ⇒ Tabella di Verità (3/4)

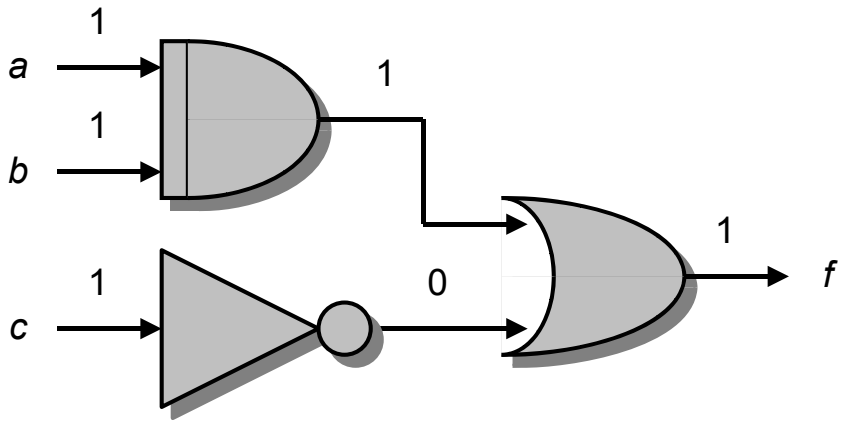
(corrisponde alla riga 2 della tabella)



risultato della simulazione: $f(0, 1, 0) = 1$

Rete ⇒ Tabella di Verità (4/4)

(corrisponde alla riga 7 della tabella)



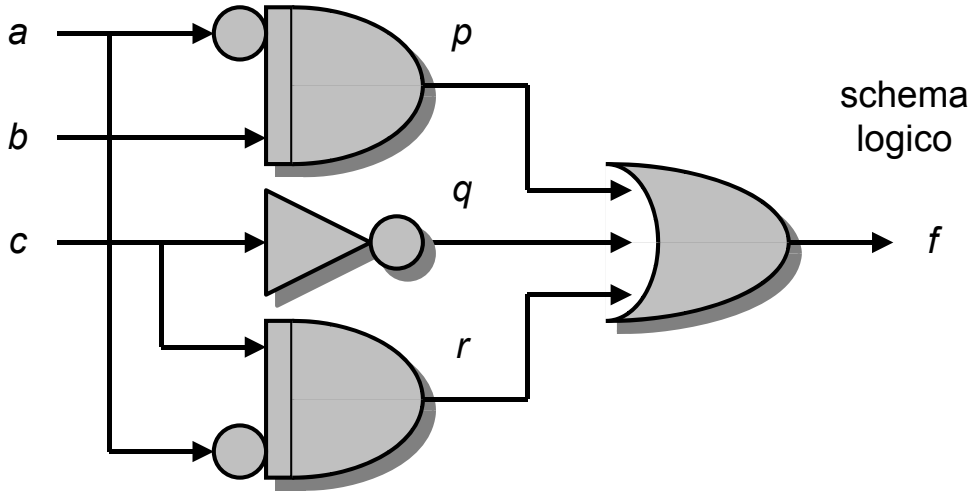
risultato della simulazione: $f(1, 1, 1) = 1$

Analisi formale

- Si può descrivere la rete combinatoria dandone l'espressione booleana corrispondente.
- L'espressione booleana è strutturalmente legata alla rete combinatoria cui corrisponde.
- Reti combinatorie strutturalmente diverse hanno espressioni booleane differenti.
- Reti combinatorie strutturalmente diverse ma equivalenti, hanno espressioni booleane differenti ma trasformabili l'una nell'altra.
- Si ricava l'espressione booleana della rete esaminando formalmente come si propagano i segnali interni alla rete stessa.

Rete ⇒ Espressione (1/3)

1. Data una rete si applicano i nomi ai segnali interni: *p*, q e *r*



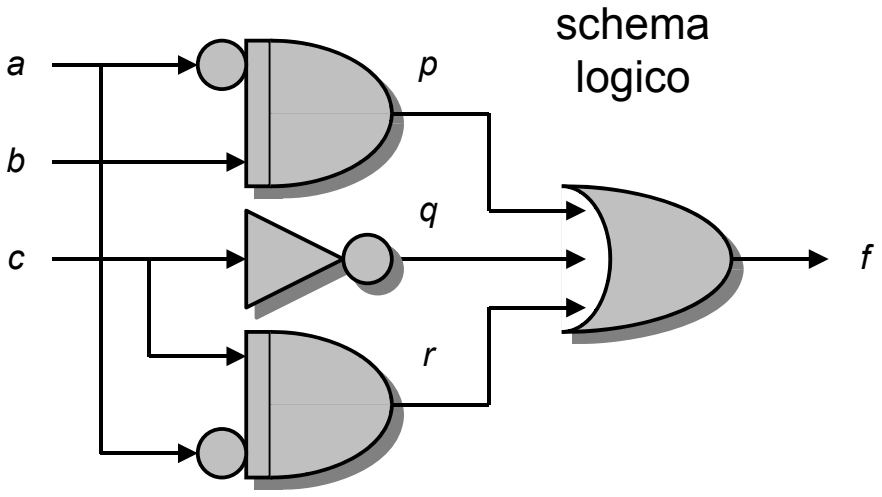
Rete ⇒ Espressione (2/3)

2. Si ricavano le espressioni booleane dei segnali interni

$p = /a\ b$

$q = /c$

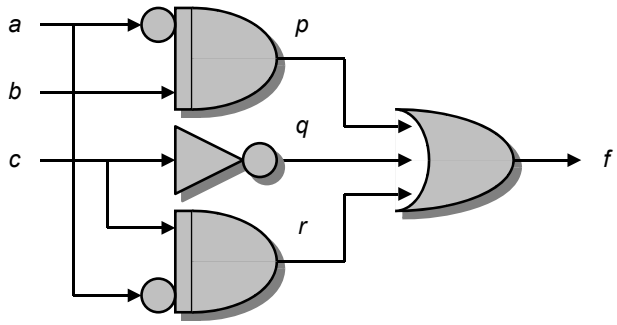
$r = /a\ c$



Rete ⇒ Espressione (3/3)

3. Si ricava l'uscita come espressione booleana dei segnali interni:
   $f = p + q + r$
4. Quindi, per sostituzione, si ricava l'uscita come espressione booleana:
   $f = p + q + r$
   $f(a, b, c) = /a\ b + /c + /a\ c$

l'espressione booleana così trovata ha una struttura conforme allo schema logico di partenza



Espressione ⇒ Tabella di Verità

# riga	a	b	c	/a b + /c + /a c	f
0	0	0	0	/0 0 + /0 + /0 0	1
1	0	0	1	/0 0 + /1 + /0 1	1
2	0	1	0	/0 1 + /0 + /0 0	1
3	0	1	1	/0 1 + /1 + /0 1	1
4	1	0	0	/1 0 + /0 + /1 0	1
5	1	0	1	/1 0 + /1 + /1 1	0
6	1	1	0	/1 1 + /0 + /1 0	1
7	1	1	1	/1 1 + /1 + /1 1	0

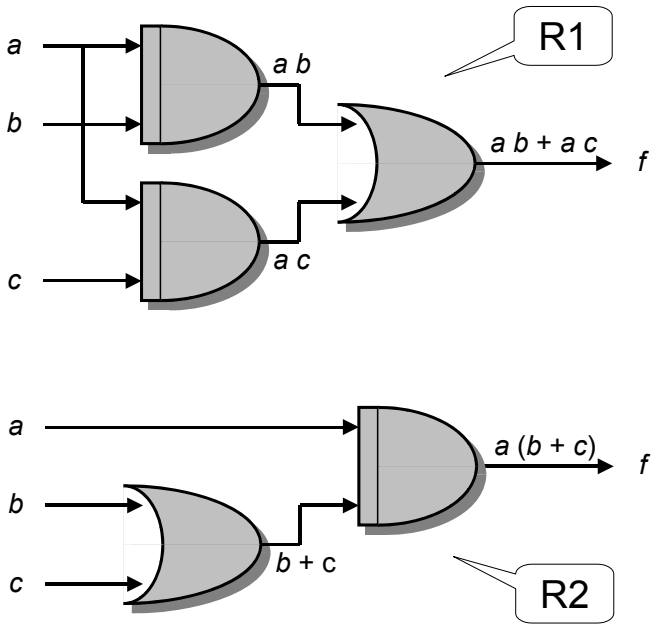
Equivalenza tra Reti Combinatorie

- Una funzione combinatoria può ammettere più reti combinatorie differenti che la sintetizzano.
- Reti combinatorie che realizzano la medesima funzione combinatoria si dicono equivalenti.

- Esse hanno tutte la stessa funzione (tabella di verità), ma possono avere struttura (e costo) differente.

- Per vedere se due reti combinatorie siano equivalenti, basta confrontarne le tabelle di verità, oppure cercare di trasformare l'una nell'altra.

Esempio di Reti Equivalenti



$f_1 = a\ b + a\ c$

$f_2 = a\ (b + c)$

Trasformazione:

$f_1 = a\ b + a\ c =$

$= a\ (b + c) =$

$= f_2$

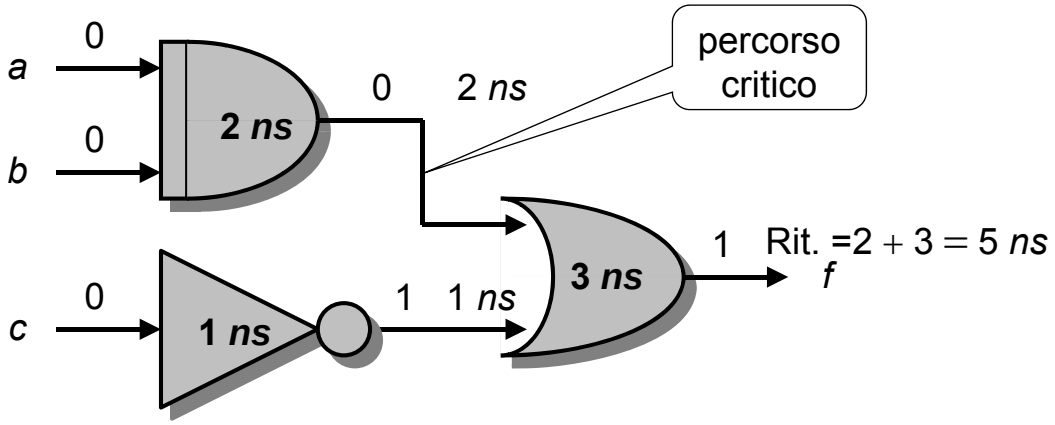
(proprietà distributiva)

Velocità e costo di una rete combinatoria

- La velocità di una rete combinatoria è misurata dal tempo che una variazione di ingresso impiega per modificare l'uscita (ritardo di propagazione):
  - per calcolare la velocità di una rete combinatoria, occorre conoscere i ritardi di propagazione delle porte componenti la rete, e poi esaminare i percorsi ingresso-uscita.

- Il costo di una rete combinatoria si valuta in vari modi (criteri di costo):
  - numero di porte, per tipo di porta e per quantità di ingressi
  - numero di porte universali (NAND o NOR)
  - numero di transistori
  - complessità delle interconnessioni
  - e altri ancora …

Ritardo e Frequenza



ritardo totale = 5 ns = 5 * 10^{-9} s

frequenza di commutazione = 1 / 5 ns = 200 MHz

Criterio dei Letterali

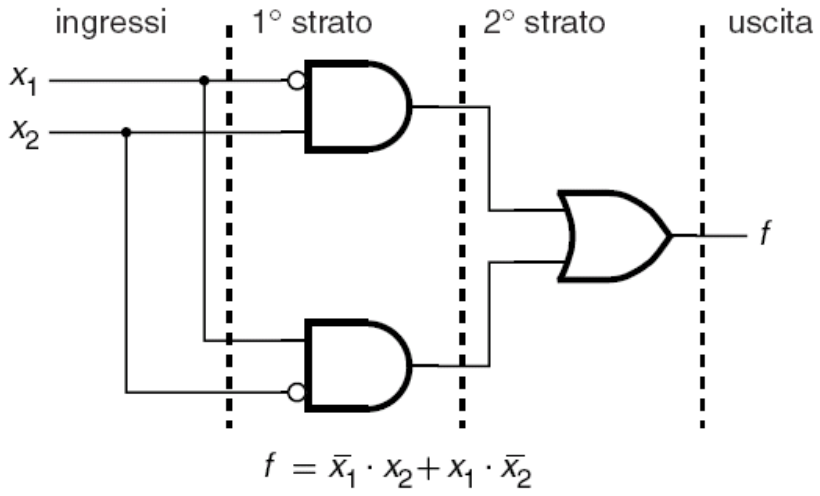
- Il **criterio di costo** più comune è quello cosiddetto **dei letterali** (o degli ingressi).
- Vale *unicamente per le reti combinatorie a due livelli*, cioè con due soli strati di porte logiche.

- Rete **Somma di Prodotti** (*Sum-of-Products*, SoP, SP):
  - porte AND seguite da una porta OR
- Rete **Prodotto di Somme** (*Product-of-Sums*, PoS, PS):
  - porte OR seguite da una porta AND

- Il *costo della rete è il numero di comparse di variabili* che figurano nell'espressione booleana della rete.
- Non si fa distinzione tra variabili dirette e negate.

Costo in Letterali

rete combinatoria a due livelli di tipo SP



costo della rete: 4 letterali

Forme canoniche

SINTESI DI RETI COMBINATORIE

Forma canonica

- Data una funzione booleana, la soluzione iniziale al problema di determinarne un'espressione booleana equivalente, consiste nel ricorso alle forme canoniche.
- Le forme canoniche sono la forma **somma di prodotti** (SP) (*sintesi in 1ª forma canonica*) e quella **prodotto di somme** (PS) (*sintesi in 2ª forma canonica*), rispettivamente.
- Data una funzione booleana esistono una e una sola forma canonica SP, e una e una sola forma canonica PS, che la rappresentano.

Prima forma canonica

- Data una funzione ***f*** delle *n* variabili $(x_1, x_2, \ldots, x_n)$, la **prima forma canonica** si ottiene come ***somma di un numero di termini pari al numero di righe in cui la funzione vale 1***,
  ciascuno dei quali è costituito dal prodotto di tutte le variabili $(x_1, x_2, \ldots x_n)$;
  nel prodotto, ciascuna variabile appare in forma diretta o complementata a seconda del fatto che, sulla corrispondente riga su cui la funzione vale 1, la variabile valga 1 o 0.
- Un tale termine è detto *prodotto fondamentale* (*o mintermine*).

Prima forma canonica (def. alternativa)

- Data una tabella di verità a $n \geq 1$ ingressi della funzione da sintetizzare, la funzione *f* che la realizza può essere specificata come:
  - la *somma logica* (OR) di *tutti* (e soli) i termini *prodotto* (AND) delle variabili di ingresso *corrispondenti agli 1 della funzione*
  - ogni termine prodotto (o *mintermine*) è costituito dal *prodotto logico delle variabili di ingresso* (*letterali*) prese:
    - in forma naturale se valgono 1
    - in forma complementata se valgono 0

Esempio 1 (1/3)

- Considera l'esempio seguente:

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

- È intuitivo osservare come la funzione possa essere ottenuta dallo OR delle funzioni f_1 e f_2 seguenti:

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

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	0
1 1	0

+

a b	f_2(a,b)
0 0	0
0 1	0
1 0	0
1 1	1

Esempio 1 (2/3)

- Per cui, intuitivamente, si ottiene:

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

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	0
1 1	0

+

a b	f_2(a,b)
0 0	0
0 1	0
1 0	0
1 1	1

$f_1(a,b) = /ab$ $f_2(a,b) = ab$

- Infatti, quando $a = 0$ e $b = 1$ il prodotto $/a\ b$ assume valore 1, mentre vale 0 in tutti gli altri casi.

Esempio 1 (3/3)

- Ne consegue:

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

=

a b	$f_1(a,b)$
0 0	0
0 1	1
1 0	0
1 1	0

+

a b	$f_2(a,b)$
0 0	0
0 1	0
1 0	0
1 1	1

f(a,b) = /ab + ab

- Mettendo in OR i mintermini della funzione si ottiene un'espressione booleana della funzione stessa espressa come somma di prodotti:
  - nel mintermine (prodotto) una variabile compare nella forma naturale (*x*) se nella configurazione di ingresso ha valore 1, nella forma complementata (!*x*) se ha valore 0.

Esempio 2 (1/2)

- Sia data una funzione in forma tabellare.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Da questa *tabella di verità* è sempre possibile giungere alle forme canoniche.
  - Si considerino tutte le righe in cui z=1
  - Si consideri il **prodotto delle varabili che danno z=1** (detto *prodotto fondamentale* o ***mintermine***)
  - Le variabili possono essere in forma diretta o complementata.
  - **Le somme di questi prodotti** costituiscono la ***prima*** *forma canonica o forma* ***SP*** *– Somme di Prodotti.*

Esempio 2 (2/2)

$\mathbf{f(x_1,x_2,x_3)} =$
$/x_1/x_2/x_3 + /x_1x_2x_3 + x_1x_2/x_3 + x_1x_2x_3$

Somma di prodotti fondamentali o mintermini

Oppure in modo sintetico:

$\mathbf{y = \sum_3(0,3,6,7)}$

si dice che y, funzione di 3 variabili, vale 1 per le configurazioni di peso 0,3,6,7.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Esercizio: funzione *maggioranza*

- Si chiede di sintetizzare (in 1ª forma canonica) una funzione combinatoria dotata di 3 ingressi *a*, *b* e *c*, e di un'uscita *f*, funzionante come segue:
  - se la maggioranza (stretta) degli ingressi vale 0, l'uscita vale 0,
  - se la maggioranza (stretta) degli ingressi vale 1, l'uscita vale 1.

Funzione maggioranza: tabella di Verità

l'uscita vale 1 se e solo se 2 o tutti e 3 gli ingressi valgono 1 (cioè se e solo se il valore 1 è in maggioranza)

# riga	*a*	*b*	*c*	*f*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

mintermini

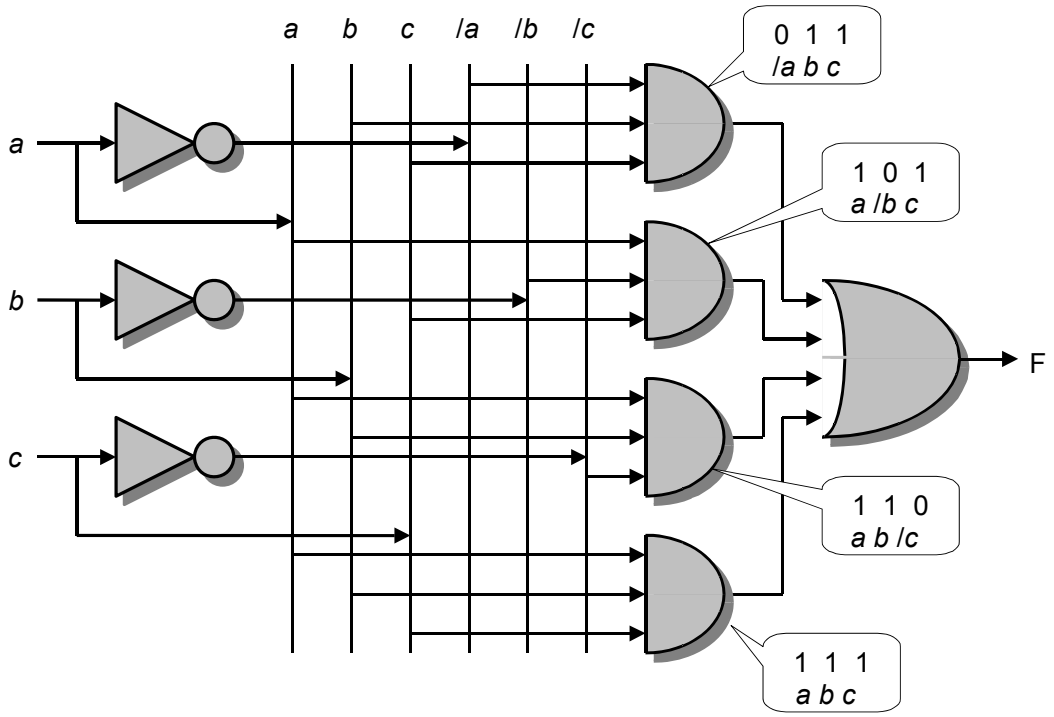
Funzione maggioranza: espress. booleana

- Dalla tabella di verità si ricava, tramite la sintesi in 1ª forma canonica (somma di prodotti), l'espressione booleana che rappresenta la funzione:

$$f(a, b, c) = /a\ b\ c + a\ /b\ c + a\ b\ /c + a\ b\ c$$

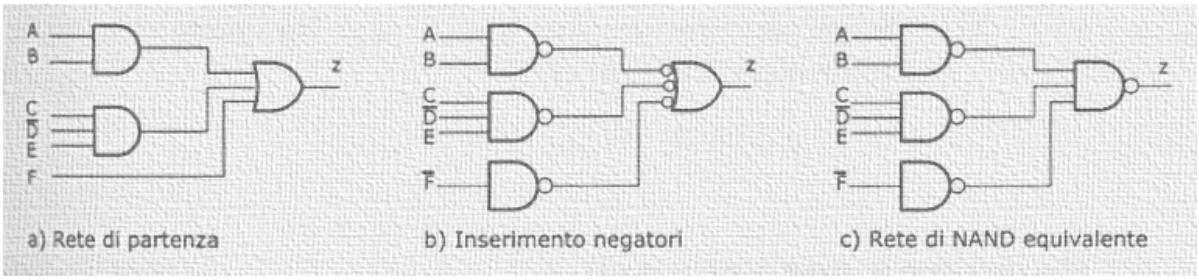
- Dall'espressione booleana si ricava la rete combinatoria (il circuito), che è sempre a 2 livelli.

Funzione maggioranza: schema logico



Passaggio da rete SP a rete NAND

- Si inseriscono sui rami che collegano le uscite delle porte AND con gli ingressi della porta OR, coppie di NOT.
- Si sostituisce il simbolo della porta di uscita con quello del NAND.



Seconda forma canonica

- Data una funzione ***f*** delle *n* variabili $(x_1, x_2, \ldots, x_n)$, la **seconda forma canonica** si ottiene **come *prodotto di un numero di termini pari al numero di righe in cui la funzione vale 0***, ciascuno dei quali è costituito dalla somma di tutte le variabili $(x_1, x_2, \ldots x_n)$, in forma diretta o complementata a seconda del fatto che, sulla corrispondente riga su cui la funzione vale 0, la variabile valga 0 oppure 1.

- I termini della somma vengono detti *somme fondamentali.*

Seconda Forma Canonica (def. alternativa)

- Data una tabella di verità, a $n \geq 1$ ingressi, della funzione da sintetizzare, la funzione *f* che la realizza può essere specificata come:
  - il prodotto logico (AND) di tutti (e soli) i termini somma (OR) delle variabili di ingresso corrispondenti agli 0 della funzione
  - ogni termine somma (o *maxtermine*) è costituito dalla somma logica delle variabili di ingresso (letterali) prese:
    - in forma naturale se valgono 0
    - in forma complementata se valgono 1

Esempio 1 (1/3)

- Si consideri nuovamente lo stesso esempio:

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

- È intuitivo osservare come la funzione possa essere ottenuta dallo AND delle funzioni f_1 e f_2 seguenti:

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

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	1
1 1	1

*

a b	f_2(a,b)
0 0	1
0 1	1
1 0	0
1 1	1

Esempio 1 (2/3)

- Per cui, intuitivamente, si ottiene:

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

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	1
1 1	1

*

a b	f_2(a,b)
0 0	1
0 1	1
1 0	0
1 1	1

$f_1(a,b) = /(/a/b)$ $f_2(a,b) = /(a/b)$

  - infatti, per esempio, quando *a* = 0 e *b* = 0 il termine /(/a/b) assume valore 0, mentre vale 1 in tutti gli altri casi

  - o anche:

$f_1(a, b) = a + b$ $f_2(a, b) = /a + b$

Esempio 1 (3/3)

- Applicando le leggi di De Morgan, si ottiene le trasformazione seguente:

f_1(a,b)=/(/a/b) f_2(a,b)=/(a!b)

f_1(a,b)=(a + b) f_2(a,b)=(/a + b)

f(a,b) = (a + b)*(/a + b)

- Mettendo in AND i maxtermini della funzione si ottiene un'espressione booleana della funzione stessa espressa come prodotto di somme:
  - nel maxtermine (somma) una variabile compare nella forma naturale (*x*) se nella configurazione di ingresso ha valore 0, nella forma complementata (/*x*) se ha valore 1.

Esempio 2: forma PS da tabella di Verità

maxtermini

# riga	*a*	*b*	*c*	*f*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$f = (a + b + c)\ (a + b + !c)\ (a + /b + c)\ (/a + b + c)$

Esempio 3 (1/2)

#	x_1	x_2	x_3	termine	f_1	termine	f_2
0	0	0	0	m_0	1	m_0	1
1	0	0	1	m_1	1	m_1	1
2	0	1	0	M_2	0	m_2	1
3	0	1	1	m_3	1	M_3	0
4	1	0	0	M_4	0	m_4	1
5	1	0	1	M_5	0	m_5	1
6	1	1	0	M_6	0	M_6	0
7	1	1	1	m_7	1	M_7	0

m_i: i^{esimo} mintermine

M_i: i^{esimo} maxtermine

Esempio 3 (2/2)

$$f_1 = m_0 + m_1 + m_3 + m_7$$

SP

rappresentazione compatta

$$f_2 = m_0 + m_1 + m_2 + m_4 + m_5 \qquad f_2 = \sum_1 \underbrace{(0,1,2,4,5)}_{\text{mintermini}}$$

$$f_1 = M_2 \cdot M_4 \cdot M_5 \cdot M_6$$

PS

rappresentazione compatta

$$f_2 = M_3 \cdot M_6 \cdot M_7 \qquad f_2 = \prod_0 \underbrace{(3,6,7)}_{\text{maxtermini}}$$

Esempio 4 (1/2)

- Sia data una funzione in forma tabellare.
- Da questa *tabella di verità* è sempre possibile giungere alle forme canoniche.
  - Si considerino tutte le righe in cui z=0
  - Si consideri **la somma delle varabili che danno z=0** (detta somma *fondamentale* o ***maxtermine***)
  - Le variabili possono essere in forma diretta o complementata.
  - **I prodotti di queste somme** costituiscono la ***seconda*** *forma canonica o forma* ***PS*** *– Prodotti di Somme.*

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Esempio 4 (2/2)

$f(x_1,x_2,x_3)$ =
$(x_1+x_2+/x_3)*(x_1+/x_2+x_3)*(/x_1+x_2+/x_3)*(/x_1+x_2+/x_3)$

Prodotto di somme fondamentali o maxtermini

Oppure in modo sintetico:

$y = \prod_3(1,2,4,5)$

si dice che y, funzione di 3 variabili, vale 0 per le configurazioni di peso 1,2,4,5.

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Forme canoniche: reti a 2 livelli

- Le forme canoniche danno luogo a reti a due livelli:
  - SP: 1° liv. AND, 2° liv. OR
  - PS: 1° liv. OR, 2° liv. AND

- Nota: al primo livello sono sempre disponibili gli ingressi anche in forma complementata.

Dualità delle forme canoniche

- Le forme SP e PS sono duali tra loro e pertanto è sempre possibile passare da una forma all'altra
- Esempio: $y = \sum_3(0,3,6,7) = \prod_3(1,2,4,5)$

  Consideriamo quando /y = 1

→ $/y = /x_1/x_2x_3 + /x_1x_2/x_3 + x_1/x_2/x_3 + x_1/x_2x_3$

→ $//y = /(/x_1/x_2x_3 + /x_1x_2/x_3 + x_1/x_2/x_3 + x_1/x_2x_3)$

→ Applico De Morgan alla somma, ossia /(x + y) = /x * /y :

→ $y = /(/x_1/x_2x_3) * /(/x_1x_2/x_3) * /(x_1/x_2/x_3) * /(x_1/x_2x_3)$

→ Applico De Morgan ai singoli prodotti, ossia /(x * y) = /x + /y :

→ $y = (//x_1+//x_2+/x_3)*(//x_1+/x_2+//x_3)*(/x_1+//x_2+//x_3)*(/x_1+//x_2+/x_3)$

→ $y = (x_1+x_2+/x_3) * (x_1+/x_2+x_3) * (/x_1+x_2+x_3) * (/x_1+x_2+x_3)$

forma minima
metodo di Karnaugh

MINIMIZZAZIONE DI RETI COMBINATORIE

Costo in Letterali

- Sintesi in 1ª forma canonica: la funzione logica ottenuta è costituita dagli elementi seguenti:
  - n° di mintermini pari agli 1 della funzione
  - per ogni mintermine, n° di letterali pari al numero delle variabili di ingresso
  - dualmente per la 2ª forma canonica
- Il costo della rete è funzione del numero totale di letterali, cioè del n° di mintermini e del numero di letterali costituenti ciascun mintermine.

Riduzione Algebrica

- L'uso delle proprietà e dei teoremi dell'algebra di commutazione consente (talvolta) di semplificare l'espressione logica:
  - ridurre sia il n° di mintermini sia il numero di letterali
- La semplificazione tramite le proprietà e i teoremi dell'algebra:
  - non è esprimibile tramite un algoritmo di riduzione
  - non garantisce di arrivare alla forma minima dell'espressione, cioè a quella costituita dal minimo numero di mintermini, ciascuno con il minimo numero di letterali (sempre espressa come forma di prodotti)

Esempio

costo: 12

$$
\begin{aligned}
f_1 &= \overline{x}_1\overline{x}_2\overline{x}_3 + \overline{x}_1\overline{x}_2 x_3 + \overline{x}_1 x_2 x_3 + x_1 x_2 x_3 = \\
&= \overline{x}_1\overline{x}_2\left(\overline{x}_3 + x_3\right) + \left(\overline{x}_1 + x_1\right) x_2 x_3 = \\
&= \overline{x}_1\overline{x}_2 \cdot 1 + 1 \cdot x_2 x_3 = \\
&= \overline{x}_1\overline{x}_2 + x_2 x_3 \qquad \text{(forma minima)}
\end{aligned}
$$

costo: 4

qui basta effettuare un doppio raccoglimento a fattore comune (si usano la proprietà distributiva, di complemento e dello 1)

Considerazioni: mintermini adiacenti

- Dati due mintermini che differiscono per un solo letterale (che compare dunque in forma naturale e in forma complementata).
- Applicando le proprietà distributiva, di complemento e dello 1, si ottiene un solo termine prodotto costituito da tutti e soli i letterali identici nei due mintermini:

- /x /y z + /x y z = /x z (/y + y) = /x z 1 = /x z

- I due mintermini si dicono logicamente **adiacenti**.

Espressioni adiacenti e minimizzazione

- Facciamo riferimento alle sole forme SP (più semplici).
- Se supponga di avere l'espressione:

  A/a + Aa

  ove: A è un qualsiasi prodotto di termini, e
  a è una variabile booleana.

  Allora: A/a + Aa = A(/a+a) = A*1 = A
- Si dice che le espressioni A/a e Aa sono ***adiacenti*** perché le configurazioni delle variabili che le individuano hanno forma unitaria ossia *differiscono solo per la variabile che appare sia in forma diretta che in forma complementata*.
- La minimizzazione delle funzioni SP consiste nell'individuazione delle configurazioni adiacenti e nell'applicazione sistematica delle precedente proprietà.

Considerazioni: 2 mintermini adiacenti

- Dati 2 ($= 2^1$) mintermini logicamente adiacenti è possibile sostituirli con un unico termine prodotto dove è stato eliminato un letterale, qualunque sia il n° di letterali (≥ 1) che li compongono originariamente.
- Esempio:

$$/x\ /y\ z + /x\ y\ z =$$
$$= /x\ z\ (/y + y) =$$
$$= /x\ z\ 1 =$$
$$= /x\ z$$

Considerazioni: 4 mintermini adiacenti

- Dati 4 ($= 2^2$) mintermini logicamente adiacenti è possibile sostituirli con un solo termine prodotto dove sono stati eliminati due letterali, qualunque sia il n° di letterali (≥ 2) che li compongono originariamente:

$$/x\ /y\ z + /x\ y\ z + /x\ /y\ /z + /x\ y\ /z =$$
$$= /x\ z\ (/y + y) + /x\ /z\ (/y + y) =$$
$$= /x\ z\ 1 + /x\ /z\ 1 =$$
$$= /x\ (z + /z) =$$
$$= /x$$

Mappe di Karnaugh

- Le tabelle di verità non permettono di individuare i termini adiacenti

  ⇒ servono le ***mappe di Karnaugh***.
- Mappe di ordine 1, 2 e 3.

x	0	1
	f(0)	f(1)

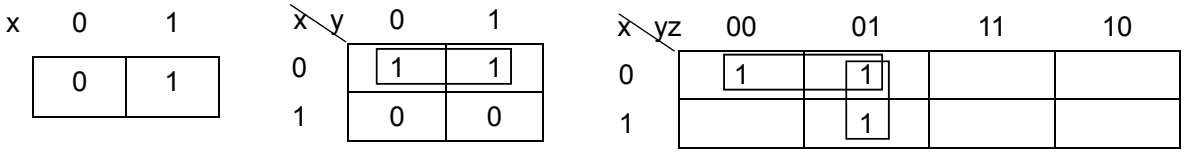
x \ y	0	1
0	f(0,0)	f(0,1)
1	f(1,0)	f(1,1)

x \ y z	00	01	11	10
0	f(0,0,0)	f(0,0,1)	f(0,1,1)	f(0,1,0)
1	f(1,0,0)	f(1,0,1)	f(1,1,1)	f(1,1,0)

Generalità sulle mappe di Karnaugh

- Una funzione di *n* variabili $f(x_1, \ldots, x_n)$ viene rappresentata su una mappa di Karnaugh di ordine *n*;
- *una mappa di ordine n contiene 2^n celle*;
- le coordinate delle 2^n celle corrispondono alle 2^n possibili configurazione delle *n* variabili;
- sulla mappa di ordine *n* le celle sono disposte in modo tale che ogni cella è adiacente a *n* celle;
- su una mappa di ordine *n* si definisce **sottocubo di ordine m** con *$m \leq n$ un insieme di 2^m celle tale per cui ciascuna casella del sottocubo è adiacente a m caselle del sottocubo stesso*;
- *se su tutte le celle di un sottocubo di ordine m la funzione vale 1*, **il contributo complessivo delle 2^m celle è dato dal prodotto delle (*n* - *m*) variabili che non variano nel sottocubo**. Nel prodotto la variabile compare in forma diretta se nel sottocubo vale 1, in forma complementata se nel sottocubo vale 0.

Esempi di sottocubi



- f(x)=x f(x,y)=/x f(x,y,z)=/x/y+/yz

Metodo di Karnaugh

- La mappa di Karnaugh è un metodo tabellare di rappresentazione della tabella di verità.
  - Sfrutta, nella rappresentazione, l'adiacenza logica tra mintermini.
  - Consente di derivare un algoritmo per semplificare in modo automatico l'espressione data come somma di prodotti (o prodotto di somme).

- **Algoritmo:**
  - ricerca gli implicanti
  - identifica gli implicanti primi
  - copri la funzione

Metodo di Karnaugh: implicante

- Si definisce **implicante** ***il prodotto di variabili corrispondente ad un sottocubo in cui la funzione vale 1***, ossia:
  - è un **sottogruppo di 2^n celle adiacenti dove la funzione assume valore 1**, indicizzate da configurazioni di ingresso di n variabili, tali che
    le n variabili assumono valore 0 e 1 in tutti i modi possibili (esattamente 2^n);
  - nel termine prodotto corrispondente è dunque possibile semplificare *n* letterali di ingresso.

Metodo di Karnaugh: implicante primo

- Un **implicante** si dice **primo** se ***corrisponde ad un sottocubo non completamente coperto da un altro sottocubo in cui la funzione è 1***, ossia:
  - è un implicante di dimensione massima, cioè non è contenuto in nessun altro implicante di dimensione superiore;
  - dunque genera il minimo numero di letterali nel termine prodotto corrispondente.

Metodo di Karnaugh: implicanti essenziali

- Gli *implicanti primi* che fanno *necessariamente parte della copertura minima* della funzione si dicono **essenziali.**

Metodo di Karnaugh: copertura

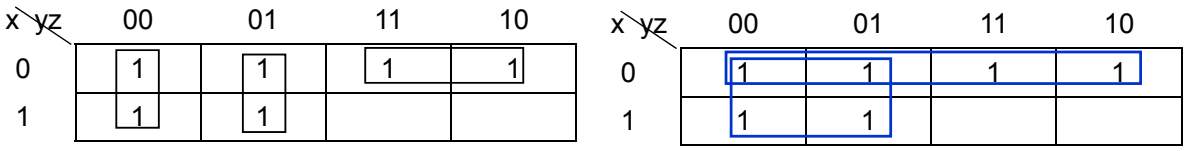
- Si definisce ***copertura*** della funzione **un insieme di sottocubi tale da *coprire* tutti gli 1 della funzione stessa**, ossia:
  - ricerca del numero minimo di implicanti primi che coprono tutti gli 1 della funzione (si genera così il numero minimo di termini prodotto);
  - gli implicanti primi possono anche essere parzialmente sovrapposti.

Metodo di Karnaugh: minimizzazione

- La ***minimizzazione*** consiste nel **trovare una copertura formata da un insieme di sottocubi, ciascuno dei quali sia il più ampio possibile e non sia contenuto in altri sottocubi**.

→**La minimizzazione di una funzione booleana in forma SP si risolve con la ricerca di un insieme di implicanti primi che coprono la funzione.**

Esempio di copertura e minimizzazione



- La funzione è coperta in entrambi i casi.
- La copertura di destra fornisce la minima espressione SP, in quanto formata da sottocubi più ampi.

Metodo di Karnaugh: esempio a 2 var.

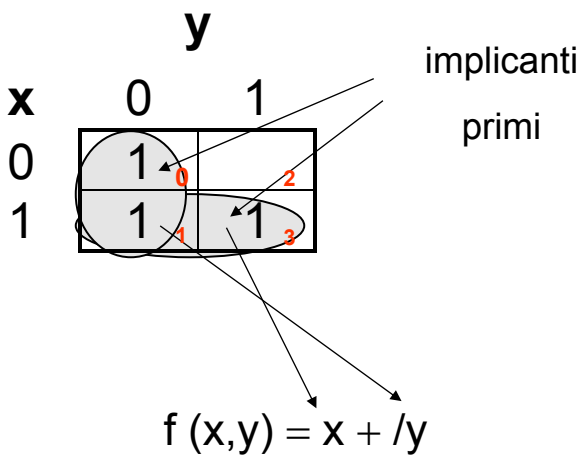
- Sia data la funzione seguente:

  f (x,y) = f (0, 2, 3) = /x /y + x /y + x y

m	x	y	f
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

tabella di verità

mappa di Karnaugh SP



f (x,y) = x + /y

Metodo di Karnaugh: esempio a 3 var. (1/3)

f (x,y,z) = f (0,4,6,7) = x y z + x y /z + x /y /z + /x /y /z

forma SP

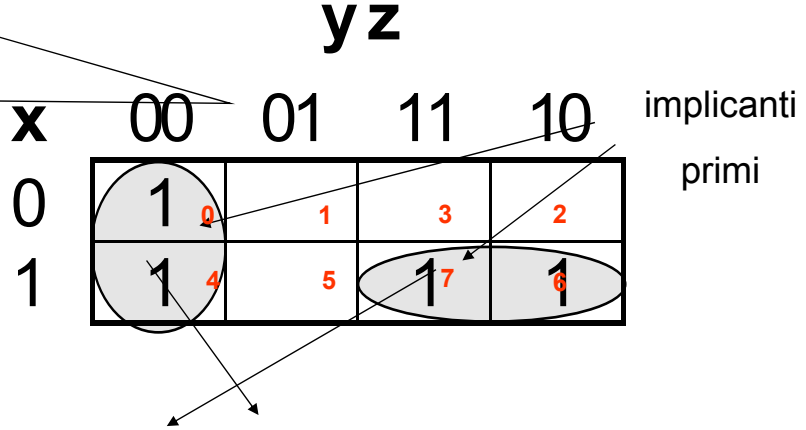
m	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

tabella di verità

Metodo di Karnaugh: esempio a 3 var. (2/3)

la disposizione delle configurazioni per colonna rispetta il criterio di adiacenza

mappa di Karnaugh SP



$$f(x,y,z) = x\,y + /y\,/z$$

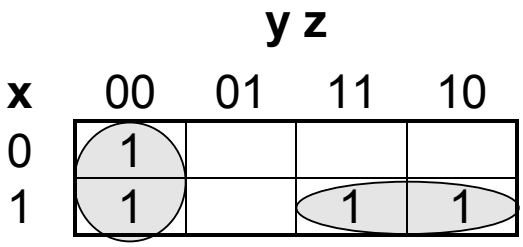
Metodo di Karnaugh: esempio a 3 var. (3/3)

$$f(x,y,z) = x\,y + x\,/z + /x\,/y\,/z$$

m	x	y	z	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

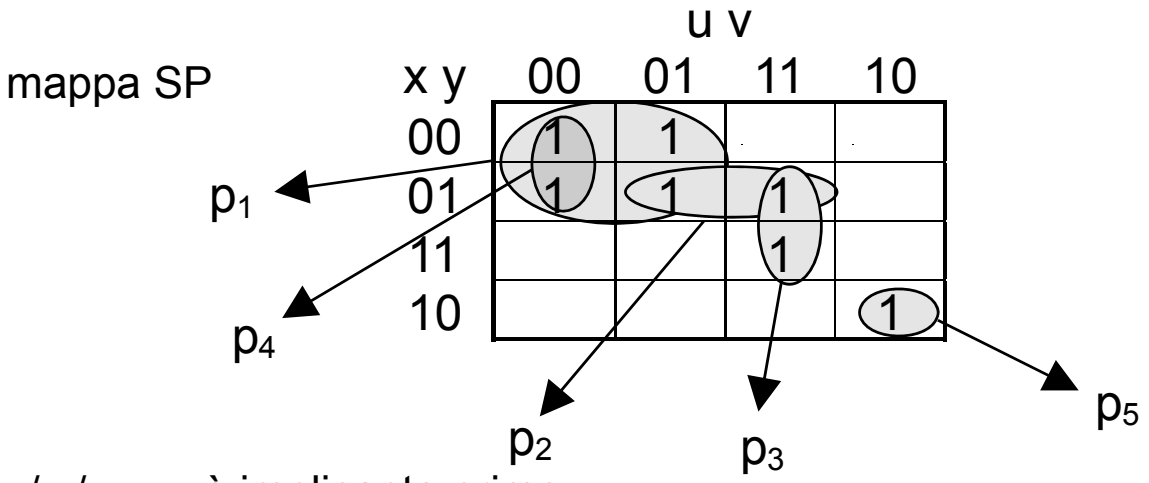
tabella di verità

forma SP



$$f(x,y,z) = x\,y + /y\,/z$$

Metodo di Karnaugh: esempio a 4 var.



$p_1 = /x\ /u$ è implicante primo

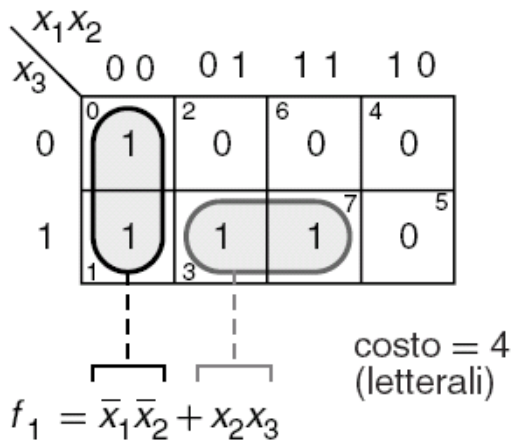
$p_2 = /x\ y\ v$ è implicante primo

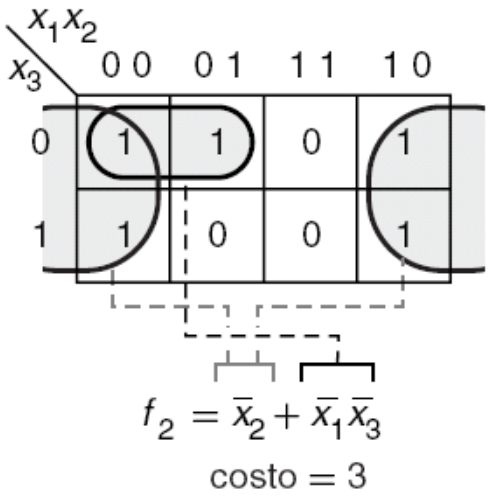
$p_3 = y\ u\ v$ è implicante primo

$p_4 = /x\ /u\ /v$ è implicante (ma non primo, perché è contenuto in p_1) → non va usato (giacché non è primo)

$p_5 = x\ /y\ u\ /v$ è implicante primo (ed è anche mintermine)

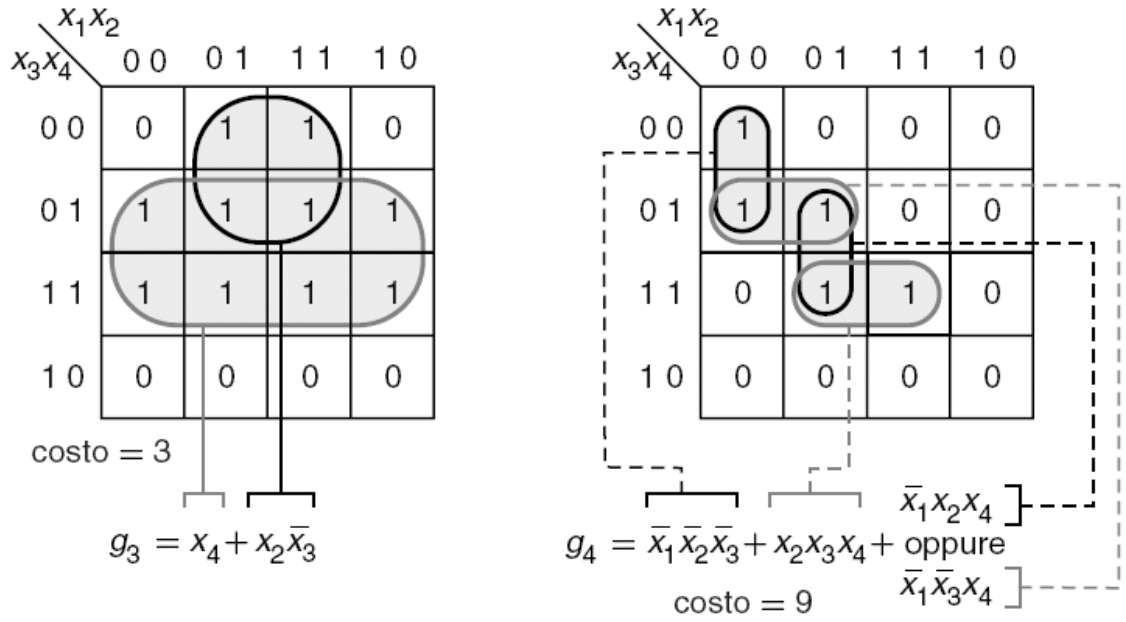
Altri esempi di mappa di Karnaugh a 3 var.





Costi delle forme minime

Mappa di Karnaugh a 4 var.



- Costi delle forme minime (non necessariamente uniche)

Forma Mimina PS

- Il metodo di Karnaugh si estende senza difficoltà per costruire la forma minima PS (prodotto di somme) della funzione.
- Sulla mappa, si cercano i gruppi di 0, invece che i gruppi di 1.
- Da ogni gruppo si deriva un *maxtermine*, invece che un *mintermine*, coprendo gli 0 della funzione.
- Nota bene: *data una funzione, le forme minime SP e PS potrebbero avere costo differente*;

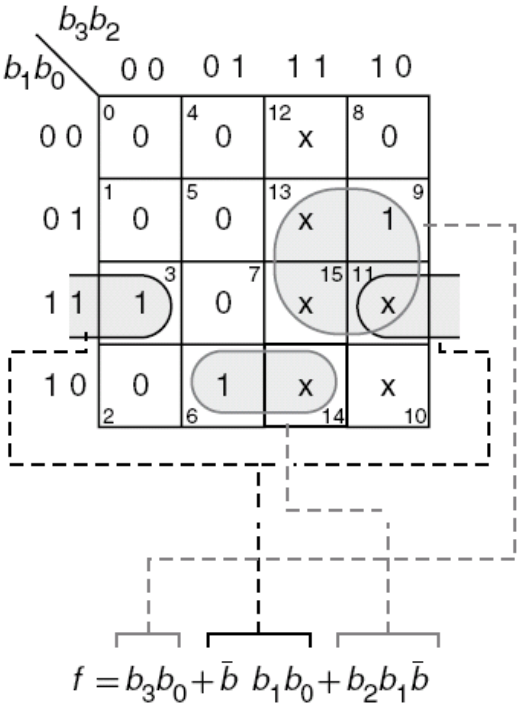
  $\Rightarrow$ in generale vanno ricavate entrambe, poi si sceglie la migliore.

Condizione di indifferenza

- Alcuni valori di uscita della funzione possono non essere specificati.
- Si chiamano *condizioni di indifferenza* e si indicano con il simbolo '–' oppure '×'.
- Ciascuna condizione di indifferenza può essere risolta a 0 o a 1, l'una in modo indipendente dall'altra, secondo come convenga per ottenere una forma minima di costo basso (SP o PS, a scelta).

Esempio - Cifra divisibile per 3

cifra decimale	#	codifica binaria b_3	b_2	b_1	b_0	f
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	1
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	1
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	1
non usate	10	1	0	1	0	x
	11	1	0	1	1	x
	12	1	1	0	0	x
	13	1	1	0	1	x
	14	1	1	1	0	x
	15	1	1	1	1	x



Metodi algoritmici

- Quando il numero della variabili supera 5 si usano metodi algoritmici (es. metodo di Quine-McCluskey)