Corso di Calcolatori Elettronici I
A.A. 2012-2013

Minimizzazione di circuiti combinatori

ing. Alessandro Cilardo

Accademia Aeronautica di Pozzuoli
Corso Pegaso V “GArn Elettronici”

Funzioni logiche elementari

Operazioni fondamentali sui valori booleani (bit)

Corrispondono ai *circuiti elementari* che possiamo usare come "mattoncini" per costruire circuiti più complessi

a	***b***	**y**
0	0	*0*
0	1	*0*
1	0	*0*
1	1	*1*

AND
Congiunzione (o prodotto)
(vale '1' solo se entrambe *a* e *b* sono '1')
a **AND** *b* si indica anche con $a \cdot b$

a	***b***	**y**
0	0	*0*
0	1	*1*
1	0	*1*
1	1	*1*

OR
Disgiunzione (o somma)
(vale '0' solo se entrambe *a* e *b* sono '0')
a **OR** *b* si indica anche con $a + b$

a	**y**
0	*1*
1	*0*

NOT
Negazione

NOT(*a*) si indica anche con $\bar{a}$

Funzioni logiche elementari

AND e **OR** generalizzate (a più di due ingressi)

a	***b***	***c***	**y**
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*

AND
Congiunzione (o prodotto)
(vale ‘1’ solo se **tutte** le variabili di ingresso sono ‘1’)
a **AND** *b* **AND** c si indica anche con $a \cdot b \cdot c$

a	***b***	***c***	**y**
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*

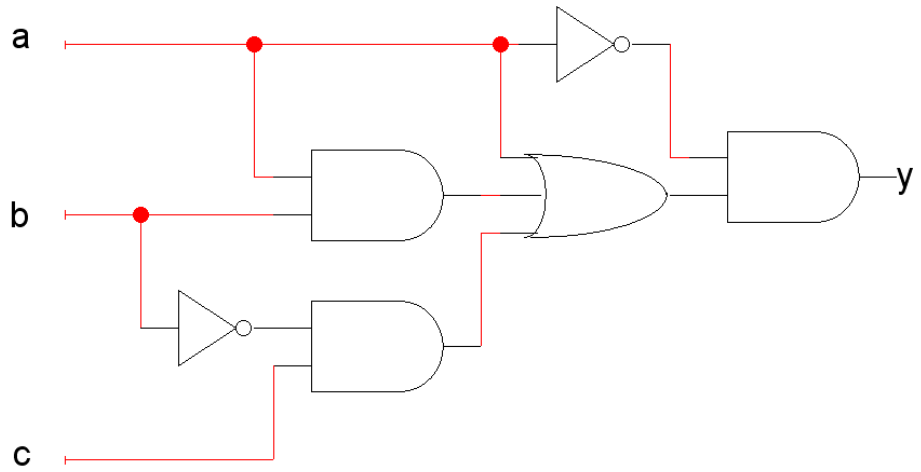
OR
Disgiunzione (o somma)
(vale ‘0’ solo se **tutte** le variabili di ingresso sono ‘0’)
a **OR** *b* **OR** *c* si indica anche con $a + b + c$

Progetto di reti logiche

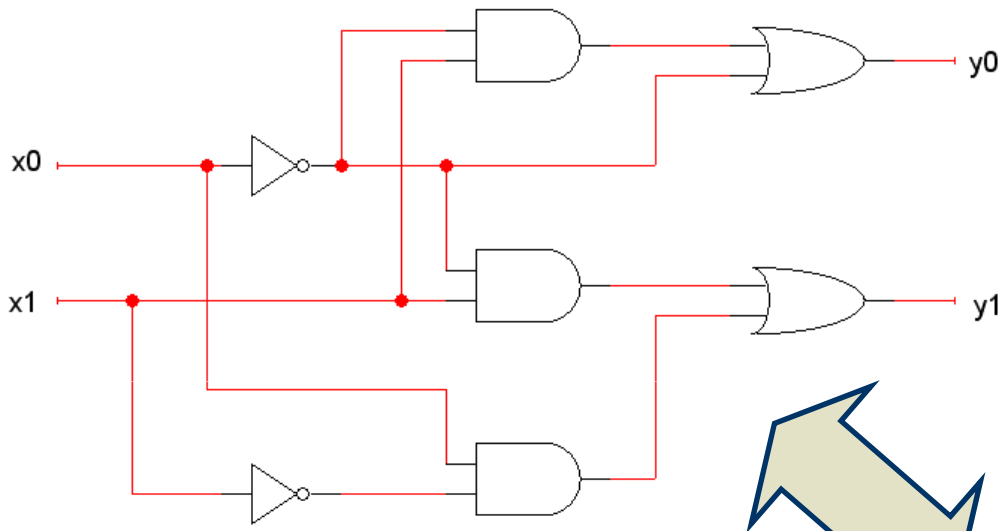
Una funzione logica generica è esprimibile attraverso la *tabella di verità*, che elenca i valori che le uscite devono avere per ciascuna possibile combinazione degli ingressi.

a	***b***	***c***	**y**
0	0	0	*0*
0	0	1	*1*
0	1	0	*0*
0	1	1	*0*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*0*

Una tabella di verità può essere realizzata dalla combinazione di circuiti (o porte) elementari

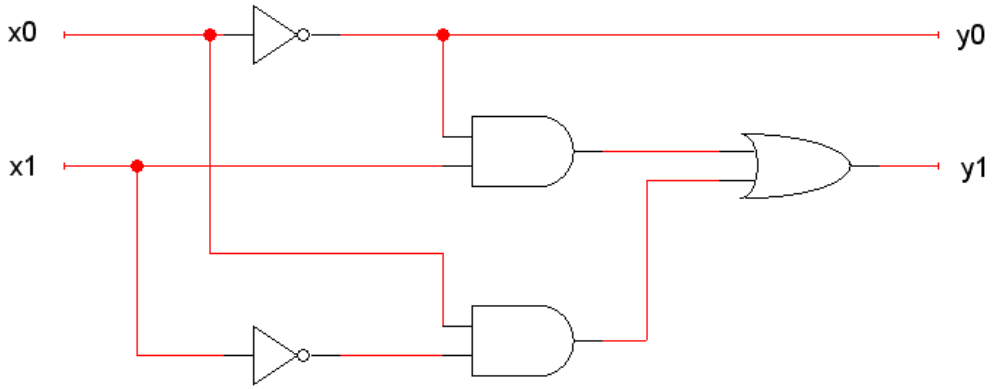


Equivalenza tra reti logiche



Una stessa funzione logica può essere realizzata da **diversi circuiti**

I due circuiti realizzano la stessa funzione, ma hanno costi differenti !



Progetto di reti logiche

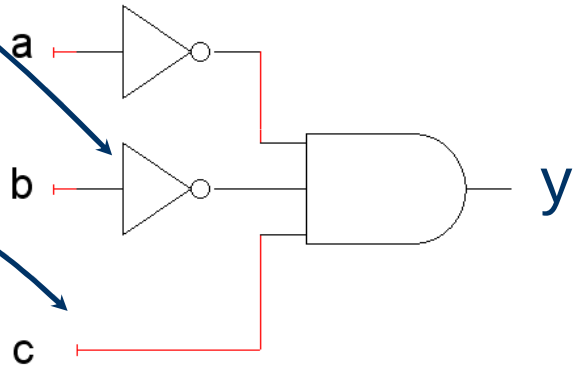
a	*b*	*c*	y
0	0	0	*0*
0	0	1	*1*
0	1	0	*0*
0	1	1	*0*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*0*

Come realizzare la corrispondenza tra una generica tabella e una sua possibile realizzazione circuitale?

Progetto di reti logiche

Se è presente un unico 1 nella tabella…

a	*b*	*c*	y
0	0	0	*0*
0	0	1	*1*
0	1	0	*0*
0	1	1	*0*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*0*



la tabella suggerisce che *y* deve essere 1 **solo quando** *a*=0, *b*=0 e *c*=1

→ usiamo la porta AND con delle NOT laddove gli ingressi devono essere 0

Progetto di reti logiche

a	*b*	*c*	y	
0	0	0	*0*	
0	0	1	*1*	⇦
0	1	0	*0*	
0	1	1	*0*	
1	0	0	*0*	
1	0	1	*1*	⇦
1	1	0	*1*	⇦
1	1	1	*1*	⇦

come realizzare una rete corrispondente alla tabella data se sono presenti diversi 1, come accade qui?

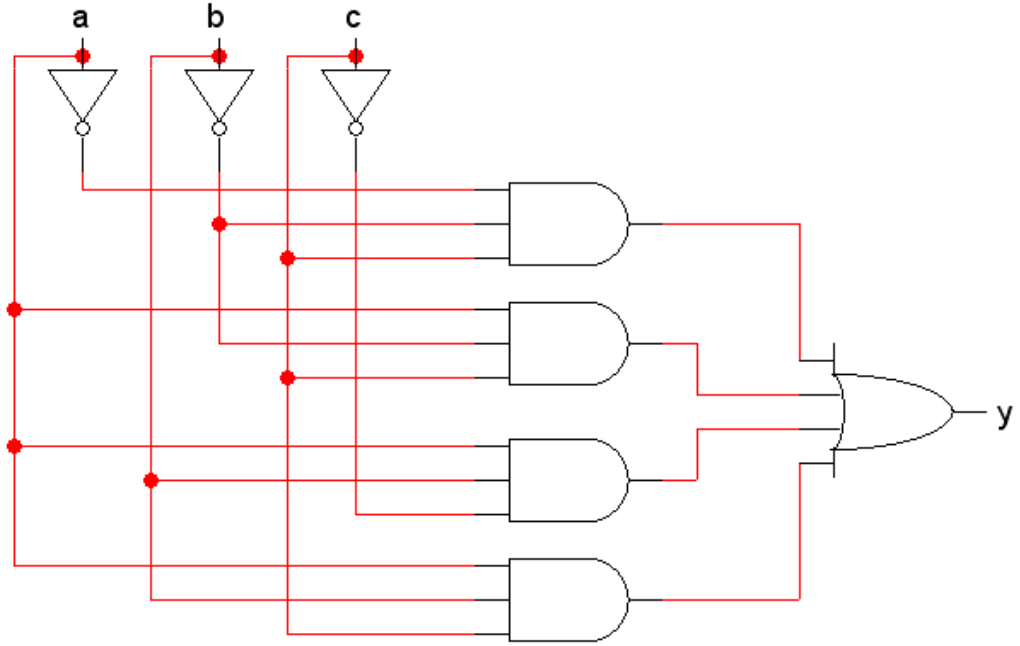
osservazione:
y deve essere alta quando
a=0 **e** *b*=0 **e** *c*=1, ***oppure***
a=1 **e** *b*=0 **e** *c*=1, ***oppure***
a=1 **e** *b*=1 **e** *c*=0, ***oppure***
a=1 **e** *b*=1 **e** *c*=1

usiamo tante AND per ogni **1** nella tabella ed una porta OR (che realizza l'***oppure***) con le AND messe in ingresso

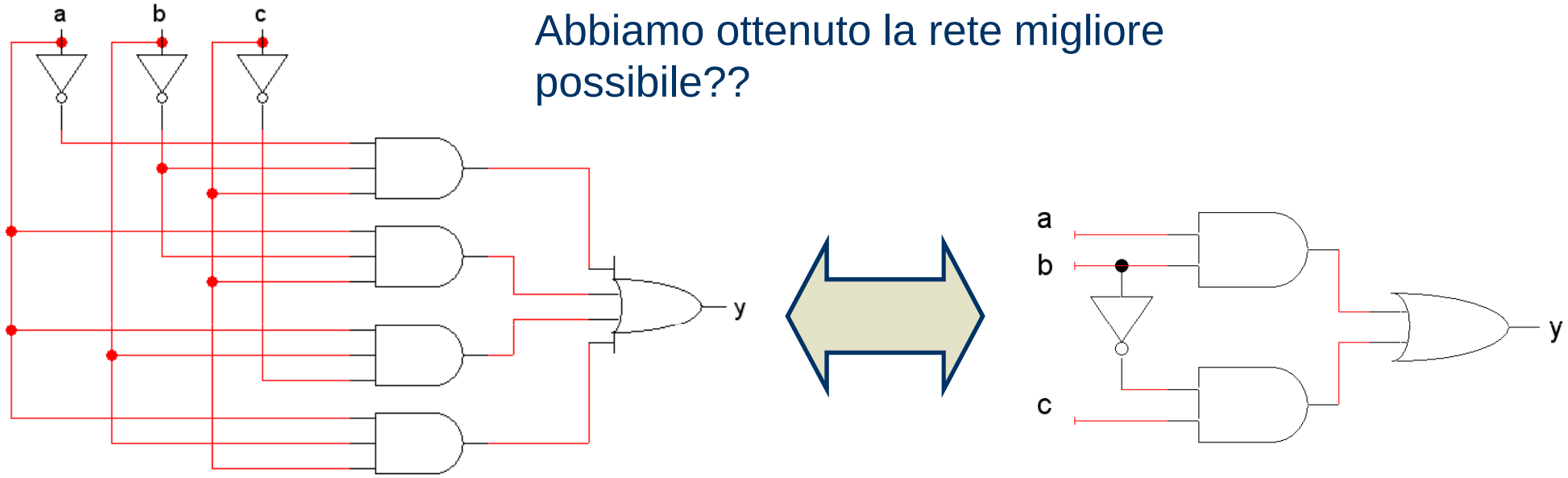
Progetto di reti logiche

a	*b*	*c*	y
0	0	0	*0*
0	0	1	*1*
0	1	0	*0*
0	1	1	*0*
1	0	0	*0*
1	0	1	*1*
1	1	0	*1*
1	1	1	*1*

Otteniamo la rete come struttura di tipo NOT-AND-OR (detta anche *somma di prodotti*)



Progetto di reti logiche



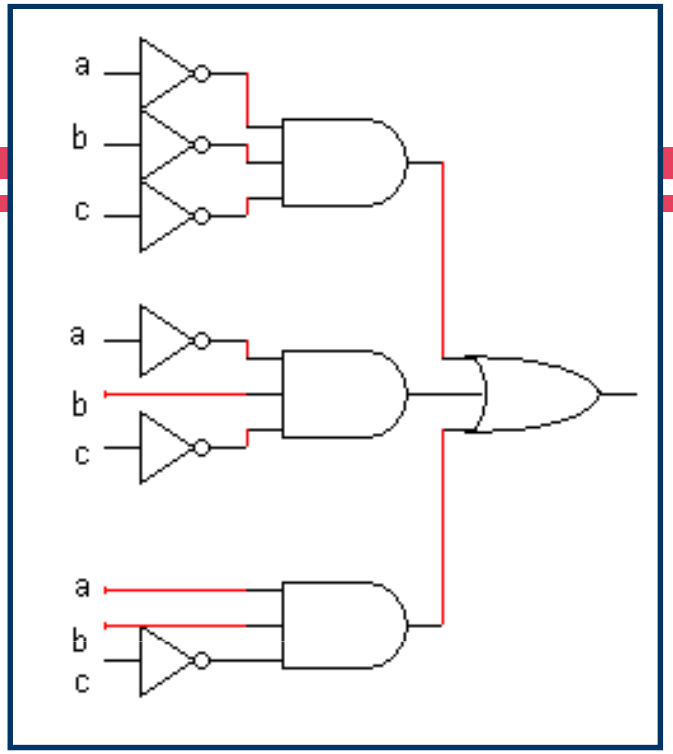
Abbiamo ottenuto la rete migliore possibile??

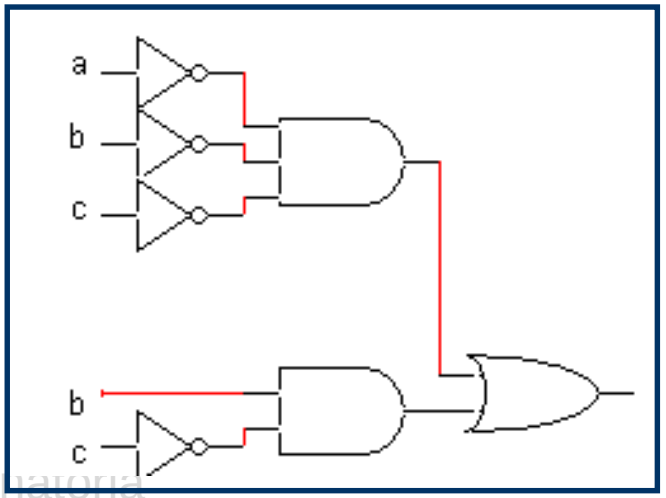
Osservazione: la rete sulla destra risponde agli ingressi nella stessa maniera di quella che abbiamo trovato (basta verificare il valore generato per ciascuna combinazione degli ingressi), ma è sicuramente più piccola e meno costosa!!

Sintesi di Reti Logiche

Altro esempio:

a	b	c	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0





I due circuiti realizzano la stessa funzione, ma hanno costi differenti !

Teoria delle Reti Logiche

- Come identificare circuiti, o reti, *minimi* ?
- risulta chiaro come sia necessario uno strumento *formale* per l'analisi ed il progetto delle reti logiche…
- … usiamo l'Algebra di Boole!

Alcune definizioni…

$y = a\bar{b} + b\bar{d}x$

SOMMA DI PRODOTTI

b, x, ¬d sono *letterali*

$y = a \cdot (b + \bar{c}) \cdot (\bar{a} + x)$

PRODOTTO DI SOMME

termine elementare (*clausola*)

$f(a,b,c) = \ldots + a\bar{b}\bar{c} + \ldots$

fattore elementare

$f(a,b,c) = \ldots(\ldots) \cdot (a + \bar{b} + \bar{c}) \cdot (\ldots)\ldots$

mintermine (P_i)

maxtermine (S_i)

Mintermini e Maxtermini

mintermine di una funzione di *n* ingressi: indica uno qualsiasi dei 2^n prodotti in cui figurano tutte le *n* le variabili

maxtermine di una funzione di *n* ingressi: indica una qualsiasi delle 2^n somme in cui figurano tutte le *n* le variabili

$$P_0 = \bar{a}\bar{b}\bar{c} \quad P_5 = a\bar{b}c$$

$$S_0 = a+b+c \quad S_5 = \bar{a}+b+\bar{c}$$

osservazione:
0 in binario su tre bit è **000**, 5 è **101**.

In generale: Nei mintermini Pi si nega in corrispondenza dello zero, nei maxtermini Si si nega in corrispondenza degli uno

$$\bar{P}_i = S_i \quad \text{(dai teoremi di de Morgan)}$$

$$\forall i \neq j \; P_i \cdot P_j = 0, \; S_i + S_j = 1$$

$$\sum P_i = 1, \; \prod S_i = 0$$

forma normale di tipo *P*

$$f(x_1,\ldots\ldots,x_n)=\overline{x_1}\,f(0,x_2,\ldots\ldots,x_n)+x_1 f(1,x_2,\ldots\ldots,x_n)=$$

$$=\overline{x_1}[\overline{x_2}\,f(0,0,x_3,\ldots x_n)+x_2 f(0,1,x_3,\ldots x_n)]+$$

$$x_1[\overline{x_2}\,f(1,0,x_3,\ldots x_n)+x_2 f(1,1,x_3,\ldots x_n)]$$

……

$$=f(0,0,\ldots 0)\overline{x_1}\,\overline{x_2}\cdots\overline{x_n}+\ldots+f(1,1,\ldots 1)x_1x_2\cdots x_n=\sum_{i=0}^{2^n-1}\alpha_i P_i$$

Forma canonica: definisce univocamente la funzione

mintermini

dove

$$\alpha_0=f(0,0,\ldots 0),\ \alpha_1=f(0,0,\ldots 1),\ldots\ldots\alpha_{2^n-1}=f(1,1,\ldots 1),$$

"uscite" della funzione: sono gli '1' e '0' della tabella di verità, non sono variabili!

forma normale di tipo *P*

- una funzione di *n* variabili, assegnata mediante una tabella di verità, può essere espressa algebricamente da una **somma di prodotti**.
- Ciascun termine della somma è associato ad un "1" presente nella colonna della tabella ed è un prodotto delle *n* variabili, ciascuna delle quali nella forma negata o non a seconda che nelle colonne corrispondenti sia presente uno "0" o un "1".
- Qualsiasi funzione è pertanto **algebrica**

forma normale di tipo *P*

- Esempio:

a	b	c	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

nella forma algebrica si scrive
un mintermine per ogni 1
presente nella tabella di verità

$$y = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + abc$$

forma normale di tipo *P*

- Viceversa, qualsiasi funzione algebrica può essere posta in forma normale *P* “aggiungendo” i letterali mancanti
- Basta sviluppare tutte le operazioni fino ad ottenere una somma di prodotti
- Le clausole che non siano mintermini (ovvero che non contengano tutti le variabili della funzione) possono essere moltiplicate per la somma di tutte le possibili clausole ottenibili con le variabili assenti
- *Esempio….*

forma normale di tipo *P*

- Partendo da:

$$y(a,b,c,d)=\bar{a}b+bc+bd$$

- si può ricavare la forma P moltiplicando
  il primo termine per $(c+\bar{c})(d+\bar{d})$
  il secondo per $(a+\bar{a})(d+\bar{d})$
  ed il terzo per $(a+\bar{a})(c+\bar{c})$
- Si otterrà una forma normale di tipo P con 12 mintermini

forma normale di tipo *S*

$$f(x_1 \dots x_n) = \prod_{i=0}^{2^n - 1} (\alpha_i + S_i)$$

- si può ottenere con il procedimento duale di quello usato per la forma di tipo *P*
- In alternativa, si può negare la forma di tipo *P* e poi applicare de Morgan

forma normale di tipo *S*

- una funzione di *n* variabili può essere espressa algebricamente da un prodotto di somme;
- ciascun fattore del prodotto è associato ad uno *0* presente nella colonna della tabella ed è una somma delle *n* variabili, ciascuna delle quali nella forma negata o non a seconda che nelle colonne corrispondenti sia presente un *1* o uno *0*.

forma normale di tipo *S*

- Esempio

a	b	c	y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

nella forma algebrica si scrive un maxtermine per ogni 0 presente nella tabella di verità

$$y = (\bar{a} + b + c) \cdot (\bar{a} + b + \bar{c}) \cdot (\bar{a} + \bar{b} + c)$$

Implicanti di una funzione

- Un implicante di *f* è una funzione *h* tale che

$$\bar{h} + f = 1$$

In altri termini: se *h* è 1, anche *f* ***deve*** necessariamente essere 1

- Esempi:

$$f = ab + abc + cd$$

tutte le clausole in una forma SOP (somma di prodotti) sono implicanti!

$$f = a + ((a \oplus d) \uparrow b) \equiv (cd)$$

a è sicuramente un implicante

Implicanti di una funzione

- in una funzione scritta come *somma di prodotti*, ciascun prodotto rappresenta un implicante
  - *infatti in una somma, se è alto anche solo un termine, è alta l'intera somma*

$$y = \underbrace{\bar{a}\bar{b}\bar{c}}_{} + \underbrace{\bar{a}bc}_{} + \underbrace{abc}_{}$$

sono tutti implicanti. Ad esempio, se è alto abc, è alta anche y

Implicanti primi

- Un implicante di una funzione *f* si dice **primo** se non implica a sua volta nessun altro implicante della funzione

$$f = ab + abc + cd$$

abc è un implicante non primo

ab è un implicante **primo**

- Una funzione scritta come somma solo di implicanti primi si dice scritta in **forma PI**

Implicanti primi

- I termini di una somma sono sempre implicanti, ma non è detto che siano implicanti primi:

$$y = \bar{a}\bar{b}\bar{c} + \bar{a}bc + abc =$$

$$\bar{a}\bar{b}\bar{c} + bc(\bar{a} + a) = \bar{a}\bar{b}\bar{c} + bc$$

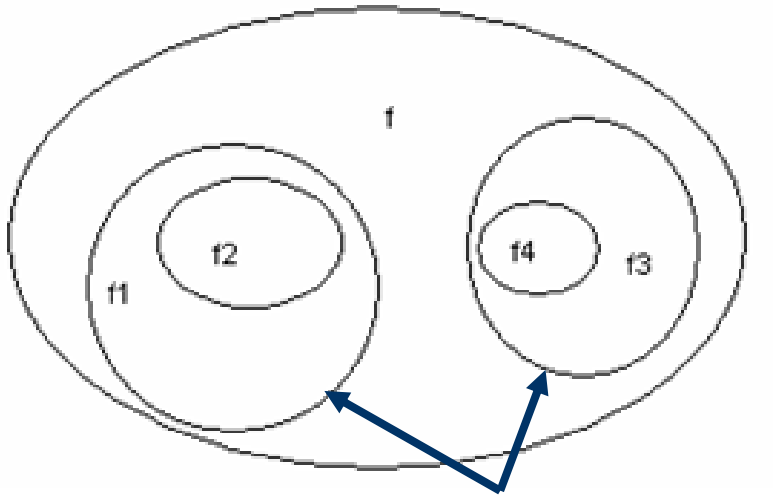
implicazioni:

$$abc \rightarrow bc \rightarrow y$$

Esempio: abc, è un implicante di y, ma anche di bc, quindi non è primo. bc, invece, non implica nient'altro a parte la funzione y stessa, quindi è *primo*

Implicanti primi

- Un’interpretazione insiemistica:
- Tutti i sottoinsiemi di *f* sono suoi implicanti



- Solo f_1 ed f_3 sono implicanti primi

Proprietà degli implicanti

1. Una clausola Y ne implica un'altra X se e solo se Y contiene tutti i letterali di X

$$bc = bc(\bar{a} + a) = \bar{a}bc + abc$$

Esempio:
abc implica bc poiché ne contiene tutti i letterali

2. La somma di due clausole di ordine *n* che contengono *n-1* letterali uguali ed in cui un letterale dell'una sia il complemento di quello dell'altra è la clausola di ordine *n-1* formata dai letterali comuni (detta ***consenso***)

   nell'esempio al punto 1 c'è un consenso che genera bc

Funzioni di costo

$$f = bc(a\bar{d} + \bar{b} + c) + \bar{c}(d + \bar{a})(b + c)$$

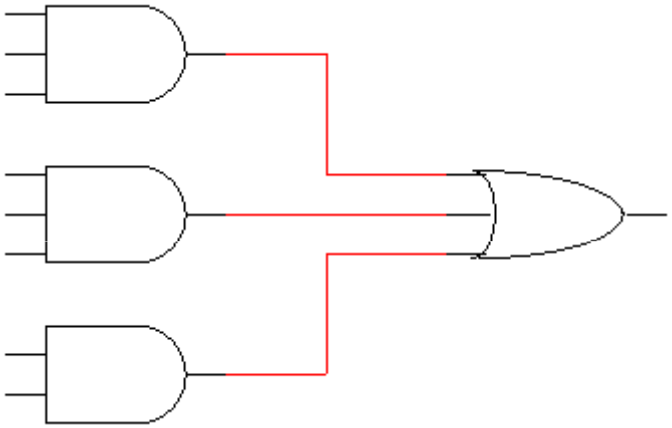
Costo di letterali (CL)	**11**
Costo di funzioni o di porte (CP)	**7**
Costo di ingressi (CI)	**17**

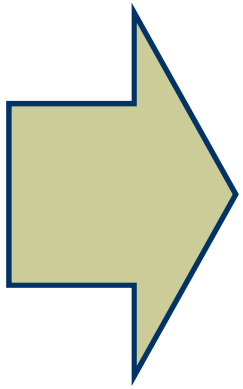
Implicanti e funzioni di costo

- una forma elementare che minimizzi i valori dei costi CL e CI è una forma PI
- fra le forme minime a 2 livelli che minimizzano CP ne esiste almeno una PI
- sotto il vincolo di rete a 2 livelli, la forma minima va allora cercata tra le forme PI

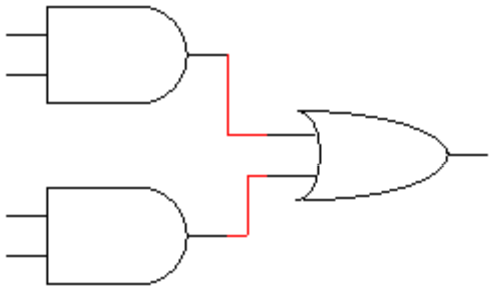
Implicanti e funzioni di costo

$$ab\bar{c} + abc + cd$$





$$ab + cd$$



Forma **non-PI**
CL = 8
CI = 11
CP = 4

Forma **PI**
CL = 4
CI = 6
CP = 3

Ricerca di forme minime

funzione *f* da minimizzare

riscrittura della funzione *f* in forma PI

scelta del sottoinsieme ottimo di PI la cui somma dà ancora *f*

Alcuni definizioni

- Un implicante primo E_i di una funzione *f* è detto *essenziale* se è l'unico ad essere implicato da una mintermine di *f*
- In altri termini, E_i è l'unico a "coprire" un determinato mintermine della funzione
- Il *nucleo N* della funzione è la somma dei suoi implicanti primi essenziali

$$N = \sum_{i=1}^{k} E_i$$

- Ogni forma minima di f è del tipo $f = N+R$

  con N e R (insieme degli implicanti primi non essenziali) eventualmente nulli

Ricerca di Implicanti Primi (PI)

a	b	c	y
0	0	0	*0*
0	0	1	*0*
0	1	0	*0*
0	1	1	*1*
1	0	0	*1*
1	0	1	*1*
1	1	0	*0*
1	1	1	*1*

$$f(a,b,c) = \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + abc$$

$$= bc(\bar{a} + a) + a\bar{b}(\bar{c} + c)$$

$$= bc + a\bar{b}$$

scriviamo la funzione come somma di *mintermini*, ciascuno corrispondente ad un ‘1’ nella tabella di verità. Cerchiamo poi di semplificare l’espressione con passaggi algebrici

Ricerca di Implicanti Primi (PI)

a	b	c	y
0	0	0	*0*
0	0	1	*0*
0	1	0	*0*
0	1	1	*1*
1	0	0	*1*
1	0	1	*1*
1	1	0	*0*
1	1	1	*1*

$$f(a,b,c) = \bar{a}bc + \ldots + abc$$

$$= bc(\bar{a} + a) + \ldots$$

$$= bc + \ldots$$

scriviamo la funzione come somma di *mintermini*, ciascuno corrispondente ad un '1' nella tabella di verità

$\bar{a}bc$

abc

abbiamo potuto raccogliere due termini e semplificare l'espressione poiché tra le due clausole è presente un **consenso**

Rappresentare le funzioni booleane

- Consenso:
  - presenza di due clausole (non necessariamente mintermini) aventi gli stessi letterali, tranne uno presente negato in una clausola e non negato nell'altra
- Esempi…

$$\bar{a}bc + \ldots + abc = \ldots + bc + \ldots \qquad \ldots + a\bar{b}\bar{c} + \ldots + ab\bar{c} = \ldots + a\bar{c}$$

Ricerca di PI: metodo agebrico (Quine)

- Scrivere la funzione in forma P (somma di mintermini)
- cercare tutti i possibili consensi tra coppie qualsiasi di clausole (aventi lo stesso numero di letterali)
- al passaggio successivo, eliminare clausole che sono state “assorbite” in clausole più piccole (poiché corrispondono ad implicanti non primi)
- iterare i raccoglimenti per clausole di dimensione via via minore
- quando non sarà più possibile individuare consensi, la funzione sarà scritta in forma PI

Metodo agebrico: esempio

$$f(a,b,c) = \bar{a}(b+c) + ac$$

$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

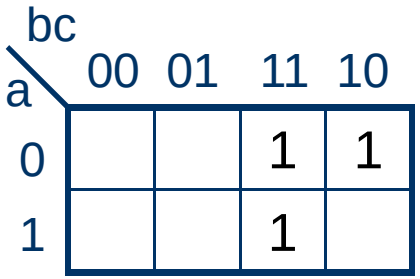
Forma P

$$\bar{a}b + ac + bc + \bar{b}c + ac =$$

$$\bar{a}b + c + c = \bar{a}b + c$$

Mappe di Karnaugh

a	*b*	*c*	y
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	*0*
1	1	1	*1*



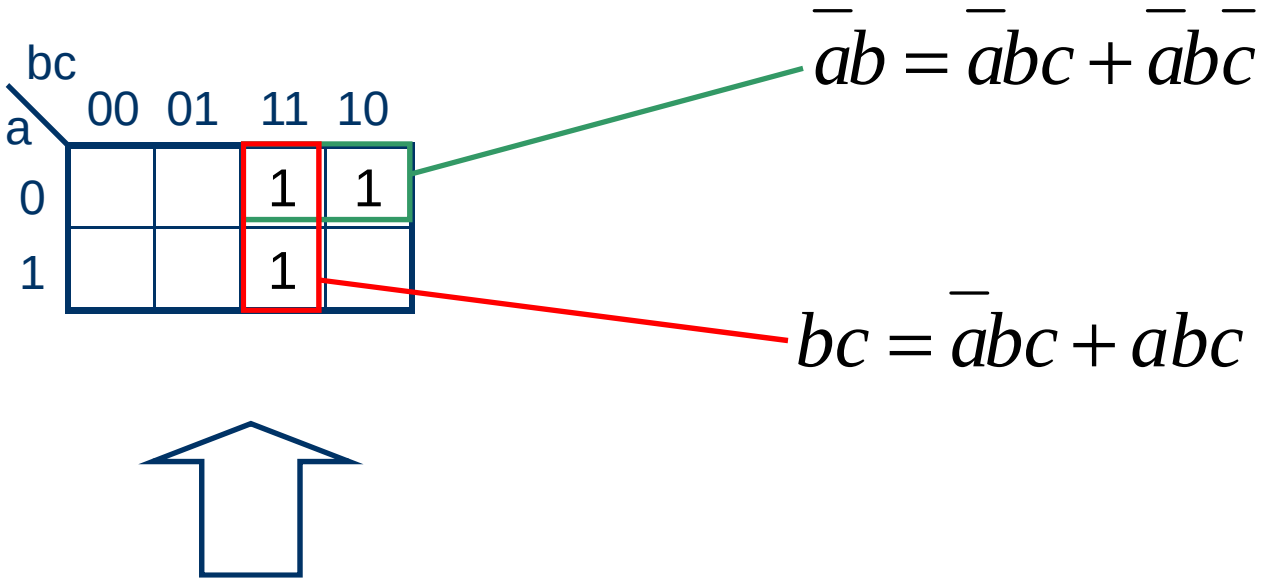
E’ semplicemente un altro modo di disporre la tabella di verità

a differenza della tabella di verità, tutte le celle adiacenti corrispondono ad un consenso

(è il motivo per cui si sceglie la sequenza di valori 00, 01, 11, 10)

Mappe di Karnaugh

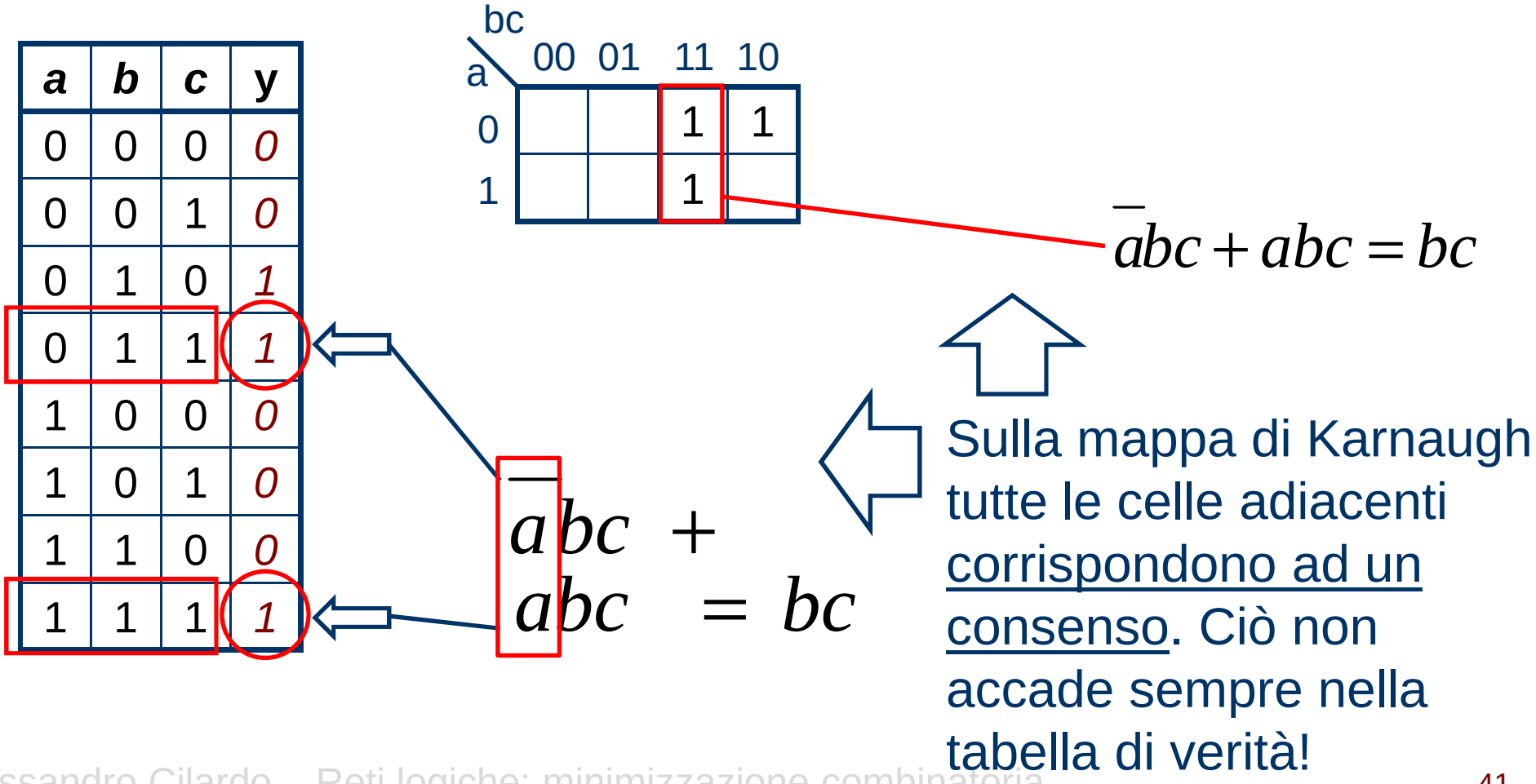
a	*b*	*c*	y
0	0	0	*0*
0	0	1	*0*
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*



a differenza della tabella di verità, tutte le celle adiacenti corrispondono ad un consenso

(è il motivo per cui si sceglie la sequenza di valori 00, 01, 11, 10)

Mappe di Karnaugh



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

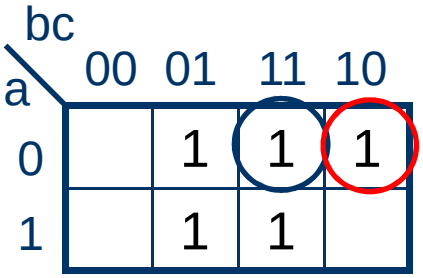
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}bc + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

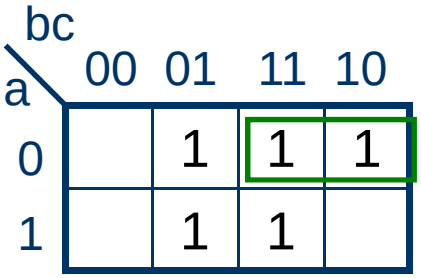
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}bc + \bar{a}bc + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

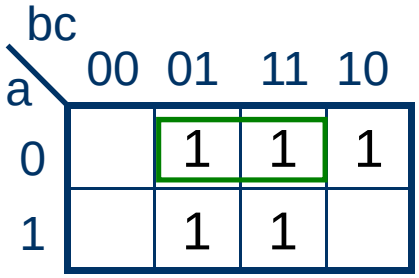
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \boxed{\bar{a}c} + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

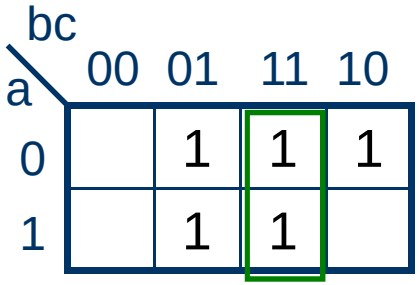
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

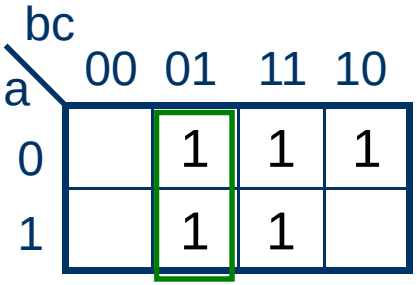
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

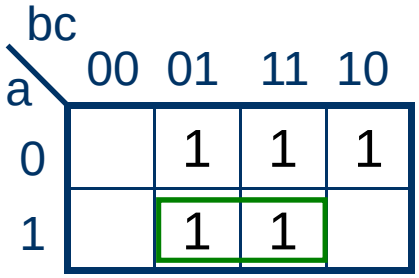
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

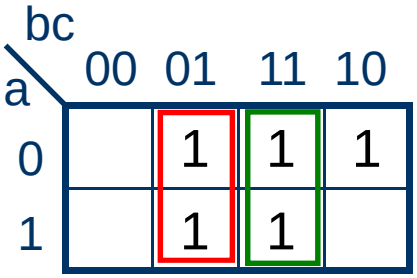
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



Mappe di Karnaugh

$$f(a,b,c) = \bar{a}(b+c) + ac$$

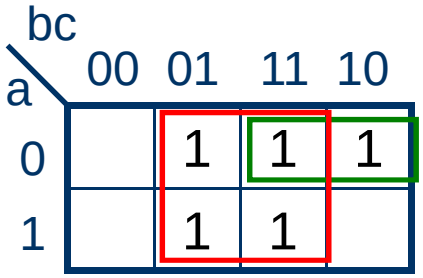
$$\bar{a}b + \bar{a}c + ac =$$

$$\bar{a}b(c+\bar{c}) + \bar{a}(b+\bar{b})c + a(b+\bar{b})c =$$

$$\bar{a}bc + \bar{a}b\bar{c} + a\bar{b}c + abc + a\bar{b}c =$$

$$\bar{a}b + \bar{a}c + bc + \bar{b}c + ac =$$

$$\bar{a}b + c =$$



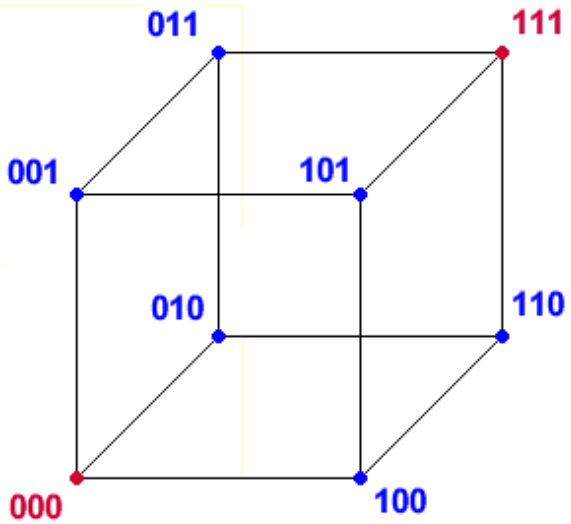
Gruppi di celle sulle mappe di Karnaugh che corrispondono ad una clausola (ad es. $\bar{a}b$ oppure c) sono detti ***cubi***

Mappe di Karnaugh

a	*b*	*c*	y
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	*0*
1	1	1	*1*

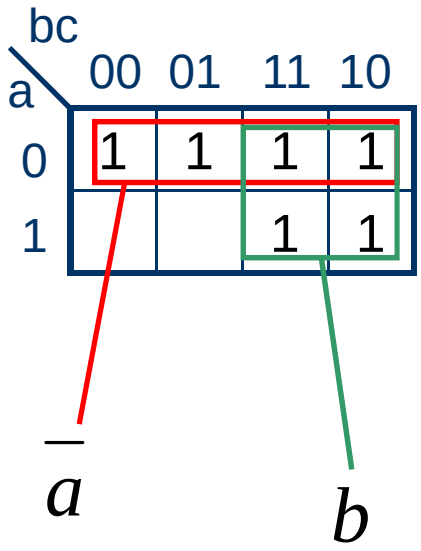
a \ bc	00	01	11	10
0			1	1
1			1	

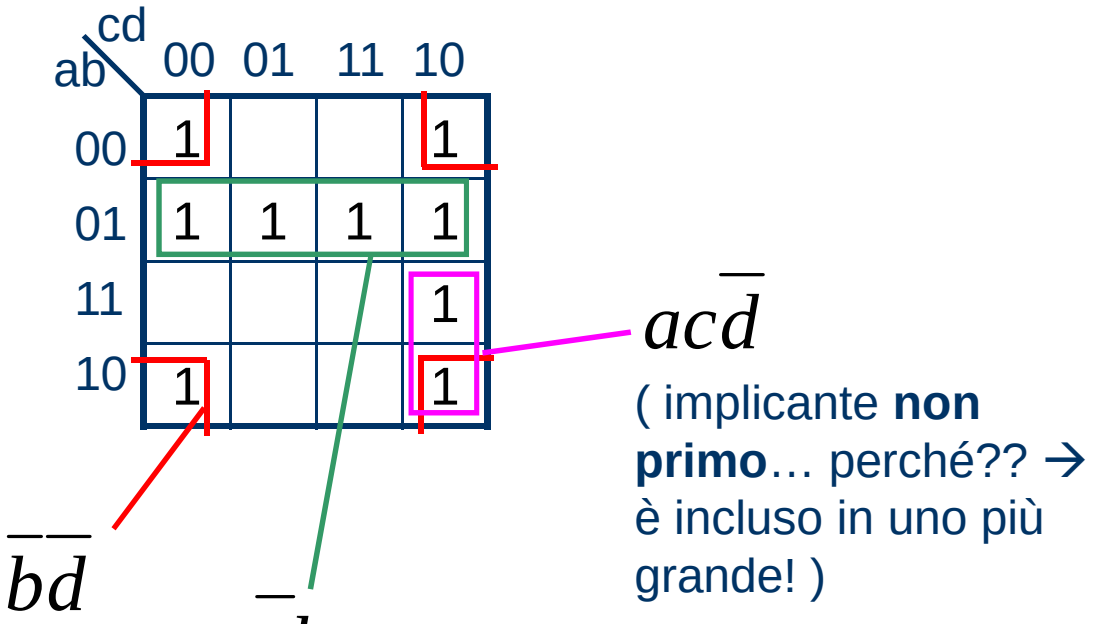
tutte le celle adiacenti corrispondono ad un consenso



Mappe di Karnaugh

Vari modi per identificare **cubi** su una mappa di Karnaugh.
La caratteristica essenziale di un cubo è quella di contenere alcune variabili sempre con lo stesso segno, mentre le rimanenti assumo tutte le possibili combinazioni di valori negati o non negati (un cubo conterrà quindi sempre un numero di celle pari ad una potenza di 2)

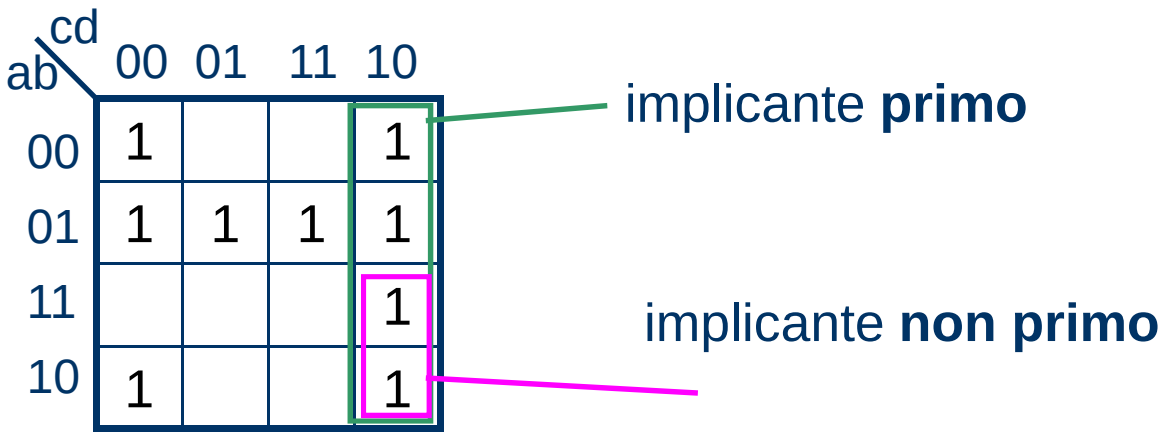




(implicante **non primo**… perché?? → è incluso in uno più grande!)

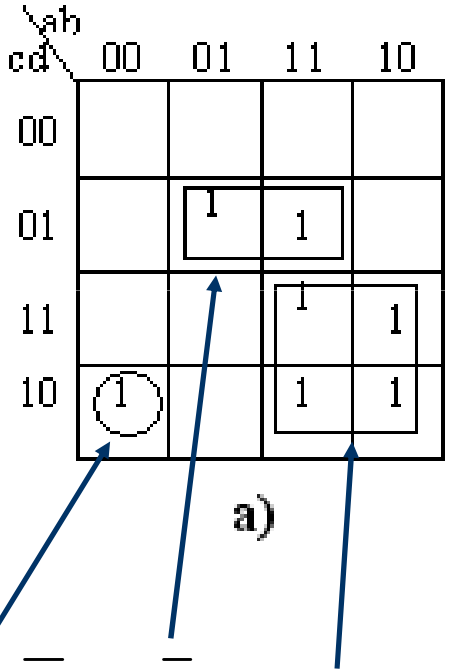
PI su mappe di Karnaugh

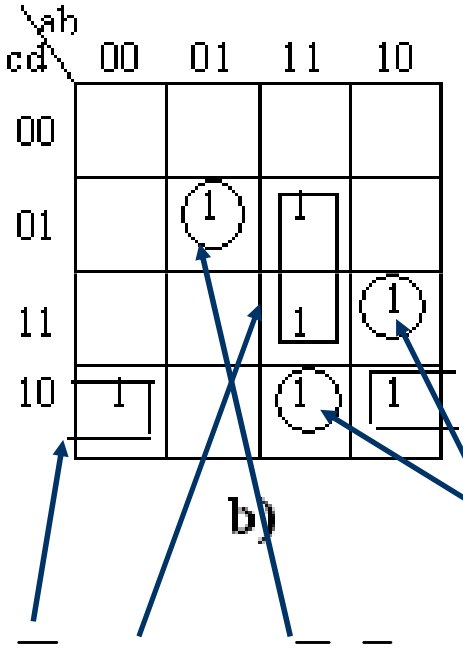
- La ricerca di Implicanti Primi sulle mappe di Karnaugh è effettuata semplicemente cercando tutti i cubi più larghi (corrispondenti a clausole più piccole) che coprano celle con ‘1’



Mappe di Karnaugh

Due modi per rappresentare la stessa funzione:



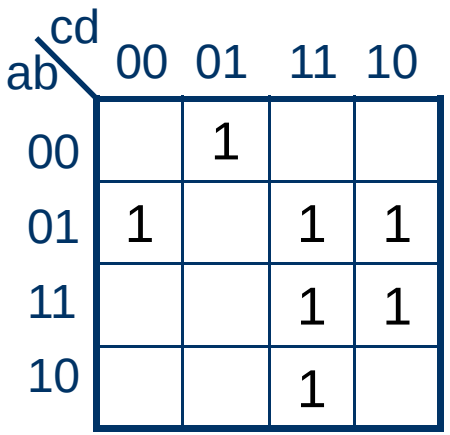


$a)\ y_1 = \bar{a}bc\bar{d} + b\bar{c}d + a\bar{c} \quad b)\ y_2 = \bar{b}\bar{c}d + a\bar{b}d + a\bar{b}cd + ab\bar{c}d + abc\bar{d}$

La forma a) contiene solo implicanti primi ed è quindi migliore in termini di costo

Rappresentare le funzioni booleane

a	*b*	*c*	*d*	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*0*
0	1	0	0	*1*
0	1	0	1	*0*
0	1	1	0	*1*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*1*
1	1	0	0	*0*
1	1	0	1	*0*
1	1	1	0	*1*
1	1	1	1	*1*



Esercizio: individuare la forma PI (somma di implicanti primi) con metodo grafico

Esercizio

- Scrivere la seguente funzione in forma PI usando il metodo algebrico:

$$f(a,b,c) = (a \equiv c) \cdot b + \overline{a}b$$

Esercizio

- Scrivere la seguente funzione in forma PI usando il metodo algebrico:

$$f(a,b,c) = (a \equiv c) \cdot b + \bar{a}b =$$
$$(ac + \bar{a}\bar{c}) \cdot b + \bar{a}b(c + \bar{c}) =$$
$$abc + \bar{a}b\bar{c} + \bar{a}bc + \bar{a}b\bar{c} =$$
$$bc + \bar{a}b$$

Esercizio

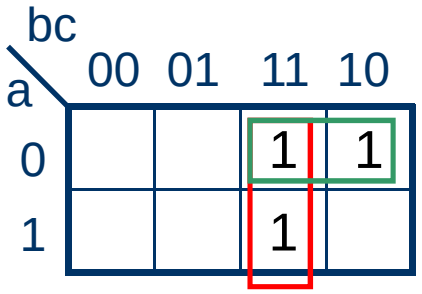
- Scrivere la seguente funzione in forma PI usando il metodo algebrico:

$$f(a,b,c) = (a \equiv c) \cdot b + \bar{a}b =$$

$$(ac + \bar{a}\bar{c}) \cdot b + \bar{a}b(c + \bar{c}) =$$

$$abc + \bar{a}b\bar{c} + \bar{a}bc + \bar{a}b\bar{c} =$$

$$bc + \bar{a}b$$



Metodo di McCluskey

- ciascuna clausola della funzione viene indicata da una stringa di *1*, *0* e ‘-’ :
  - ***1*** per le variabili in forma affermata,
  - ***0*** per quelle in forma negata
  - **-** per le variabili che non compaiono nel prodotto

Esempi di clausole (McCluskey)

- Convenzioni usate per indicare clausole nel metodo di McCluskey:

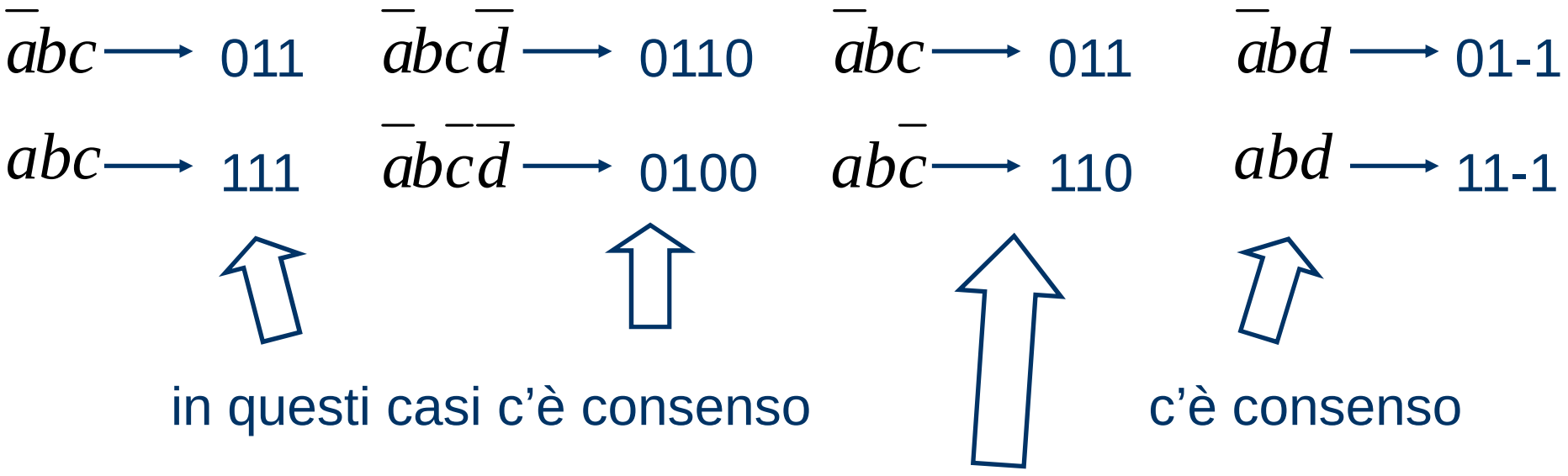
$\bar{a}bc \rightarrow 011$ $\bar{a}bd \rightarrow 01\text{-}1$

$\bar{a}b\bar{c}d \rightarrow 0101$ $\bar{a}bc \rightarrow 011\text{-}$

in questo caso, stiamo considerando una funzione avente quattro variabili *a*, *b*, *c*, *d,* e non tre come nel primo esempio → i simboli rappresentati sono quattro

Perché questa convenzione?

- stessa idea delle mappe di Karnaugh:
  → identificare facilmente consensi tra clausole



Perché questa convenzione?

$\bar{a}bc + abc = bc$	011	+	111	=	-11
$\bar{a}bc\bar{d} + \bar{a}bcd = \bar{a}b\bar{d}$	0110	+	0100	=	01-0
$\bar{a}bc + ab\bar{c}$	011	+	110		
$\bar{a}bd + abd = bd$	01-1	+	11-1	=	-1-1

- Osservazione: coppie di clausole che differiscono per un solo simbolo (**incluso il trattino -**) generano un consenso
- ogni coppia si può sostituire con un'unica clausola avente un trattino al posto del simbolo discordante

Metodo di McCluskey

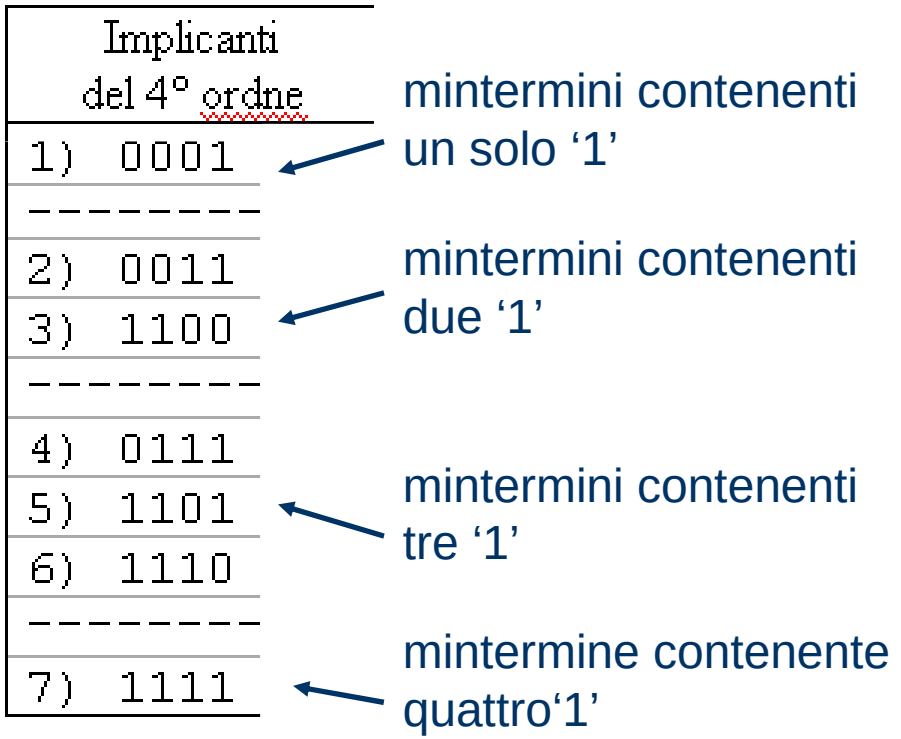
- Si scrive la funzione in forma P (somma di mintermini)
- ciascun mintermine (in quanto particolare clausola) viene denotato con la convenzione vista prima (*1*, *0* o ‘-’ per ciascuna variabile della funzione)
- Le clausole individuate vengono suddivise in classi contenenti elementi con egual numero di *1*

  → questa scelta facilita l’individuazione di eventuali consensi, riducendo il numero di confronti

Metodo di McCluskey

a	*b*	*c*	*d*	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*1*
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*0*
1	1	0	0	*1*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*1*

esempio di inizializzazione della tabella di McCluskey



Metodo di McCluskey

- Per ogni *i* da 0 a *k-1*: si generano i consensi di ordine *k-1* accoppiando le clausole della "classe *i*" con quelle della "classe *i*+1"; si marcano le clausole che generano consenso
- Le clausole non marcate sono **implicanti primi** di ordine *k*. Nei consensi generati si eliminano gli eventuali doppioni; i consensi stessi sono ordinati e riorganizzati da "classe 0" a "classe *k-1*".
- Si pone *k=k-1* e si ripetono i precedenti passi finché non sia *k≥0*.

Implicanti del 4° ordne
1) 0001 √
----
2) 0011 √
3) 1100 √
----
4) 0111 √
5) 1101 √
6) 1110 √
----
7) 1111 √

Metodo di McCluskey: esempio

Implicanti del 4° ordne
1) 0001 √
----
2) 0011 √
3) 1100 √
----
4) 0111 √
5) 1101 √
6) 1110 √
----
7) 1111 √

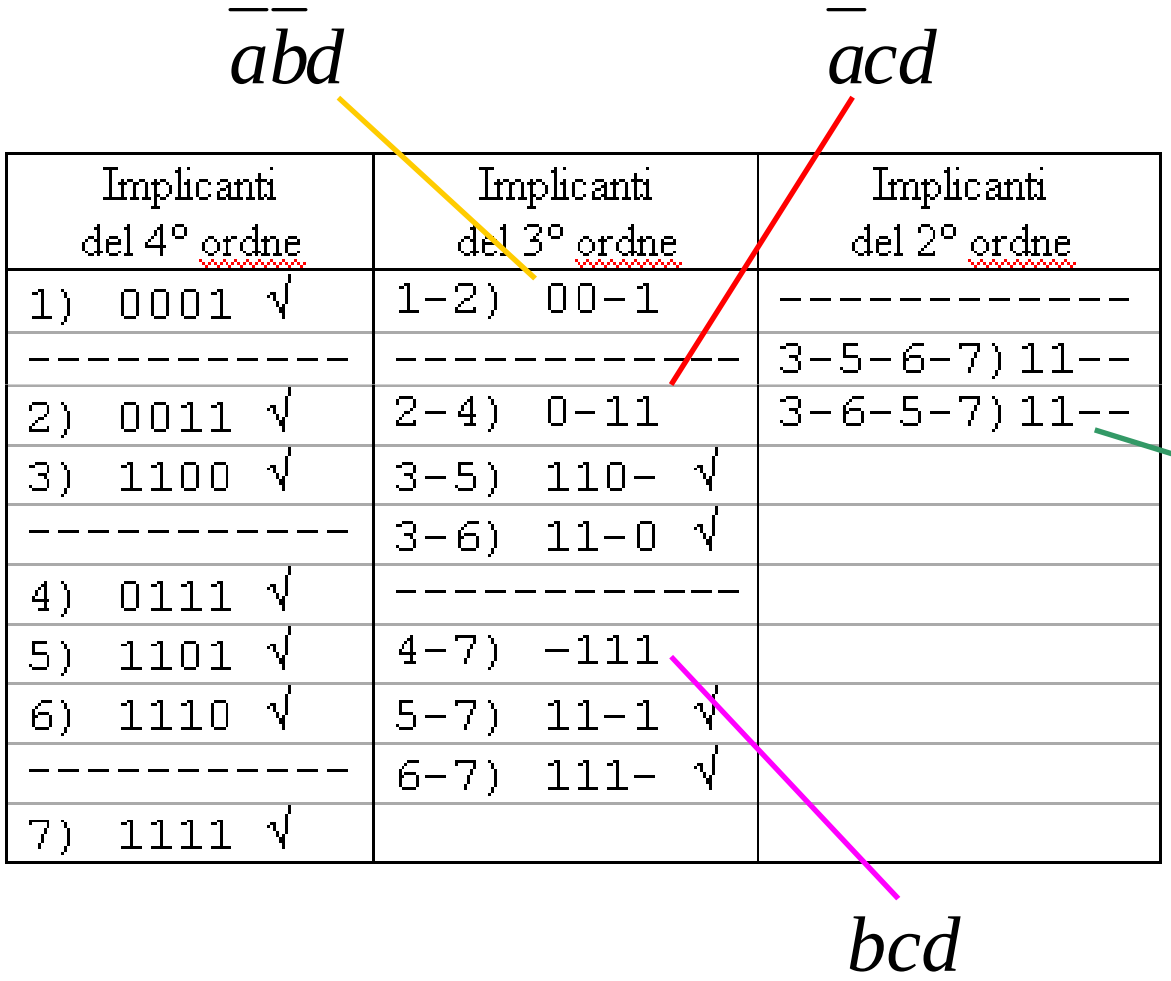
Metodo di McCluskey: esempio

Implicanti del 4° ordne	Implicanti del 3° ordne
1) 0001 √	1-2) 00-1
----------	----------
2) 0011 √	2-4) 0-11
3) 1100 √	3-5) 110- √
----------	3-6) 11-0 √
4) 0111 √	----------
5) 1101 √	4-7) -111
6) 1110 √	5-7) 11-1 √
----------	6-7) 111- √
7) 1111 √	

Metodo di McCluskey: esempio

Implicanti del 4° ordne	Implicanti del 3° ordne	Implicanti del 2° ordne
1) 0001 √	1-2) 00-1	----------
----------	----------	3-5-6-7) 11--
2) 0011 √	2-4) 0-11	3-6-5-7) 11--
3) 1100 √	3-5) 110- √	
----------	3-6) 11-0 √	
4) 0111 √	----------	
5) 1101 √	4-7) -111	
6) 1110 √	5-7) 11-1 √	
----------	6-7) 111- √	
7) 1111 √		

Metodo di McCluskey: esempio



le clausole non marcate sono implicanti primi

L’esempio sulla mappa di Karnaugh

- Il metodo di McCluskey ci ha consentito di trovare **tutti** gli implicanti primi della funzione:

$$\bar{a}\bar{b}d,\ \bar{a}cd,\ bcd,\ ab$$

- Sappiamo che una forma che minimizzi le diverse funzioni di costo dovrà essere una somma di tali PI
- **non necessariamente, però, li includerà tutti**
- Esempio:

certamente questa forma non è minima!

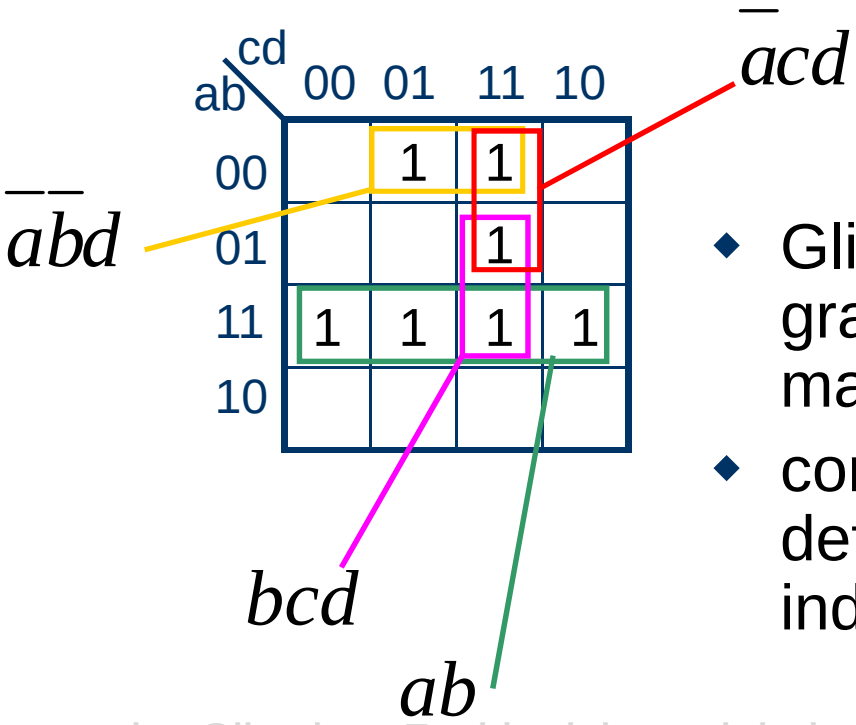
$$f = \bar{a}\bar{b}d + \bar{a}cd + bcd + ab =$$

$$\bar{a}\bar{b}d + \bar{a}\bar{b}cd + \bar{a}bcd + bcd + ab \;=\; \bar{a}\bar{b}d + bcd + ab$$

L’esempio sulla mappa di Karnaugh

- Il metodo di McCluskey ci ha consentito di trovare **tutti** gli implicanti primi della funzione:

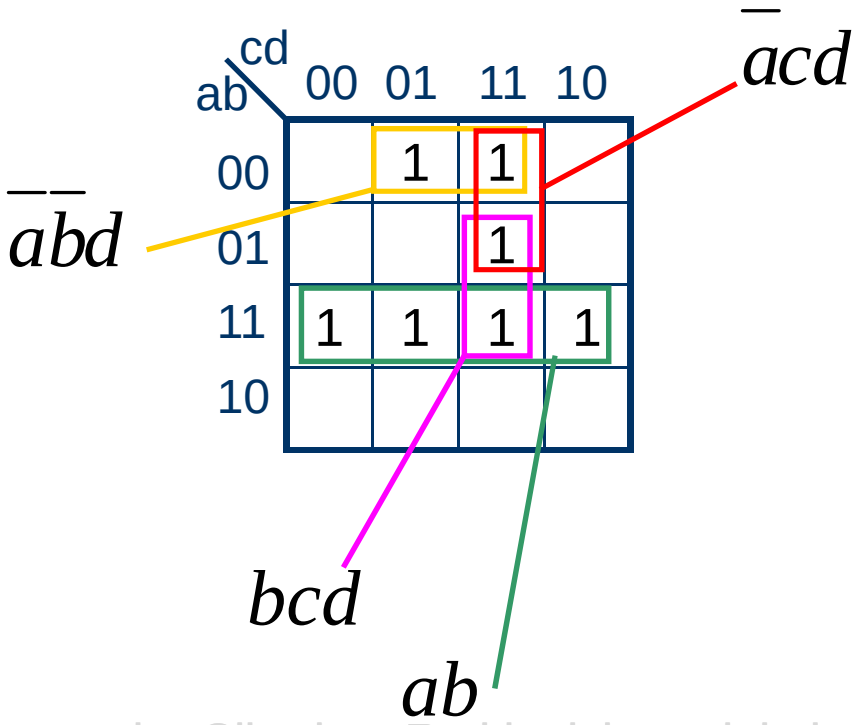
$$\bar{a}\bar{b}d,\ \bar{a}cd,\ bcd,\ ab$$



- Gli implicanti primi sono individuati graficamente come sottocubi di area massima nella mappa di Karnaugh
- come evidenziato dalla mappa, non è detto che tutti gli implicanti primi siano indispensabili!

L’esempio sulla mappa di Karnaugh

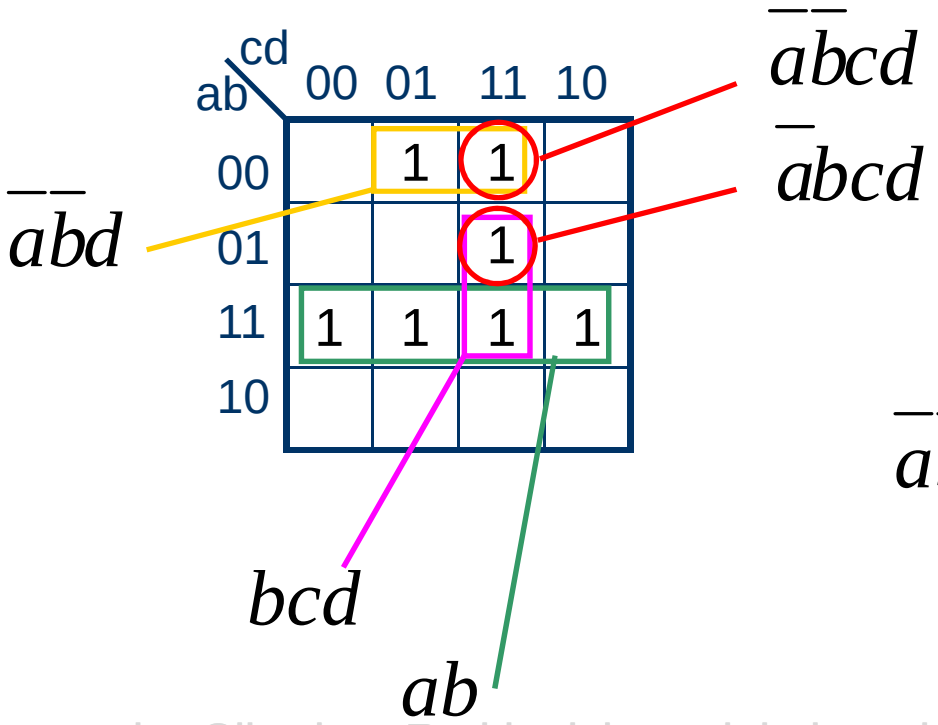
- Il metodo di McCluskey ci ha consentito di trovare **tutti** gli implicanti primi della funzione:



$$f = \bar{a}\bar{b}d + \bar{a}cd + bcd + ab$$

L’esempio sulla mappa di Karnaugh

- Il metodo di McCluskey ci ha consentito di trovare **tutti** gli implicanti primi della funzione:

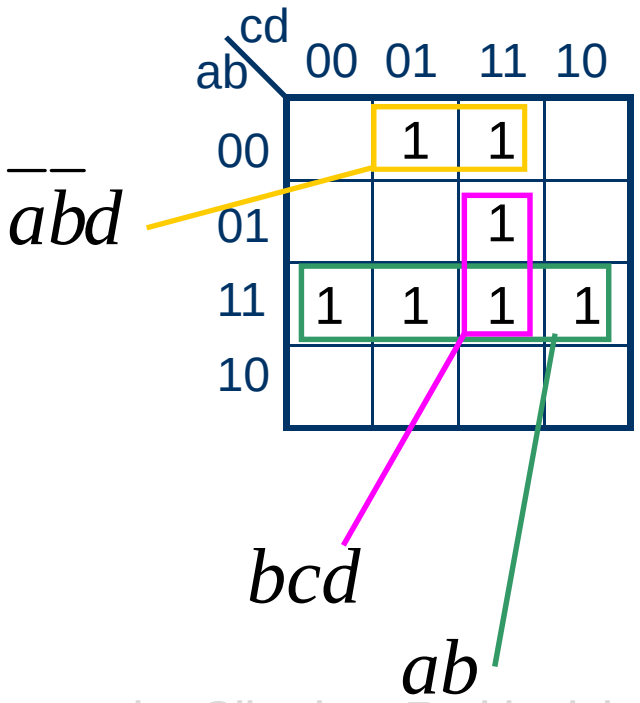


$$f = \bar{a}\bar{b}d + \bar{a}cd + bcd + ab =$$

$$\bar{a}\bar{b}d + \bar{a}\bar{b}cd + \bar{a}bcd + bcd + ab$$

L’esempio sulla mappa di Karnaugh

- Il metodo di McCluskey ci ha consentito di trovare **tutti** gli implicanti primi della funzione:



$$f = \bar{a}\bar{b}d + \bar{a}cd + bcd + ab =$$

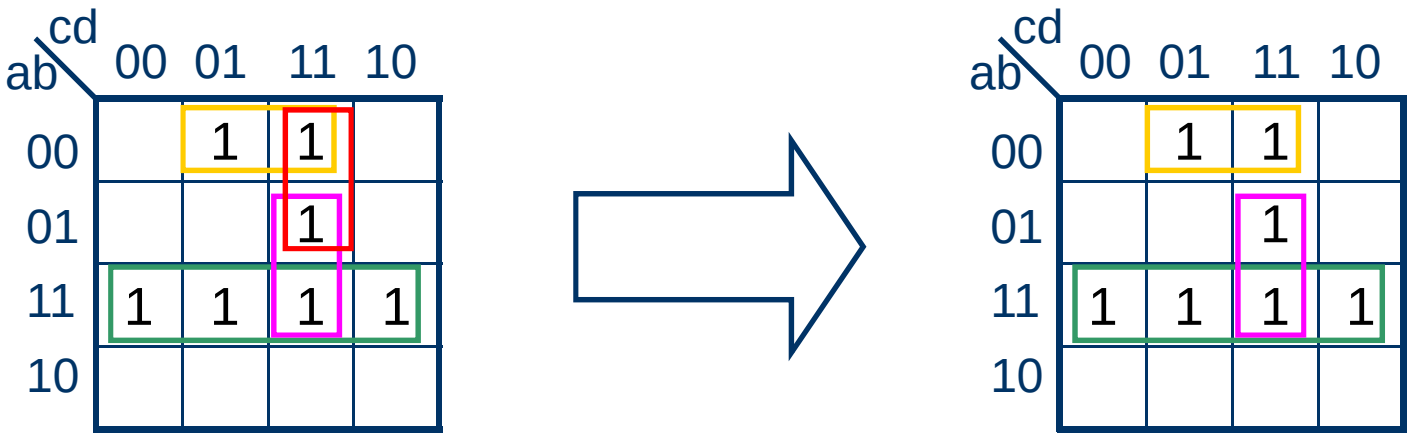
$$\underbrace{\bar{a}\bar{b}d + \bar{a}\bar{b}cd} + \underbrace{\bar{a}bcd + bcd} + ab =$$

$$\bar{a}\bar{b}d + bcd + ab$$

Copertura minima

- Individuati gli implicanti primi, occorre scegliere tra di essi **un insieme minimo** che consenta di "coprire" ancora la funzione

esempio:



Problema della copertura minima

- Il problema precedente si può ricondurre alla seguente formulazione:
- Data una matrice di *N* righe e *M* colonne, i cui elementi siano $a_{ij} = 1$ oppure $a_{ij} = 0$, si dice che una riga *i copre* una colonna *j* se $a_{ij} = 1$.
- Problema della *copertura minima*:
  - selezionare il numero **minimo** di righe che coprano tutte le colonne.
- Il problema della copertura è di interesse generale in molti settori differenti (ad esempio, in applicazioni di testing)

Copertura minima

	C_1	C_2	C_3	C_4	C_5	C_6
R_1		X				
R_2	X		X	X		
R_3			X			X
R_4					X	

- la riga R_1 ***copre*** la colonna C_2
- la riga R_2 ***copre*** le colonne C_1, C_3 e C_4
- etc..
- …
- la colonna C_3 ***è coperta*** dalle righe R_2 e R_3
- etc..

Copertura minima

colonne: *mintermini, ovvero '1' della funzione*

	C_1	C_2	C_3	C_4	C_5	C_6
R_1		X				
R_2	X		X	X		
R_3			X			X
R_4					X	

righe: *implicanti primi precedentemente trovati con metodo di McCluskey o con mappe di Karnaugh*

- nel nostro caso:
  - mettiamo sulle righe gli implicanti primi
  - e sulle colonne i mintermini che implicano gli implicanti primi:

    "la riga *X* copre la colonna *i*" significa quindi:

    l'implicante primo X è implicato dal mintermine i

Copertura minima

Ad esempio:

$\bar{a}\bar{b}cd$ $\bar{a}bcd$

	C_1	C_2	C_3	C_4	C_5	C_6
R_1		X				
R_2	X		X	X		
R_3			X			X
R_4					X	

$\bar{a}\bar{b}d$

a	b	c	d	y
0	0	0	0	*0*
0	**0**	**0**	**1**	***1***
0	0	1	0	*0*
0	**0**	**1**	**1**	***1***
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*

mintermini

⇦ $\bar{a}\bar{b}\bar{c}d$

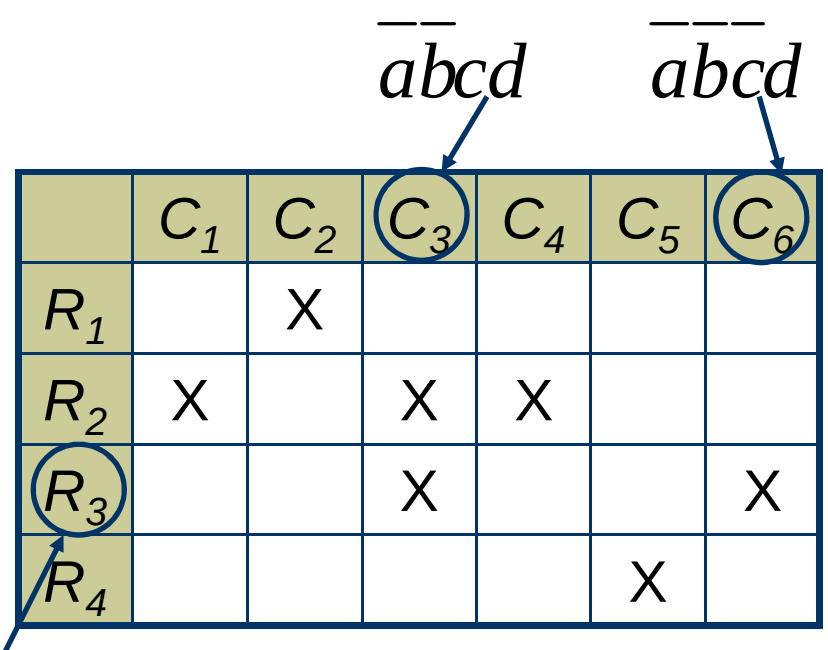
⇦ $\bar{a}\bar{b}cd$

. . . . . .

$$f = \ldots + \bar{a}\bar{b}cd + \bar{a}\bar{b}\bar{c}\bar{d} + \ldots = \ldots + \bar{a}\bar{b}d + \ldots$$

implicante primo ⇧

Copertura minima



a	b	c	d	y	
0	0	0	0	*0*	
0	**0**	**0**	**1**	***1***	⇦
0	0	1	0	*0*	
0	**0**	**1**	**1**	***1***	⇦
0	1	0	0	*0*	
0	1	0	1	*0*	

la tabella ci dice che la presenza dell'implicante (**riga R_3**) copre quei due ***1*** nella tabella di verità

$$f = \ldots + \bar{a}\bar{b}cd + \bar{a}\bar{b}\bar{c}\bar{d} + \ldots = \ldots + \bar{a}\bar{b}d + \ldots$$

Costruzione della matrice

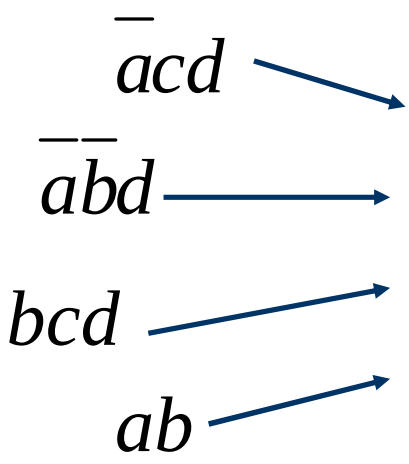
a	*b*	*c*	*d*	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*1*
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*0*
1	1	0	0	*1*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*1*

Ricordate che i mintermini si indicano con $\mathbf{P_i}$ *dove* i *è il valore decimale corrispondente alla configurazione degli ingressi relativi a quel mintermine*

	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
R_1							
R_2							
R_3							
R_4							

Costruzione della matrice

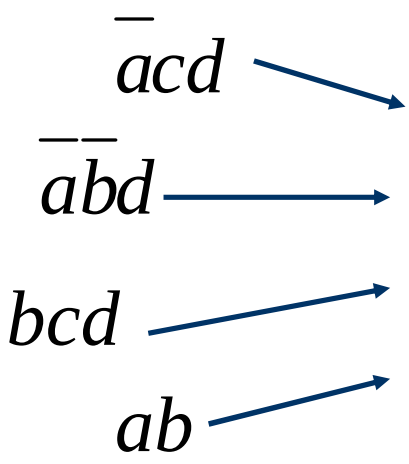
- Mettiamo sulle righe tutti gli implicanti primi precedentemente individuati tramite il metodo di McCluskey



	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11							
B= 00-1							
C= -111							
D= 11--							

Costruzione della matrice

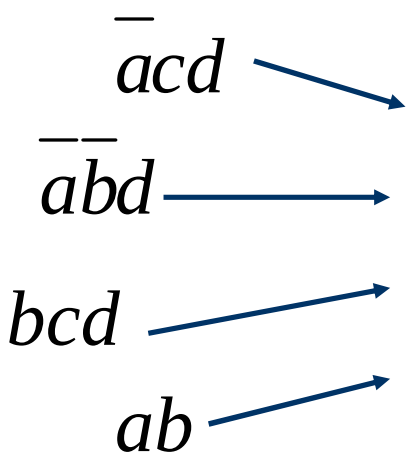
- Mettiamo sulle righe tutti gli implicanti primi precedentemente individuati tramite il metodo di McCluskey



	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1							
C= -111							
D= 11--							

Costruzione della matrice

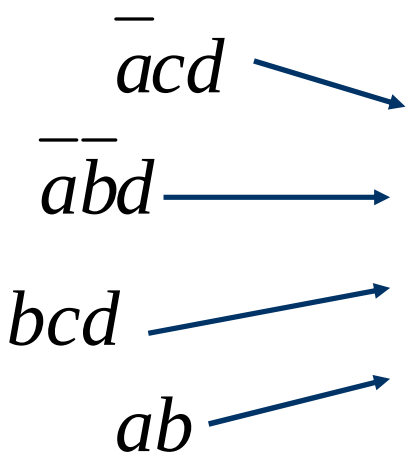
- Mettiamo sulle righe tutti gli implicanti primi precedentemente individuati tramite il metodo di McCluskey



	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1	X	X					
C= -111							
D= 11--							

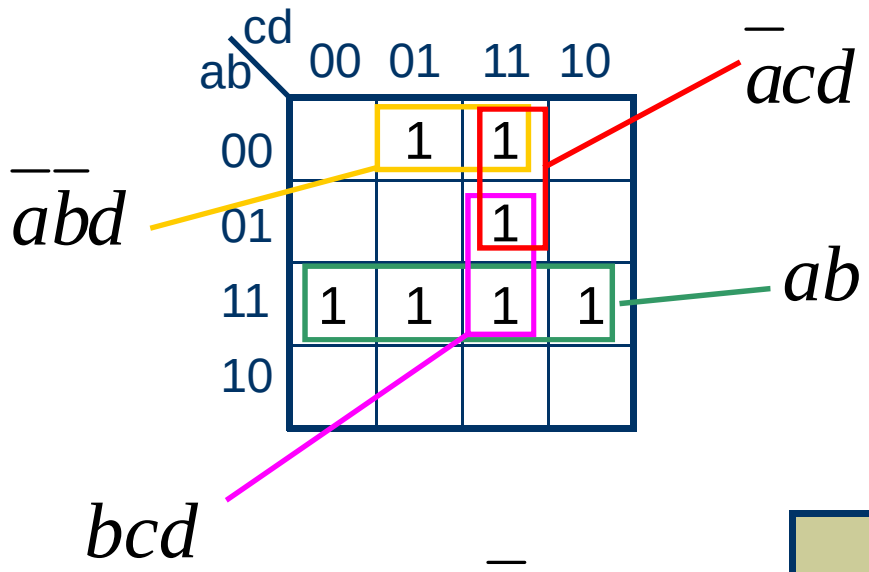
Costruzione della matrice

- Mettiamo sulle righe tutti gli implicanti primi precedentemente individuati tramite il metodo di McCluskey

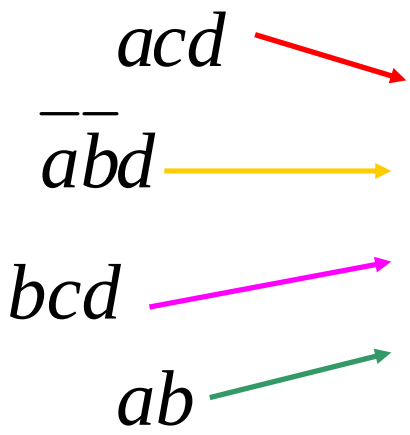


	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1	X	X					
C= -111			X				X
D= 11--				X	X	X	X

Costruzione della matrice



confrontate la matrice di copertura con la Mappa di Karnaugh

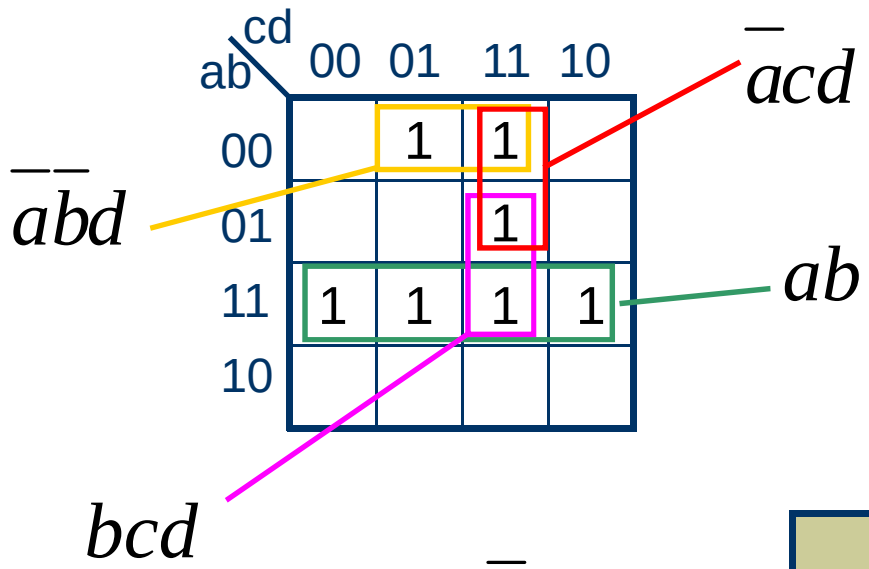


	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1	X	X					
C= -111			X				X
D= 11--				X	X	X	X

Individuazione del nucleo

- Per quanto visto prima, il primo passo da fare per individuare la copertura è selezionare l'insieme degli implicanti primi **essenziali** (il *nucleo*).
- Tali implicanti devono infatti essere sempre presenti e la loro selezione è univoca
- Nella matrice di copertura vengono individuati in corrispondenza di colonne con un unico '1'

Costruzione della matrice



l'implicante *ab* è sicuramente **essenziale** (è infatti l'unico a "coprire" degli 1 della funzione). Lo stesso vale per $\bar{a}\bar{b}d$

$\bar{a}cd$

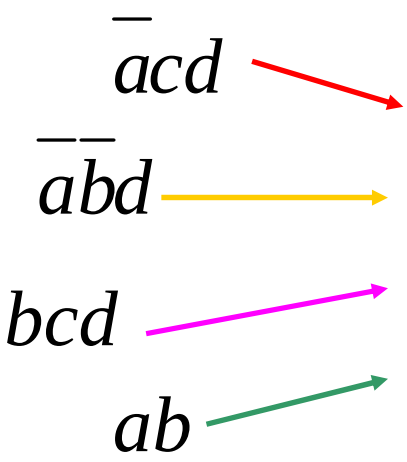
$\bar{a}\bar{b}d$

bcd

ab

	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1	X	X					
C= -111			X				X
D= 11--				X	X	X	X

Semplificazione della matrice



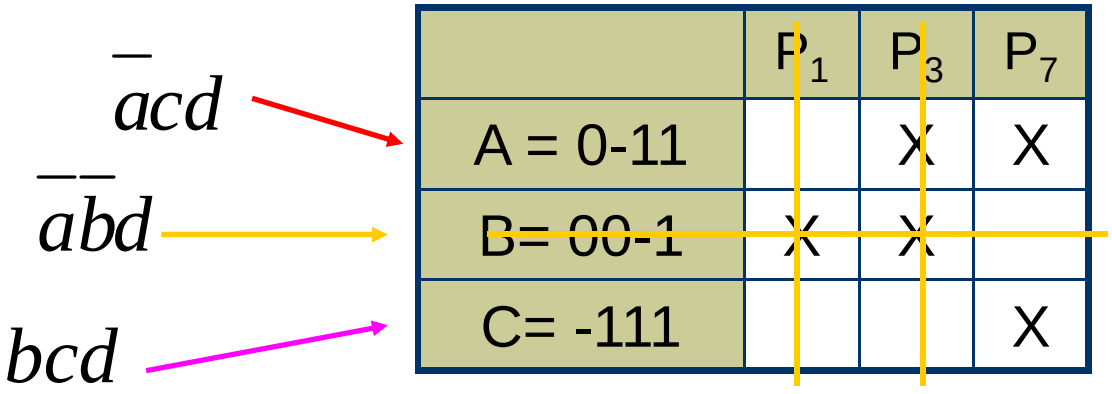
	P_1	P_3	P_7	P_{12}	P_{13}	P_{14}	P_{15}
A = 0-11		X	X				
B= 00-1	X	X					
C= -111			X				X
D= 11--				X	X	X	X

Il nucleo ***N*** è costituito dagli implicanti ***B*** e ***D***: ***N***={***B***, ***D***}. In altri termini, la funzione minima, scritta come somma di implicanti primi, conterrà certamente ***B*** ed ***D***:

$$f = \ldots + \bar{a}\bar{b}d + ab + \ldots$$

Avendo incluso ***D*** nella funzione, i mintermini P_{12}, P_{13}, P_{14} e P_{15} risultano coperti. Il problema si riduce alla coperatura di P_1, P_3 e P_7.

Semplificazione della matrice



Il nucleo ***N*** è costituito dagli implicanti ***B*** e ***D***: ***N***={***B***, ***D***}. In altri termini, la funzione minima, scritta come somma di implicanti primi, conterrà certamente ***B*** ed ***D***:

$$f = \ldots + \bar{a}\bar{b}d + ab + \ldots$$

Avendo incluso ***B*** nella funzione, i mintermini P_1 e P_3 risultano coperti. Il problema si riduce alla coperatura di P_7.

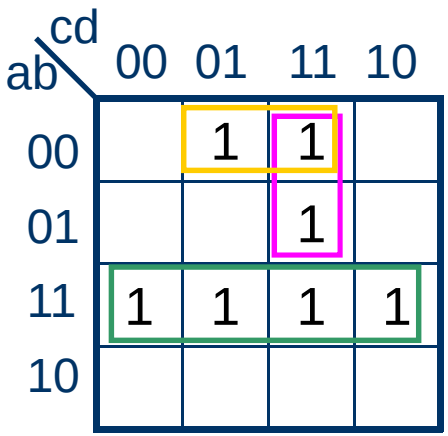
E' indifferente a questo punto scegliere ***A*** oppure ***C***

Forma minima della funzione

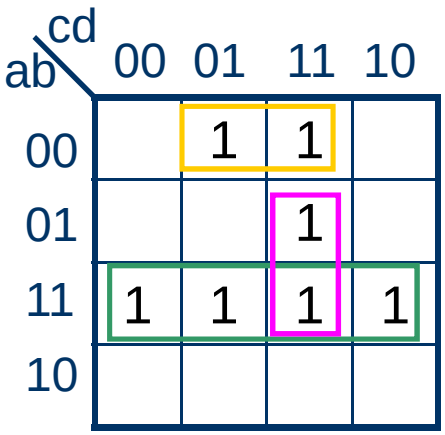
La ***f*** può quindi esser scritta come: $f = ab + \bar{a}\bar{b}d + \bar{a}cd$

oppure $f = ab + \bar{a}\bar{b}d + bcd$

…risultato a cui si può giungere intuitivamente sulla Mappa di Karnaugh:



oppure



Metodi di Copertura minima

Materiale facoltativo

- La selezione degli implicanti primi non essenziali è **arbitraria**, a differenza di quelli essenziali.

	P_1	P_8	P_9	P_{24}	P_{27}
A= -100-		X	X	X	
B= --001	X		X	X	
C= 0-00-	X	X	X		
D= 11-11					X
F= 110-1				X	X

Metodi di Copertura minima

Materiale facoltativo

- La selezione degli implicanti primi non essenziali è **arbitraria**, a differenza di quelli essenziali.
- La scelta deve essere:
  - Ottima secondo le prefissate funzioni di costo
  - ottenibile in maniera computazionalmente efficiente
- Due fondamentali alternative
  - Tabellare (righe/colonne dominanti)
  - Algebrica (Petrick)

Metodo di Petrick

Materiale facoltativo

- Esprime la condizione algebrica secondo la quale ***TUTTE*** le colonne devono essere coperte da ***ALMENO UN*** implicante.
- Può allora essere espressa come una AND di OR.
- Tale forma può essere trasformata in una OR di AND che esprime tutte le possibili alternative per la soluzione del problema di copertura

Metodo di Petrick: esempio

Materiale facoltativo

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
A	1		1		1		1	1	1
B	1	1		1			1	1	
C		1				1			1
D			1				1		
E				1		1			
F					1			1	
G						1			1

$$(A+B)(B+C)(A+D)(B+E)(A+F)(C+E+G)(A+B+D)(A+B+F)(A+C+G)=$$

$$=ACE + ABC + ABG + ABE + BCDF + BDFG$$

per risolvere il problema di copertura possono essere usati {A,C,E}, oppure {A,B,C}, oppure…

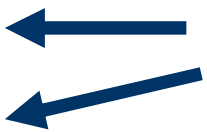
Righe/Colonne dominanti

Materiale facoltativo

- Chiamiamo “linea” indifferentemente una riga o una colonna
- Una linea *L* domina la linea *K* se la “include”, ovvero se contiene tutti i suoi *1*

La colonna **3** domina la **1**

	1	2	3	4	5	6
A		X				
B	X		X	X		
C	X		X		X	X
D			X		X	



La riga **C** domina la **D**

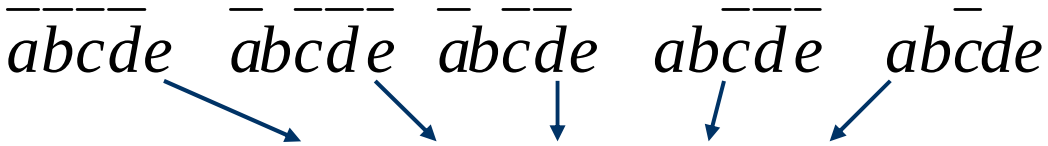
Righe/Colonne dominanti

Materiale facoltativo

- Se si eliminano le righe dominate e le colonne dominanti, da una matrice di copertura, se ne trae una equivalente (che rappresenta, cioè, il medesimo problema di copertura)

Righe/Colonne dominanti

Materiale facoltativo



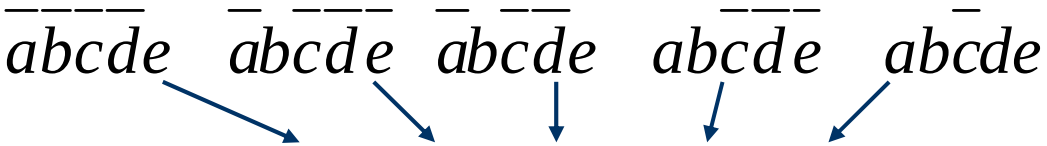
	P_1	P_8	P_9	P_{24}	P_{27}
$b\bar{c}\bar{d}$ A= -100-		X	X	X	
$\bar{c}\bar{d}e$ B= --001	X		X	X	
$\bar{a}\bar{c}\bar{d}$ C= 0-00-	X	X	X		
$abde$ D= 11-11					X
$ab\bar{c}e$ F= 110-1				X	X

Eliminazione delle **colonne dominanti**:

se trovo un sottoinsieme di {A,B,C,D,F} che copre P_8, lo stesso sottoinsieme coprirà anche P_9

Righe/Colonne dominanti

Materiale facoltativo



	P_1	P_8	P_9	P_{24}	P_{27}
$b\bar{c}\bar{d}$ A= -100-		X	X	X	
$\bar{c}de$ B= --001	X		X	X	
$\bar{a}\bar{c}\bar{d}$ C= 0-00-	X	X	X		
$abde$ ~~D= 11-11~~					~~X~~
$ab\bar{c}e$ F= 110-1				X	X

Eliminazione delle **righe dominate**:

il sottoinsieme {A,B,C,F} copre almeno gli stessi mintermini di {A,B,C,D,F} ma con un numero minore di implicanti

Righe/Colonne dominanti: riepilogo

Materiale facoltativo

- Il metodo tabellare per righe/colonne dominanti procede allora come segue:
  - Si ricercano gli implicanti primi (*PI*) e si individuano quelli essenziali;
  - Si includono nella forma minima i *PI* essenziali, eliminandoli dalla matrice, unitamente con i mintermini ricoperti;
  - Si eliminano le righe dominate e le colonne dominanti;
  - Si individuano i *PI* essenziali “secondari” della matrice così ridotta;
  - Si ripetono i passi 2, 3, 4 finché è possibile.

Esempio

Materiale facoltativo

$$f(x_1, x_2, x_3, x_4, x_5) = \sum(0,1,2,8,9,15,17,21,24,25,27,28,31) =$$

$$= \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5} + \overline{x_1}\,x_2\,\overline{x_3}\,\overline{x_4}\,\overline{x_5} + \overline{x_1}\,x_2\,x_3\,x_4\,\overline{x_5} + \ldots\ldots$$

Esempio

Materiale facoltativo

Implicanti del 5° ordne	Implicanti del 4° ordne	Implicanti del 3° ordne
1) 00000 √ ---------- 2) 00001 √ 3) 00010 √ 4) 01000 ---------- 5) 01001 6) 10001 7) 11000 ---------- 8) 10101 9) 11001 10) 11100 ---------- 11) 01111 12) 11011 ---------- 13) 11111	1-2 0000- 1-3 000-0 … … … …	

Esempio

Materiale facoltativo

Implicanti del 5° ordne	Implicanti del 4° ordne	Implicanti del 3° ordne
1) 00000 √	1-2 0000-	
----------	1-3 000-0	
2) 00001 √	1-4 0-000	
3) 00010 √	----------	
4) 01000 √	2-5 0-001	
----------	2-6 -0001	
5) 01001 √	4-5 0100-	
6) 10001 √	4-7 -1000	
7) 11000 √	----------	
----------	5-9 -1001	
8) 10101 √	6-8 10-01	
9) 11001 √	6-9 1-001	
10) 11100 √	7-9 1100-	
----------	7-10 11-00	
11) 01111 √	9-12 110-1	
12) 11011 √	----------	
----------	11-13 -1111	
13) 11111 √	12-13 11-11	

Esempio

Materiale facoltativo

Implicanti del 5° ordine	Implicanti del 4° ordine	Implicanti del 3° ordine
1) 00000 √	1-2 0000- √	1-2/4-5 0-00-
----------	1-3 000-0	1-4/2-5 0-00-
2) 00001 √	1-4 0-000 √	----------
3) 00010 √	----------	2-5/6-9 --001
4) 01000 √	2-5 0-001 √	2-6/5-9 --001
----------	2-6 -0001 √	4-5/7-9 -100-
5) 01001 √	4-5 0100- √	4-7/5-9 -100-
6) 10001 √	4-7 -1000 √	
7) 11000 √	----------	
----------	5-9 -1001 √	
8) 10101 √	6-8 10-01	
9) 11001 √	6-9 1-001 √	
10) 11100 √	7-9 1100- √	
----------	7-10 11-00	
11) 01111 √	9-12 110-1	
12) 11011 √	----------	
----------	11-13 -1111	
13) 11111 √	12-13 11-11	

Esempio

Materiale facoltativo

Implicanti del 5° ordine	Implicanti del 4° ordine	Implicanti del 3° ordine
1) 00000 √	1-2 0000- √	1-2/4-5 0-00- <== (C)
----------	1-3 000-0 <== (J)	1-4/2-5 0-00-
2) 00001 √	1-4 0-000 √	----------
3) 00010 √	----------	2-5/6-9 --001 <== (B)
4) 01000 √	2-5 0-001 √	2-6/5-9 --001
----------	2-6 -0001 √	4-5/7-9 -100- <== (A)
5) 01001 √	4-5 0100- √	4-7/5-9 -100-
6) 10001 √	4-7 -1000 √	
7) 11000 √	----------	
----------	5-9 -1001 √	
8) 10101 √	6-8 10-01 <== (G)	
9) 11001 √	6-9 1-001 √	
10) 11100 √	7-9 1100- √	
----------	7-10 11-00 <== (H)	
11) 01111 √	9-12 110-1 <== (F)	
12) 11011 √	----------	
----------	11-13 -1111 <== (E)	
13) 11111 √	12-13 11-11 <== (D)	

Esempio

Materiale facoltativo

Implicanti	Mintermini coperti
A = -100-	8,9,24,25
B = --001	1,9,17,25
C = 0-00-	0,1,8,9
D = 11-11	27,31
E = -1111	15,31
F = 110-1	25,27
G = 10-01	17,21
H = 11-00	24,28
J = 000-0	0,2

Esempio

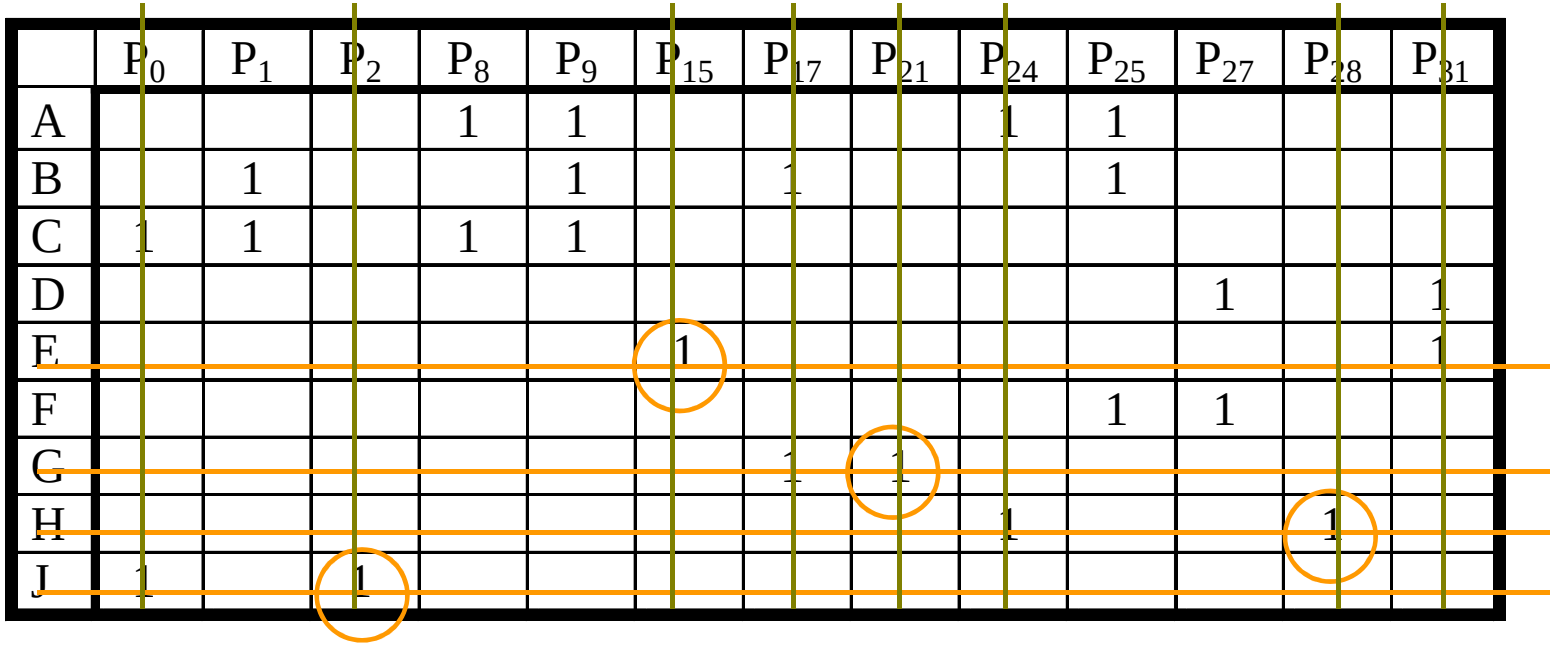
Materiale facoltativo

	P_0	P_1	P_2	P_8	P_9	P_{15}	P_{17}	P_{21}	P_{24}	P_{25}	P_{27}	P_{28}	P_{31}
A				1	1				1	1			
B		1			1		1			1			
C	1	1		1	1								
D											1		1
E						1							1
F										1	1		
G							1	1					
H									1			1	
J	1		1										

Primi implicanti essenziali: J, E, G, H

Esempio

Materiale facoltativo



	P_0	P_1	P_2	P_8	P_9	P_{15}	P_{17}	P_{21}	P_{24}	P_{25}	P_{27}	P_{28}	P_{31}
A				1	1				1	1			
B		1			1		1			1			
C	1	1		1	1								
D											1		1
E						1							1
F										1	1		
G							1	1					
H									1			1	
J	1		1										

Gli implicanti primi essenziali J,E,G,H coprono i mintermini: 0, 2, 15, 17, 21, 24, 28, 31

Esempio

Materiale facoltativo

	P_1	P_8	P_9	P_{25}	P_{27}
A		1	1	1	
B	1		1	1	
C	1	1	1		
D					1
F				1	1

riga D dominata dalla F

	P_1	P_8	P_{25}	P_{27}
A		1	1	
B	1		1	
C	1	1		
F			1	1

F implicante primo essenziale secondario:
copre P_{25} e P_{27}

	P_1	P_8
A		1
B	1	
C	1	1

Righe A e B dominate dalla **C**

Esempio

Materiale facoltativo

$$f(x_1, x_2, x_3, x_4, x_5) = \sum(0,1,2,8,9,15,17,21,24,25,27,28,31) =$$

$$= \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5} + x_1\overline{x_2}\,\overline{x_3}\,\overline{x_4}\,\overline{x_5} + \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,x_4\,\overline{x_5} + \ldots\ldots =$$

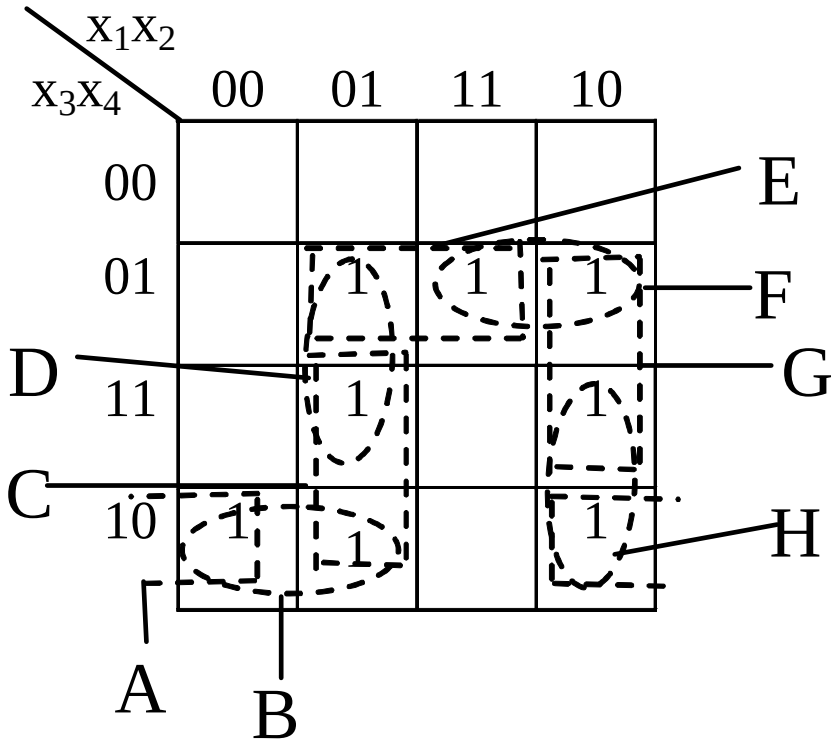
$$= E + G + H + J + F + C =$$

$$= x_2 x_3 x_4 x_5 + x_1 \overline{x_2}\,\overline{x_4}\,x_5 + x_1 x_2 \overline{x_4}\,x_5 + \overline{x_1}\,\overline{x_2}\,\overline{x_3}\,\overline{x_5}$$

$$+ x_1 x_2 \overline{x_3} x_5 + x_1 \overline{x_3}\,\overline{x_4}$$

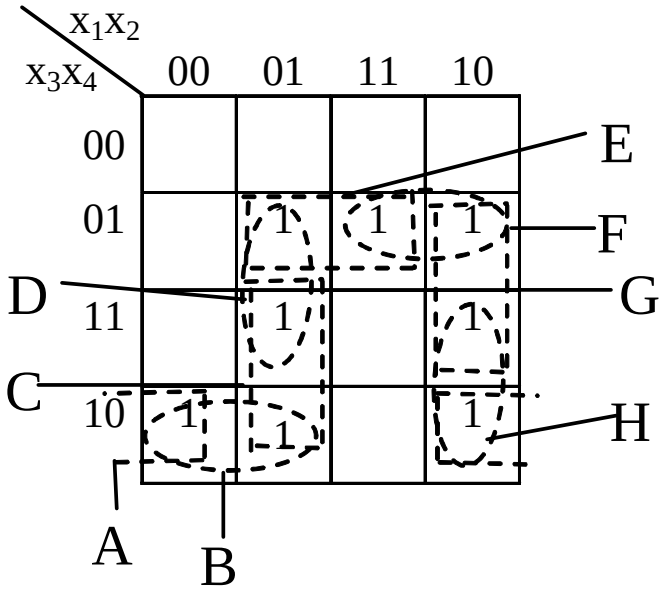
Altro esempio

Materiale facoltativo



Altro esempio

Materiale facoltativo



	P_3	P_6	P_7	P_8	P_9	P_{10}	P_{13}	P_{14}
A				1		1		
B				1	1			
C					1		1	
D	1						1	
E	1		1					
F		1	1					
G		1						1
H						1		1

Altro esempio

Materiale facoltativo

- Metodo di Petrick:

  (A+B)(B+C)(C+D)(D+E)(E+F)(F+G)(G+H)(H+A)=

  =ABCDEFGH+BCDEFGH+…+BDFH+…+ACEG+…=

  = BDFH+ACEG

  e quindi si individuano due coperture minime:

  {B,D,F,H} e {A,C,E,G}.

Ricapitolando

Procedimento completo per la minimizzazione di una funzione logica

Specifica della funzione
(tabella di verità)

↓

Ricerca di tutti gli implicanti primi

↓

Estrazione degli implicanti primi essenziali

↓

Selezione di un sottoinsieme dei restanti implicanti che determina una copertura minima

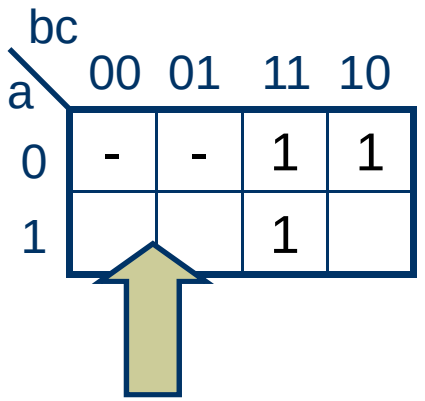
} *metodo algebrico (Quine), tabellare (McCluskey), mappe di Karnaugh*

} *metodo algebrico, righe/colonne dominanti, metodo di Petrick*

Punti di indeterminazione (don’t care)

- rappresentano combinazioni di ingresso per le quali non ci interessa il comportamento della funzione
- indicati con **–**
- ha senso usarli per la **specifica** di una funzione, ovvero in fase di **sintesi**

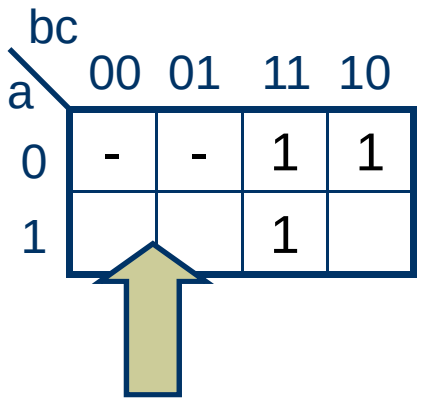
a	*b*	*c*	y
0	0	0	-
0	0	1	-
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*



Punti di indeterminazione (don’t care)

- nell’esempio, non ci interessa il comportamento della funzione quando gli ingressi sono ‘000’ oppure ‘001’
- in effetti, la tabella specifica non una singola funzione ma **4** differenti funzioni tra loro **compatibili**
- se ne può scegliere una qualsiasi in fase di sintesi

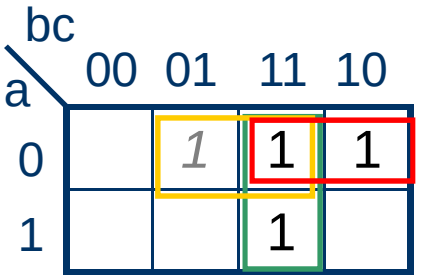
a	*b*	*c*	y
0	0	0	-
0	0	1	-
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*



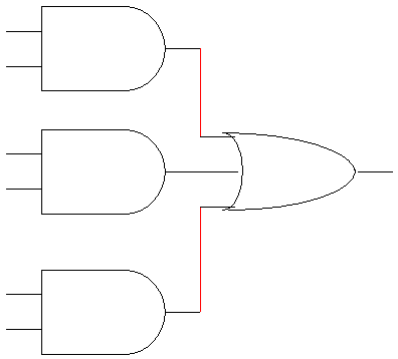
Punti di indeterminazione (don’t care)

due differenti funzioni **compatibili** in virtù dei don’t care:

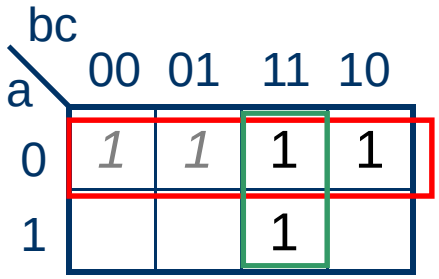
a	*b*	*c*	y
0	0	0	*0*
0	0	1	*1*
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*



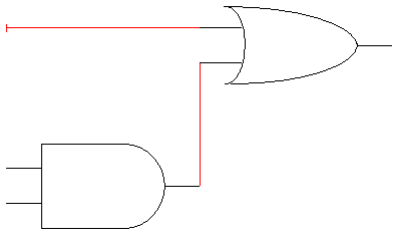
$$f = \bar{a}c + bc + \bar{a}b$$



a	*b*	*c*	y
0	0	0	*1*
0	0	1	*1*
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*



$$f = \bar{a} + bc$$



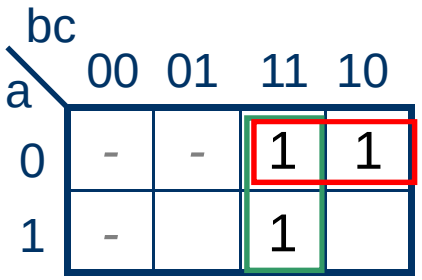
Minimizzazione con don’t care

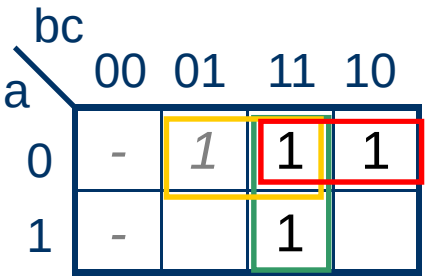
- I don’t care possono essere sfruttati per minimizzare ulteriormente la struttura di una funzione logica

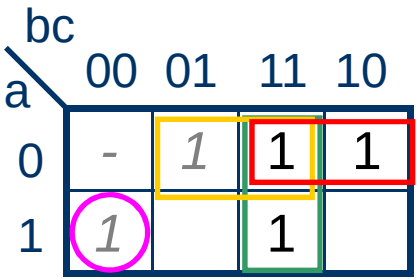
  → *si può cercare tra tutte le funzioni compatibili quella che ha costo minimo*

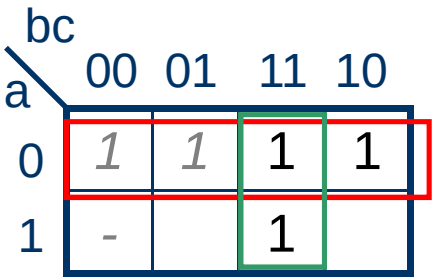
Presenza di don’t care

- Notate che più ‘1’ nella tabella di verità consentono di ottenere implicanti più “larghi”
- D’altro canto, un maggior numero di ‘0’ nella tabella di verità riduce il numero di mintermini da coprire









Presenza di don’t care

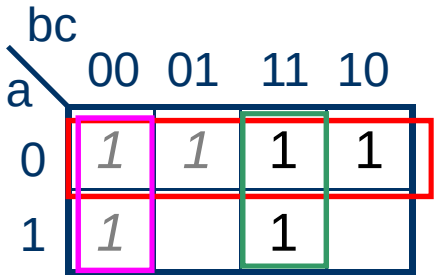
- Notate che più ‘1’ nella tabella di verità consentono di ottenere implicanti più “larghi”
- D’altro canto, un maggior numero di ‘0’ nella tabella di verità riduce il numero di mintermini da coprire

  → conviene considerare i don’t care come ‘1’ quando si cercano gli implicanti primi, e come ‘0’ quando si ricerca la copertura

Presenza di don’t care

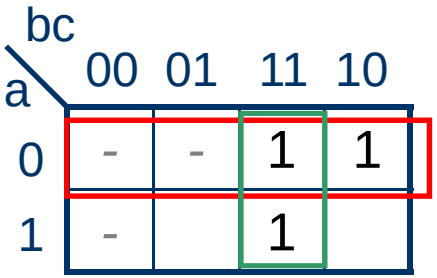
- conviene considerare i don’t care come ‘1’ quando si cercano gli implicanti primi, e come ‘0’ quando si ricerca la copertura

ricerca degli implicanti primi



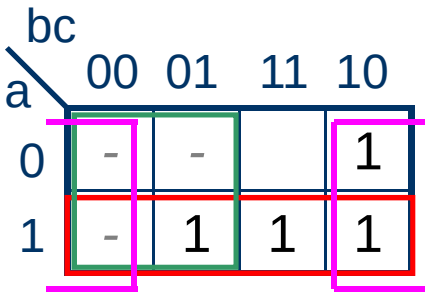
$$\{\overline{a}, \overline{b}\overline{c}, bc\}$$

copertura



$$f = \overline{a} + bc$$

Esempio



	P_2	P_5	P_6	P_7
a				
A = 1--		X	X	X
$\bar{b}$				
B= -0-		X		
$\bar{c}$				
C= --0	X		X	

implicanti primi ottenuti considerando i don't care pari a 1

matrice di copertura considerando i don't care pari a 0 (presenti solo **4** mintermini)

Esempio

C è un implicante primo essenziale: sarà quindi incluso nella forma minima della funzione, coprendo i mintermini $\mathbf{P_2}$ e $\mathbf{P_6}$

	P_2	P_5	P_6	P_7
A = 1--		X	X	X
B= -0-		X		
C= --0	X		X	

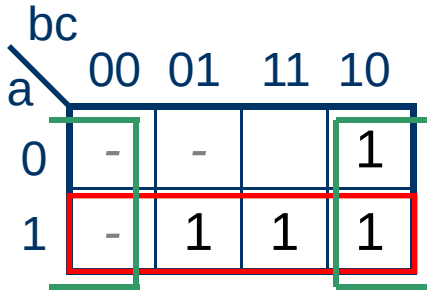
Esempio

La riga **A** domina **B**, che può quindi essere eliminata. **A** copre tutti i restanti mintermini **P_5** e **P_7**

	P_5	P_7
A = 1--	X	X
B= -0-	X	

f può quindi essere scritta come somma di **C** ed **A**:

$$f = a + \bar{c}$$

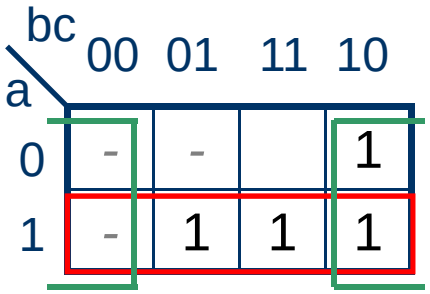


Esempio

Sulla mappa di Karnaugh (utilizzabile per funzioni con pochi ingressi) il procedimento è ancora una volta molto intuitivo:

cercare gli implicanti primi più grandi, coprendo anche i *don't care*, e scegliere per la forma minima solo quelli che effettivamente coprono degli '1'

$$f = a + \bar{c}$$



Minimizzazione in forma S

- procede in maniera ***duale*** a quanto fatto per la forma P
- *maxtermini* in luogo di *mintermini*
- *somme elementari* in luogo di clausole
- al posto degli implicanti consideriamo in questo caso somme che rendono zero la funzione:

$$f(a,b,c,d) = (a+\bar{b})(\bar{a}+c+d)\ldots(\bar{a}+\bar{b}+c+\bar{d})(a+\bar{d})$$

somma elementare

maxtermine

Minimizzazione in forma S

- i consensi sono generati come segue:

$$f(a,b,c,d) = (a+\bar{b})\ (\bar{a}+b+\bar{c}+\bar{d})...(\bar{a}+b+c+\bar{d})\ (a+\bar{d}) =$$

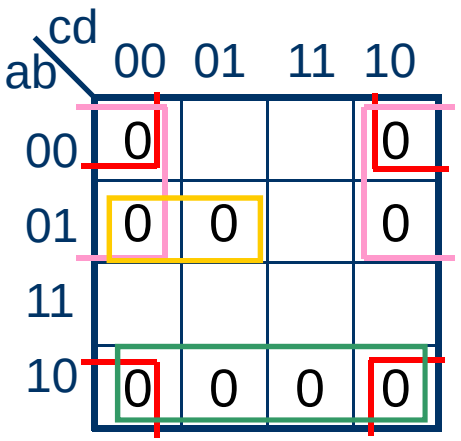
$$(a+\bar{b})\ (c\bar{c}+\bar{a}+b+\bar{d})...(a+\bar{d}) = (a+\bar{b})(\bar{a}+b+\bar{d})...(a+\bar{d})$$

a	b	c	d	y	
...	...	...	....	...	
1	0	0	1	0	⇦
1	0	1	0	0	
1	0	1	1	0	⇦
1	1	0	0	1	
...	...	...	....	...	
...	...	...	....	...	

notare che maxtermini e somme sono definiti in maniera duale rispetto a mintermini e clausole, e corrispondono agli zero della funzione

Minimizzazione in forma S

a	*b*	*c*	*d*	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*1*
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*0*
1	1	0	0	*1*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*1*

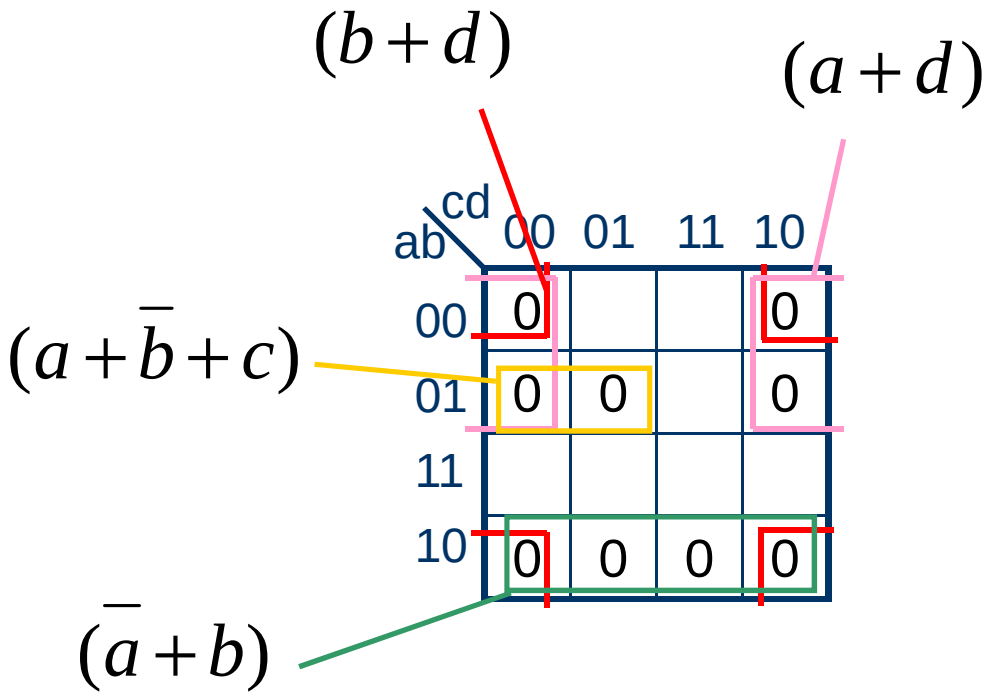


è possibile individuare sulla mappa l'equivalente degli Implicanti Primi per la forma P:

i cubi di area massima che coprano soltanto zero della funzione

Minimizzazione in forma S

a	b	c	d	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*0*
0	0	1	1	*1*
0	1	0	0	*0*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*0*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*0*
1	1	0	0	*1*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*1*



$$f = (\bar{a}+b)(a+d)(a+\bar{b}+c)$$

Minimizzazione in forma S

- mappe di Karnaugh
- metodo di Quine/McCluskey
- metodo di Petrick
- matrice di copertura

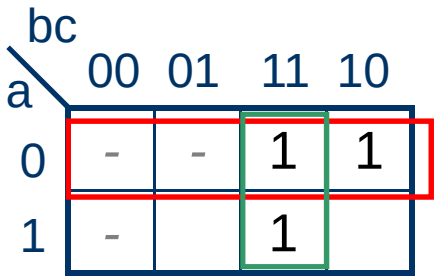
  etc…

possono essere applicati in maniera identica al caso di minimizzazione in forma S

Minimizzazione con don’t care

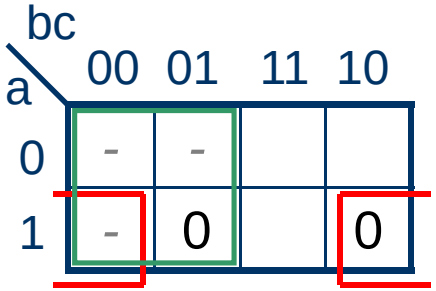
a	*b*	*c*	y
0	0	0	-
0	0	1	-
0	1	0	*1*
0	1	1	*1*
1	0	0	*0*
1	0	1	*0*
1	1	0	*0*
1	1	1	*1*

minimizzazione in forma P



$$f = \bar{a} + bc$$

minimizzazione in forma S



$$f = b(\bar{a} + c)$$

$$= \bar{a}b + bc$$

sono due funzioni diverse…

… ma **compatibili**

Funzioni NAND e NOR

$$x \uparrow y = \overline{x \cdot y} = \overline{x} + \overline{y}$$

$$x \downarrow y = \overline{x + y} = \overline{x} \cdot \overline{y}$$

De Morgan

$$x_1 \uparrow x_2 \uparrow \ldots \uparrow x_n = \overline{x_1 \cdot x_2 \cdot \ldots \cdot x_n} = \overline{x_1} + \overline{x_2} + \ldots \overline{x_n}$$

$$x_1 \downarrow x_2 \downarrow \ldots \downarrow x_n = \overline{x_1 + x_2 + \ldots + x_n} = \overline{x_1} \cdot \overline{x_2} \cdot \ldots \cdot \overline{x_n}$$

Funzioni NAND e NOR

- NAND e NOR ***non*** godono della proprietà associativa

$$(x_1 \uparrow x_2) \uparrow x_3 \neq x_1 \uparrow (x_2 \uparrow x_3) \neq x_1 \uparrow x_2 \uparrow x_3$$

$$(x_1 \downarrow x_2) \downarrow x_3 \neq x_1 \downarrow (x_2 \downarrow x_3) \neq x_1 \downarrow x_2 \downarrow x_3$$

- E’ possibile ottenere una NOT tramite NAND e NOR

$$x \uparrow 1 = \overline{x \cdot 1} = \overline{x}$$

$$x \downarrow 0 = \overline{x + 0} = \overline{x}$$

Funzioni NAND e NOR

- Riassumendo, le NAND permettono di ottenere una NOT, una AND e, usando De Morgan, una OR
- Similmente per la NOR
- ricordiamo che {AND,OR,NOT} è un insieme funzionalmente completo, quindi →

{NAND} e {NOR} sono due
insiemi funzionalmente completi

Forme NAND e NOR di una funzione

- una forma elementare di tipo P si trasforma in una forma NAND a due livelli operando come segue:
  - tutti gli operatori si trasformano in NAND, rispettando le priorità;
  - le clausole costituite da un solo letterale vengono negate.

$$f = \gamma_1 + \gamma_2 + \ldots + \gamma_n = \overline{\overline{\gamma_1} \cdot \overline{\gamma_2} \cdot \ldots \cdot \overline{\gamma_n}} = \overline{\gamma_1} \uparrow \overline{\gamma_2} \uparrow \ldots \uparrow \overline{\gamma_n}$$

- Dualmente per la forma di tipo S

Funzione implicazione

- ha due ingressi, *a* e *b*, ordinati
- pari ad 1 se l'ingresso *a* implica l'ingresso *b*
  - non può accadere che *a* sia 1 senza che lo sia *b*

a	*b*	y
0	0	*1*
0	1	*1*
1	0	*0*
1	1	*1*

$$f(a,b) = \bar{a}\bar{b} + \bar{a}b + ab$$
$$= (\bar{a} + b)$$

Forma S

$$a \Rightarrow b$$

Funzioni XOR e EQ

XOR (funzione OR esclusivo, anche detta somma modulo 2)

$$x \oplus y = \overline{x \equiv y} = x\bar{y} + \bar{x}y = (x + y)(\bar{x} + \bar{y})$$

x	*y*	x XOR y
0	0	*0*
0	1	*1*
1	0	*1*
1	1	*0*

EQ (funzione equivalenza)

$$x \equiv y = \overline{x \oplus y} = xy + \bar{x}\bar{y} = (x + \bar{y})(\bar{x} + y)$$

x	*y*	x EQ y
0	0	*1*
0	1	*0*
1	0	*0*
1	1	*1*

Funzione XOR

- Forme alternative:

$$x \oplus y = (x \uparrow \overline{y}) \uparrow (\overline{x} \uparrow y)$$

forma a 2 livelli (costo di porte **5** includendo anche la NOT)

$$x \oplus y = x(\overline{x} + \overline{y}) + y(\overline{x} + \overline{y}) =$$
$$= (x \uparrow (x \uparrow y)) \uparrow (y \uparrow (x \uparrow y))$$

forma a 3 livelli (costo di porte **4** riutilizzando un fattore comune)

Proprietà della XOR

- a differenza di NAND e NOR, la XOR non costituisce un insieme funzionalmente completo
- Permette di ottenere una NOT

$$x \oplus 1 = x \cdot 0 + \overline{x} \cdot 1 = \overline{x}$$

$$x \equiv 0 = x \cdot 1 + \overline{x} \cdot 0 = \overline{x}$$

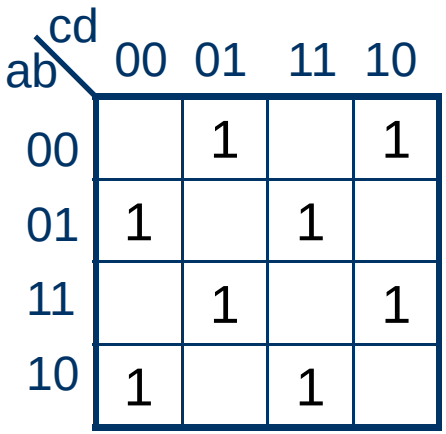
Funzioni Parità e Disparità

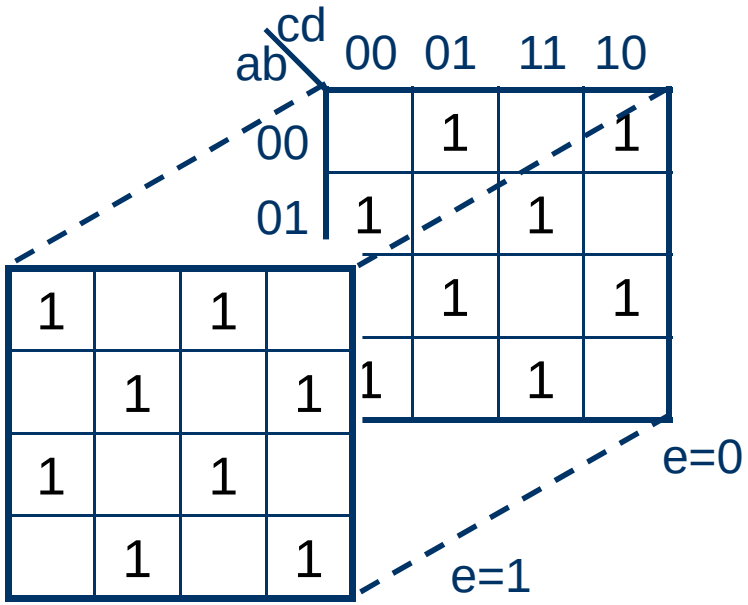
- ***p(X)***: vera se e solo se il numero di ‘1’ presenti in ***X*** è pari
- ***d(X)***: vera se e solo se il numero di ‘1’ presenti in ***X*** è dispari
- si ha ovviamente: $d(X) = \overline{p(X)}$
- Per funzioni a due variabili

$$d(x_1, x_2) = x_1 \oplus x_2$$
$$p(x_1, x_2) = x_1 \equiv x_2$$

Funzioni Parità e Disparità

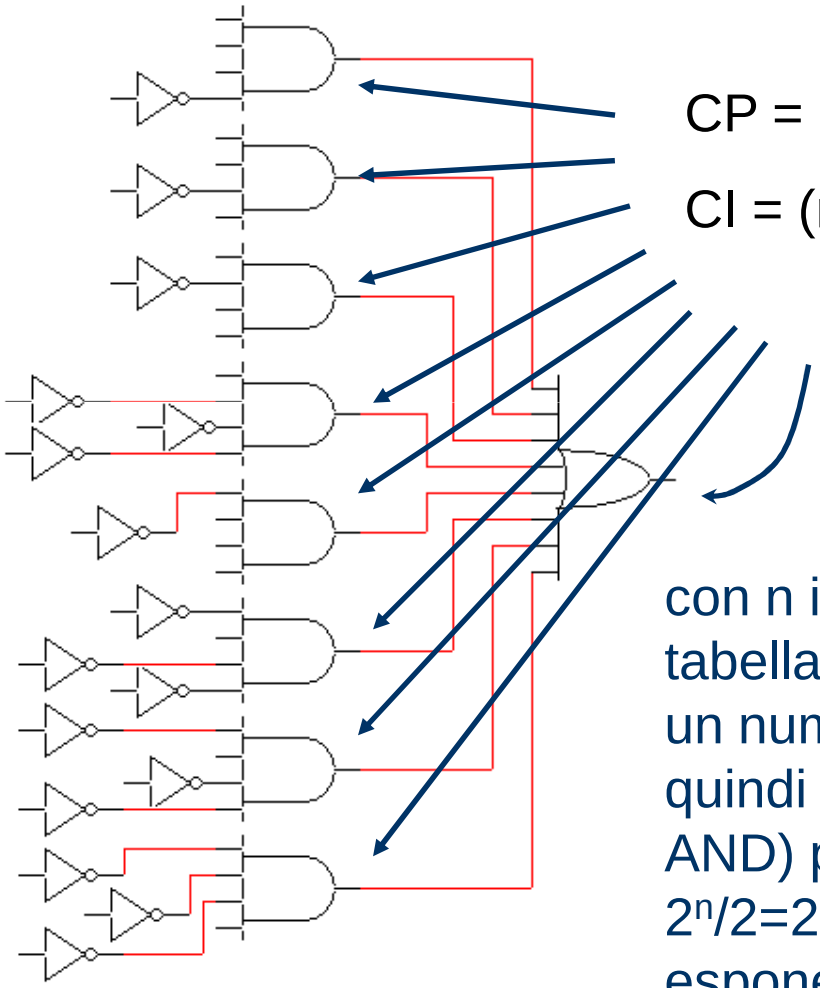
- Si noti che la forma minima di queste due funzioni coincide con la forma canonica
  - Mappa di Karnaugh “a scacchiera”





Funzioni Parità e Disparità

a	*b*	*c*	*d*	y
0	0	0	0	***0***
0	0	0	1	***1***
0	0	1	0	***1***
0	0	1	1	***0***
0	1	0	0	***1***
0	1	0	1	***0***
0	1	1	0	***0***
0	1	1	1	***1***
1	0	0	0	***1***
1	0	0	1	***0***
1	0	1	0	***0***
1	0	1	1	***1***
1	1	0	0	***0***
1	1	0	1	***1***
1	1	1	0	***1***
1	1	1	1	***0***



$CP = 2^{n-1} + 1$

$CI = (n + 1)\ 2^{n-1}$

con n ingressi, la tabella di verità ha un numero di **1** (e quindi di porte AND) pari a $2^n/2=2^{n-1}$, (cresce esponenzialmente!)

Funzioni Parità e Disparità

- La funzione disparità può essere scritta come sommatoria dei suoi mintermini *dispari*
- dualmente, la funzione parità è pari alla sommatoria dei suoi mintermini *pari*
- Le funzioni di costo assumono valori inaccettabili per un numero di ingressi alto:
  - Costo di porte: $CP = 2^{n-1} + 1$
  - Costo di ingressi: $CI = (n + 1)\ 2^{n-1}$

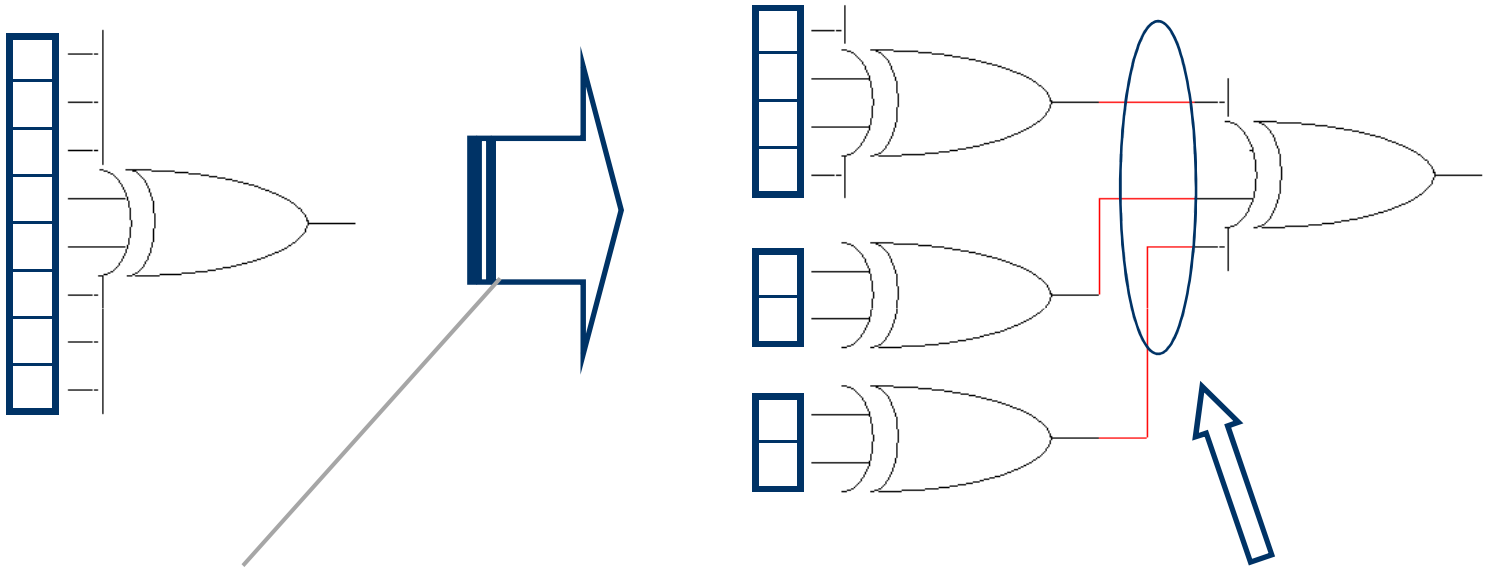
Funzioni Disparità

- La funzione di disparità (a differenza di quella di parità) gode della proprietà **associativa**

$$d(X) = d\left[d(X^{(1)}),\ldots,\ d(X^{(m)})\right]$$

- possiamo partizionare gli ingressi $\boldsymbol{X}$ in gruppi $\boldsymbol{X^{(i)}}$
- la funzione disparità sugli ingressi $\boldsymbol{X}$ è vera se e solo se è dispari il numero di gruppi $\boldsymbol{X^{(i)}}$ con un numero dispari di ‘1’
- Funzioni disparità a più ingressi si possono ottenere come un albero di funzioni disparità “più piccole”

Proprietà associativa della disparità



Proprietà associativa della XOR:si può scomporre una XOR di molti ingressi in una XOR di porte XOR aventi ciascuna pochi ingerssi

il numero di uscite alte è dispari se e solo se è dispari il numero di ingressi alti, indipendentemente da come questi sono partizionati

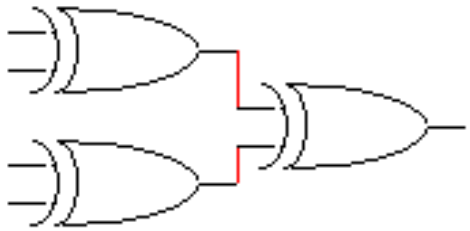
Funzioni Disparità

a	*b*	*c*	*d*	y
0	0	0	0	*0*
0	0	0	1	*1*
0	0	1	0	*1*
0	0	1	1	*0*
0	1	0	0	*1*
0	1	0	1	*0*
0	1	1	0	*0*
0	1	1	1	*1*
1	0	0	0	*1*
1	0	0	1	*0*
1	0	1	0	*0*
1	0	1	1	*1*
1	1	0	0	*0*
1	1	0	1	*1*
1	1	1	0	*1*
1	1	1	1	*0*

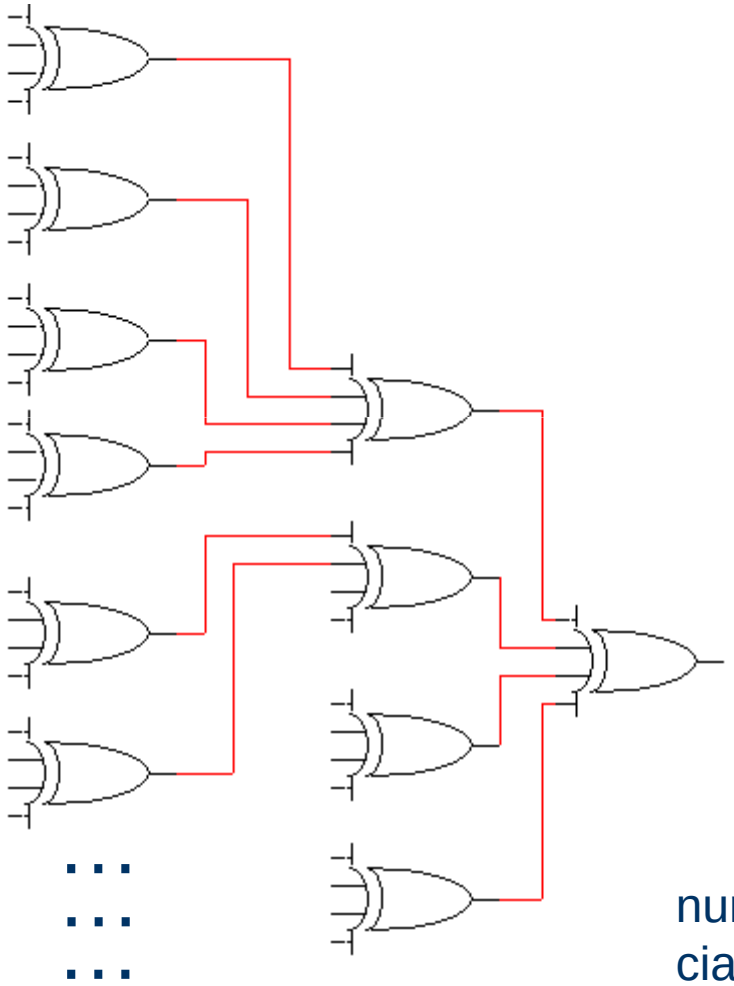
Una XOR a 4 ingressi può quindi essere ottenuta interconnettendo più XOR a 2 ingressi

Il numero di componenti necessarie non cresce più esponenzialmente con il numero di ingressi, ma **linearmente**!

Il **ritardo** della rete però tende ad aumentare, poiché in questo caso occorre attraversare un numero maggiore di porte tra ingresso e uscita



Funzioni Disparità



Esempio di albero di **3** livelli, composto soltanto di XOR di **4** ingressi

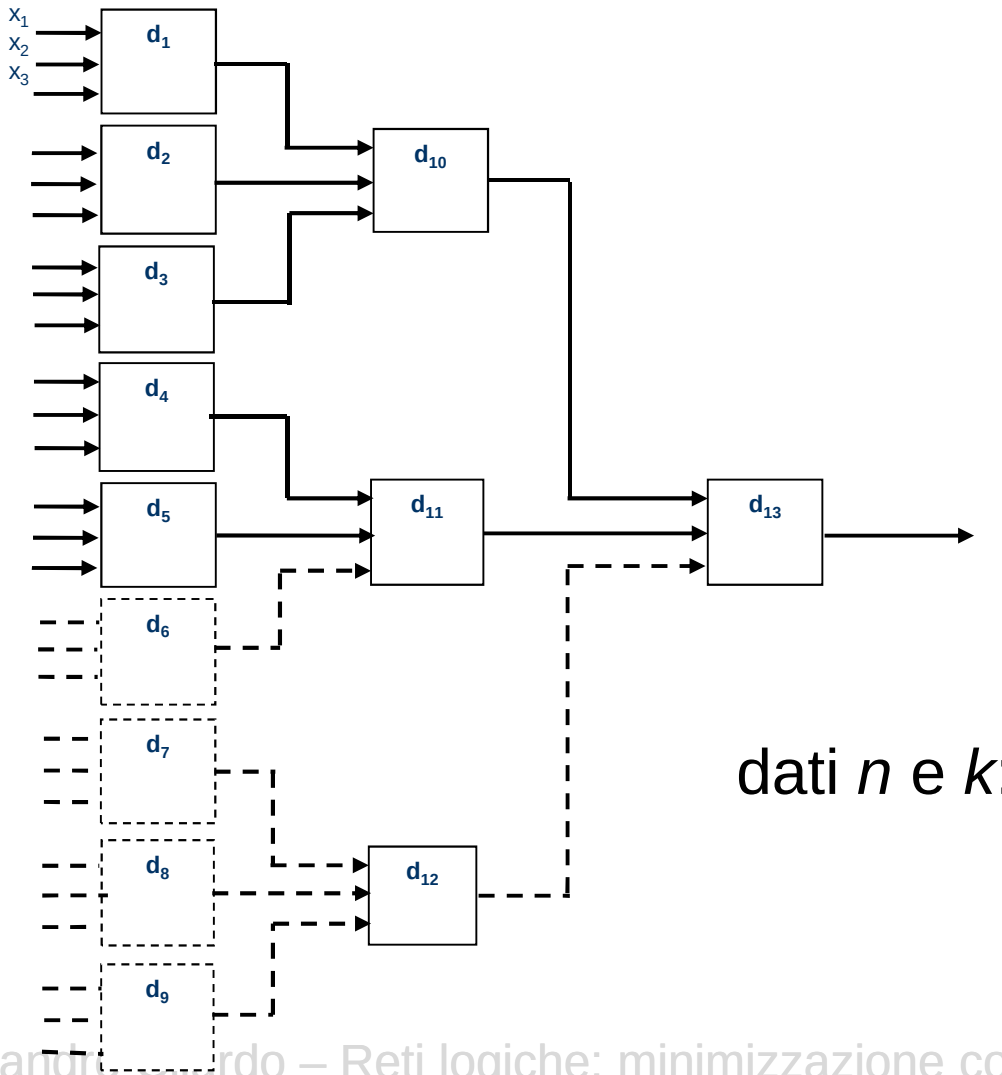
E' possibile ottenere una XOR complessiva avente 4 x 4 x 4 =

$$\mathbf{4^3} = 64 \text{ ingressi}$$

numero di ingressi di ciascuna XOR (4)

numero di livelli (3)

Funzioni Disparità



- *n*: numero di ingressi complessivi
- *k*: numero di ingressi della singola funzione
- *l*: numero di livelli
- *q*: numero di porte

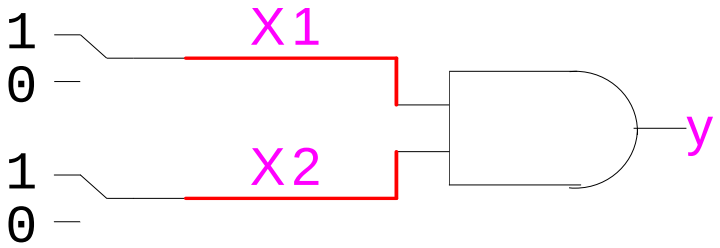
dati *n* e *k*:

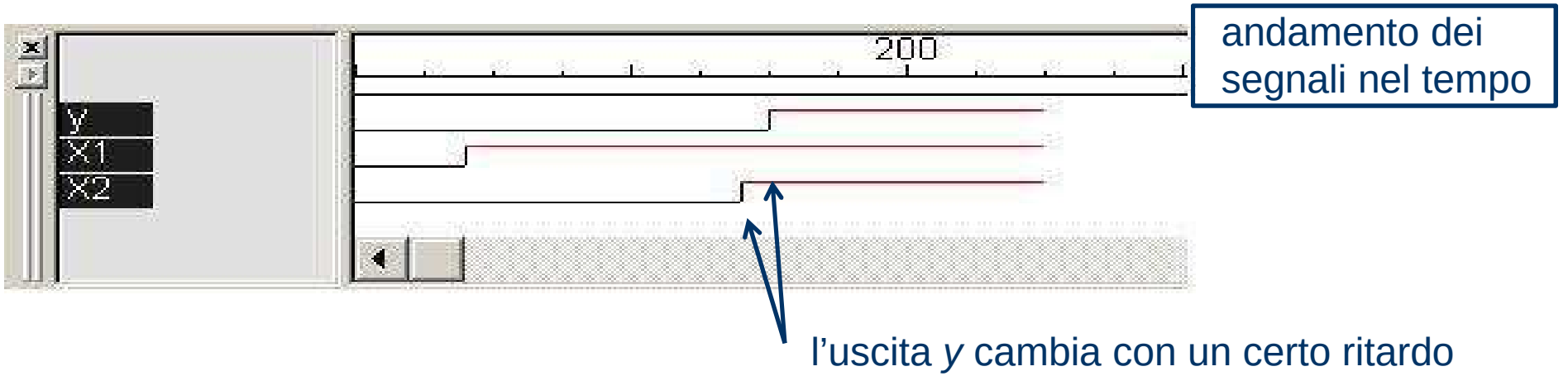
$$l = \lceil \log_k n \rceil$$

$$q = \left\lceil \frac{n-1}{k-1} \right\rceil$$

Il Tempo di risposta

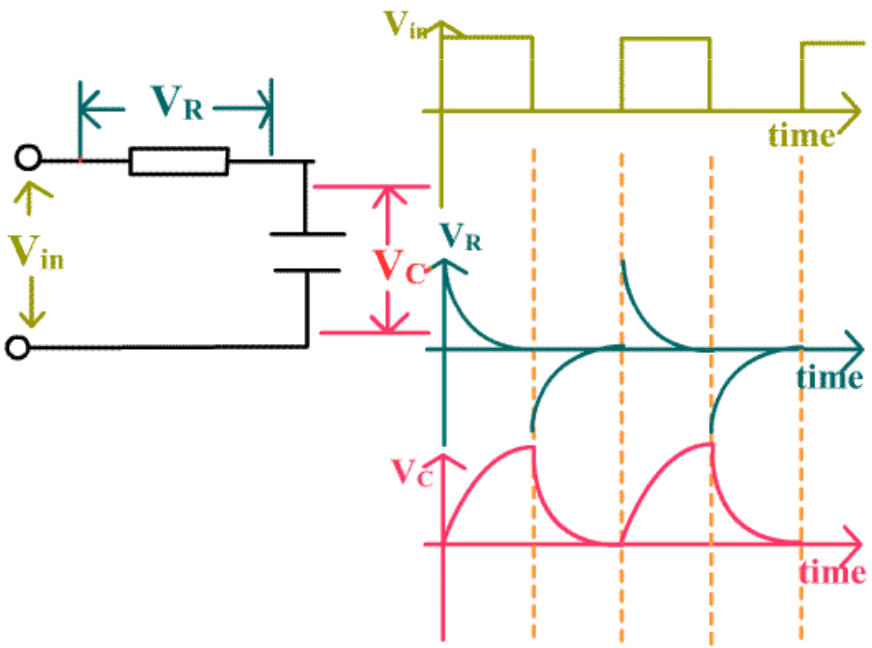
Ritardo $d=t_f - t_i$ con il quale una variazione sull'ingresso è seguita da una variazione sull'uscita





Il Ritardo Inerziale

- ritardo dovuto a fenomeni fisici dinamici causati dalla realizzazione delle porte che “smussano” le transizioni



segnale “logico”: esprime la sequenza 1→0→1→0→1 …

segnale fisico: a causa dell’andamento smussato, potrebbe essere interpretato come una sequenza errata di ‘1’ e ‘0’

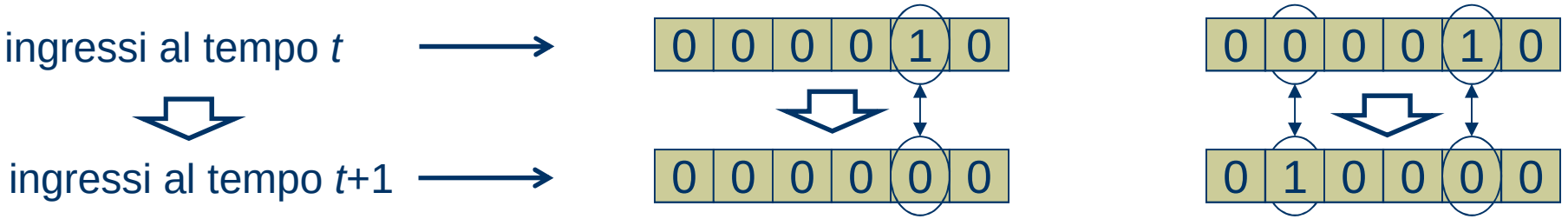
Le Alee (o “*hazards*”)

- La presenza di ritardi nei dispositivi utilizzati può avere l’effetto di **modificare** il comportamento delle uscite in alcuni casi
- Si chiamano **Alee** (*hazards*) quelle situazioni a causa delle quali, per brevi intervalli di tempo (*alee transitorie*) o in maniera permanente (*alee di regime*), le uscite assumono dei valori imprevisti.

Tipologie di transizioni in ingresso

- Per lo studio delle alee è importante valutare il modo nel quale si passa da una configurazione dei valori di ingresso a quella successiva nel tempo.
- **Valori adiacenti** di una codifica:

  Data una codifica, due configurazioni di ingressi si dicono adiacenti se differiscono per una sola variabile

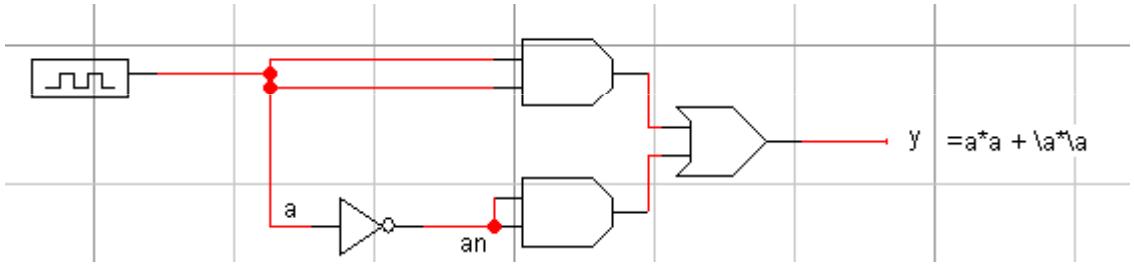


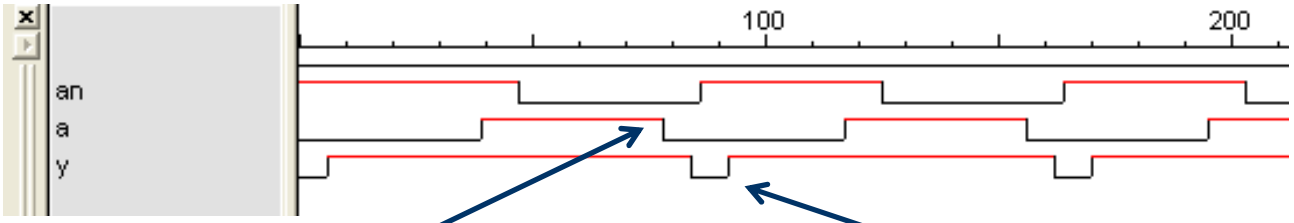
Alee Statiche

Si ha un'**Alea Statica** se avendo due ingressi i_1 e i_2 in sequenza con uscite uguali $f = f(i_1) = f(i_2)$, l'uscita assume per un breve istante di tempo nel passaggio $i_1 \rightarrow i_2$, il valore *NOT(f)* (sbagliato)

Es.

$Y = a\,a + \overline{a}\,\overline{a}$





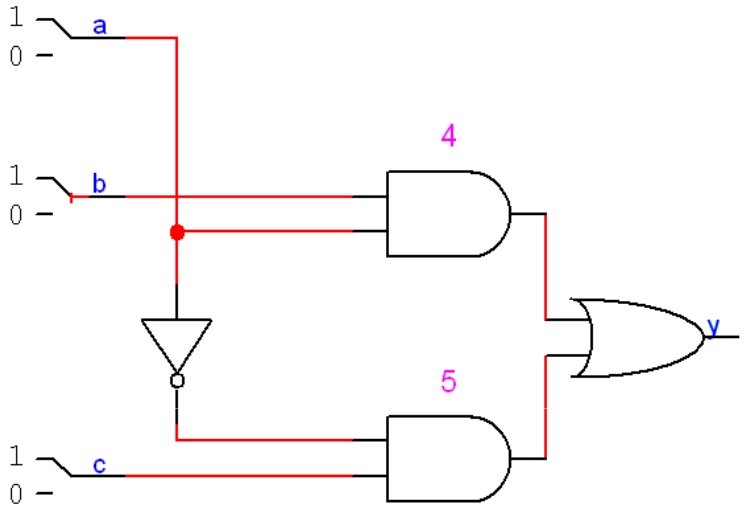
Si manifesta anche per transizioni tra ingressi **adiacenti**!

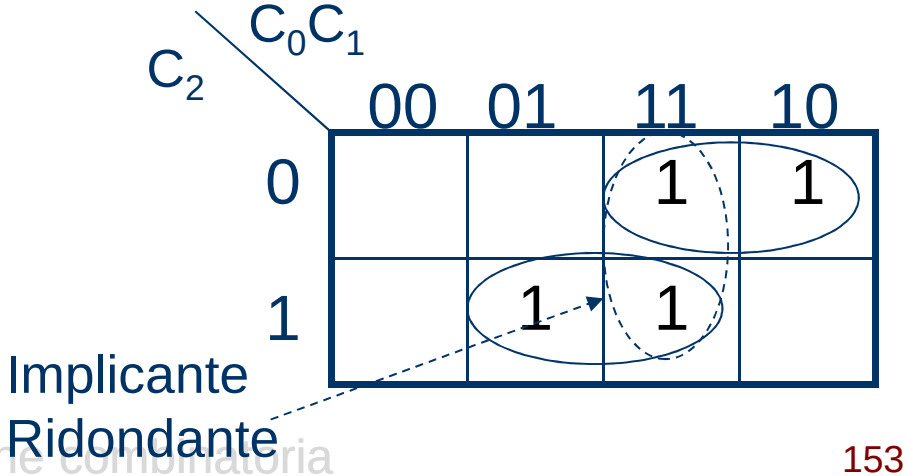
a passa da 1 a 0, quindi $\overline{a}$ passa da 1 a 0, ma con un certo ritardo…

…questo fa sì che *y* assuma un valore sbagliato per un certo (breve) tempo

Alea Statica - Soluzione

- Il problema è legato ad una doppia variazione dei valori interni della rete a partire dalla variazione di un singolo ingresso
- Aggiungendo gli implicanti ridondanti si “coprono” le variazioni che determinano l’alea
- Non è una buona idea, invece, cercare di bilanciare i ritardi circuitali
  - da un punto di vista elettronico è molto difficile controllare i ritardi delle diverse parti di un circuito



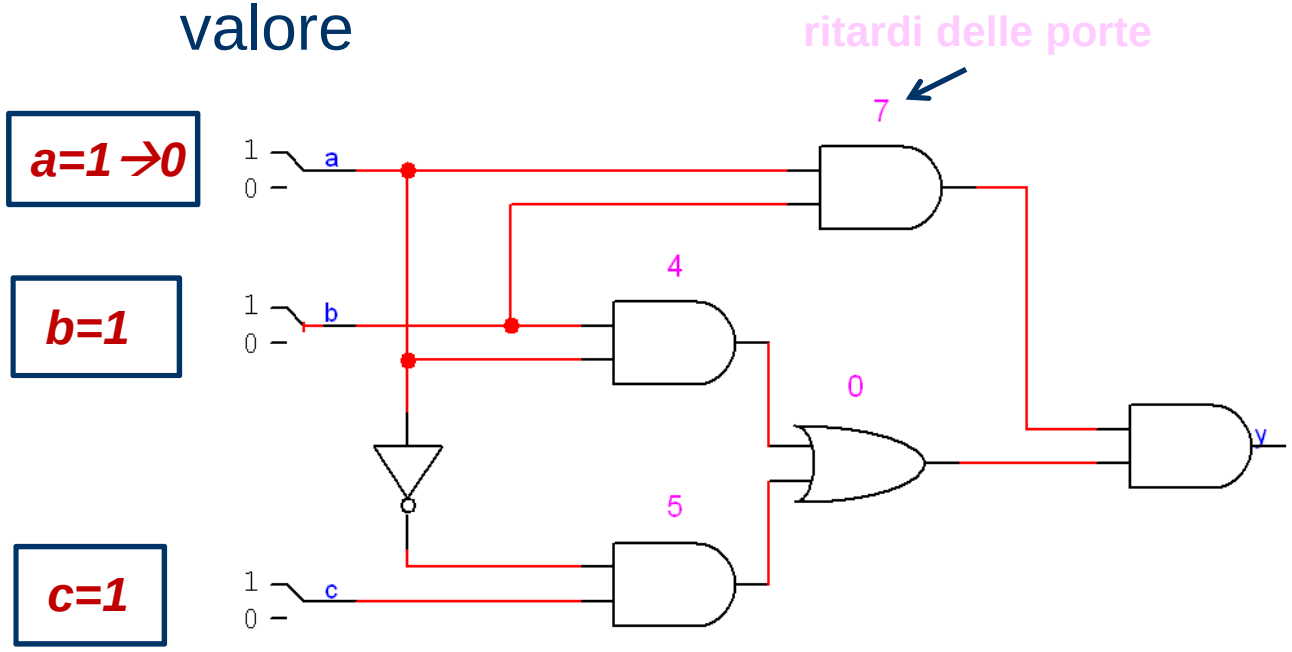


Alea Statica - Soluzione

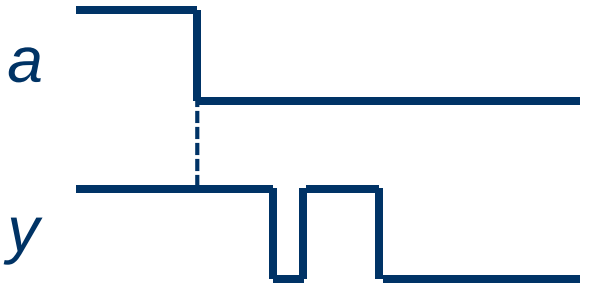
- Una rete AND-OR è esente da alee statiche se nel primo livello della rete sono presenti tutti gli implicanti che coprano transizioni 1→1
- La rete è esente da alee per transizioni 0→0
- Dualmente per le reti OR-AND
- Le alee statiche si possono eliminare aggiungendo *ridondanza* alla rete, come visto prima

Alee Dinamiche

Le alee dinamiche riguardano situazioni in cui, in corrispondenza di un cambiamento degli ingressi, si ha un cambiamento dell'uscita, ma…
prima di assumere il valore corretto, l'uscita cambia più volte valore

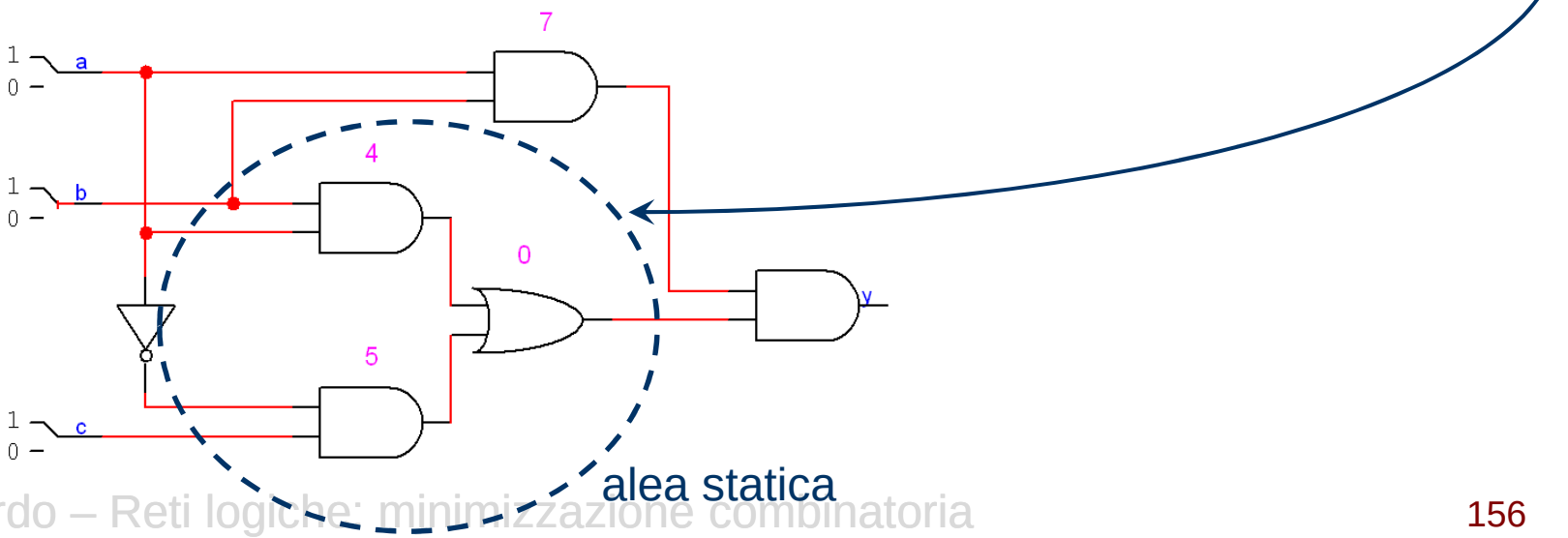


Ciò accade per esempio qui quando i tre ingressi ***abc*** passano da **111** a **011**
A causa dei ritardi delle porte, l'uscita ***y*** avrà il seguente andamento:



Alee Dinamiche

- Si verificano solo in reti a più di due livelli.
- Sono dovute ad alee statiche nei livelli precedenti
- Si eliminano eliminando le alee statiche nelle sottoreti componenti



Alee Multiple

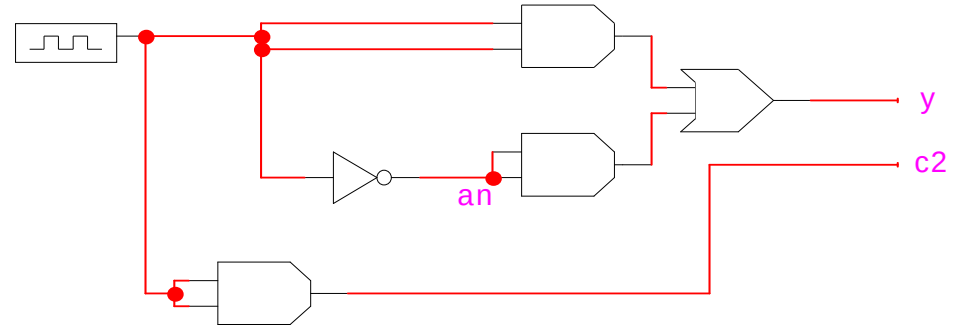
Si ha in generale un'*Alea Multipla* in corrispondenza di una variazione di ingressi in cui almeno due bit cambino contemporaneamente (gli ingressi non sono adiacenti)

Es.

$Y = X_1X_2 + \overline{X}_1\overline{X}_2$

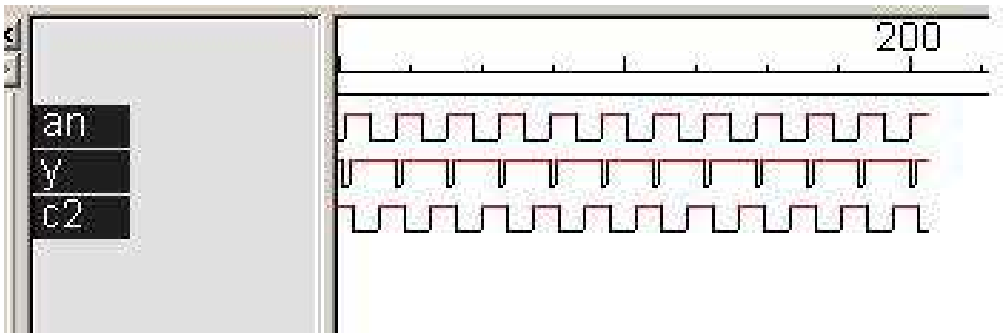
quando:

$X_1X_2 = 00 \rightarrow 11$



Unica Soluzione:

Imporre come vincolo l'applicazione di sequenze di ingressi sempre **adiacenti**

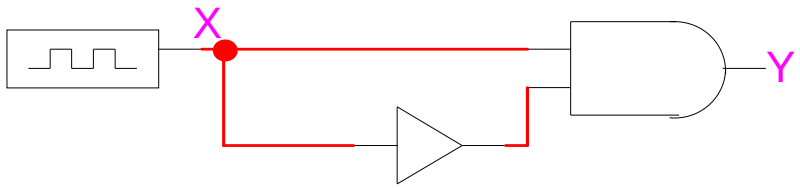


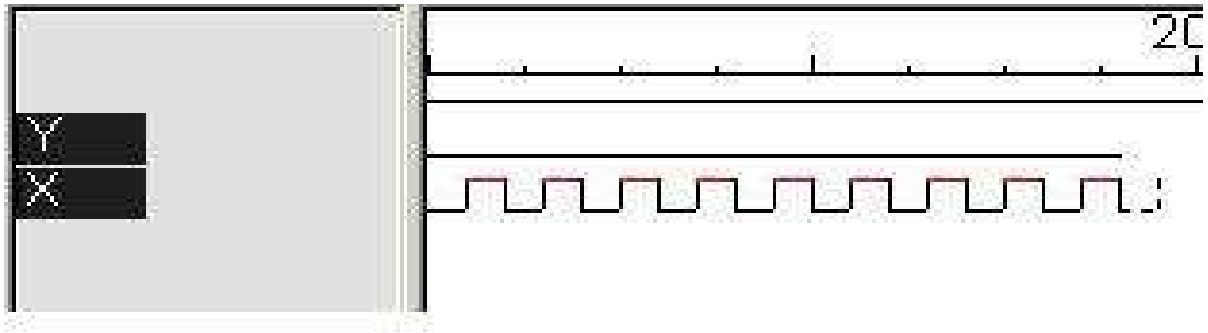
Impulsi Concomitanti

Si ha un'*Alea da impulsi concomitanti* quando il funzionamento del sistema dipende dal fatto che due brevi impulsi brevi si presentino nello stesso momento su due distinti segnali del sistema

Es.

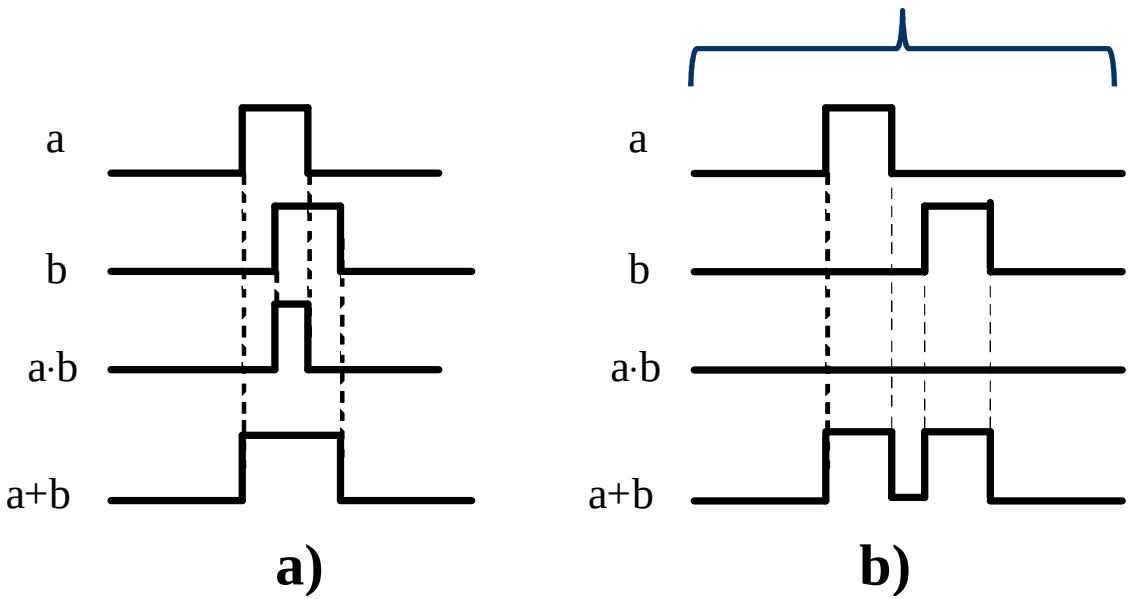
$Y = X_1X_2$





Impulsi Concomitanti

- bastano piccoli disallineamenti accidentali dei due impulsi per alterare la sequenza dei valori prodotti



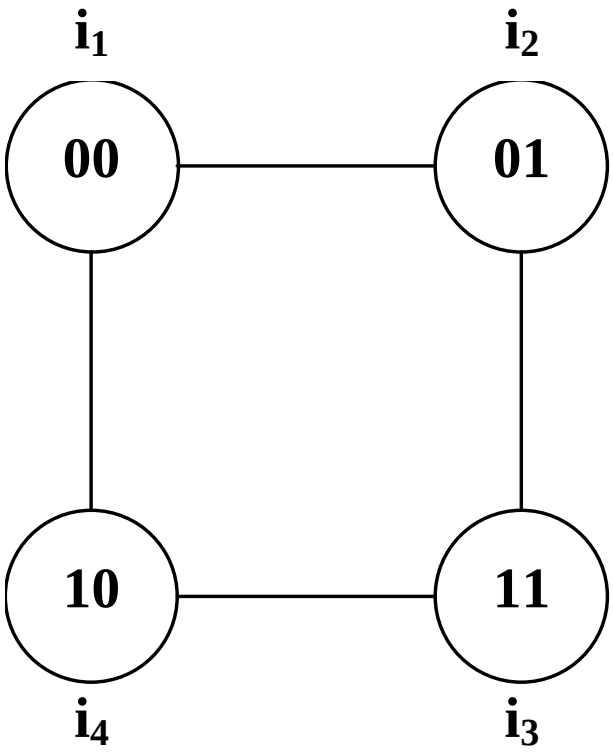
disallineamento trascurabile disallineamento più pronunciato

in questo caso, ad esempio, il disallineamento dei due impulsi su **a** e **b** è tale per cui la loro AND (ab) e la loro OR (a+b) sono totalmente stravolte rispetto al funzionamento corretto

Progetto orientato all'affidabilità

- La presenza di alee *influenza* il progetto delle reti logiche
  - Aggiunta di ridondanza (alee statiche e dinamiche)
  - Codifica degli ingressi (alee multiple)
    - forzare transizioni tra ingressi adiacenti
    - eventualmente aggiungendo variabili (e quindi codifiche dell'ingresso)

Grafo di transizione degli ingressi

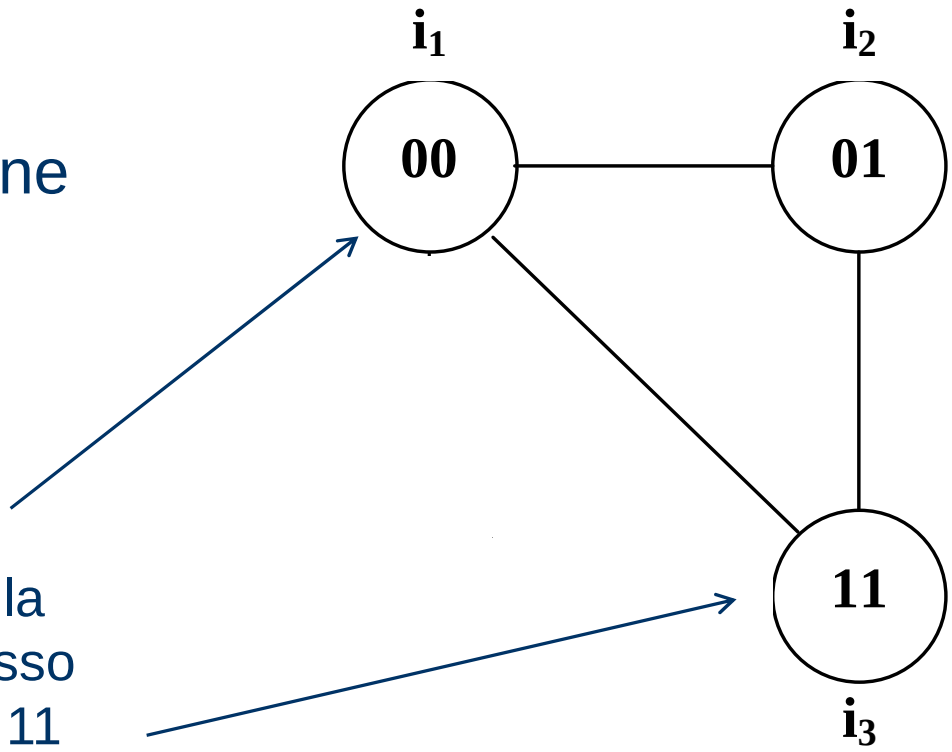


Il grafo esprime tutti i possibili modi in cui da un ingresso si può passare ad un altro

In questo esempio, le transizioni di ingresso ammesse sono tutte tra **ingressi adiacenti** (ovvero, ingressi che differiscono per un solo bit)
→ non ci sono *Alee Multiple*

Grafo di transizione degli ingressi

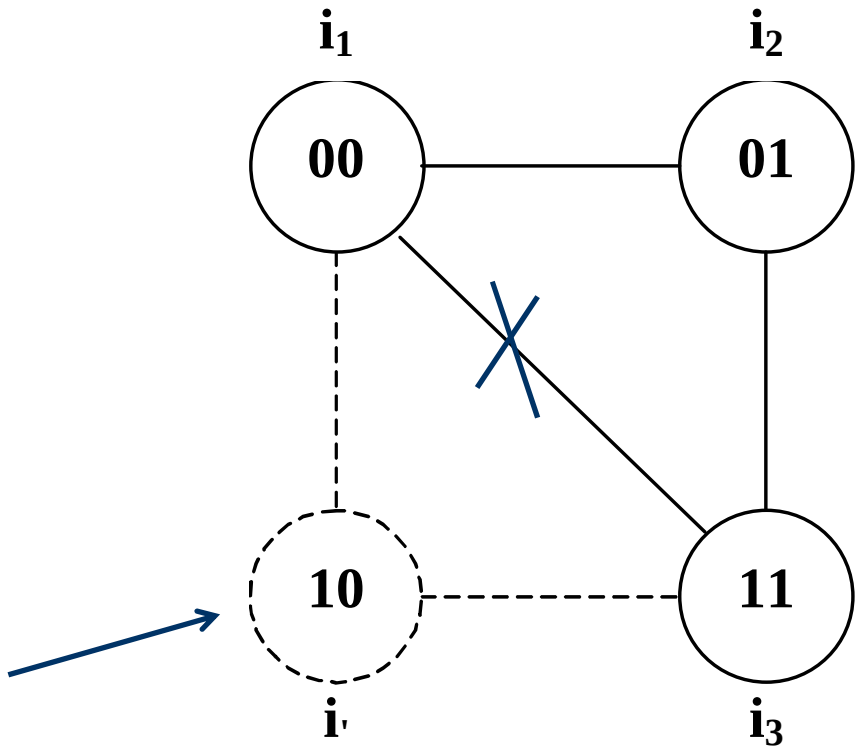
Supponiamo di avere questo grafo di transizione degli ingressi



Il passaggio diretto dalla configurazione d'ingresso 00 e la configurazione 11 introduce un'**alea multipla**

Aggiunta di configurazioni di ingresso

per ottenere sempre transizioni tra **ingressi adiacenti**, aggiungiamo “artificialmente” un passaggio intermedio tra **00** e **11** attraverso la configurazione **10**



Aggiunta di configurazioni di ingresso

Se il grafo iniziale prevede molte transizioni, può diventare necessario aggiungere molti passaggi in modo da avere solo transizioni tra ingressi adiacenti