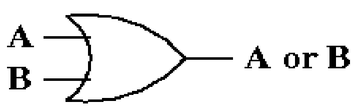
Esercitazioni su circuiti combinatori

Salvatore Orlando
&
Marta Simeoni

Algebra Booleana: funzioni logiche di base

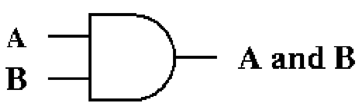
OR (somma): l'uscita è 1 se almeno uno degli ingressi è 1

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



AND (prodotto): l'uscita è 1 se tutti gli ingressi sono 1

A	B	(A · B)
0	0	0
0	1	0
1	0	0
1	1	1



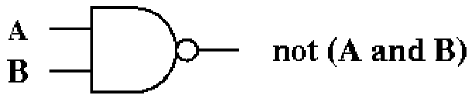
Algebra Booleana: funzioni logiche di base

NOT (complemento): l’uscita è il complemento dell’ingresso

A	~A
0	1
1	0

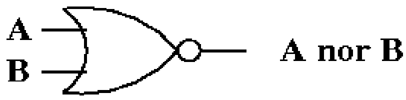
NAND

A	B	~(A · B)
0	0	1
0	1	1
1	0	1
1	1	0



NOR

A	B	~(A + B)
0	0	1
0	1	0
1	0	0
1	1	0



Algebra booleana: equazioni

Come si dimostra che due funzioni logiche sono uguali?

Ci sono due metodi:

- Costruire la tabella di verità delle due funzioni e verificare che, per gli stessi valori dei segnali di ingresso, siano prodotti gli stessi valori dei segnali di uscita

- Sfruttare le proprietà dell’algebra booleana per ricavare una funzione dall’altra (tramite sequenze di equazioni)

Algebra booleana: equazioni

Come si dimostra che due funzioni logiche sono uguali?

Esempio: considerare le leggi di De Morgan

~(A•B) = (~A) + (~B)

A	B	(A·B)	~(A·B)	~A	~B	(~A)+(~B)
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Algebra booleana: equazioni

Come si dimostra che due funzioni logiche sono uguali?

Esempio: considerare le leggi di De Morgan

~(A+B) = (~A) • (~B)

~A~B = ~A~B + 0 = ~A~B + [~(A+B) · (A+B)] =
[~A~B + ~(A+B)] · [~A~B + (A+B)] =
[~A~B + ~(A+B)] · [(~A + A) · (~B + A) + B] =
[~A~B + ~(A+B)] · [~B + A + B] = (~A~B) + ~(A+B)
~A~B = ~A~B · 1 = ~A~B · [~(A+B) + (A+B)] =
(~A~B) · ~(A+B) + (~A~B) · (A+B) =
(~A~B) · ~(A+B) + [~A~B A + ~A~B B] =(~A~B) · ~(A+B)
~A~B = ~A~B + ~(A+B) = ((~A~B)· ~(A+B)) + ~(A+B) =
~(A+B) · [(~A~B) + 1] = ~(A+B)

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

Intuitivamente le equazioni sono:

$$D = A + B + C$$
$$F = ABC$$
$$E = (AB + BC + AC) \cdot \sim(ABC)$$

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Tabella di verità

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Prodotti di somme (PS):

D = A+B+C

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Prodotti di somme (PS):

D = A+B+C

E = (A+B+C) (A+B+~C) (A+~B+C) (~A+B+C) (~A+~B+~C)

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Prodotti di somme (PS):

D = A+B+C

E = (A+B+C) (A+B+~C) (A+~B+C) (~A+B+C) (~A+~B+~C)

F = (A+B+C) (A+B+~C) (A+~B+C) (A+~B+~C)(~A+B+C) (~A+B+~C)(~A+~B+C)

Realizzazione di circuiti combinatori

Esercizio: Dati tre ingressi A, B, C realizzare un circuito che fornisca in uscita tre segnali

D è vera se almeno uno degli ingressi è vero
E è vera se esattamente due input sono veri
F è vera se tutti e tre gli input sono veri

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	0	1

Somme di Prodotti (SP):

D = (~A~BC)+(~AB~C)+(~ABC)+ (A~B~C)+(A~BC)+(AB~C)+(ABC)

E = (~ABC)+(A~BC)+(AB~C)

F = ABC

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione D dell’esercizio precedente

D = (~A~BC)+(~AB~C)+(~ABC)+ (A~B~C)+(A~BC)+(AB~C)+(ABC)

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	1

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione D dell’esercizio precedente

D = (~A~BC)+(~AB~C)+(~ABC)+ (A~B~C)+(A~BC)+(AB~C)+(ABC)

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	1

Si può considerare un rettangolo più grande di quello a sinistra, che include anche quello selezionato

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione D dell'esercizio precedente

**D = (~A~BC)+(~AB~C)+(~ABC)+
(A~B~C)+(A~BC)+(AB~C)+(ABC)**

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	1

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione D dell'esercizio precedente

**D = (~A~BC)+(~AB~C)+(~ABC)+
(A~B~C)+(A~BC)+(AB~C)+(ABC)**

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	1

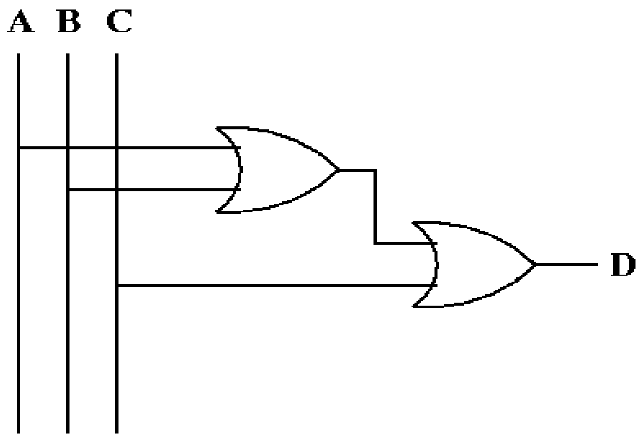
Errore!
si deve raccogliere un p-sottocubo
(rettangolo di celle adiacenti)
di **2^p^** celle

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione D dell'esercizio precedente

D = (~A~BC)+(~AB~C)+(~ABC)+ (A~B~C)+(A~BC)+(AB~C)+(ABC)

A \ BC	00	01	11	10
0		1	1	1
1	1	1	1	1



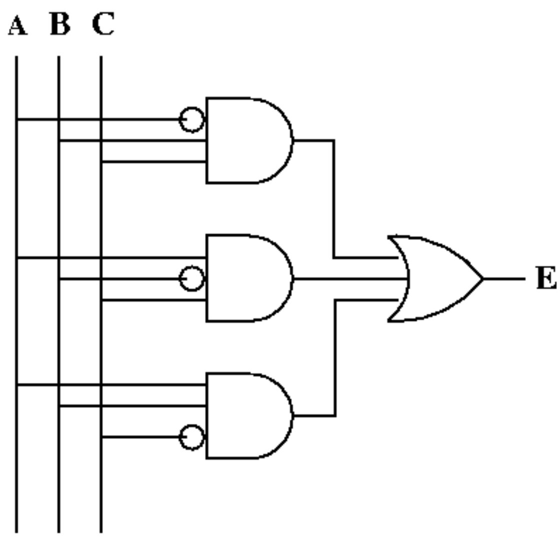
D = A + B + C

Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione E dell'esercizio precedente

E = (~ABC)+(A~BC)+(AB~C)

A \ BC	00	01	11	10
0			1	
1		1		1



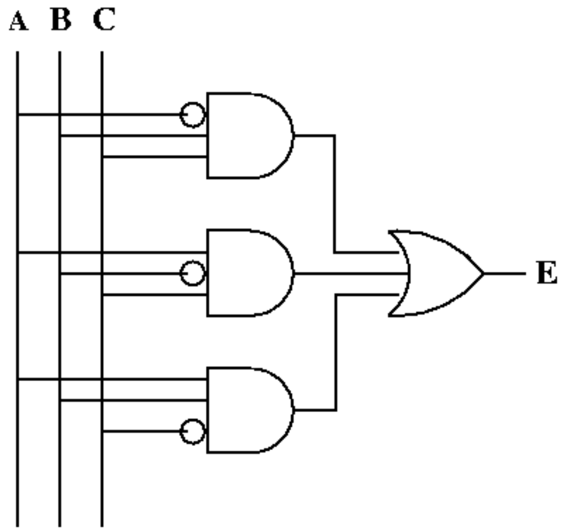
E = (~ABC)+(A~BC)+(AB~C)

Realizzazione di circuiti combinatori

Esercizio: Realizzare il circuito precedente (riportato qui in figura) nei seguenti casi:

1. utilizzando porte AND e OR a due ingressi

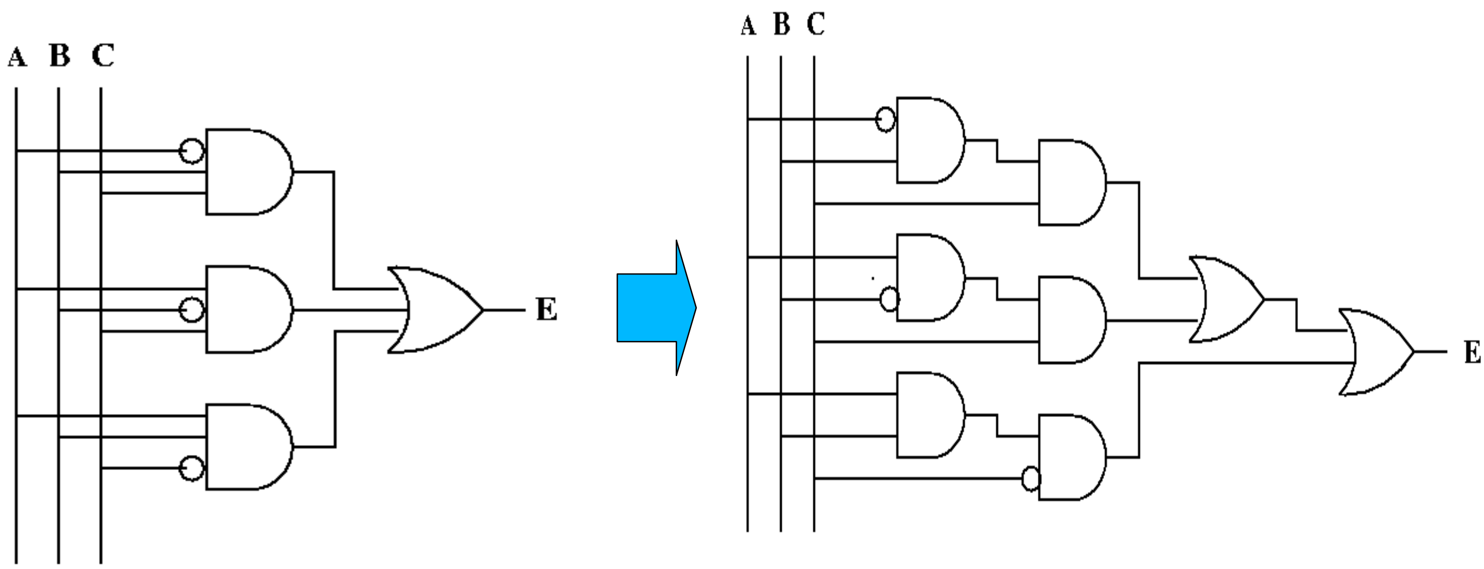
2. utilizzando porte NAND a tre ingressi



Realizzazione di circuiti combinatori

Esercizio: (continua)

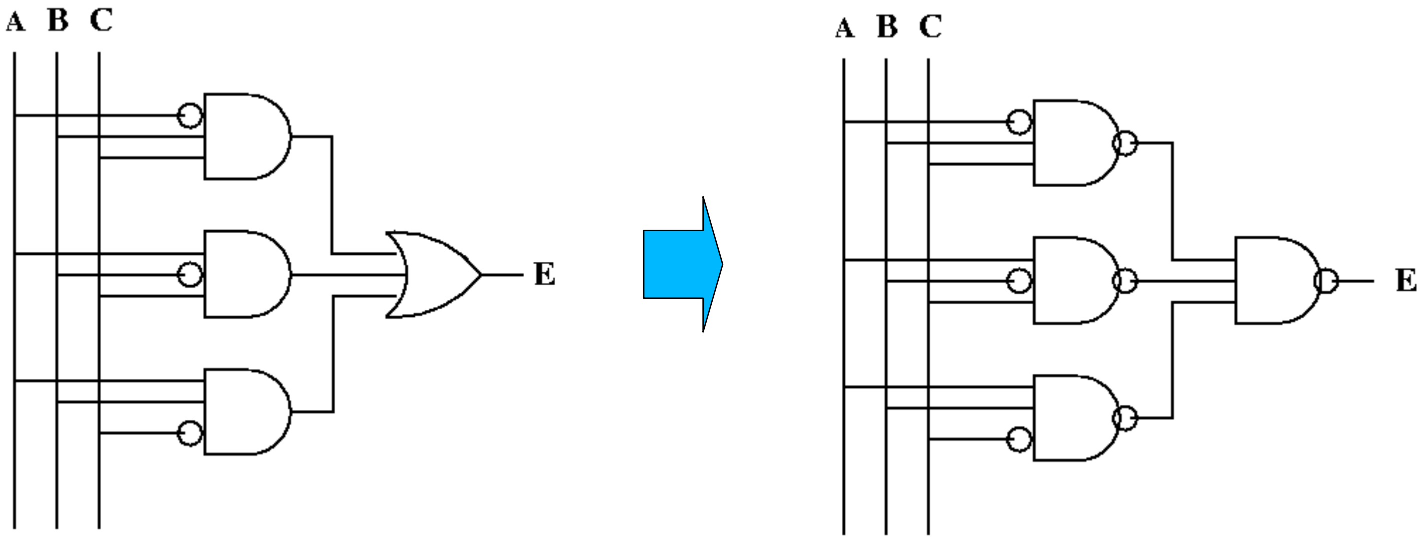
Realizzazione utilizzando porte AND e OR a due ingressi



Realizzazione di circuiti combinatori

Esercizio: (continua)
Realizzazione utilizzando porte NAND a tre ingressi

E = (~ABC)+(A~BC)+(AB~C) = [*applico De Morgan*]
~[~(~ABC) • ~(A~BC) • ~(AB~C)]



Realizzazione di circuiti combinatori

Esercizio: Minimizzare la funzione F dell'esercizio precedente espressa come prodotto di somme (PS)

F = (A+B+C) (A+B+~C) (A+~B+C) (A+~B+~C)(~A+B+C) (~A+B+~C)(~A+~B+C)

A \ BC	00	01	11	10
0	0	0	0	0
1	0	0		0

F = B • A • C

p-sottocubi composti da zeri. Per ottenere le varie somme (PS), in ogni somma devono apparire solo le variabili che rimangono invariate in ogni p-sottocubo. Le variabili sono negate sono quelle valori uguali ad 1.

Realizzazione di circuiti combinatori

Esercizio: Dati quattro ingressi A, B, C, D realizzare un circuito che fornisca in uscita il segnale E definito come segue:

- il valore di E è indifferente se gli ingressi sono tutti 0 o tutti 1
- E è 1 se gli ingressi contengono un numero dispari di 1
- E è 0 se gli ingressi contengono un numero pari di 1

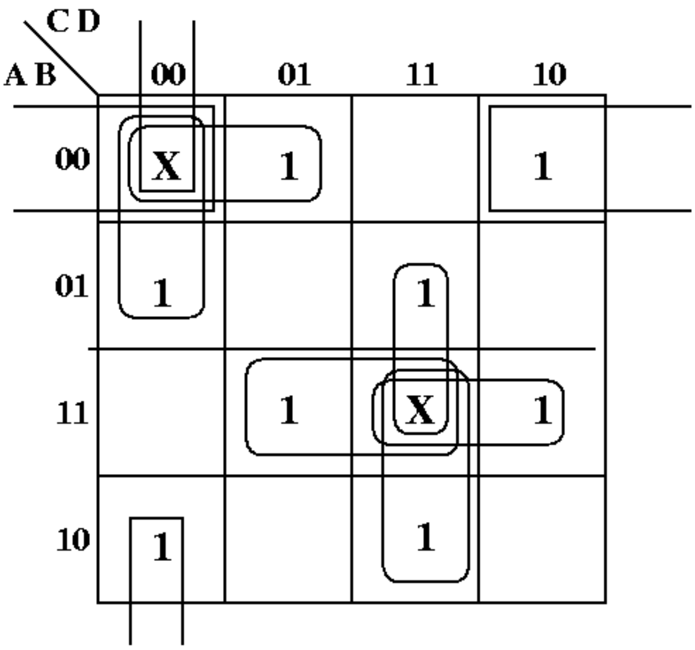
A	B	C	D	E
0	0	0	0	X
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	X

Tabella di verità

Realizzazione di circuiti combinatori

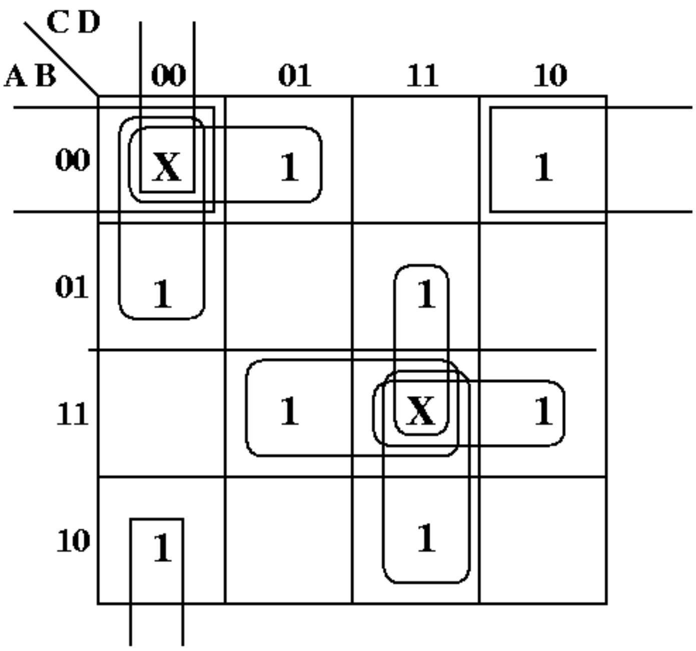
A	B	C	D	E
0	0	0	0	X
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	X

Tabella di verità



Mappa di Karnaugh

Realizzazione di circuiti combinatori



Realizzare il circuito usando porte AND e OR a due soli ingressi

E = ~A~B~C + ~A~C~D + ~B~C~D + ~A~B~D +
BCD + ABC + ABD + BCD

Sintesi di funzioni logiche

<u>Algoritmo di Quine McCluskey</u>

- Le mappe di Karnaugh servono per la minimizzazione "a mano" delle funzioni (fini a 5 variabili)
- L'algoritmo di Quine McCluskey serve per sintetizzare funzioni logiche *minime* in maniera "automatica"

Sintesi di funzioni logiche: Algoritmo di Quine McCluskey

Considerare la funzione logica rappresentata dalla tabella di verità seguente:

A	B	C	D	E	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	--> 5
0	1	1	0	1	--> 6
0	1	1	1	1	--> 7
1	0	0	0	1	--> 8
1	0	0	1	1	--> 9
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	1	--> 12
1	1	0	1	1	--> 13
1	1	1	0	1	--> 14
1	1	1	1	0	

Sintesi di funzioni logiche: Algoritmo di Quine McCluskey

Prima fase: riportare le combinazioni che danno uscita "1" in tabella, suddividendole rispetto al PESO, cioè al numero di "1" presenti in ciascuna combinazione.

```
   A B C D
8  1 0 0 0
-----------
5  0 1 0 1
6  0 1 1 0
9  1 0 0 1
12 1 1 0 0
-----------
7  0 1 1 1
13 1 1 0 1
14 1 1 1 0
```

Sintesi di funzioni logiche: Algoritmo di Quine McCluskey

Prima fase: Confrontare poi le configurazioni di una sezione con tutte le combinazioni della sezione successiva.
Individuiamo così eventuali coppie con distanza di Hamming uguale a 1
Nella nuova tabella, i bit differenti tra ogni coppia diventano DON’T CARE

	A	B	C	D	
8	1	0	0	0	√

5	0	1	0	1	√
6	0	1	1	0	√
9	1	0	0	1	√
12	1	1	0	0	√

7	0	1	1	1	√
13	1	1	0	1	√
14	1	1	1	0	√

	A	B	C	D
8/9	1	0	0	_
8/12	1	_	0	0

5/7	0	1	_	1
5/13	_	1	0	1
6/7	0	1	1	_
6/14	_	1	1	0
9/13	1	_	0	1
12/13	1	1	0	_
12/14	1	1	_	0

Sintesi di funzioni logiche: Algoritmo di Quine McCluskey

Prima fase: **Iteriamo** il procedimento sulle nuove tabelle, fino a quando non è più possibile individuare coppie di righe con distanza di Hamming uguale ad 1.

	A	B	C	D	
8	1	0	0	0	√

5	0	1	0	1	√
6	0	1	1	0	√
9	1	0	0	1	√
12	1	1	0	0	√

7	0	1	1	1	√
13	1	1	0	1	√
14	1	1	1	0	√

	A	B	C	D	
8/9	1	0	0	_	√
8/12	1	_	0	0	√

5/7	0	1	_	1	
5/13	_	1	0	1	
6/7	0	1	1	_	
6/14	_	1	1	0	
9/13	1	_	0	1	√
12/13	1	1	0	_	√
12/14	1	1	_	0	

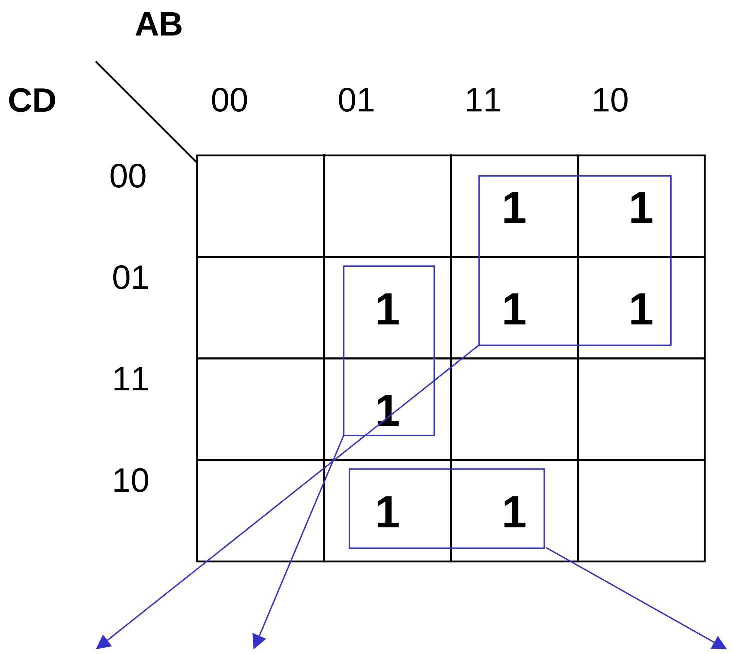
	A	B	C	D
8/9/12/13	1	_	0	_

Solo le righe non flagged concorrono a determinare l’equazione

E = A~C + ~ABD + B~CD + ~ABC + BC~D + AB~D

Sintesi di funzioni logiche: Algoritmo di Quine McCluskey

Applichiamo le mappe di Karnaugh.



E = A~C + ~ABD + B~CD + ~ABC + BC~D + AB~D

Nell'equazione di sopra abbiamo quindi alcuni p-sottocubi ridondanti !!

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: Costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

Le colonne 8 e 9 si possono "coprire" solo usando A~C, che quindi diventa un termine indispensabile

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

A~C copre le colonne 8 e 9, ma anche le colonne 12 e 13

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

	5	6	7	14
~ABD	X		X	
B~CD	X			
~ABC		X	X	
BC~D		X		X
AB~D				X

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	14
~ABD	X		X	
B~CD	X			
~ABC		X	X	
BC~D		X		X
AB~D				X

le righe relative a B~CD e AB~D sono dominate dalle righe relative a ~ABD e BC~D, rispettivamente

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

	5	6	7	14
~ABD	X		X	
B~CD	X			
~ABC		X	X	
BC~D		X		X
AB~D				X

	5	6	7	14
~ABD	X		X	
~ABC		X	X	
BC~D		X		X

Le colonne 5 e 14 si possono rispettivamente "coprire" solo usando ~ABD e BC~D
Quindi entrambi diventano termini indispensabili

Sintesi di funzioni logiche: Algoritmo di Quine-McCluskey

Seconda fase: costruzione della tabella di copertura

	5	6	7	8	9	12	13	14
A~C				X	X	X	X	
~ABD	X		X					
B~CD	X						X	
~ABC		X	X					
BC~D		X						X
AB~D						X		X

	5	6	7	14
~ABD	X		X	
B~CD	X			
~ABC		X	X	
BC~D		X		X
AB~D				X

	5	6	7	14
~ABD	X		X	
~ABC		X	X	
BC~D		X		X

E = A~C + ~ABD + BC~D