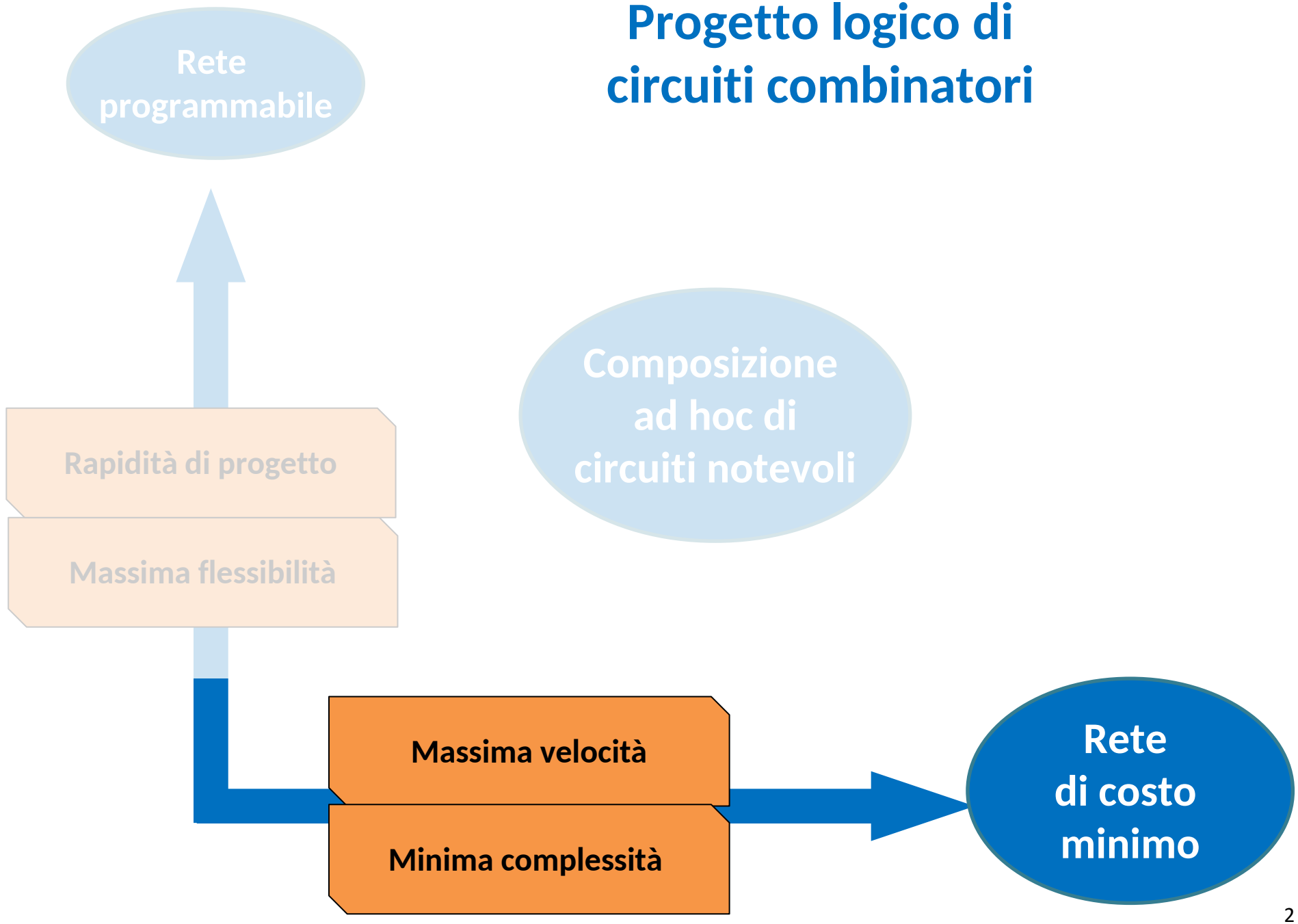
Reti di Costo Minimo

Reti Logiche T

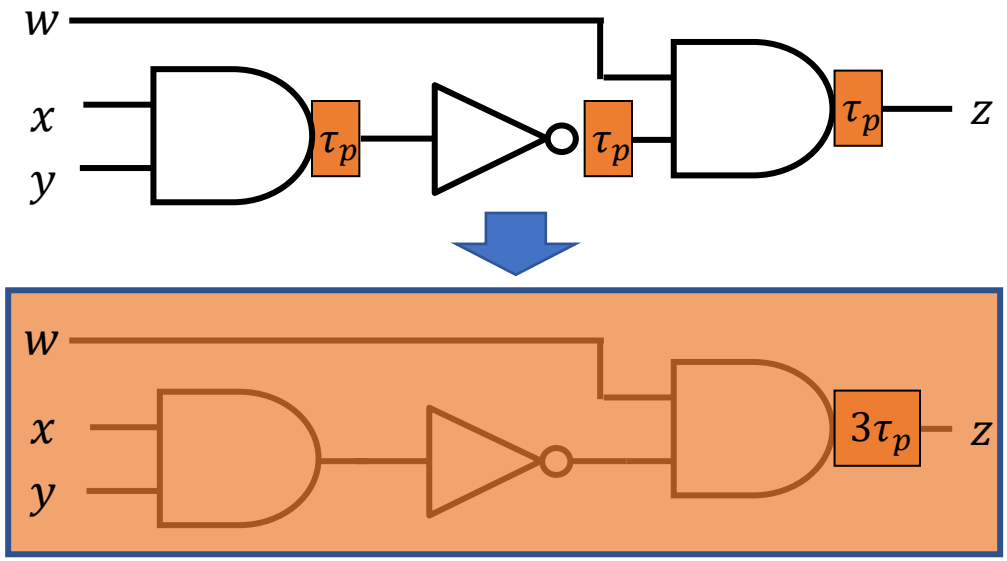
Ingegneria Informatica

Progetto logico di circuiti combinatori



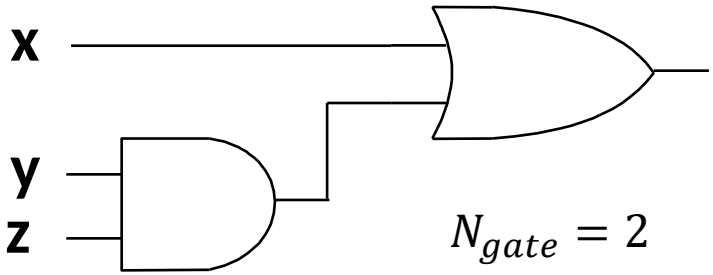
Ritardi e velocità

- Abbiamo già visto che la differenza principale tra l'astrazione fornita dalla tabella della verità e un gate reale è il **ritardo di propagazione**
- Quando cambia un ingresso di un gate, **l'uscita non cambia istantaneamente**, ma dopo un tempo τ_p che dipende dalla tecnologia utilizzata
- τ_p è diverso da gate a gate e anche se il segnale passa da **H** a **L** o viceversa. In prima approssimazione, assumeremo che ogni gate introduca un ritardo τ_p.
- Quindi, il ritardo complessivo introdotto da una rete combinatoria è dato, nel **caso peggiore**, dal numero di gate sul percorso più lungo tra ingressi e uscite
- Questo è anche il ritardo che assegneremo alla rete complessiva.



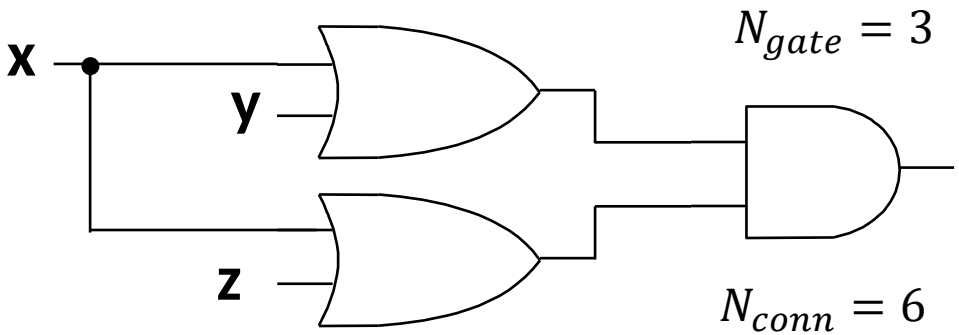
Complessità e velocità

- Per confrontare **complessità** e **velocità di risposta** di reti combinatorie **equivalenti** useremo **3 indicatori** :
  - N_{gate} = **numero di gate** - maggiore è N_{gate}, maggiore la complessità
  - N_{conn} = **numero di connessioni**, ovvero ingressi – maggiore è N_{conn}, maggiore la complessità
  - N_{casc} = **max numero di gate disposti in cascata**, ovvero in serie tra ingressi e uscita – minore è N_{casc}, maggiore la velocità



$N_{gate} = 2$
$N_{conn} = 4$
$N_{casc} = 2$

x + yz



$N_{gate} = 3$
$N_{conn} = 6$
$N_{casc} = 2$

(x+y) (x+z)

Le due reti sono equivalenti (prop. distrib., E3), hanno la stessa velocità di elaborazione, ma la rete di sinistra è meno complessa.

Reti di “costo minimo”

Ipotesi
- ingressi disponibili in forma vera e negata
- fan-in dei gate grande quanto serve

Rete combinatoria di costo minimo (tipo SP e tipo PS)

Schema logico ed espressione (**espressione minima**) che realizzano una funzione qualsiasi con

1. non più di 2 gate in cascata tra ingressi e uscita
2. minimo numero di gate
3. minimo numero di ingressi per gate.

N.B. - Il numero di gate e/o di connessioni della rete di costo minimo di tipo SP è in generale diverso da quello della rete di costo minimo di tipo PS.
N.B. 2 – E’ possibile che più espressioni dello stesso tipo (SP o PS) siano minime (abbiano cioè eguali valori di N_{gate} e N_{conn}; N_{casc} è sempre ≤ 2).

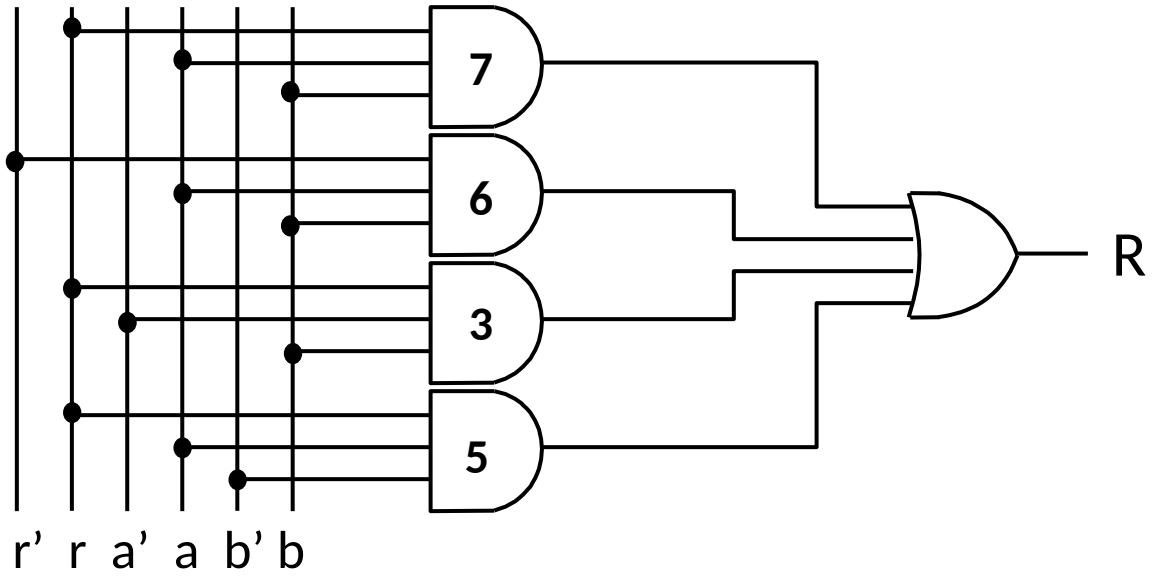
Manipolare espressioni canoniche

- Abbiamo già visto che ogni funzione combinatoria può essere realizzata tramite le espressioni canoniche **a 2 livelli**.
- Il vincolo su N_{casc} e la velocità della rete è già soddisfatto quindi con espressioni canoniche.
- Abbiamo anche visto che attraverso manipolazione algebrica si possono ridurre N_{gate} e N_{conn}
- Obiettivo di questa lezione è fornire uno strumento per rendere più semplice e ripetibile l'individuazione delle manipolazioni che conducono all'espressione minima: **le mappe di Karnaugh**.

Richiamo: Sintesi canonica del Full Adder

i	a	b	r	S	R
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

R = a' b r + a b' r + a b r' + a b r



- Nelle espressioni canoniche usiamo **i mintermini per «riconoscere» una particolare configurazione** degli ingressi in cui la funzione vale «1» e attivare l'uscita.
- L'idea base per ottenere espressioni minime SP è quella di creare dei **termini prodotto che «riconoscano» più di una configurazione** in cui la funzione vale «1»
- Vale il ragionamento duale nel caso PS

Implicanti e implicanti primi

- **Implicante**: termine prodotto di n **o meno** variabili che assume il valore 1 solo per configurazioni in cui la funzione *vale 1 o indifferenza*

3 ingressi (mintermini) $a'b'c', a'b'c, a'bc', a'bc, ab'c', abc$

2 ingressi $a'b', a'b, a'c, a'c', b'c', bc$

1 ingresso a'

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	–
1	0	0	–
1	0	1	0
1	1	0	0
1	1	1	1

- **Implicante primo**: implicante che cessa di essere tale rimuovendo un suo letterale

$$a', b'c', bc$$

Implicanti primi essenziali

- **Implicante primo essenziale**: implicante che è **l'unico** ad assumere il valore 1 per alcune configurazioni delle variabili di ingresso in cui la funzione **assume il valore 1** (cioè non vi sono altri implicanti primi che possono coprire uno o più «uni» della funzione)

$$a', bc$$

- **L'espressione minima SP è la somma di implicanti primi essenziali**

$$F(a, b, c) = a' + bc$$

a	b	c	z	Implicanti primi attivi
0	0	0	1	a' $b'c'$
0	0	1	1	a'
0	1	0	1	a'
0	1	1	–	a' bc
1	0	0	–	$b'c'$
1	0	1	0	
1	1	0	0	
1	1	1	1	bc

- a' è l'unico a valere 1 per le conf. di ingresso 001 e 010 **quindi è essenziale**
- bc è l'unico ad assumere valore 1 per 111, **quindi è essenziale**
- $b'c'$ assume valore 1 per 000, dove anche a' vale 1,**non è essenziale**

Implicati e implicati primi

- **Implicato**: termine **somma** di n *o meno* variabili che assume il valore **0** solo per configurazioni in cui la funzione *vale* **0** *o indifferenza*

$$a + b' + c', a' + b + c\ , a' + b + c', a' + b' + c$$
$$a' + c\ , a' + b$$

- **Implicato primo**: implicato che cessa di essere tale rimuovendo un suo letterale

$$a + b' + c', a' + c\ , a' + b$$

- **Implicato primo essenziale**: implicato che è l'unico ad assumere il valore **0** per alcune configurazioni delle variabili di ingresso in cui la funzione **assume il valore 0** (cioè non vi sono altri implicati primi che possono coprire uno o più **zeri** della funzione)

$$a' + c\ , a' + b$$

- **L'espressione minima PS è il prodotto di implicati minimi essenziali**

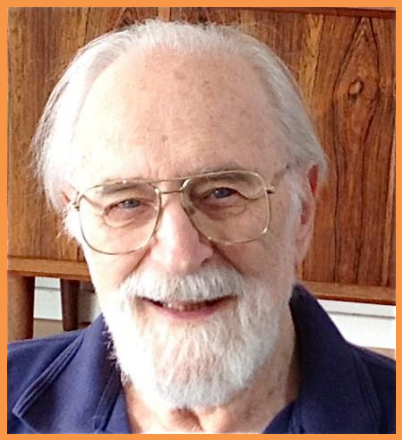
$$F = (a' + c)(a' + b)$$

a	b	c	z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	–
1	0	0	–
1	0	1	0
1	1	0	0
1	1	1	1

Metodi per la determinazione dell'espressione minima

Metodi algoritmici
(Quine-Mc Cluskey, Petrick)
consentono di trattare funzioni con un numero qualsiasi di variabili e vengono tipicamente eseguiti da un calcolatore.

Metodo grafico
(Mappe di Karnaugh)
consente di trattare agevolmente funzioni fino a 6 variabili e viene eseguito manualmente.



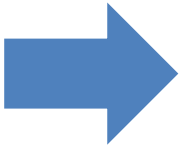
"The Map Method for Synthesis of Combinational Logic Circuits", Trans. AIEE. pt I, 72(9):593-599, Nov. 1953

Mappe di Karnaugh

Mappa di Karnaugh - Rappresentazione **bidimensionale** della **tabella della verità** di una funzione di 2,3,4 variabili, i cui valori sono elencati sui bordi in maniera tale che due configurazioni consecutive differiscano per il valore di un solo bit (**codice di Gray**).

Esempi: Uscita R del Full Adder

a	b	r	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



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

Mappe di Karnaugh

Mappa di Karnaugh - Rappresentazione **bidimensionale** della **tabella della verità** di una funzione di 2,3,4 variabili, i cui valori sono elencati sui bordi in maniera tale che due configurazioni consecutive differiscano per il valore di un solo bit (**codice di Gray**).

Altri esempi:

a \ b	0	1
0	1	0
1	0	1

Equivalence

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

Uscita S del Full Adder

ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

Exor di 4 variabili

Le mappe permettono di identificare graficamente (dunque in maniera agevole) configurazioni **adiacenti**, ovvero che differiscono per un solo bit, **aventi il medesimo valore di uscita** (cosa a sua volta utile per trovare **implicanti/implicati primi essenziali**)

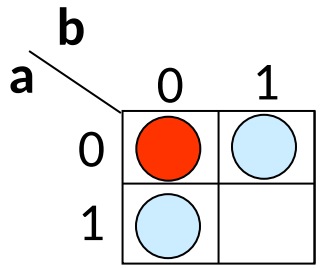
Adiacenza tra celle

Coppia di celle adiacenti su mappe di Karnaugh : due celle le cui coordinate **differiscono per un solo bit**. In una mappa che descrive una funzione di *n* variabili **ogni cella ha *n* celle adiacenti**.

Regola grafica per l'adiacenza - Sono adiacenti celle aventi un lato in comune **o poste all'estremità di una stessa riga o colonna**.

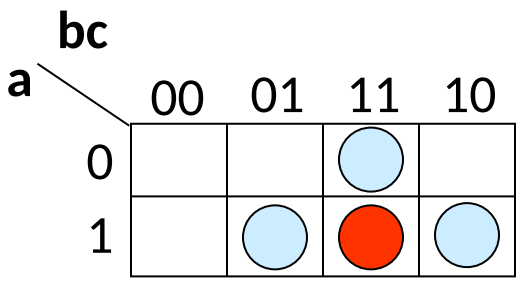
cella scelta come esempio

celle adiacenti



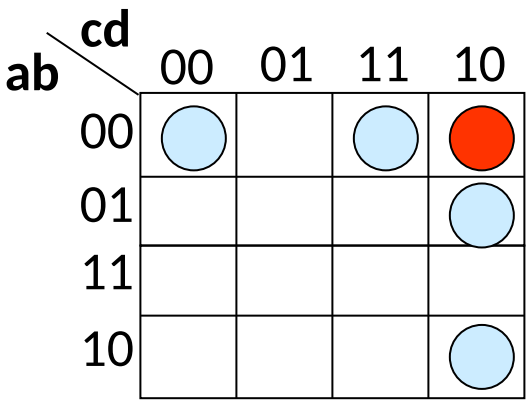
2 variabili

ab=00 adiacente a:
01, 10



3 variabili

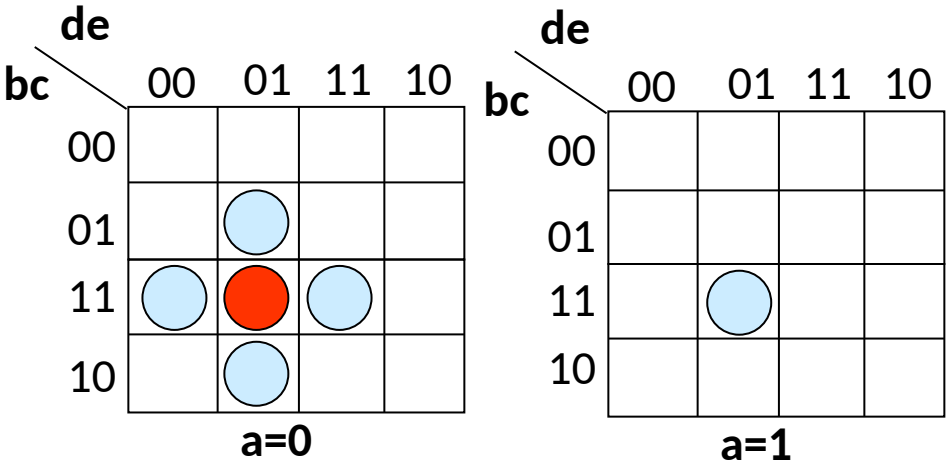
abc=111 adiacente a:
101, 011, 110



4 variabili

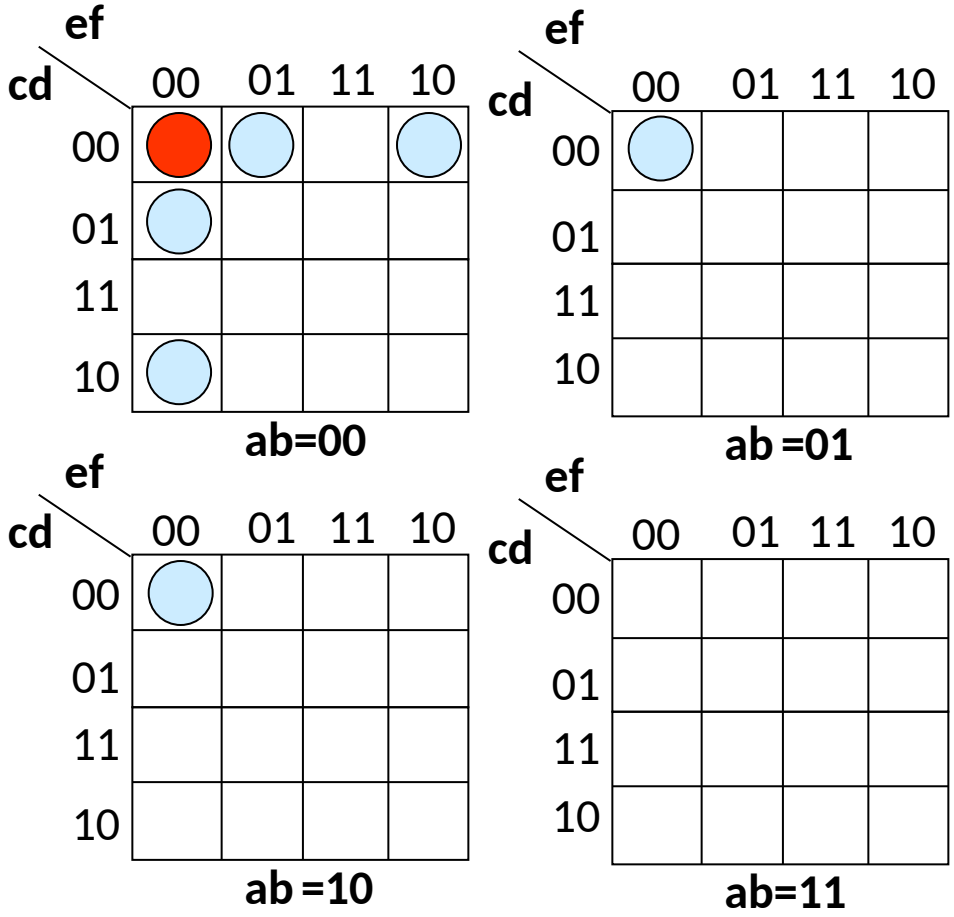
abcd=0010 adiacente a: 0011, 0110
ma anche a: 0000, 1010

Estensione delle mappe a 5 e a 6 variabili



5 variabili

abcde=0 1101 adiacente a:
0 1100, 0 0101, 0 1001, 0 1111
ma anche a: 1 1101



6 variabili

abcdef=00 0000 adiacente a:
00 0100, 00 0001, 00 1000, 00 0010
ma anche a: 10 0000, 01 0000

Ulteriore regola di adiacenza - Sono adiacenti celle che occupano la stessa posizione **in sotto-mappe adiacenti**.

Manipolazione algebrica per via grafica (1)

Perché è così importante individuare celle adiacenti?

ab \ cd	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	1
10	X	X	X	1

canonica SP:
ab'cd' + abcd' + …

E9 COMBINAZIONE

non canonica SP: acd' + …

In questo e negli esempi che seguono, «X» indica che il valore di queste celle non è rilevante per le proprietà che vengono illustrate (**NON** indica un'indifferenza).

ab \ cd	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	0	0	X
10	X	X	X	X

canonica PS:
(a'+b'+c+d')(a'+b'+c'+d') …

E9 COMBINAZIONE

non canonica PS: (a'+b'+d')…

Due termini di una espressione canonica (SP o PS) corrispondenti a configurazioni che individuano celle adiacenti sono equivalenti ad un unico termine con un letterale in meno (**quello che cambia valore**).

Funzioni incomplete

ab \ cd	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	1
10	X	X	X	1

canonica SP:
ab'cd' + abcd' + ...

E9

non canonica SP: acd' + ...

ab \ cd	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	0	-	X
10	X	X	X	X

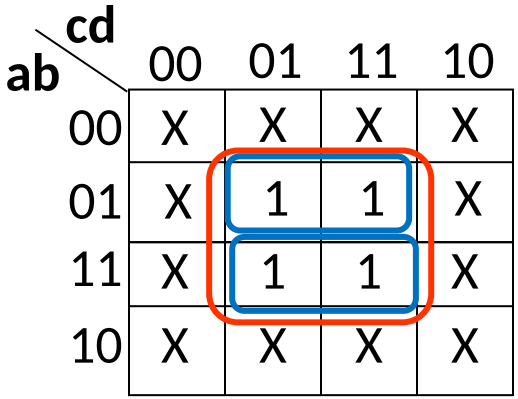
canonica PS: (a'+b'+c+d')(a'+b'+c'+d') ...

E9

non canonica PS: (a'+b'+d')...

Nel caso di funzioni non completamente specificate, si può ridurre il numero di letterali sfruttando celle con condizioni di indifferenza adiacenti a celle usate per definire l'espressione canonica.

Manipolazione algebrica per via grafica (2)



canonica SP: a'bc'd + a'bcd + abc'd + abcd + ...

E9 E9

non canonica SP: a'bd + abd + ...

E9

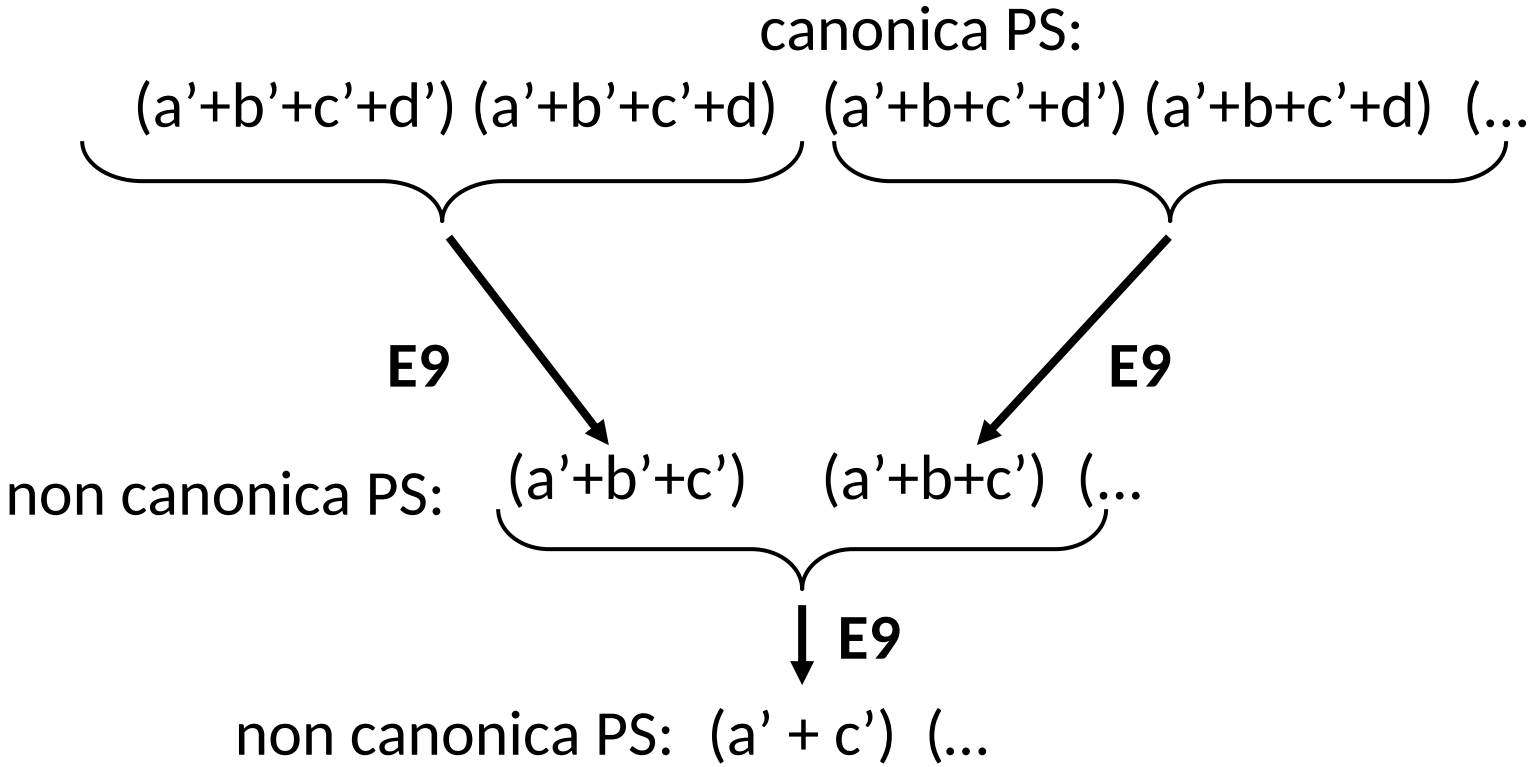
non canonica SP: bd + ...

Quattro mintermini corrispondenti a configurazioni che individuano un **raggruppamento di 4 celle** a 2 a 2 adiacenti sono equivalenti ad un unico termine con **due letterali in meno (quelli che cambiano)**.

Manipolazione algebrica per via grafica (3)

La proprietà è vera anche per quattro maxtermini

ab \ cd	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	0	0
10	X	X	0	0



Raggruppamenti Rettangolari (RR)

Raggruppamento Rettangolare (RR) di ordine p - Insieme di 2^p **celle** di una mappa all'interno del quale ogni cella ha esattamente p **celle adiacenti**.

N.B.: il numero di celle deve essere **una potenza di 2**, non sono possibili RR con 6 o 10 celle!

RR ed implicanti - **Un RR di ordine p** costituito da celle contenenti **valore «1»**, ed eventualmente condizioni di indifferenza, individua un **implicante** della funzione. Nel prodotto compaiono le sole $(n - p)$ variabili che rimangono **costanti** nelle coordinate del RR, in forma vera se valgono «1», in forma complementata se valgono «0».

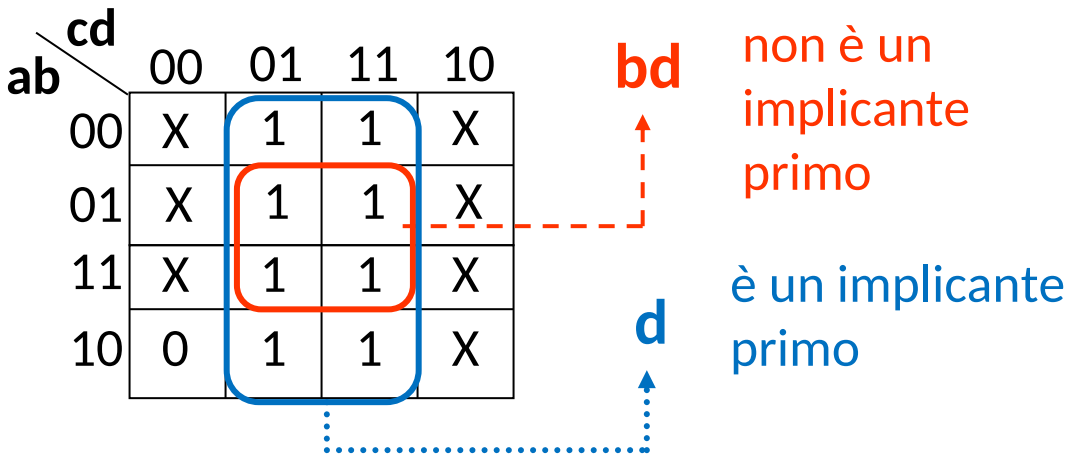
RR ed implicati - **Un RR di ordine p** costituito da celle contenenti **valore «0»**, ed eventualmente condizioni di indifferenza, individua un **implicato** della funzione. Nella somma compaiono le sole $(n - p)$ variabili che rimangono **costanti** nelle coordinate del RR, in forma vera se valgono «0», in forma complementata se valgono «1».

Raggruppamenti, Implicanti e Implicati primi

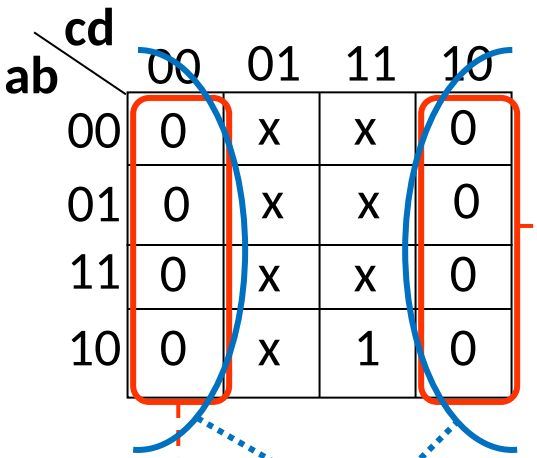
RR di dimensione massima ed implicanti primi - Un RR formato da celle contenenti valore “1” o “-” **e non interamente incluso in un RR di ordine superiore individua un implicante primo**.

RR di dimensione massima ed implicati primi - Un RR formato da celle contenenti valore “0” o “-” **e non interamente incluso in un RR di ordine superiore individua un implicato primo**.

Esempio (caso SP):



Esempio (caso PS)



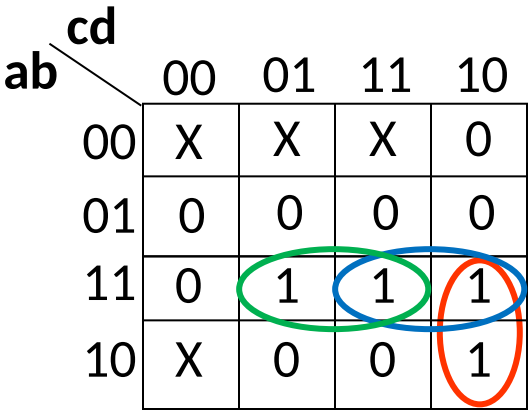
c’ + d non è un implicato primo

non è un implicato primo **c + d**

d è un implicato primo

Individuazione grafica dei termini ridondanti

Un RR le cui celle sono tutte incluse in altri RR
non deve essere usato nell'espressione minima (per E11).
Corrispondono a implicanti (implicati) primi, ma non essenziali.



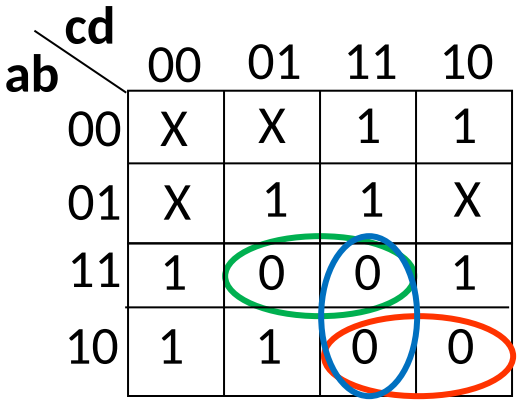
espressione SP:

acd' + abc + abd + ...

a(cd' + bc + bd) + ... **distr. (E3)**

consenso (E11)

acd' + abd + ...



espressione PS:

(a'+b'+d')(a'+c'+d')(a'+b+c') ...

a'+(b'+d')(c'+d')(b+c') ... **distr. (E3)**

consenso (E11)

(a'+b'+d')(a'+b+c') ...

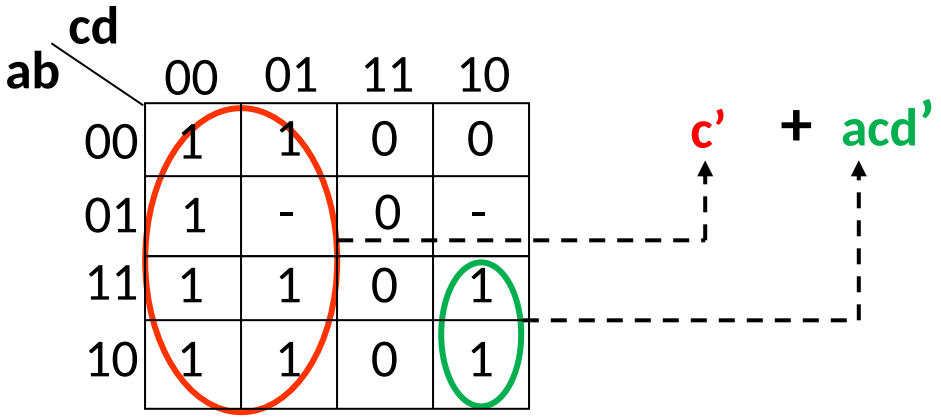
Copertura minima

Copertura di una funzione su una mappa - Insieme di RR la cui unione racchiude tutte le celle contenenti o valore 1 (copertura degli uni) o valore 0 (copertura degli zeri), ed eventualmente celle con valore indifferente.

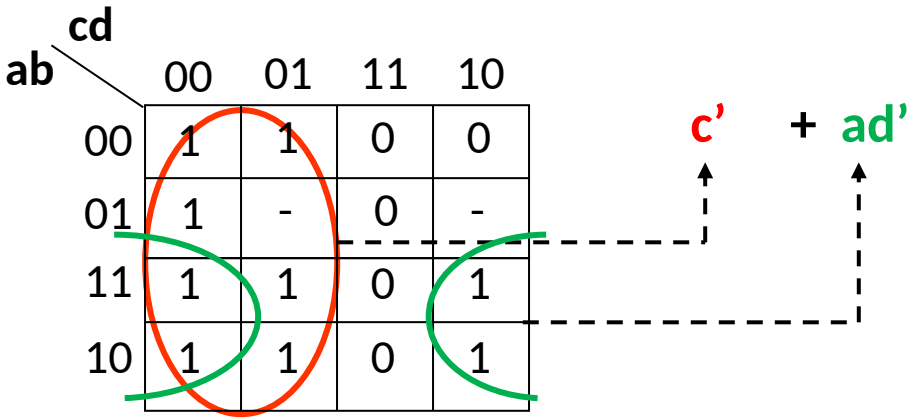
Coperture ed espressioni - **Una copertura degli uni** (zeri) individua una espressione SP (PS) che **fornisce una possibile struttura** per realizzare la funzione assegnata tramite la mappa.
Gli implicanti (implicati) che appaiono nell'espressione sono individuati dai raggruppamenti componenti la copertura.

Copertura minima - Copertura costituita dal **minimo numero possibile di RR di dimensione massima** e corrispondente all'espressione minima.

Coperture ed espressioni (1)



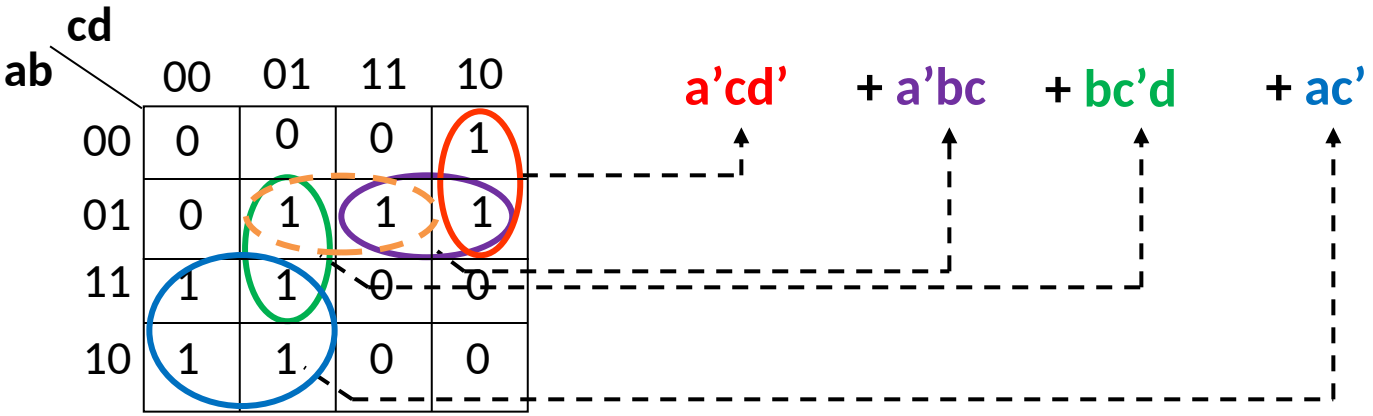
Uno dei due RR non è di dimensione massima (acd' non è un implicante primo): l'espressione non è minima.



Per derivare l'espressione minima SP (PS) celle con valore 1/– (0/-) possono essere racchiuse in più raggruppamenti

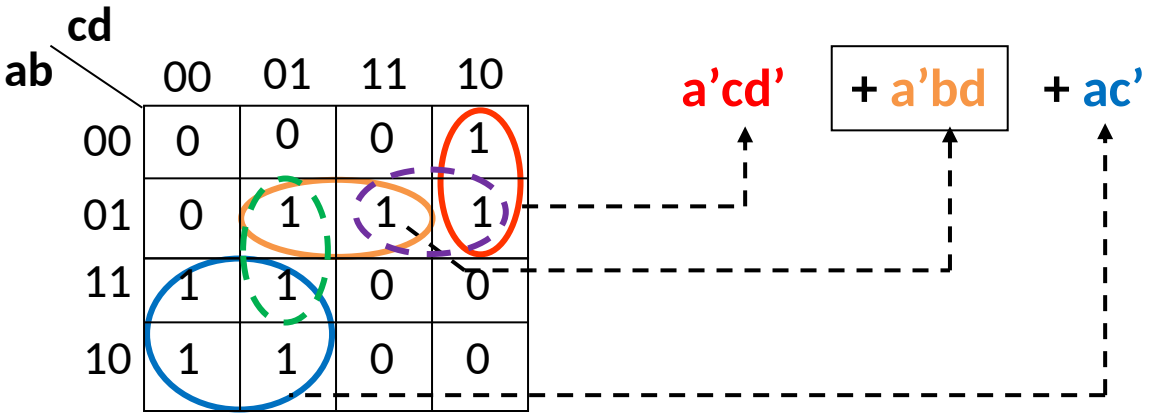
L'espressione è minima

Coperture ed espressioni (2)



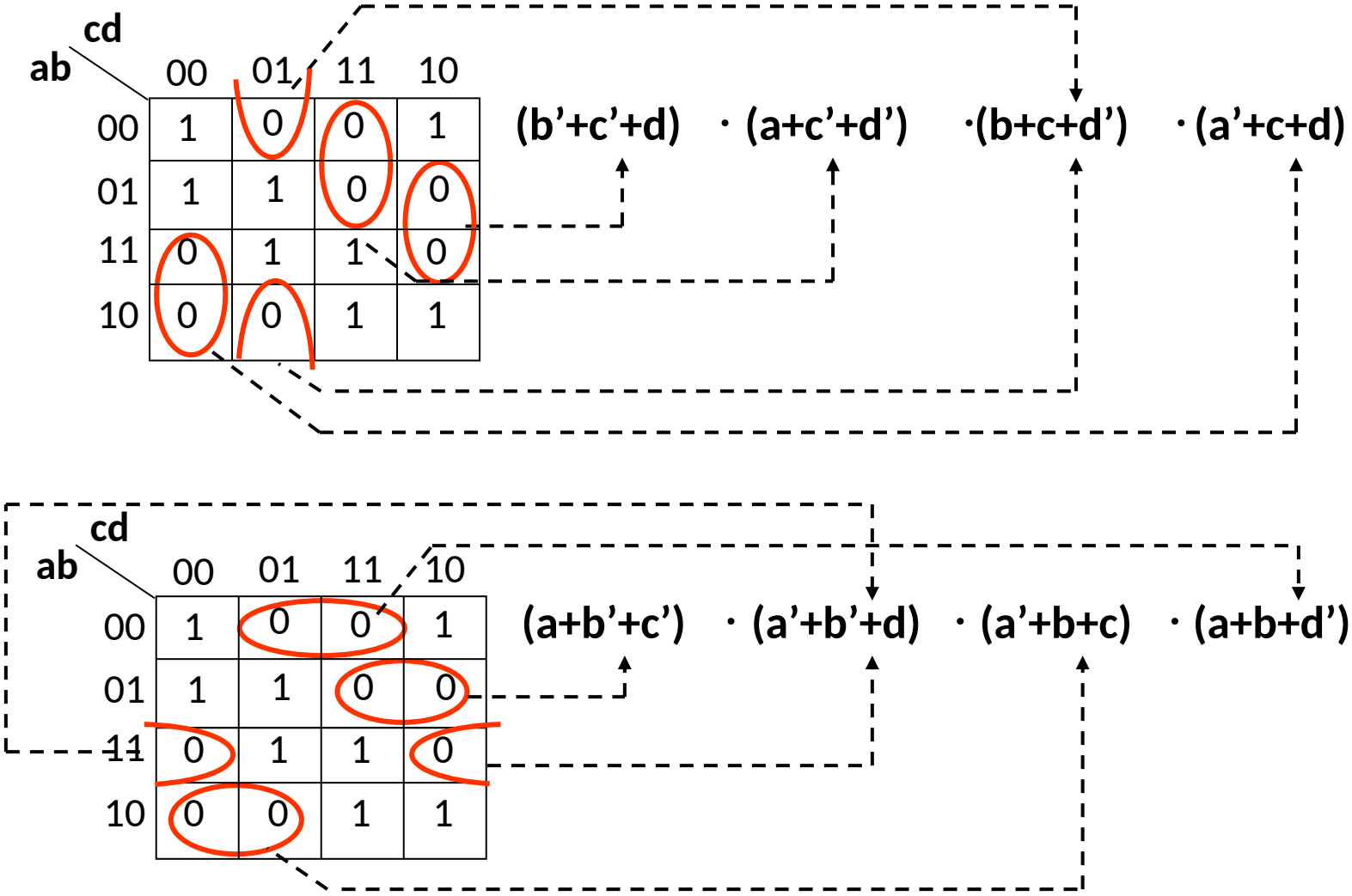
Somma di implicanti primi **ma alcuni non essenziali (a'bc , bc'd):** non è espressione (e quindi copertura) minima

Quando una o più celle possono essere coperte con più RR della stessa dimensione (celle abcd=0101 e 0111 in questo esempio), devo considerarli tutti e scegliere quelli essenziali (a'bd in questo esempio)



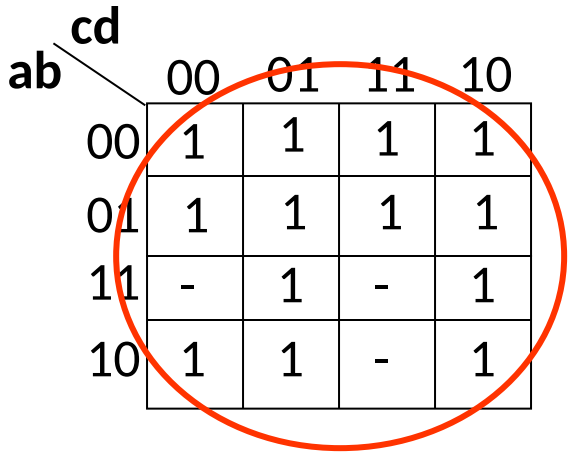
Espressione minima

Coperture ed espressioni (3)

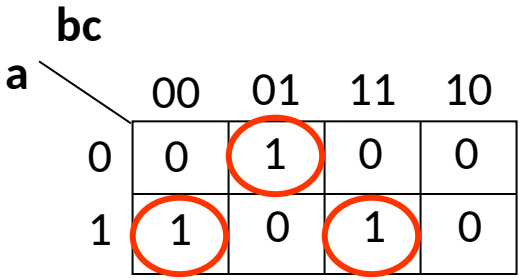


Due espressioni minime equivalenti di tipo PS

Coperture ed espressioni (4)

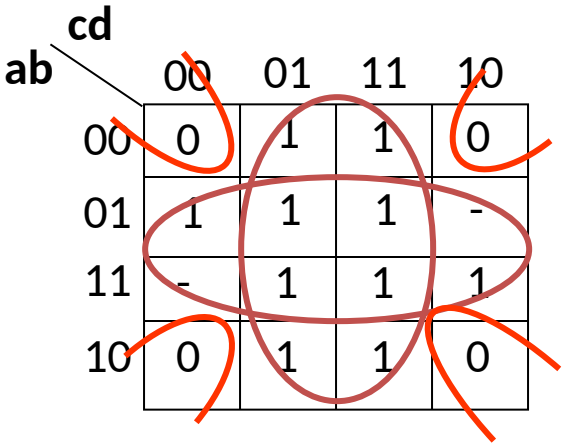


La funzione f(a, b, c, d) è identicamente uguale a 1



a' b'c + ab'c' + abc

L'espressione minima SP è l'espressione canonica



PS: **b + d**

SP: **b + d**

Le coperture minime PS ed SP portano alla stessa espressione

Individuazione grafica dell'espressione minima (1)

A partire dalla mappa che descrive la funzione occorre determinare la copertura minima e da questa la corrispondente espressione minima. Il procedimento è per sua natura non sistematico e presuppone l'abilità di chi lo esegue.

È tuttavia possibile delineare una sequenza di passi che consentono di individuare la copertura minima:

1) Si decide se cercare l'espressione di tipo SP o PS e ci si predispone di conseguenza a coprire gli uni o gli zeri.

ab \ cd	00	01	11	10
00	0	0	0	1
01	0	1	-	-
11	1	1	0	0
10	1	1	0	0

e=0

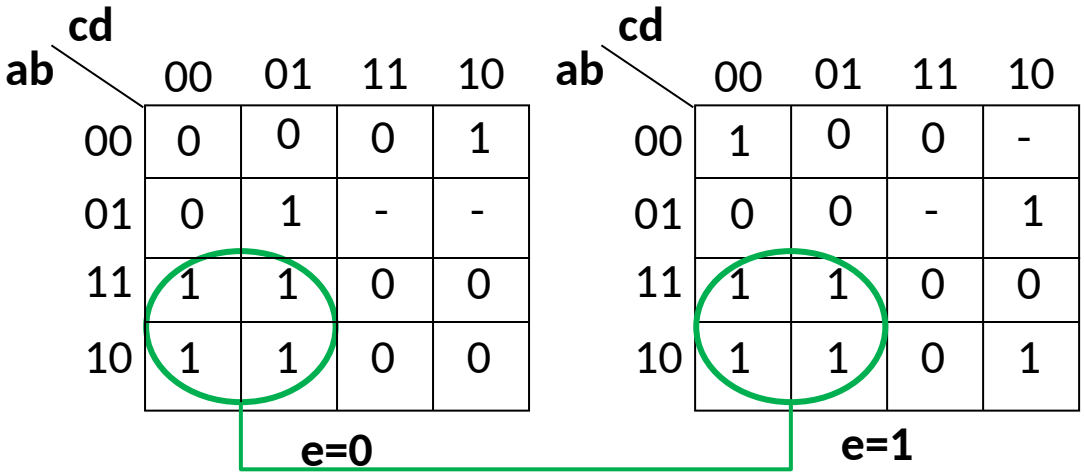
ab \ cd	00	01	11	10
00	1	0	0	-
01	0	0	-	1
11	1	1	0	0
10	1	1	0	1

e=1

1) Scegliamo (o ci viene assegnato) SP

Individuazione grafica dell'espressione minima (2)

2) Si cerca di individuare tra le celle da coprire una cella che possa essere racchiusa in un solo RR e lo si traccia di dimensione massima, annotando il termine corrispondente. Se la funzione è incompleta il RR può contenere anche condizioni di indifferenza.



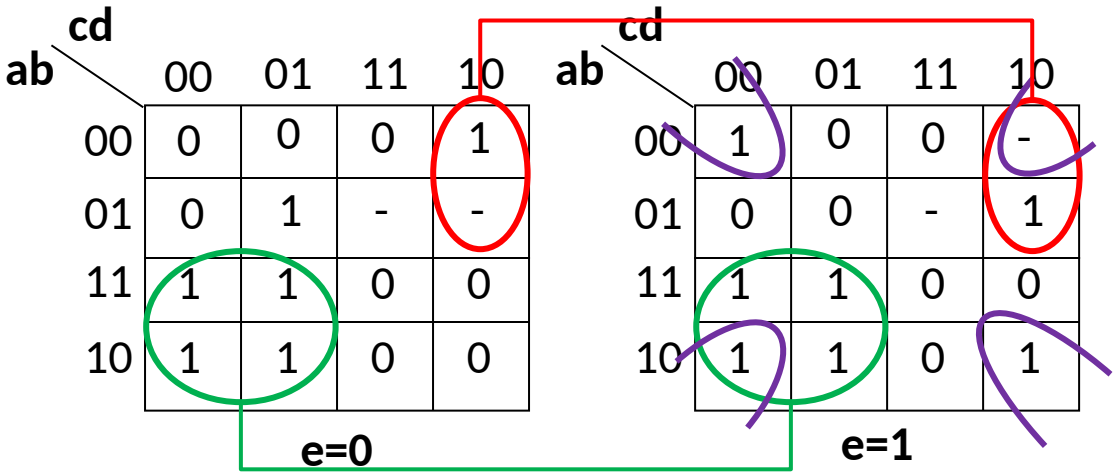
1) scegliamo SP

2) ac'

N.B.: è un unico RR di ordine 3 (8 celle)

Individuazione grafica dell'espressione minima (3)

Si ripete fino a quando è possibile il passo 2, tenendo conto della possibilità di coprire anche celle incluse in RR già tracciati.



1) scegliamo SP

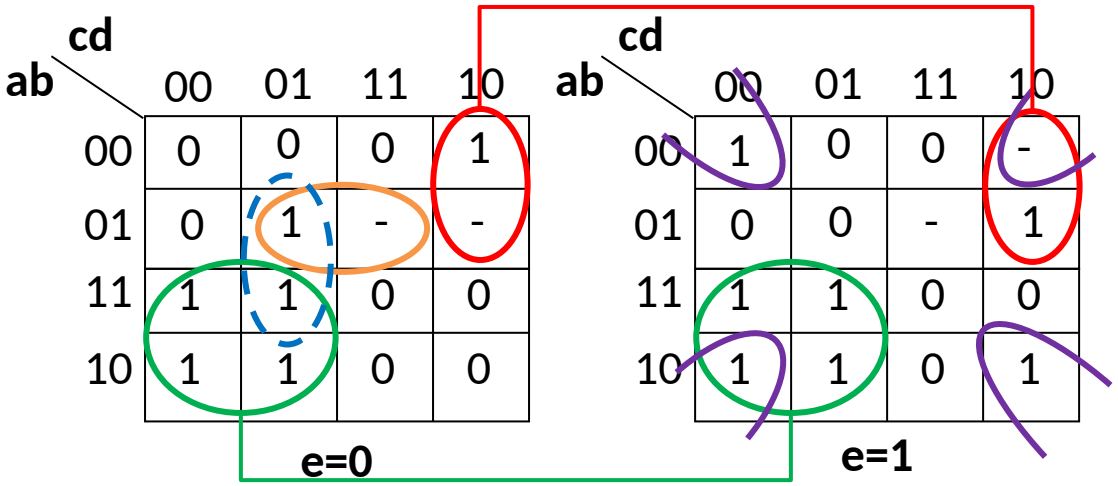
2) ac'

2) a'cd'

2) b'd'e

Individuazione grafica dell'espressione minima (4)

3) Si prendono in considerazione le celle ancora da coprire e se ne sceglie la copertura migliore, tenendo conto come al solito della possibilità di coprire celle già coperte e condizioni di indifferenza.



1) scegliamo SP

2) ac'

2) a'cd'

2) b'd'e

3) a'bde' *oppure* **bc'de'**

4) Si ripete il passo 3 fino a soddisfare la condizione di copertura. Si scrive infine l'espressione minima.

4) ac' + a'cd' + b'd'e + { **a'bde'** / **bc'de'**

Sintesi minima di un encoder

- L'encoder converte il codice «1 su N» in rappresentazione binaria
- Esempio: da codice «1 su 3» a numero binario a due cifre

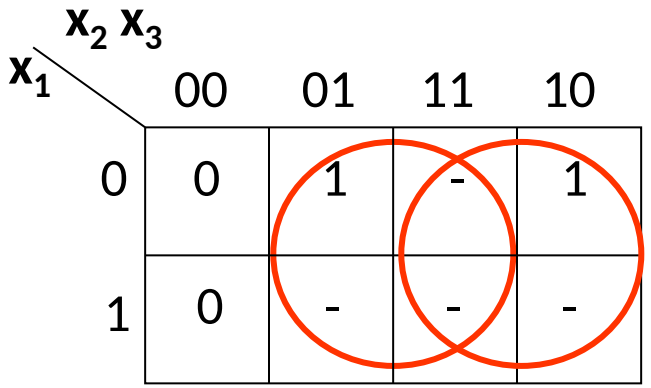
x_3	x_2	x_1	z_1	z_0
0	0	0	0	0
1	0	0	1	1
0	1	0	1	0
0	0	1	0	1
1	1	0	-	-
1	0	1	-	-
0	1	1	-	-
1	1	1	-	-

Espressioni canoniche SP:

$z_0 = x_1' x_2' x_3 + x_1 x_2' x_3'$

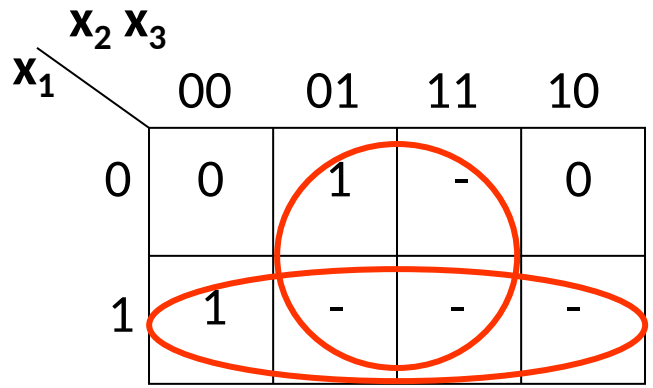
$z_1 = x_1' x_2' x_3 + x_1' x_2 x_3'$

z_1



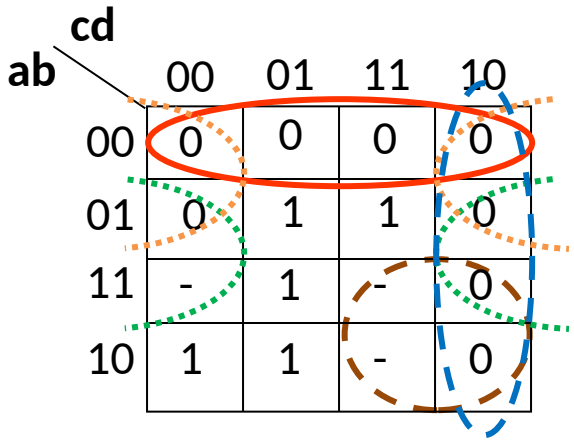
$z_1 = x_2 + x_3$

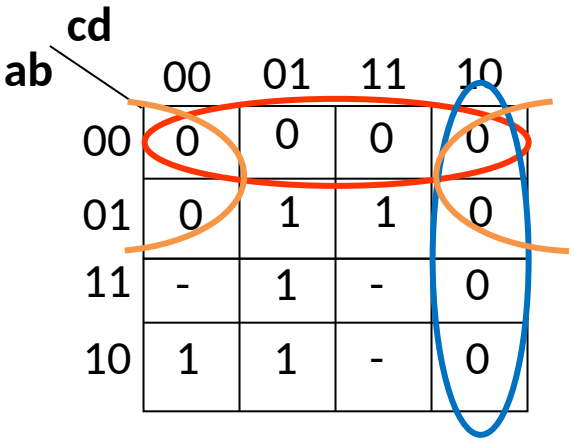
z_0



$z_0 = x_1 + x_3$

Esercizio





1) scegliamo PS

2) **a + b**

Non ci sono altri «0» che posso coprire con un solo implicato primo

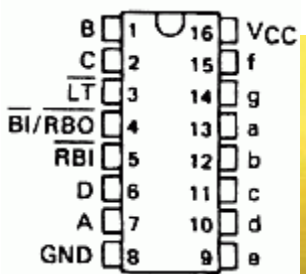
3) **b' + d** *oppure* **a + d**

a' + c' *oppure* **c' + d**

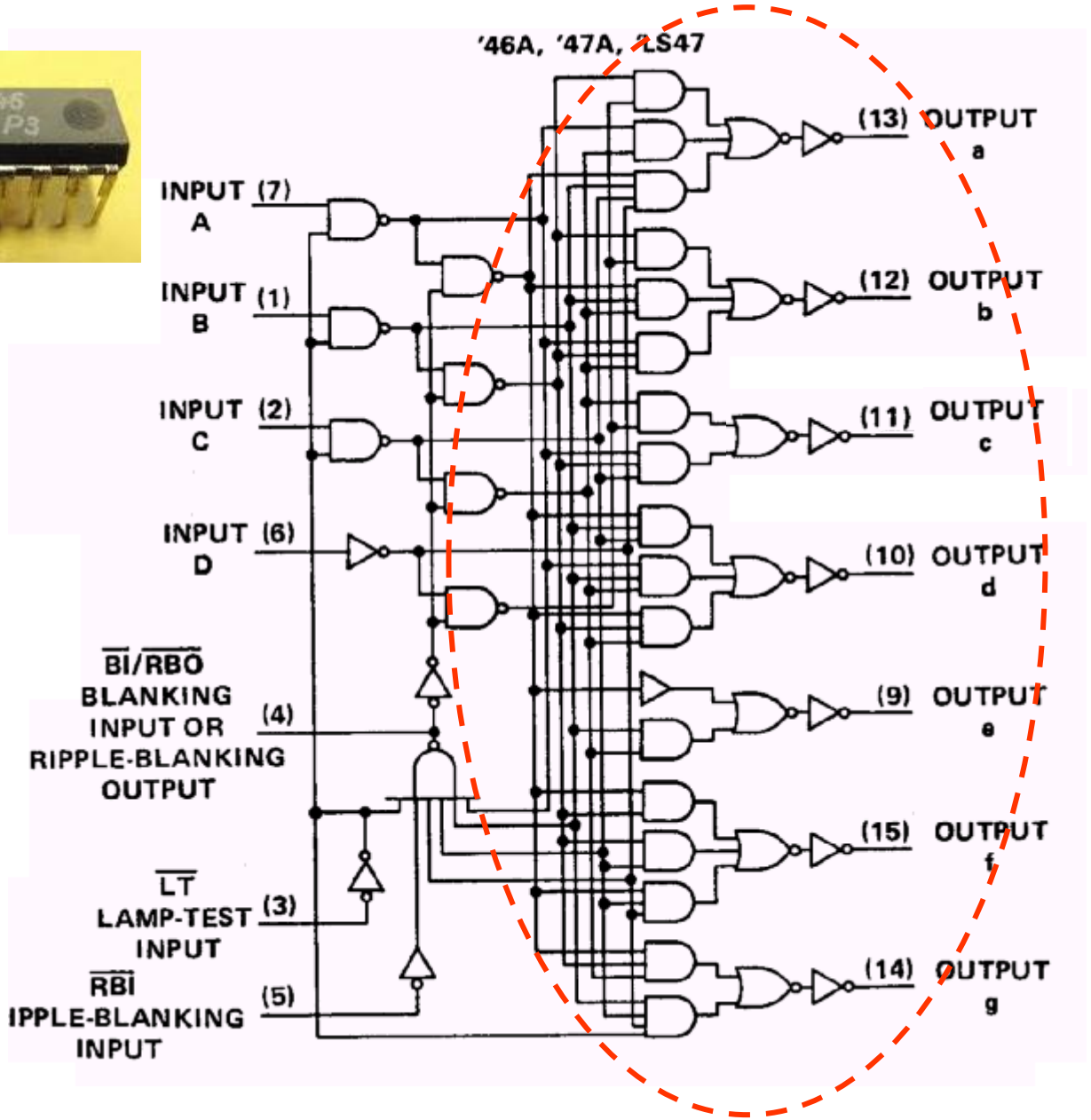
4) $(a+b) \cdot \left\{ \begin{matrix} (b' + d) \\ (a + d) \end{matrix} \right\} \cdot \left\{ \begin{matrix} (a' + c') \\ (c' + d) \end{matrix} \right\}$

$(a + b) \cdot (a + d) \cdot (c' + d)$

Esercizio: il circuito di trascodifica BCD-7 segmenti



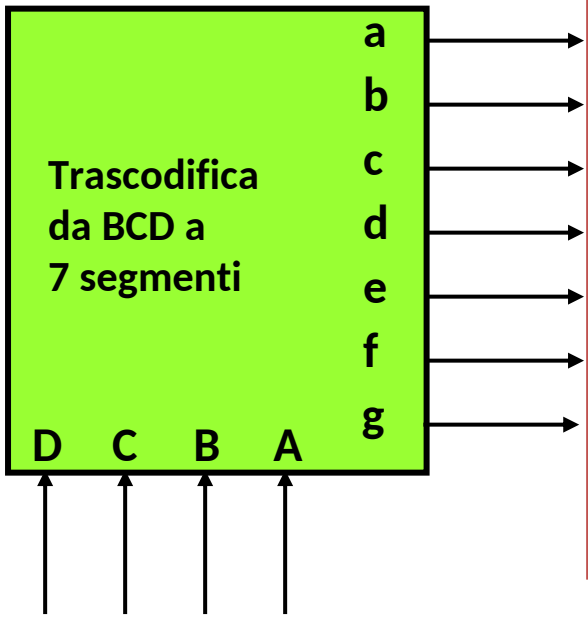
SN 7446

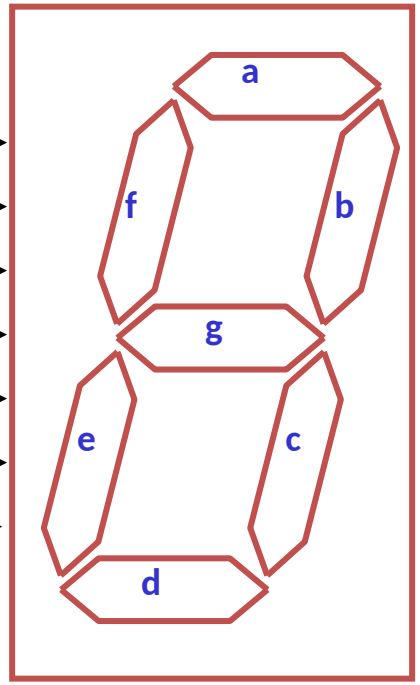


Requisito aggiuntivo, oltre la trascodifica: **la configurazione d'ingresso 1111 deve spegnere tutti i segmenti**.

Effettuare la sintesi minima SP e simularla con Digital

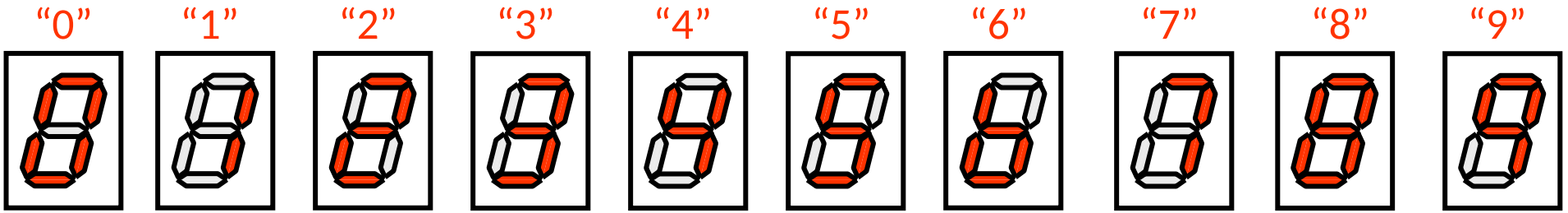
Sintesi





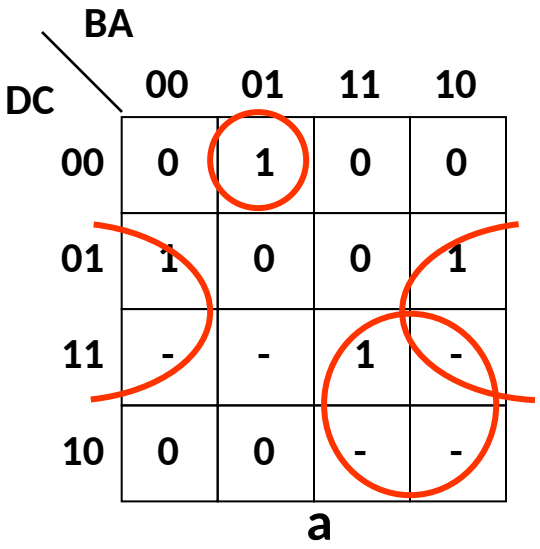
D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	1	1	1

a	b	c	d	e	f	g
0	0	0	0	0	0	1
1	0	0	1	1	1	1
0	0	1	0	0	1	0
0	0	0	0	1	1	0
1	0	0	1	1	0	0
0	1	0	0	1	0	0
1	1	0	0	0	0	0
0	0	0	1	1	1	1
0	0	0	0	0	0	0
0	0	0	1	1	0	0
1	1	1	1	1	1	1

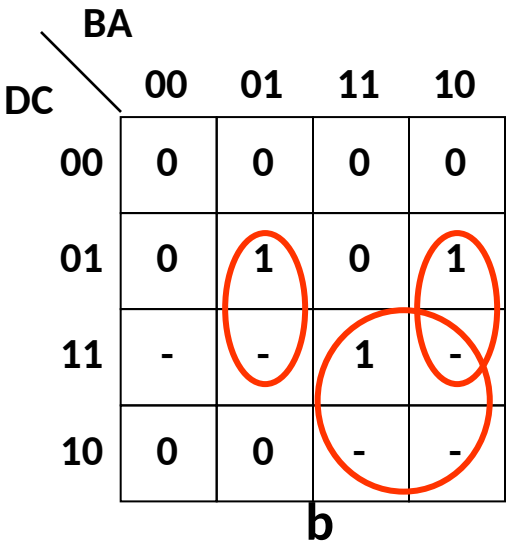


N.B.: 0 visibile, 1 non visibile

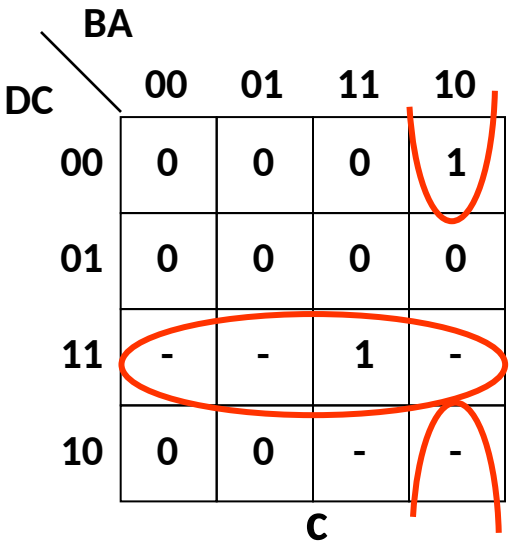
Progetto della rete di costo minimo (1)



a



b



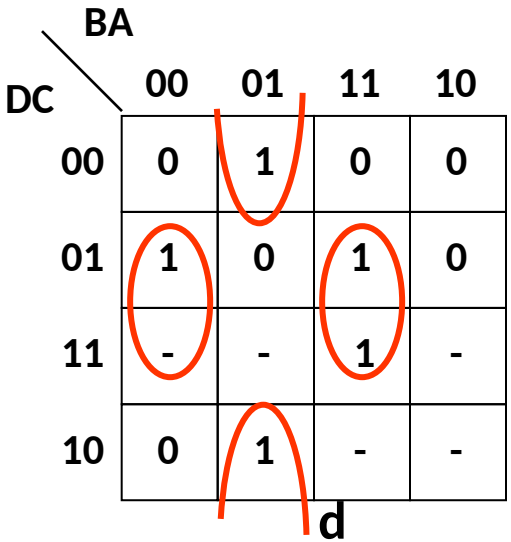
c

a = D'C'B'A + CA'+DB

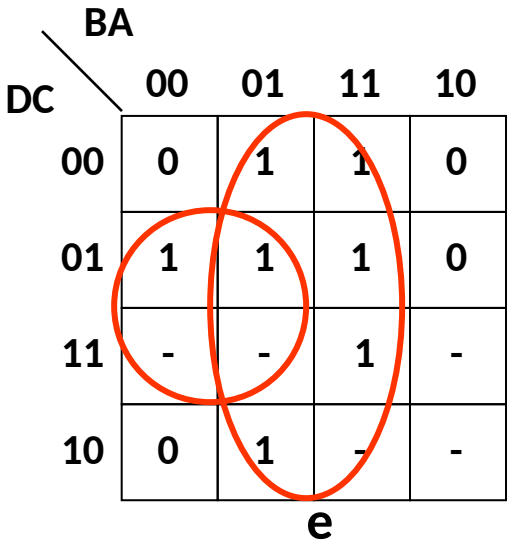
b = CB'A + CBA' +DB

c = C'BA'+DC

Progetto della rete di costo minimo (2)



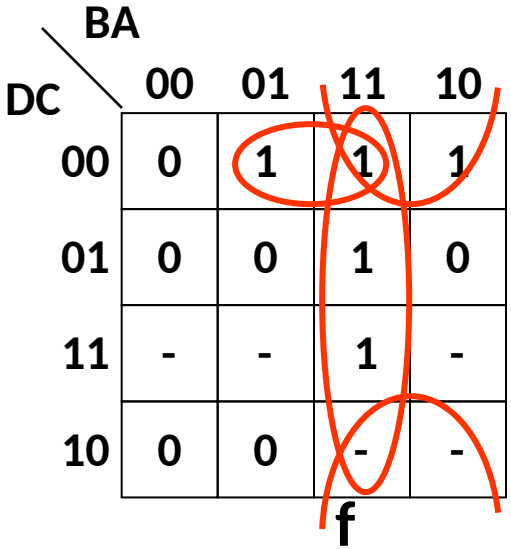
d



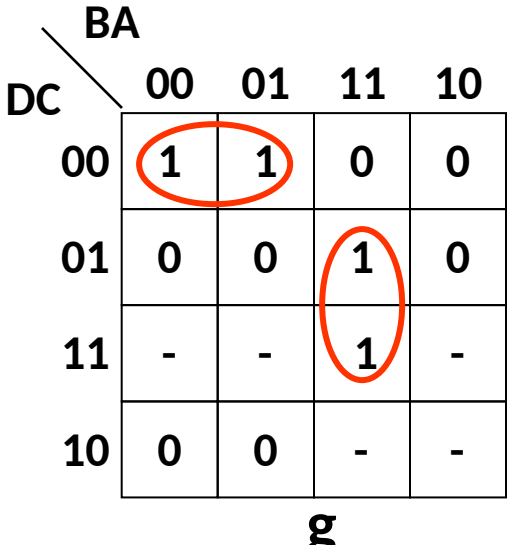
e

d = CB'A' + C'B'A + CBA

e = A + CB'



f



g

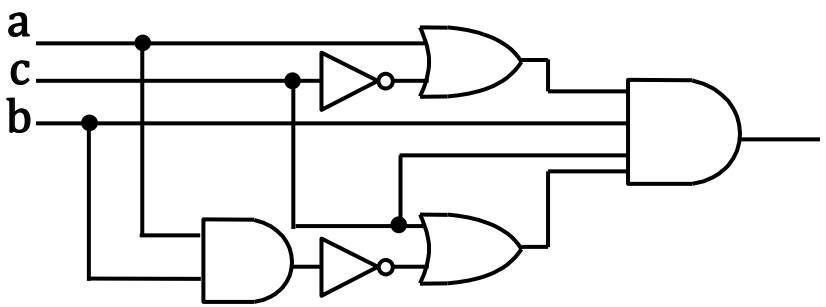
f = D'C'A + BA + C'B

g = D'C'B' + CBA

Analisi con mappe

Uso delle mappe in fase di analisi (1)

1) Si scrive l'espressione associata allo schema e, se necessario, la si manipola fino ad ottenere una espressione SP o PS:



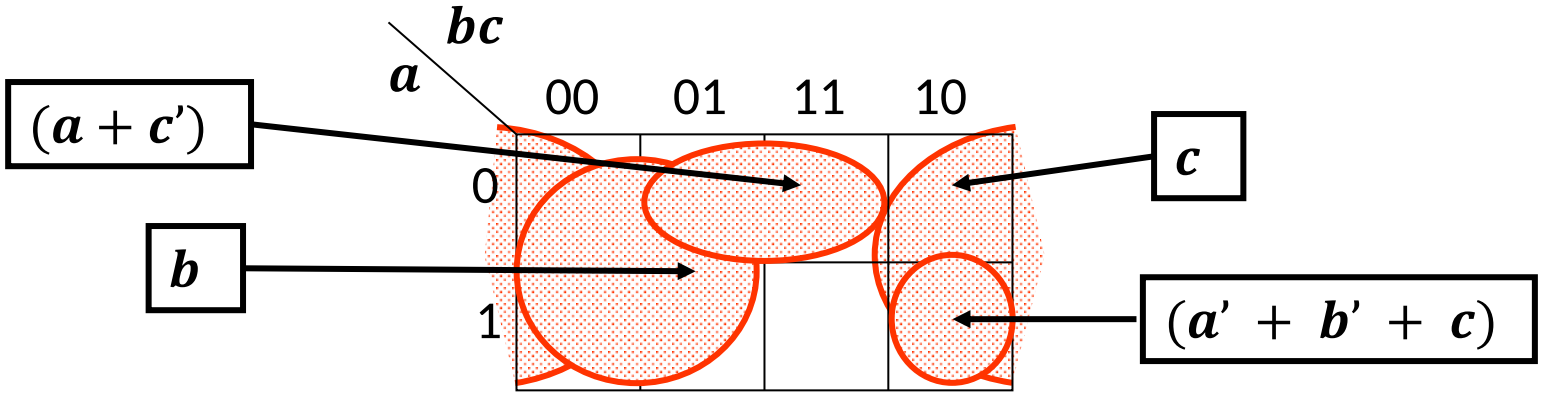
$$F = (a + c') \cdot b \cdot c \cdot ((a \cdot b)' + c)$$

De Morgan (E10)

$$F = (a + c') \cdot b \cdot c \cdot (a' + b' + c)$$

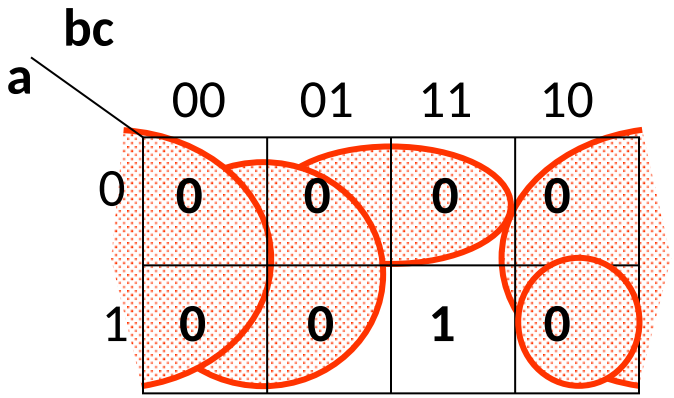
espressione PS

2) Si predispone una mappa di dimensioni adeguate e si tracciano sulla mappa i RR che corrispondono ai termini dell'espressione:



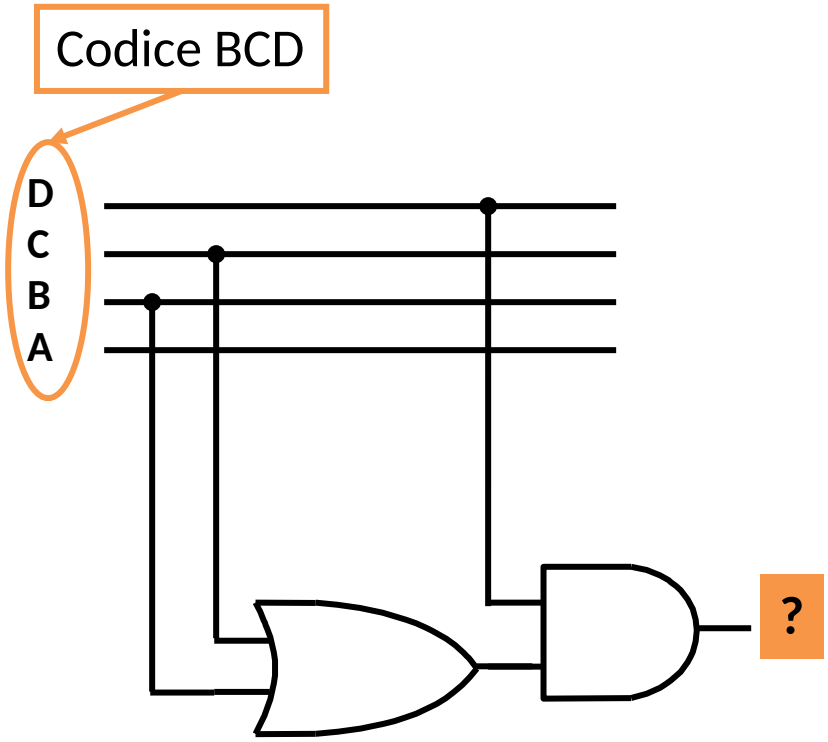
Uso delle mappe in sede di analisi (2)

3) Nelle celle coperte da un RR si indica il valore «1» se l'espressione è SP, «0» se è PS; nelle celle non coperte da RR si inserisce «0» nel caso SP, «1» nel caso PS:



N.B. - La valutazione di una espressione individua sempre una funzione completa !

ESERCIZIO



DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$(C + B) \cdot D$

Uscita «1» se configurazioni «illecite» per il codice BCD

NAND & NOR

Algebre binarie

Algebra binaria: Sistema matematico formato da un insieme di operatori definiti assiomaticamente ed atti a descrivere con una espressione ogni possibile funzione di variabili binarie

Calcolo delle proposizioni
{*vero*, *falso*} {*e*, *o*, *non*}
tre operatori

Crisippo (250 a.c.)
G. Boole (1854)

AND, OR e NOT sono sufficienti per esprimere ogni possibile funzione binaria

Algebra di commutazione
{0, 1} {+, . , ’}
tre operatori

C. Shannon (1938)

Algebra del nand
{0, 1} {↑}
un operatore

Algebra del nor
{0, 1} {↓}
un operatore

E’ possibile esprimere ogni funzione binaria con un solo operatore!

NOT, AND, OR da NAND

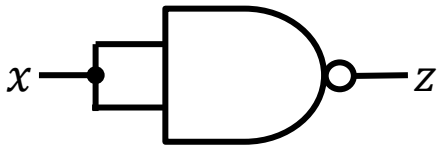
Tabella della Verità

NAND

x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

NOT

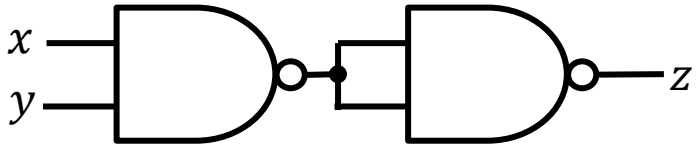
$$x \uparrow x = x'$$



AND

$$(x \uparrow y)' = xy$$

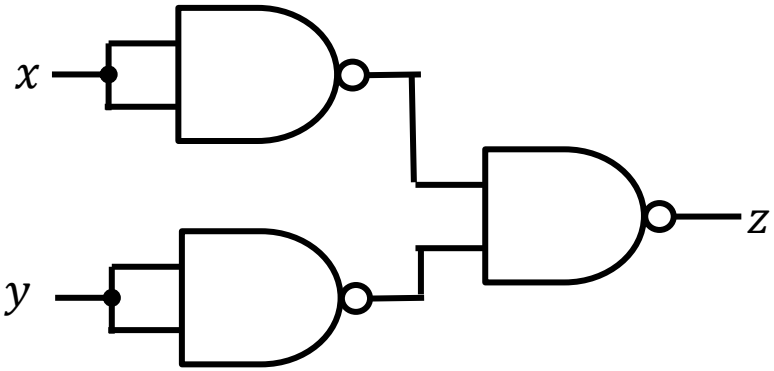
Definizione NAND



OR

$$x + y = (x'y')' = x' \uparrow y'$$

De Morgan



Sintesi con NAND

La sintesi "a NAND" può essere effettuata trasformando **un'espressione SP** (o SPS, SPSP, …) che descrive la funzione assegnata in una nuova espressione contenente esclusivamente operatori "↑":

$$F = a \cdot b + \overline{c} \cdot d + e \cdot \overline{f} + g$$

definizione dell'operatore ↑ ($\overline{a \uparrow b} = ab$)

$$F = \overline{(a \uparrow b)} + \overline{(\overline{c} \uparrow d)} + \overline{(e \uparrow \overline{f})} + g$$

E10 (IIª legge di De Morgan: $\overline{a} + \overline{b} = \overline{ab}$)

$$F = \overline{\left((a \uparrow b) \cdot (\overline{c} \uparrow d) \cdot (e \uparrow \overline{f}) \cdot \overline{g}\right)}$$

definizione dell'operatore ↑

$$F = (a \uparrow b) \uparrow (\overline{c} \uparrow d) \uparrow \left(e \uparrow \overline{f}\right) \uparrow \overline{g}$$

N.B. : stesso numero di operatori, utile per verificare la correttezza della trasformazione

File Digital: 3_NAND_associativity.dlg

N.B. 2: Il NAND **non è associativo:** le parentesi sono fondamentali

Algoritmo per la sintesi a NAND

1) Si parte da un'espressione SP, SPS, SPSP… e si introducono gli operatori "•" e le parentesi non indicati esplicitamente.

2) Si sostituisce il simbolo "↑" ad ogni simbolo "•"

3) Si sostituisce il simbolo "↑" ad ogni simbolo "+" e si **complementano** le variabili **singole** (non i prodotti) affiancate a tale simbolo senza aggiungere o togliere parentesi.

4) (Opzionale) Se compaiono segnali di ingresso in forma negata e non sono disponibili, si sostituiscono con il NAND del segnale in forma vera.

N.B. - La trasformazione dell'espressione minima SP individua l'espressione minima a NAND, se posso assumere di avere i segnali in forma vera e negata.

Esempio: sintesi a NAND di un EX-OR

$U = a\ b' + a'b$

Espressione canonica SP dell' EX-OR

passo 1:

$U = (a \cdot b') + (a' \cdot b)$

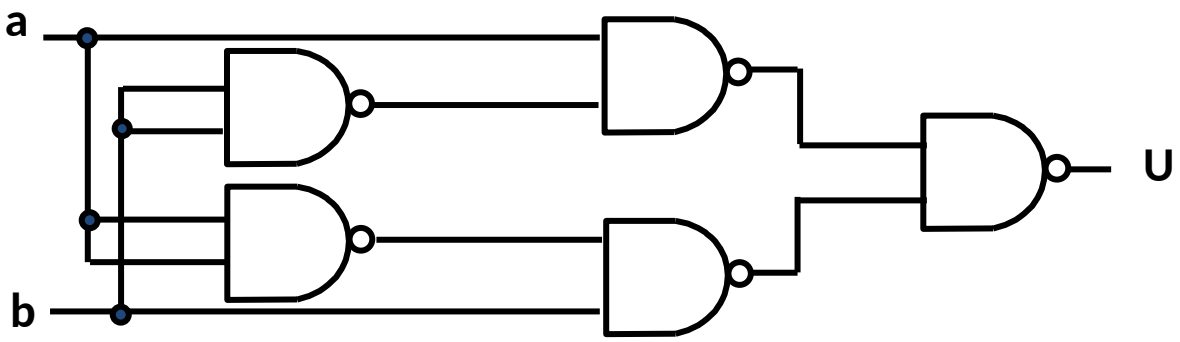
passi 2 e 3

$U = (a \uparrow b') \uparrow (a' \uparrow b)$

Compaiono delle variabili d'ingresso in forma negata. Se non ne dispongo, le sostituisco aggiungendo ulteriori gate, sfruttando le proprietà del NAND

passo 4

$U = (a \uparrow (b \uparrow b)) \uparrow ((a \uparrow a) \uparrow b)$



5 NAND

Esempio: sintesi a NAND di un EX-OR

$U = a\ b' + a'b$

Manipolo algebricamente l'espressione di partenza per avere variabili in forma negata *elaborate* direttamente dagli OR

$U = a\ b' + a'b + a'a + b'b$

Limitazione e identità

$U = a\,(a' + b') + b\,(a' + b')$

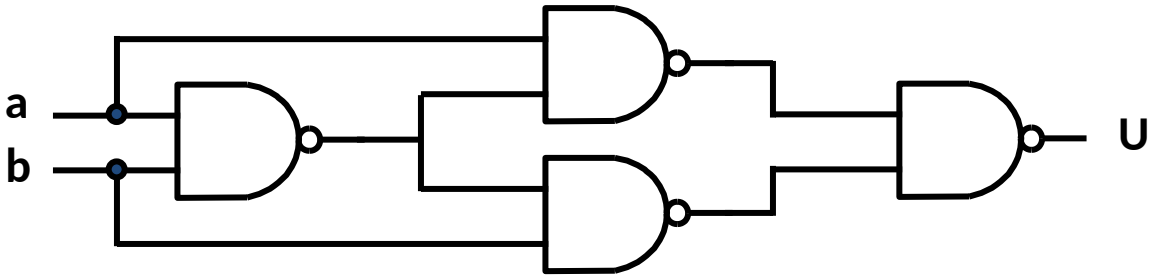
SPS

passo 1:

$U = (\,a \cdot (a' + b')\,) + (\,b \cdot (a' + b')\,)$

passi 2 e 3

$U = (\,a \uparrow (a \uparrow b)\,) \uparrow (\,b \uparrow (a \uparrow b)\,)$



4 NAND

Esempio: sintesi a NAND di un Selettore a 2 vie

$U = A' I_0 + A I_1$

passo 1:

$U = (A' \cdot I_0) + (A \cdot I_1)$

passi 2 e 3

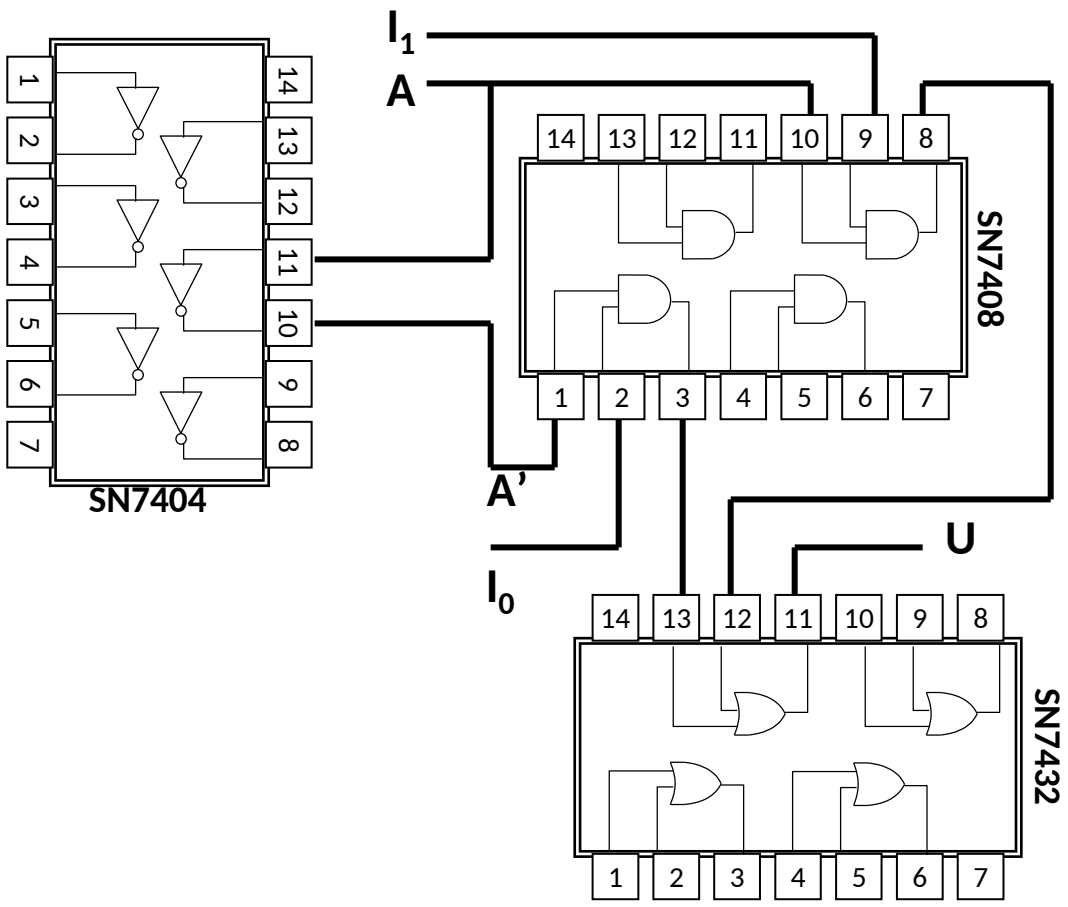
$U = (A' \uparrow I_0) \uparrow (A \uparrow I_1)$

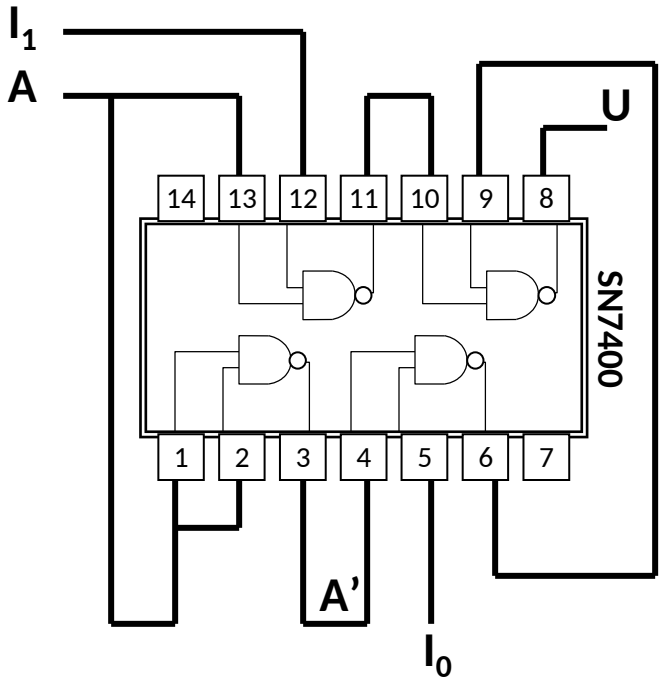
passo 4

$U = ((A \uparrow A) \uparrow I_0) \uparrow (A \uparrow I_1)$

4 NAND

Realizzazione con NAND SSI di un selettore a due vie





$U = A' \cdot I_0 + A \cdot I_1$

$U = ((A \uparrow A) \uparrow I_0) \uparrow (A \uparrow I_1)$

Reti a NAND/NOR, utilizzando un unico tipo di gate, riducono spesso il numero di circuiti SSI necessari

Sintesi con NOR

La sintesi "a NOR" può essere effettuata trasformando **un'espressione PS** (PSP, PSPS, …) che descrive la funzione assegnata in una nuova espressione contenente esclusivamente operatori "↓":

$$F = (\overline{a} + \overline{b} + c) \cdot (\overline{d} + e) \cdot \overline{f} \cdot g$$

⬇ definizione dell'operatore ↓ $(\overline{a \downarrow b} = a + b)$

$$F = \overline{(\overline{a} \downarrow \overline{b} \downarrow c)} \cdot \overline{(\overline{d} \downarrow e)} \cdot \overline{f} \cdot g$$

⬇ E10 (Iª legge di De Morgan: $\overline{a} \cdot \overline{b} = \overline{a + b}$)

$$F = \overline{(\overline{a} \downarrow \overline{b} \downarrow c) + (\overline{d} \downarrow e) + f + \overline{g}}$$

⬇ definizione dell'operatore ↓

$$F = (\overline{a} \downarrow \overline{b} \downarrow c) \downarrow (\overline{d} \downarrow e) \downarrow f \downarrow \overline{g}$$

N.B. : stesso numero di operatori, utile per verificare la correttezza della trasformazione

N.B. 2: Il NOR **non è associativo:** le parentesi sono fondamentali

Algoritmo per la sintesi a NOR

1) Si parte da un'espressione PS, PSP, PSPS… e si introducono gli operatori "•" e le parentesi non indicati esplicitamente.

2) Si sostituisce il simbolo "↓" ad ogni simbolo "**+**"

3) Si sostituisce il simbolo "↓" ad ogni simbolo "·" e si **complementano** le variabili **singole** (non le somme) affiancate a tale simbolo senza aggiungere o togliere parentesi.

4) (Opzionale) Se compaiono segnali di ingresso in forma negata e non sono disponibili, li sostituisco con il NOR del segnale in forma vera.

N.B. - La trasformazione dell'espressione minima PS individua l'espressione minima a NOR , se posso assumere di avere i segnali in forma vera e negata..

Esempio: sintesi a NOR di un "equivalence"

$U = (a + b') \cdot (a' + b)$

Espressione canonica PS dell'equivalence

passo 1,2,3

$U = (a \downarrow b') \downarrow (a' \downarrow b)$

Manipolo algebricamente l'espressione di partenza per avere sugli AND variabili in forma negata

Limitazione e identità

$U = (a + b') \cdot (a' + b) \cdot (a' + a) \cdot (b' + b)$

$U = (a + a'b') \cdot (b + a'b')$ ***PSP***

passo 1

$U = (a + (a' \cdot b')) \cdot (b + (a' \cdot b'))$

passi 2 e 3

$U = (a \downarrow (a \downarrow b)) \downarrow (b \downarrow (a \downarrow b))$

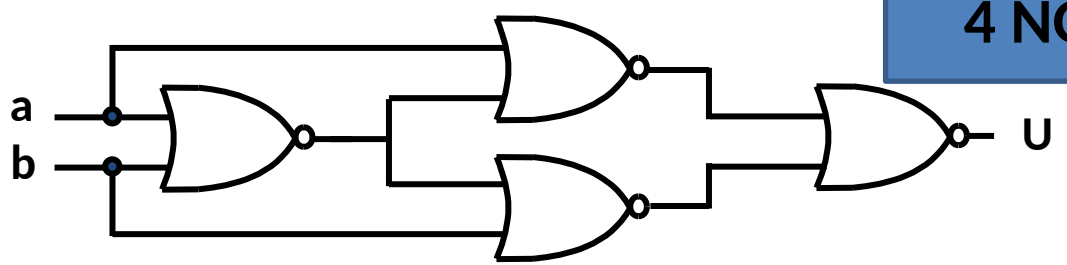
$A \downarrow A = A'$

A	A↓A
0	1
1	0

Sfrutto tale proprietà per ottenere un'espressione in cui le variabili compaiono solo in forma vera

$U = (a \downarrow (b \downarrow b)) \downarrow ((a \downarrow a) \downarrow b)$

5 NOR



4 NOR

Perché NAND/NOR?

https://www.nand2tetris.org/