Progettazione Digitale

Ottimizzazione delle reti combinatorie

A cura di:

Ph.D., Ing. **Alessandra De Benedictis**, alessandra.debenedictis@unina.it

Testi di riferimento

Franco Fummi, Mariagiovanna Sami, Cristina Silvano - Progettazione digitale - McGraw-Hill

Bolchini, Brandolese, Salice, Sciuto – Reti Logiche – Apogeo

CDL Magistrale Ing. Informatica - Prof. Antonino Mazzeo

Architettura dei Sistemi di Elaborazione

Indice Reti Combinatorie

- Ciclo di progettazione di una rete
- Formalizzazione della specifica
  - Funzioni non completamente specificate
  - On-set, DC-set e OFF-set
- Sintesi
  - Prima forma canonica
  - Seconda forma canonica
- Minimizzazione esatta
  - Metodo delle mappe di karnaugh
  - Metodo di Quine-McCluskey
  - Metodi di supporto alla copertura
- Minimizzazione euristica di reti a due livelli
  - Approccio iterativo
  - Descrizione del problema e soluzione iniziale
  - Trasformazioni
- Minimizzazione euristica su più livelli
  - Modello di riferimento
  - Trasformazioni
  - Applicazione delle trasformazioni

Ciclo di progettazione

- Comprensione e analisi della specifica semi-formale
  - Dalla descrizione verbale del funzionamento della rete si specificano gli elementi di contesto, di codifica, di errore, etc. eliminando ogni ambiguità e pervenendo ad una specifica semi-formale
- Formalizzazione della specifica della rete
  - La formalizzazione produce una tabella di verità che descrive la funzione di commutazione
- Sintesi
  - consiste nel passare dalla tabella di verità ad una espressione algebrica (una delle tante possibili) formale
- Ottimizzazione
  - consiste nel manipolare l'espressione algebrica alla luce di un criterio di qualità al fine di pervenire ad un'espressione equivalente ottima secondo il criterio prescelto

Formalizzazione della specifica

Si realizzi una rete combinatoria dotata di un ingresso a tre bit $X=\{x_0, x_1, x_2\}$ che rappresenta un numero intero nell'intervallo [0;6] in codifica binaria naturale. La rete deve essere in grado di discriminare i numeri pari dai numeri dispari.

- Dopo un'analisi della specifica verbale si ottiene la specifica semi informale seguente:
- si realizzi una rete combinatoria dotata di ingresso a tre bit $X=\{x_0, x_1, x_2\}$ che rappresenta un numero intero nell'intervallo [0;6] in codifica binaria naturale. La rete è dotata di due uscite *z* ed *e*, di un bit ciascuno. L'uscita *z* indica se il valore in ingresso è pari (z=1) o dispari (z=0), mentre l'uscita *e* indica un errore (e=1) oppure il funzionamento corretto (e=0).

Specifica formale con tabelle di verità

x_0	x_1	x_2	z
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

x_0	x_1	x_2	e
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

Funzioni non completamente specificate

In molte situazioni reali accade che una funzione logica non sia definita per tutte le 2^n possibili configurazioni degli ingressi

Quando il valore della funzione non è specificato si dice che la funzione presenta una *condizione di indifferenza* o *don't care* (indicata col simbolo -).

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

Definizione di *on-set* , *don't care set*, *off-set*

- Sia $f(x_0, x_1, \ldots, x_{n-1})=f(X)$ una generica funzione di n variabili. Si definiscono i seguenti insiemi
  - On-set $\Sigma = \{X_i | f(X_i) = 1\}$
  - Don't care-set $\Delta=\{X_i | f(X_i)= - \}$
  - Off-set $\phi=\{X_i | f(X_i) = 0\}$

per cui valgono le relazioni

$\Sigma \cup \Delta \cup \phi = B^n$; $\Sigma \cap \Delta = \emptyset$; $\Sigma \cap \phi = \emptyset$; $\Delta \cap \phi = \emptyset$

- due dei tre insiemi sono sufficienti a definire in modo completo e univoco una generica funzione

Specifica formale con tabelle di verità

Tabella delle verità di una funzione $f(B^4) \longrightarrow B$

x	y	z	v	o
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	1
0	1	1	0	1
0	1	1	1	1

x	y	z	v	o
1	0	0	0	0
1	0	0	1	1
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

- $o=f(x,y,z,v)=\Sigma\{1, 4, 5, 6, 7, 9, 11, 14, 15)$

Rappresentazione in forma canonica

PRIMA FORMA CANONICA (forma normale di tipo P o somma di prodotti)

Una generica funzione a n variabili f(x1, …, xn) può essere espansa nella forma di disgiunzione di tutti i suoi mintermini.

Un **mintermine** è un termine prodotto in cui compaiono tutte le variabili di ingresso corrispondenti alle combinazioni per cui la funzione booleana assume valore 1. In particolare il *mintermine* è composto dalle *variabili che assumono valore 1 prese in forma naturale e da quelle che assumono valore 0 prese in forma complementata*.

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

Forma P f=x’y’z + xy’z’ + xy’z + xyz’ + xyz

Rappresentazione in forma canonica

SECONDA FORMA CANONICA (forma normale di tipo S o prodotto di somme)

Una generica funzione a n variabili f(x1,…,xn) può essere espansa nella forma di congiunzione di tutti i suoi maxtermini.

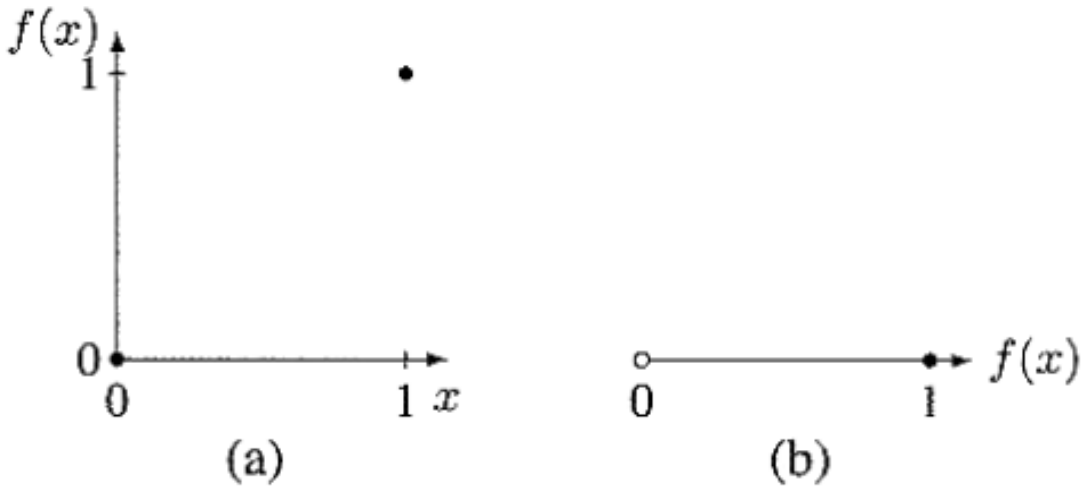
Un **maxtermine** è un termine somma in cui compaiono tutte le variabili di ingresso corrispondenti alle combinazioni per cui la funzione booleana assume valore 0. In particolare il *maxtermine* è composto dalle *variabili che assumono valore 0 prese in forma naturale e da quelle che assumono valore 1 prese in forma complementata*.

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

Forma S $f=(x+y+z)(x+y'+z)(x+y'+z')$

Rappresentazione grafica mediante ipercubi(1/3)

Ricorrendo alla rappresentazione cartesiana tradizionale, una funzione di una variabile f(x)=x può essere descritta in uno spazio bidimensionale; dato che nell'algebra di commutazione la variabile x può assumere solo i valori {0,1}, è possibile rappresentare tale funzione su una sola dimensione, in cui il valore della funzione è indicato da un punto pieno se uguale a 1 e da un punto vuoto se uguale a 0.

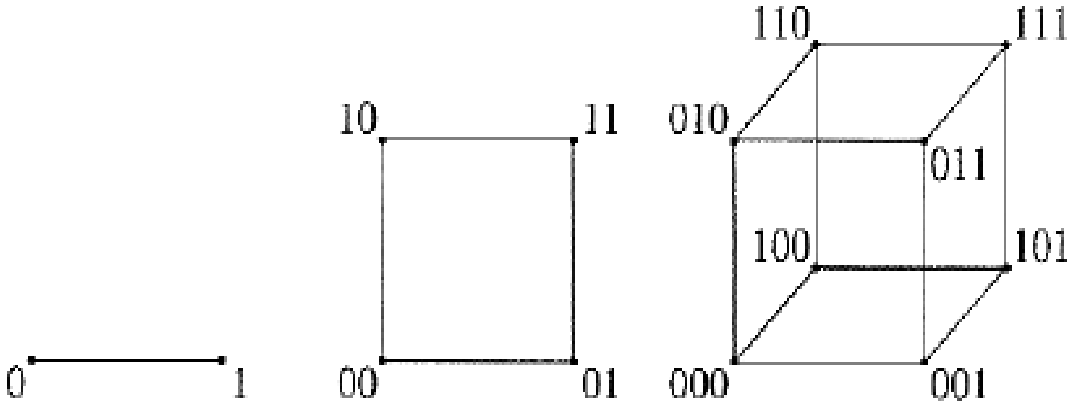


(a) rappresentazione tradizionale (b) rappresentazione semplificata

Rappresentazione grafica mediante ipercubi(2/3)

E' possibile generalizzare questa rappresentazione per funzioni di n variabili semplicemente ricorrendo ad uno spazio n-dimensionale.

Inoltre dato che sugli assi gli unici valori possibili sono 0 e 1, si usa ricorrere ad una rappresentazione ancora leggermente diversa, basata sugli n-cubi, ovvero ipercubi n-dimensionali in cui ogni vertice è annotato con le sue coordinate.

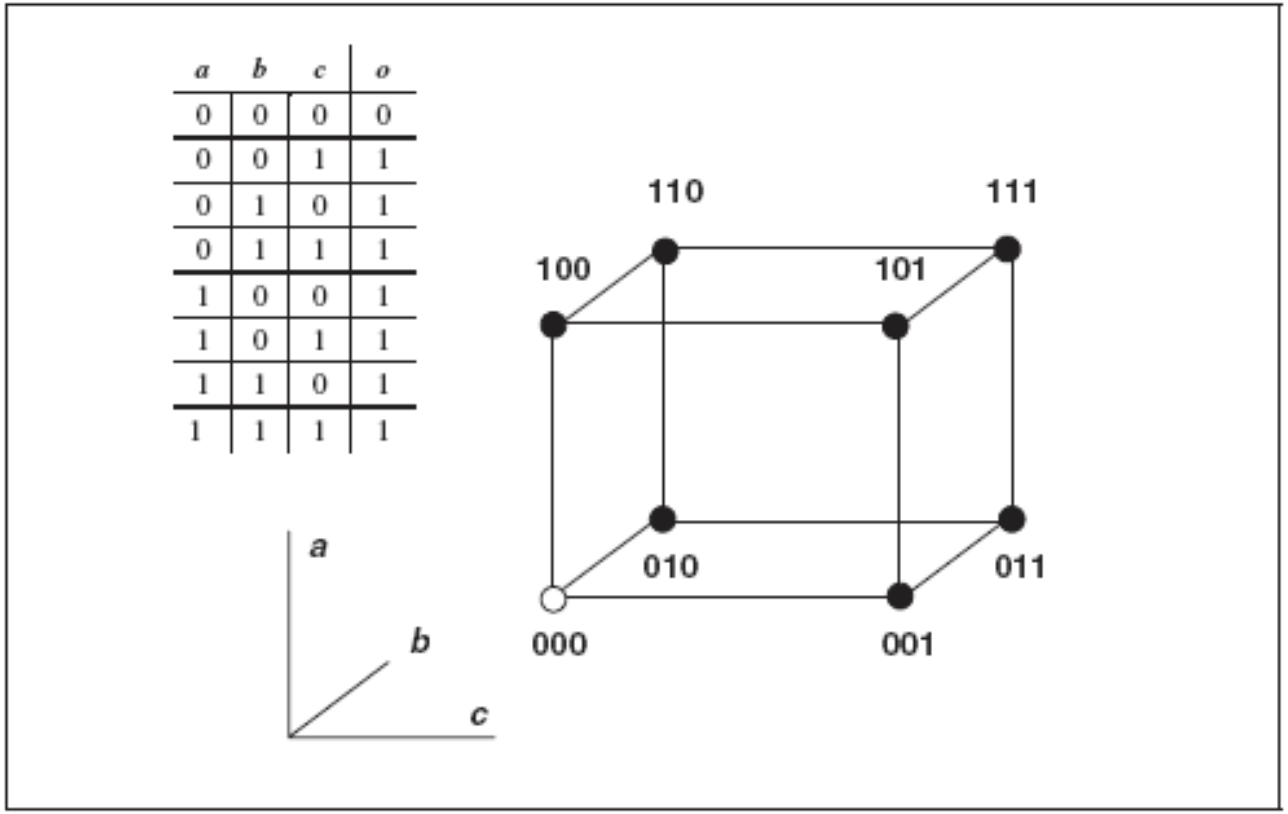


Rappresentazione di un 1-cubo, un 2-cubo e un 3-cubo

Rappresentazione grafica mediante ipercubi(3/3)

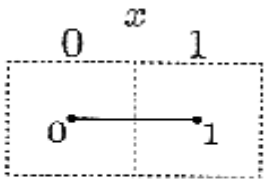
Tabella delle verità e rappresentazione cubica della funzione OR(a, b, c) descritta come $f(B^3) = B$

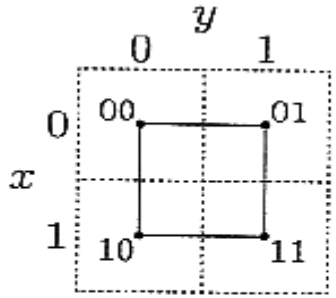
a	*b*	*c*	*o*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

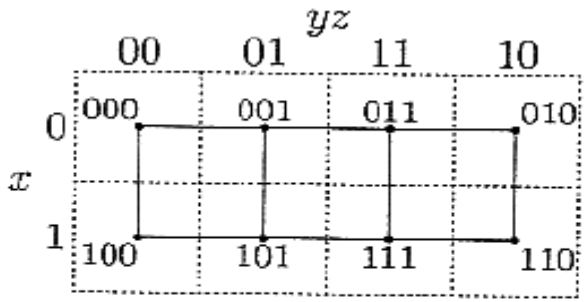


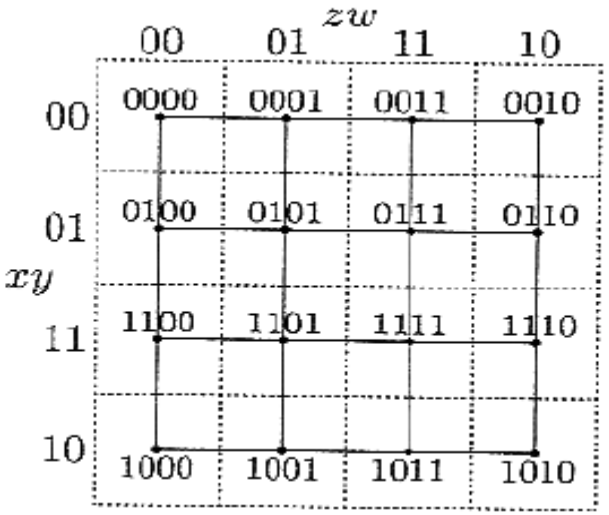
Rappresentazione mediante mappe di Karnaugh (1/2)

Una mappa di Karnaugh è una trasposizione di un n-cubo su due dimensioni che mantiene le proprietà di adiacenza che caratterizzano i vertici degli n-cubi









Rappresentazione mediante mappe di Karnaugh - esempi (2/2)

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

Mappa di Karnaugh della funzione OR(a, b, c)

x \ yz	00	01	11	10
0	0	0	0	1
1	0	0	1	1

Mappa di Karnaugh della funzione f(x,y,z)=xyz+xyz'+x'yz'

Ottimizzazione di funzioni combinatorie

Per *ottimizzazione* di una funzione si intende la sua trasformazione, attraverso passi successivi, con lo scopo di ottenere un'espressione equivalente ma migliore rispetto ad una data metrica di valutazione (area occupata, tempo necessario a produrre un dato risultato, potenza o energia assorbita ecc.).

Possibili metriche di area:

- Numero di letterali
- Numero di porte logiche
- Numero di ingressi

Costo di una funzione (1/4)

Il costo in termini di *letterali* C_L è pari al numero delle variabili indipendenti della funzione, ciascuna moltiplicata per il numero di volte che essa compare nella forma.

Il costo in termini di *funzioni* o *porte* C_P è pari al numero delle funzioni elementari f_i che la compongono, che per reti unilaterali è uguale al numero complessivo di porte adoperate.

Costo di una funzione (2/4)

Il costo in termini di *ingressi* C_I è pari al numero delle funzioni f_i che la compongono, ciascuna moltiplicata per le variabili (dipendenti o indipendenti) di cui è funzione. Per reti unilaterali tale costo equivale al numero complessivo di porte adoperate, ciascuna pesata per il numero di ingressi (fan-in).

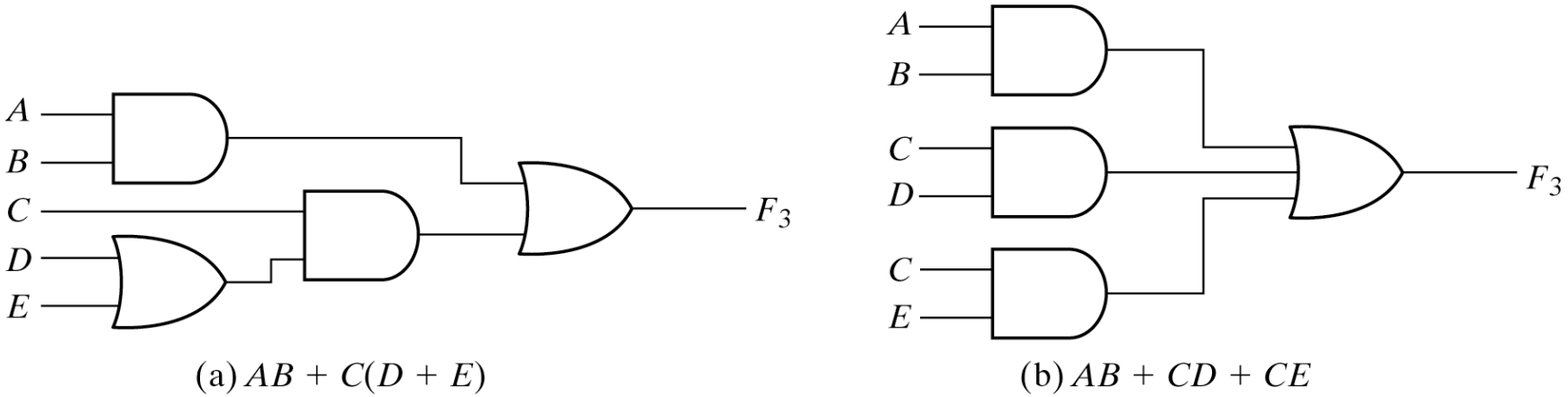
Esempio.

f=bc(a!d+!b+c)+!c(d+!a)(b+c) $\mathbf{C_L}$=11 $\mathbf{C_P}$=7 $\mathbf{C_I}$=17

f=b(!a+c+d) $\mathbf{C_L}$=4 $\mathbf{C_P}$=2 $\mathbf{C_I}$=5

Costo di una funzione (3/4)

F3=AB+C(D+E)
F3=AB+CD+CE
costo letterali=5 per entrambi i circuiti;
costo ingressi=8 per (a) e 9 per (b)



(a) $AB + C(D + E)$ (b) $AB + CD + CE$

Fig. 2-4 Three- and Two-Level implementation

Costo di una funzione (4/4)

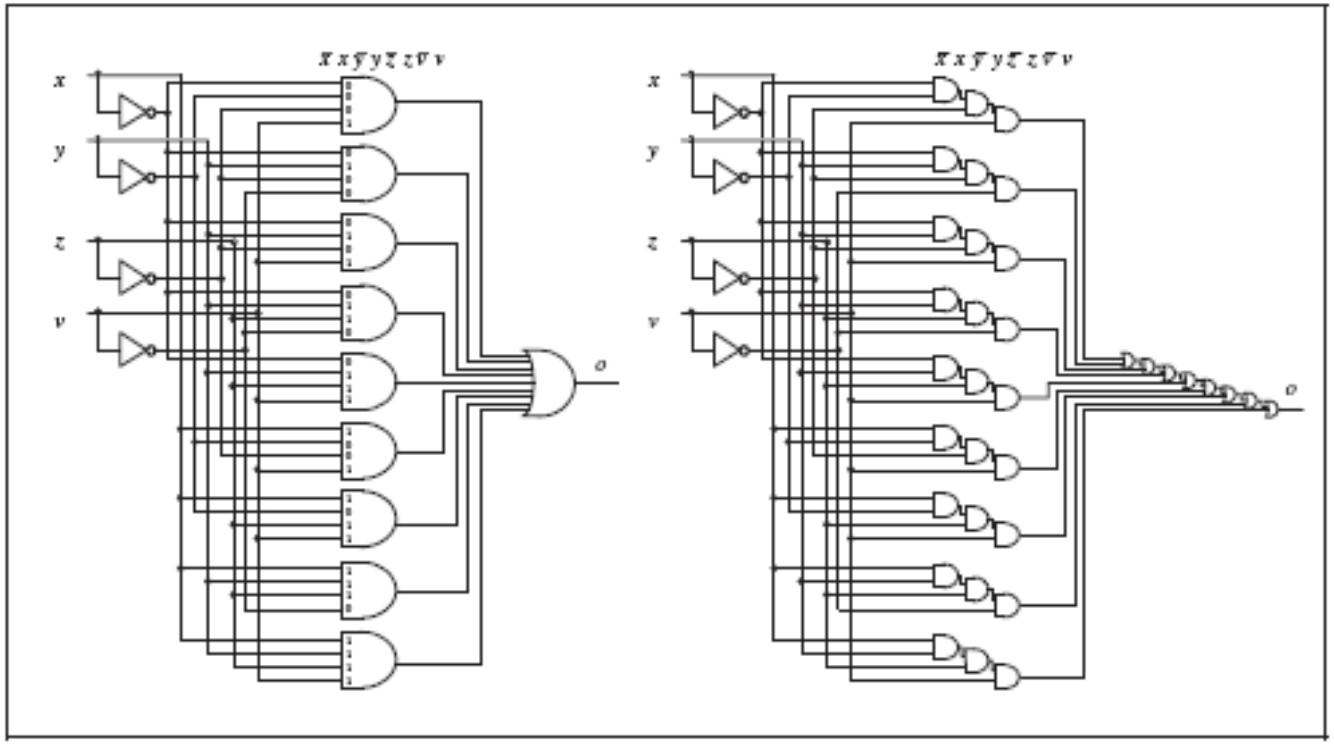
G=ABCD +!A!B!C!D (i)

G=(!A+B)(!B+C)(!C+D)(!D+A) (ii)

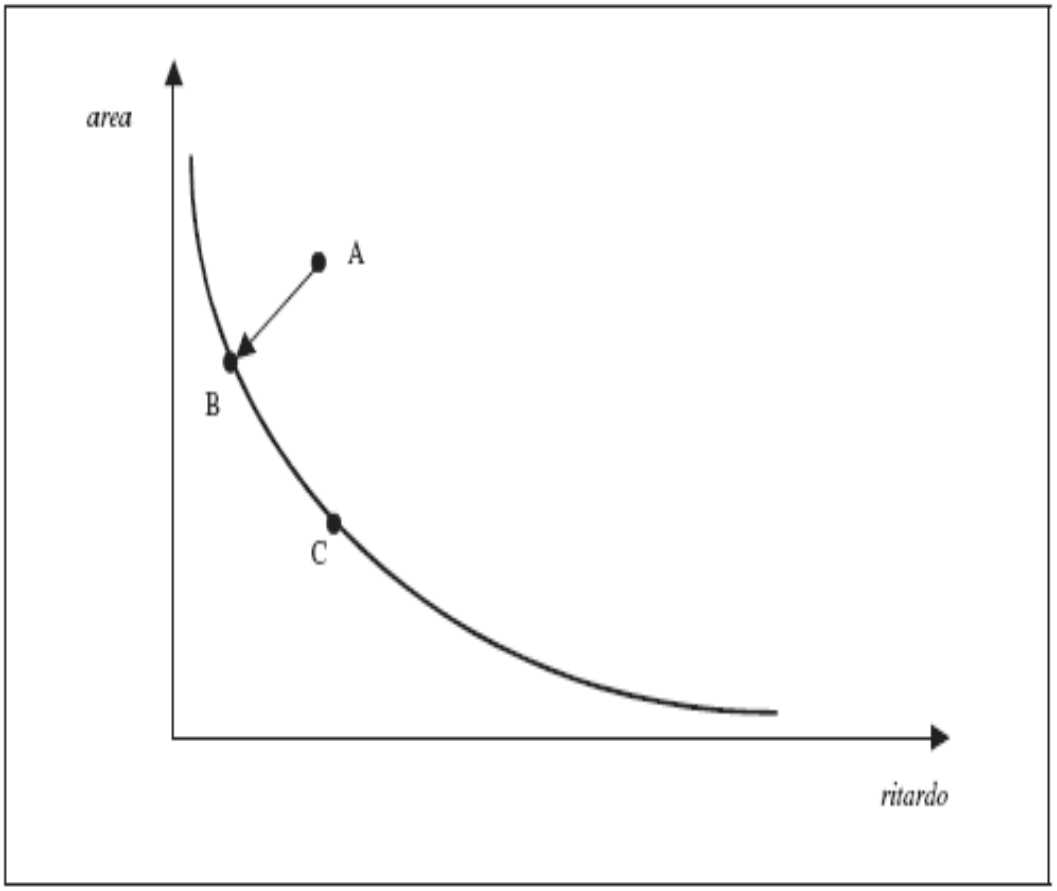
- Costo letterali (i) e (ii) è pari a 8 ma la forma (ii) occupa una maggiore area
- Costo ingressi è pari a 8+2=10 per (i)
- Costo ingressi è pari a 8+4=12 per (ii)

Trade-off fra occupazione di area e ritardo (1/2)

Vincolo tecnologico: trasformazione di un circuito in somma di prodotti a due livelli in un circuito a più livelli con sole porte a 2 ingressi



Trade-off fra occupazione di area e ritardo (2/2)



Riferendosi ai soli parametri area-tempo, idealmente i progetti che corrispondono a un bilancio ottimo si trovano su un'iperbole.

Il progettista spesso parte da un progetto sub-ottimo (punto A) per spostarsi verso soluzioni rispondenti al bilancio e che premiano uno o l'altro dei parametri di valutazione (il punto B corrisponde a un circuito più costoso in termini di area ma anche più veloce, il punto C permette un risparmio in area ma presenta un maggiore ritardo di propagazione).

Ottimizzazione di funzioni combinatorie: espansione (1/3)

I metodi di ottimizzazione delle funzioni combinatorie sono basati sull'applicazione delle proprietà dell' Algebra di Boole:

- Assorbimento:

  $P1+P2 = xP + x'P = (x+x')P = P$ *in tal caso si dice che P1 e P2 generano* ***consenso***

- Idempotenza:

  $P+P=P$

Esse consentono di semplificare l'espressione di una funzione a partire dalla sua rappresentazione in forma canonica, che ne assicura la copertura in termini di somma di mintermini (o prodotto di maxtermini).

Ottimizzazione di funzioni combinatorie: espansione (2/3)

- L'applicazione delle proprietà di assorbimento ed idempotenza è alla base del processo di espansione, volto a trasformare l'espressione algebrica di una funzione in modo da costruire termini prodotto (o somma) costituiti dal minor numero possibile di letterali.

- Si introduce così il concetto di implicante, ossia un prodotto (o una somma) di letterali risultante dal processo di espansione, che assorbe più mintermini (maxtermini) semplificando la copertura di una funzione.

Ottimizzazione di funzioni combinatorie: espansione (3/3)

- Applicando ripetutamente il processo di espansione ad una funzione è possibile determinare un insieme di implicanti primi, ossia non ulteriormente semplificabili, candidati a far parte della copertura ottima della funzione.

- Fra gli implicanti primi è possibile estrarre un set di implicanti primi essenziali, necessari alla copertura poiché sono gli unici a coprire qualche “uno” della funzione; la copertura ottima conterrà dunque tali implicanti essenziali più un sottoinsieme degli implicanti primi rimanenti, scelti secondo un criterio di costo.

Tre tipologie di ottimizzazione dei circuiti combinatori

- Circuiti a 2 livelli e 1 uscita: metodo esatto per identificare i primi implicanti essenziali e un metodo esatto o approssimato (branch & bound) per identificare una copertura ottima.
- Circuiti a 2 livelli e più uscite: come prima e metodo approssimato per trovare la copertura ottima basato sull'identificazione di implicanti primi essenziali di ogni singola uscita.
- Circuiti a più livelli e più uscite: numerosi metodi approssimati per esplorare diverse alternative di area e ritardo (i più efficaci: sintesi ottima a 2 liv. di porzioni del circuito a 1 uscita.)

Metodi di Ottimizzazione

I Metodi di Ottimizzazione possono essere classificati in due macrocategorie:

- Metodi Esatti
  - Karnaugh
  - Quine-McCluskey
- Euristiche

Applicabili entrambi a reti a due o più livelli

Metodi esatti

Metodo delle Mappe di Karnaugh

Il metodo delle Mappe di Karnaugh

Si articola in due fasi:

1) **Espansione:** consiste nella ricerca degli implicanti primi, costituiti dai sottocubi di area massima sulle mappe

2) **Copertura:** consiste nel determinare il sottoinsieme minimo di implicanti primi della funzione in grado di coprire tutti i suoi mintermini

Il metodo delle Mappe di Karnaugh - fase di espansione

I mintermini semplificabili sono rappresentati da celle adiacenti sulle mappe: l' operazione di espansione viene effettuata a partire da ogni mintermine in tutte le direzioni per raggrupparne un numero equivalente a una potenza di 2. Ciascun mintermine può appartenere a più raggruppamenti.

x \ yz	00	01	11	10
0	0	0	0	1
1	0	0	1	1

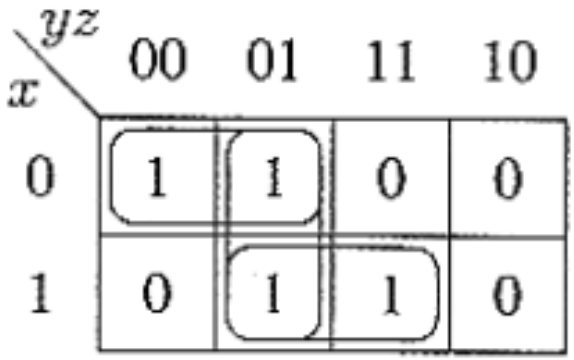
(a)

x \ yz	00	01	11	10
0	0	0	0	1
1	0	0	1	1

(b)

Mappa di Karnaugh della funzione $f(x, y, z) = xyz + xyz' + x'yz$

Il metodo delle Mappe di Karnaugh-Esempio 1



Cosiderando tutti gli implicanti primi individuati si ottiene l'espressione:

f(xyz)= x'y'+ y'z+ xz

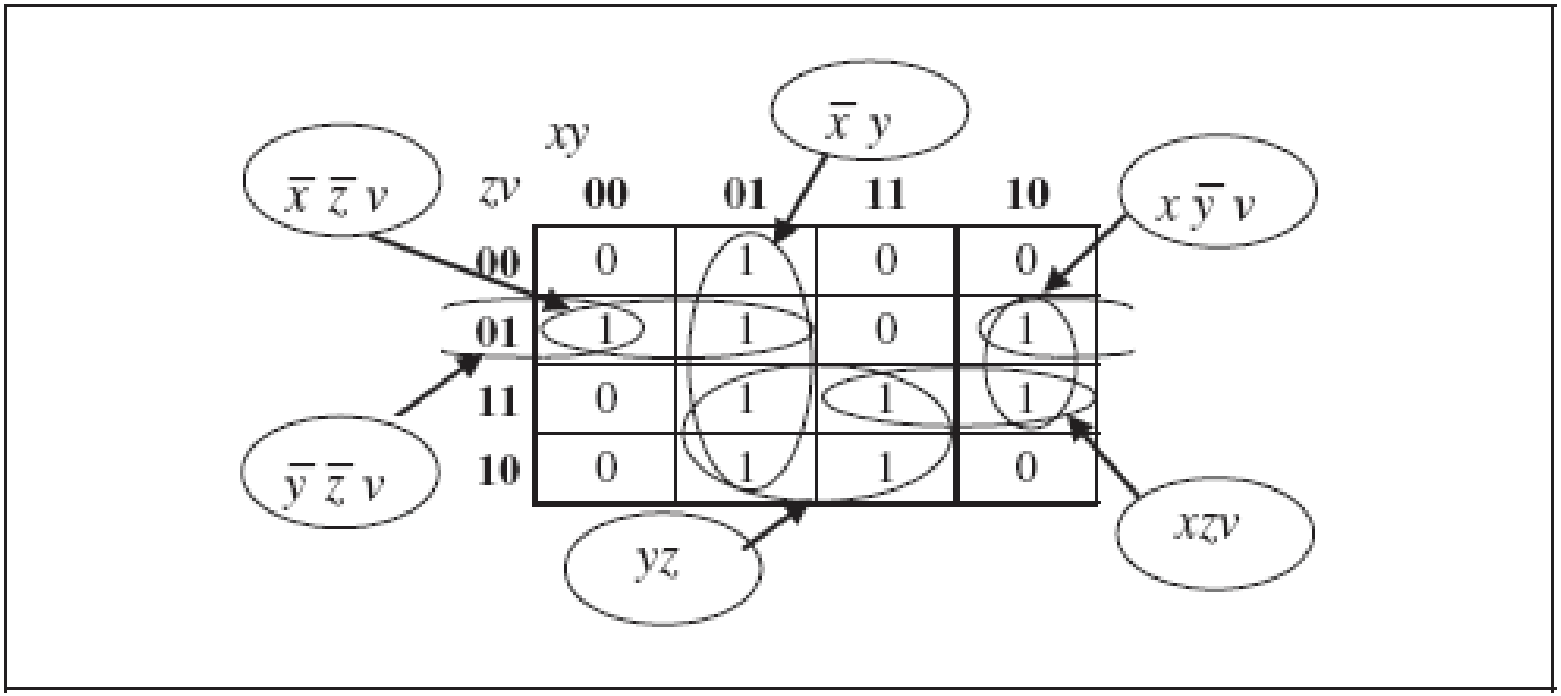
che non risulta minima.

Esaminando la mappa si nota che gli implicanti x'y' e xz sono sufficienti a coprire tutti gli 1 della funzione, e quindi fanno parte della copertura minima mentre l'implicante y'z può essere trascurato. In definitiva quindi:

f(xyz)= x'y'+ xz

Il metodo delle Mappe di Karnaugh – Esempio 2

Mappa di Karnaugh per la funzione
f(*x,y,z,v*)=Σ{1, 4, 5, 6, 7, 9, 11, 14, 15)



Criticità mappe Karnaugh

- Impraticabili per $f(B^n)$ con $n>5$
- Non forniscono informazioni utili all'identificazione di implicanti primi e essenziali utili alla copertura di $f(\bullet)$
- Inapplicabili a funzioni a più uscite
- Rappresentano un metodo grafico e non algoritmico

Metodi esatti

Metodo di Quine Mc Cluskey per funzioni ad una sola uscita su due livelli

Metodo Quine-McCluskey

- Metodo esatto per la sintesi di reti a 2 livelli
- Fattibile fino a circa 20 ingressi
- In grado di considerare funzioni a piú uscite
- Puó minimizzare sia il costo degli implicanti che quello dei letterali

L'algoritmo (facilmente implementabile) opera in due fasi distinte

1) **Espansione**
2) **Copertura**

Il metodo di Quine-McCluskey – I Fase

1. Si considerano i mintermini appartenenti all'ON-Set e al DC-Set della funzione, espressi mediante i valori dei corrispondenti letterali, e li si ordina in senso crescente in base al numero di "1" contenuti, dividendoli in classi.
2. Ogni elemento di ciascuna classe viene confrontato con tutti gli elementi della classe immediatamente successiva allo scopo di individuare consensi: la variabile eventualmente eliminata in caso di consenso viene segnata con il simbolo di don't care nel nuovo implicante generato dal processo di espansione.
   - in caso di confronto fra implicanti contenenti don't care è possibile generare espansione solo se il simbolo di don't care si trova nella stessa posizione nei due implicanti di partenza ed essi differiscono solo per un letterale.

Il metodo di Quine-McCluskey – I Fase

3. Ogni volta che due implicanti partecipano ad un raccoglimento devono essere marcati poiché non rappresentano implicanti primi, e non devono quindi essere considerati nella seconda fase.
   - Nel caso di funzioni non completamente specificate se due mintermini entrambi appartenenti al DC-Set generano espansione, il nuovo implicante viene introdotto nella successiva tabella ma viene marcato a priori, poiché non sarà necessario coprirlo nella successiva fase.
4. Il procedimento viene ripetuto finché non è più possibile determinare consensi; gli implicanti che risulteranno non marcati alla fine della prima fase sono gli implicanti primi della funzione.

Quine-McCluskey – I Fase - ESEMPIO

x	y	z	v	f
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	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
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

m_i	x	y	z	v	
1	0	0	0	1	✓
4	0	1	0	0	✓
5	0	1	0	1	✓
6	0	1	1	0	✓
9	1	0	0	1	✓
7	0	1	1	1	✓
11	1	0	1	1	✓
14	1	1	1	0	✓
15	1	1	1	1	✓

$\{m_r...m_n\}$	x	y	z	v	
1, 5	0	–	0	1	A
1, 9	–	0	0	1	B
4, 5	0	1	0	–	✓
4, 6	0	1	–	0	✓
5, 7	0	1	–	1	✓
6, 7	0	1	1	–	✓
6, 14	–	1	1	0	✓
9, 11	1	0	–	1	C
7, 15	–	1	1	1	✓
11, 15	1	–	1	1	D
14, 15	1	1	1	–	✓

$\{m_r...m_n\}$	x	y	z	v	
4, 5, 6, 7	0	–	–	–	E
6, 7, 14, 15	–	1	1	–	F

A B C D E F sono tutti gli **implicanti primi** della funzione (non sono stati marcati durante il processo di espansione)

Il metodo di Quine-McCluskey – II Fase: copertura (1/3)

1. Si costruisce la **tabella di copertura:** si conviene di porre sulle righe gli implicanti primi di f determinati nella prima fase e sulle colonne tutti i mintermini per cui la funzione vale 1; la casella in posizione (i,j) viene marcata con una "x" o un "1" se l'implicante i copre il mintermine j
   - in caso di funzioni non completamente specificate nella tabella di copertura vengono indicati solo i mintermini appartenenti al'ON-Set della funzione poiché non è necessario coprire le condizioni di indifferenza

NB: nel caso in cui si adopera la convenzione con mintermini sulle righe e implicanti sulle colonne vanno rivisti opportunamente i criteri di essenzialità e dominanza mostrati di seguito

Il metodo di Quine-McCluskey – II Fase: copertura (2/3)

2. Si individuano gli implicanti essenziali, ovvero quelli che sono i soli a coprire un dato mintermine (una sola “x” in una colonna): tali implicanti vengono inseriti nella copertura minima della funzione.

   Si genera una nuova tabella eliminando la riga corrispondente all’implicante essenziale e tutte le colonne dei mintermini da esso coperti nella tabella di partenza.

3. Si riesamina la nuova tabella prodotta finché non è più possibile individuare implicanti essenziali. A questo punto si procede col metodo della dominanza per determinare gli implicanti essenziali secondari.

Il metodo di Quine-McCluskey – criteri di dominanza

- La *riga i domina la riga j* se l'implicante Pi copre tutti i mintermini che copre l'implicante Pj più almeno uno
  - ✓ i mintermini coperti dall'implicante dominato sono un sottoinsieme dei mintermini coperti dall'implicante dominante: scegliendo di eliminare l'implicante dominato e mantere il dominante avremmo la certezza di coprire un insieme maggiore di mintermini, con un costo totale di copertura di sicuro non maggiore di quello che si avrebbe facendo la scelta opposta
- La *colonna i domina la colonna j* se il mintermine mi è coperto da un sottoinsieme degli implicanti che coprono mj
  - ✓ qualsiasi implicante copra m_i copre anche m_j : scegliendo di eliminare il mintermine mj e mantenere mi avremmo la certezza che gli implicanti selezionati per coprire quest ultimo coprono anche il primo, con un costo totale di copertura non maggiore di quello che si avrebbe facendo la scelta opposta

Il metodo di Quine-McCluskey – II Fase: copertura (3/3)

4. Si eliminano dalla tabella le righe e le colonne dominate e si cercano eventuali implicanti essenziali secondari, ovvero quelli che verificano la condizione di essenzialità nella tabella ridotta: tali implicanti vengono inseriti nella copertura della funzione

5. Si ripete il passo 4 finchè non si assicura la copertura di tutti i mintermini della funzione

Quine-McCluskey – II Fase – ESEMPIO (1/2)

	m1	m4	m5	m6	m7	m9	m11	m14	m15
A	x		x						
B	x					x			
C						x	x		
D							x		x
E		x	x	x	x				
F				x	x			x	x

Il mintermine m4 risulta coperto solo da E e il mintermine m14 solo da F:

E ed F pertanto sono implicanti essenziali e possono essere cancellati dalla tabella insieme a tutti i mintermini che coprono.

C(F)= {E,F}

Quine-McCluskey – II Fase – ESEMPIO (2/2)

	m1	m9	m11
A	x		
B	x	x	
C		x	x
D			x

Nella tabella risultante ogni mintermine è coperto almeno da due implicanti: non ci sono più implicanti essenziali e si può procedere col metodo della dominanza:

La riga C domina la riga D e la B domina la A: cancello le righe A e D.

A questo punto B e C sono diventati pseudo-essenziali perché coprono mintermini non coperti da altri implicanti: essi sono allora etichettati come implicanti essenziali *secondari* e aggiunti alla **copertura minima di f**:

C(F)= {E,F,B,C}

	m1	m9	m11
B	x	x	
C		x	x

Esercizio 1 (1/4)

- Minimizzare con il metodo di Quine-McCluskey, la rete con quattro ingressi ed una uscita specificata come segue:

  ONSet={0,2,4,5,6,7,8,9,13,15}; DCSet=∅

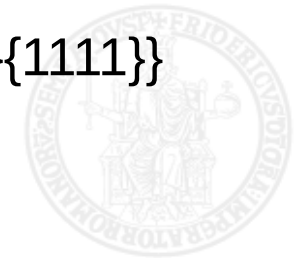
Soluzione:

- Si considerino i valori degli ingressi delle configurazioni che costituiscono l'ONSet e si ricava:

ONSet={0000,0010,0100,0101,0110,0111,1000,1001,1101,1111}

- che dà origine alla seguente partizione:

ONSet={{0000}{0010,0100,1000}{0101,0110,1001}{0111,1101}{1111}}



Esercizio 1 – I fase (2/4)

Passo 0	Passo 1	Passo 2
0000 (0) ~	00-0 (0,2) ~	0--0 (0,4,2,6)
0010 (2) ~	0-00 (0,4) ~	01-- (4,5,6,7)
0100 (4) ~	-000 (0,8)	-1-1 (5,7,13,15)
1000 (8) ~	0-10 (2,6) ~	
0101 (5) ~	010- (4,5) ~	
0110 (6) ~	100- (8,9)	
1001 (9) ~	01-0 (4,6) ~	
0111 (7) ~	01-1 (5,7) ~	
1101 (13) ~	-101 (5,13) ~	
1111 (15) ~	011- (6,7) ~	
	1-01 (9,13)	
	-111 (7,15) ~	
	11-1 (13,15) ~	

Tutte le configurazioni che non sono state marcate con il simbolo "~" sono implicanti primi. Si determina così l'elenco completo degli implicanti primi da considerare:

P1	P2	P3	P4	P5	P6
!a!d	!ab	bd	!b!c!d	a!b!c	a!cd
(0,2,4,6)	(4,5,6,7)	(5,7,13,15)	(0,8)	(8,9)	(9,13)

Seconda fase: si considera la tabella implicanti/mintermini e, applicando i tre consueti criteri, viene semplificata.

Esercizio 1 – II fase (3/4)

	0	2	4	5	6	7	8	9	13	15
P1	X	X	X		X					
P2			X	X	X	X				
P3				X		X			X	X
P4	X						X			
P5							X	X		
P6								X	X	

P1 e P3 sono essenziali: cancello le righe corrispondenti e i mintermini da essi coperti.

La copertura di f è data da:

C(f)={P1,P3}

	0	2	4	5	6	7	8	9	13	15
P1	X	X	X		X					
P2			X	X	X	X				
P3				X		X			X	X
P4	X						X			
P5							X	X		
P6								X	X	

Esercizio 1 - II fase (4/4)

	8	9
~~P4~~	~~X~~	
P5	X	X
~~P6~~		~~X~~

P5 domina P4 e P6 che vengono cancellate.

La copertura di f è data da:

C(f)={P1,P3,P5}

f = P1+P3+P5= !a!d+bd+a!b!c

Esercizio 2 (1/3)

Minimizzare con il metodo di Quine-McCluskey, la rete con quattro ingressi ed una uscita specificata come segue:

ONSet={4,10,11,13,14,15}; DCSet={3,5,6,7}

Soluzione:

Si considerino i valori degli ingressi delle configurazioni che costituiscono l'ONSet e il DCSet si ricava:

ONSet={0100,1010,1011,1101,1110,1111}

DCSet={0011,0101,0110,0111}

che dà origine alla seguente partizione:

{{0100}{1010,0011,0101,0110}{1011,1101,1110,0111}{1111}}

Esercizio 2 – I fase (2/3)

4	0100	✓
3	*0011*	✓
5	*0101*	✓
6	*0110*	✓
10	1010	✓
7	*0111*	✓
11	1011	✓
13	1101	✓
14	1110	✓
15	1111	✓

(*a*)

4,*5*	010–	✓
4,*6*	01–0	✓
3,*7*	*0–11*	✓
3,11	–011	✓
5,*7*	*01–1*	✓
5,13	–101	✓
6,*7*	*011–*	✓
6,14	–110	✓
10,11	101–	✓
10,14	1–10	✓
7,15	–111	✓
11,15	1–11	✓
13,15	11–1	✓
14,15	111–	✓

(*b*)

4,*5*,*6*,*7*	01––	***A***
3,*7*,11,15	––11	***B***
5,*7*,13,15	–1–1	***C***
6,*7*,14,15	–11–	***D***
10,11,14,15	1–1–	***E***

(*c*)

Esercizio 2 – II fase (3/3)

	4	10	11	13	14	15
A	X					
B			X			X
C				X		X
D					X	X
E		X	X		X	X

F = A+C+E=

= !xy+yv+xz

Metodo di Quine-McCluskey: metodi di supporto alla copertura

Sulla tabella non ulteriormente riducibile (non presenta più ne essenzialità ne dominanze di riga o colonna, tabella detta ***ciclica***) occorre fare una scelta di copertura in base a criteri d'ottimo euristici secondo una preassegnata funzione di costo (ad es. metodo Branch&Bound) oppure ricorrere a metodi esatti algebrici, come quello di Petrick.

	x	y	z	w	
P	×			×	1
Q	×	×			3
R		×	×		3
S			×	×	2

Figura 4.30 Tabella di copertura ciclica

Branch & Bound

- è un metodo generale per risolvere problemi di ottimizzazione combinatoria
- L' ottimizzazione combinatoria consiste nell'identificare la combinazione ottima di un numero finito di variabili discrete secondo un prefissato criterio di valutazione.
- nel caso di tabelle cicliche le variabili sono gli implicanti e il criterio di costo è la cardinalità, o il numero di letterali, della soluzione individuata, nel rispetto dei vincoli che impone la copertura di tutti i mintermini

Branch & Bound: costruzione dell'albero delle soluzioni (1/4)

	p	q	r	s
A	X			X
B			X	X
C	X	X		
D		X	X	

L'analisi esaustiva di tutte le possibili soluzioni parte dall'insieme vuoto e, ad ogni passo, aggiunge ogni possibile implicante all'insieme di copertura, fino a che la condizione di copertura non risulta soddisfatta.

Partendo dall'insieme vuoto la scelta può ricadere su uno qualsiasi dei 4 implicanti: questa soluzione può essere rappresentata dall'albero parziale in figura.

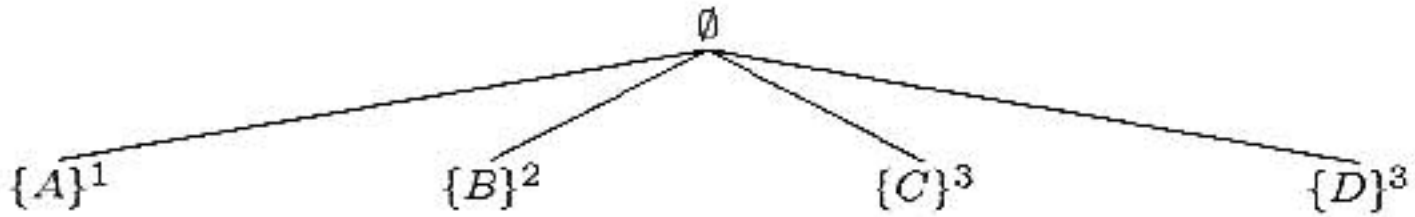


Figura 4.32 Costruzione dell'albero iniziale

Branch & Bound: costruzione dell'albero delle soluzioni (2/4)

Poiché nessuna delle coperture è completa è necessario costruire nuove soluzioni: si scelga arbitrariamente il nodo da cui partire (per es.A) e di procede costruendo le nuove possibili soluzioni parziali:

{A,B} {A,C}{A,D}

La soluzione {A,D} è completa poiché soddisfa i vincoli di copertura e il suo costo, di 4 letterali, diviene il bound iniziale

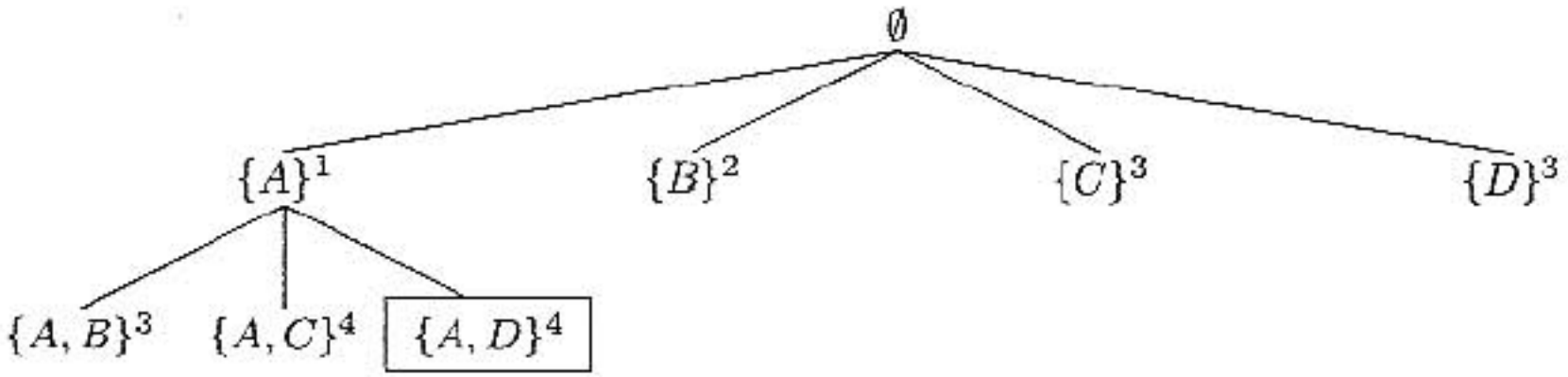


Figura 4.33 Individuazione di un bound

Branch & Bound: costruzione dell'albero delle soluzioni (3/4)

Continuando la costruzione dell'albero si nota che il costo della soluzione {A,B} è minore del bound e quindi si procede alla generazione di nuove soluzioni ottenendo {A,B,C} e {A,B,D}, entrambe a costo maggiore del bound e quindi da scartare.

L'altro ramo ancora da espandere è quello corrispondente a {A,C}: poiché il suo costo non è inferiore al bound ogni soluzione completa avrà costo maggiore e sarà quindi scartata.

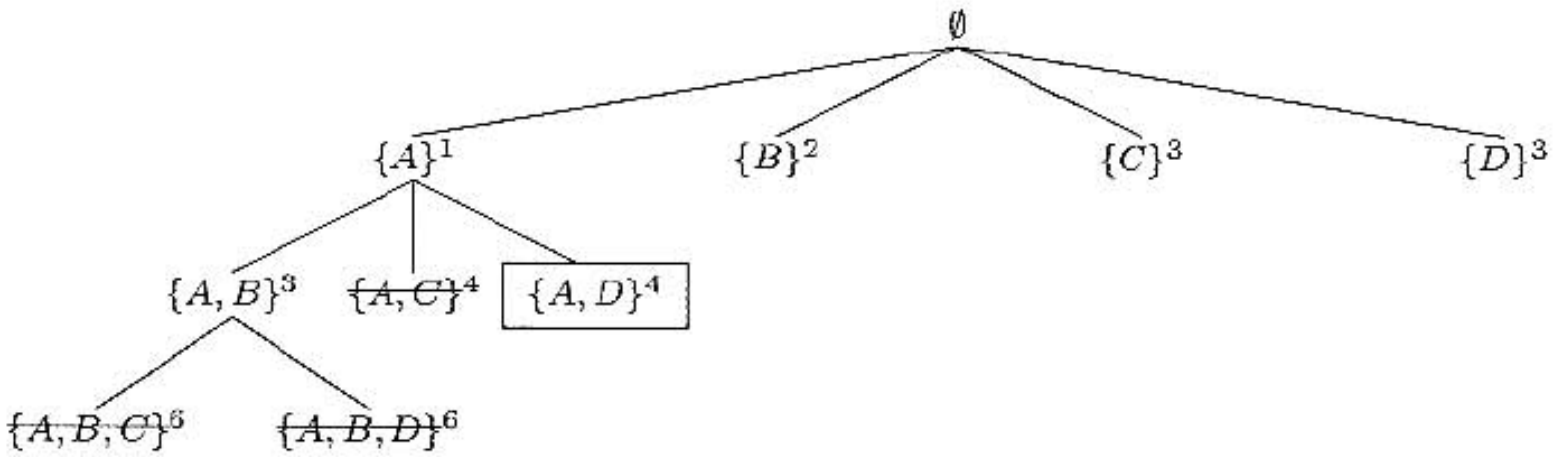


Figura 4.34 Branching limitato dal vincolo imposto dal bound

Branch & Bound: costruzione dell'albero delle soluzioni (4/4)

A questo punto tutto il sottoalbero dipendente dalla scelta iniziale di A è stato esaminato e la ricerca continua a partire dalla nuova scelta iniziale dell'implicante B. Si ottiene così il nuovo albero che non porta ad alcuna soluzione e si arresta alle soluzioni parziali {B,C} e {B,D}.

La copertura {B,A} essendo già stata considerata non viene ripetuta poiché supera il bound. Infine rimane la scelta iniziale di C che porta alla copertura parziale {C,D}, anche questa scartata poiché maggiore del bound.

In definitiva la ***soluzione ottima è data da {A,D} con costo 4***

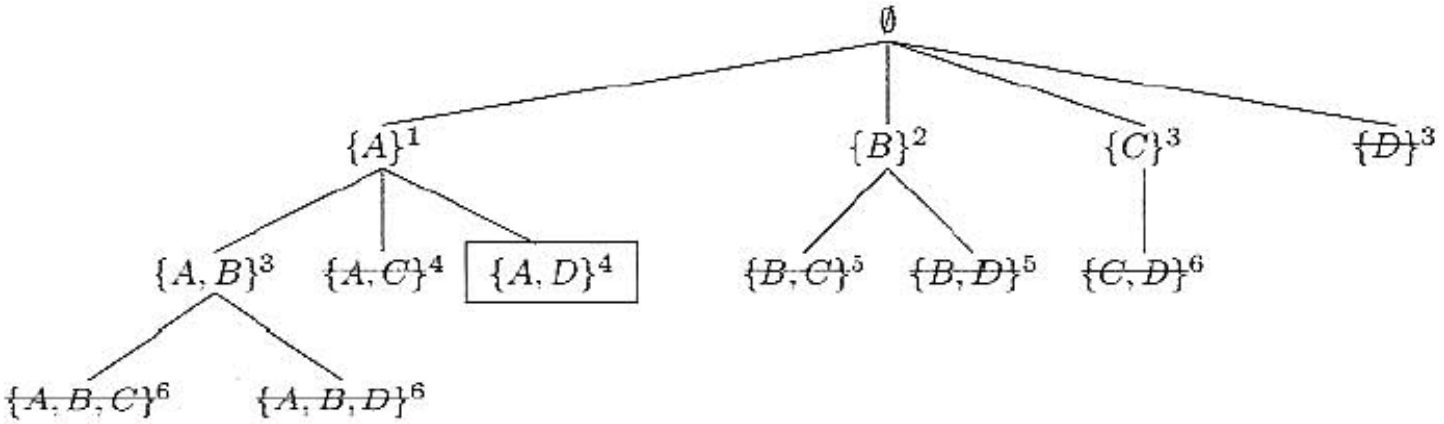


Figura 4.35 Albero finale

Metodo Quine-McCluskey: Branch&Bound (1/2)

- Si sceglie un implicante primo `Pi` come appartenente alla soluzione e si elimina la riga corrispondente e le colonne coperte da `Pi` dalla tabella di copertura.
- La tabella ridotta viene esaminata per altre possibili semplificazioni (righe essenziali o relazioni di dominanza) che **possono** portare ad una soluzione finale `Si` di costo `Ci`.
- Il processo viene ripetuto per tutte le possibili scelte di `Pi` come implicante primo appartenente alla soluzione finale.
- Si mantiene sempre la soluzione a costo minore (*bound*) e si confronta il costo parziale ottenuto con il costo minore, quando lo si supera quella soluzione viene abbandonata.
- Se la selezione di un implicante primo non porta ad una soluzione e si arriva ad una tabella ridotta ciclica, si seleziona un secondo implicante primo `Pj` tra quelli rimasti e si calcolano tutte le possibili soluzioni con `Pi` e `Pj` come elementi della soluzione. Si itera per tutti i possibili `Pj`.

Metodo Quine-McCluskey: Branch&Bound (2/2)

- Metodo che genera molte possibili soluzioni attraverso un processo di ricerca che può crescere esponenzialmente con le dimensioni della funzione.
- Ottimalità garantita solo se si esaminano tutte le possibili alternative.

Metodo di Petrick

- trasforma il problema della copertura in un problema algebrico:
- si devono coprire tutti i mintermini
- ogni mintermine può essere coperto da più implicanti

Metodo di Petrick

- Metodo sistematico e non esaustivo per identificare coperture di implicanti primi minimi.
- Esprimere tutte le condizioni di copertura che devono essere soddisfatte dagli implicanti primi e dai mintermini in forma di prodotto di somme:
  - Un termine somma per ogni mintermine
  - Il termine somma è composto dagli implicanti che rappresentano una copertura del mintermine
- Convertire il prodotto di somme in somma di prodotti applicando le leggi dell'algebra Booleana.
- Ogni termine prodotto che contiene il minimo numero di letterali specifica una copertura di implicanti primi minima e rappresenta quindi una possibile soluzione di copertura.

Metodo di Petrick - esempio

	A	B	C	D	E
P0	x	x			
P1		x	x		x
P2			x	x	
P3	x			x	x

Il significato della tabella di copertura è il seguente:
per rispettare la funzionalità (*vincolo*)
si deve coprire Il *mintermine* 0, mediante P0 o (OR) mediante P3, e (AND)
si deve coprire Il *mintermine* 3, mediante P0 o (OR) mediante P1, e (AND)
si deve coprire Il *mintermine* 10, mediante P1 o (OR) mediante P2, e ...

Da un *prodotto di somme*

$$(P_0+P_3) * (P_0+P_1) * (P_1+P_2) * (P_2+P_3) * (P_1+P_3) = 1$$

$$(P_0+P_3\ P_1) * (P_1P_3+P_2) * (P_1+P_3) = 1$$

$$(P_0P_2+P_3\ P_1) * (P_1+P_3) = 1$$

Ad una *somma di prodotti*

$$(P_0P_2P_1+P_0P_2P_3+P_3P_1) = 1$$

Gruppi di implicanti primi: $P_0P_2P_1$; $P_0P_2P_3$; $P_3\ P_1$

Metodi esatti

Metodo di Quine Mc Cluskey
per funzioni a più uscite
su due livelli

Minimizzazione di funzioni a più uscite

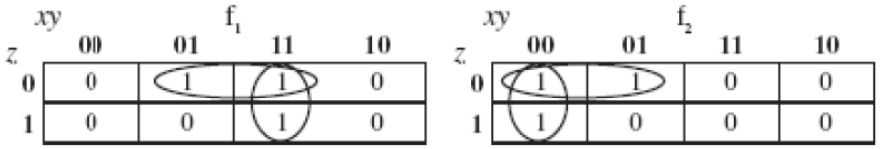
Un possibile modo di procedere all'ottimizzazione di una funzione a più uscite consiste nel minimizzare individualmente le singole uscite con uno degli algoritmi visti

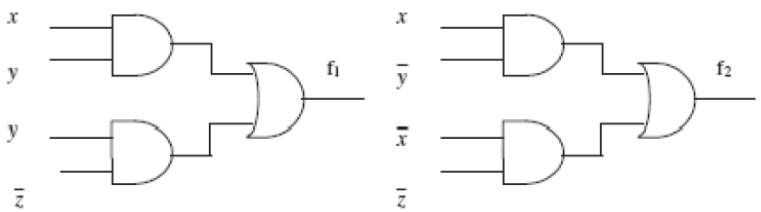
La minimizzazione individuale in questo caso porta alle espressioni:

f1=xy+yz'

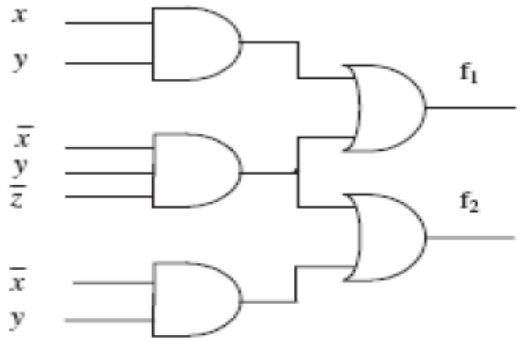
f2=x'y'+x'z'

Esempio





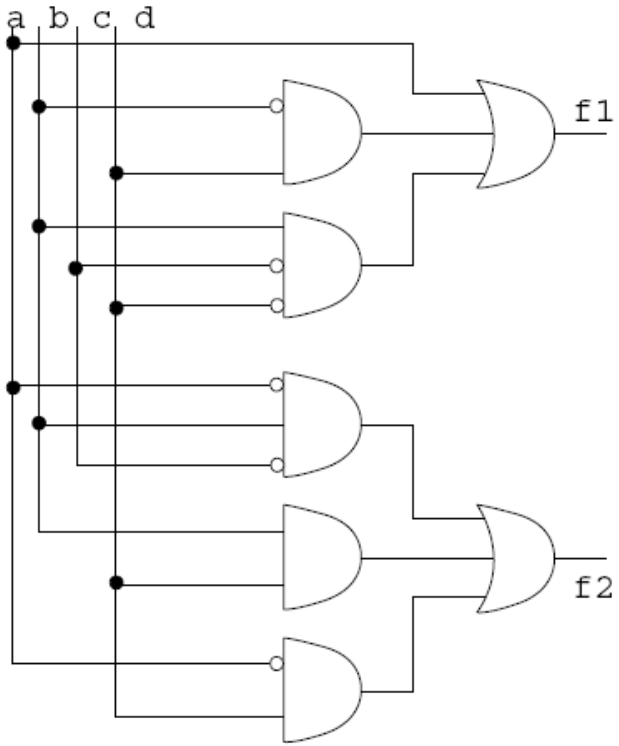
Questa scelta tuttavia non consente di evidenziare eventuali porzioni comuni ad alcune delle funzioni, limitando il riuso degli implicanti



La rete in figura realizza le stesse funzioni f1 e f2 con un costo complessivo inferiore della precedente soluzione

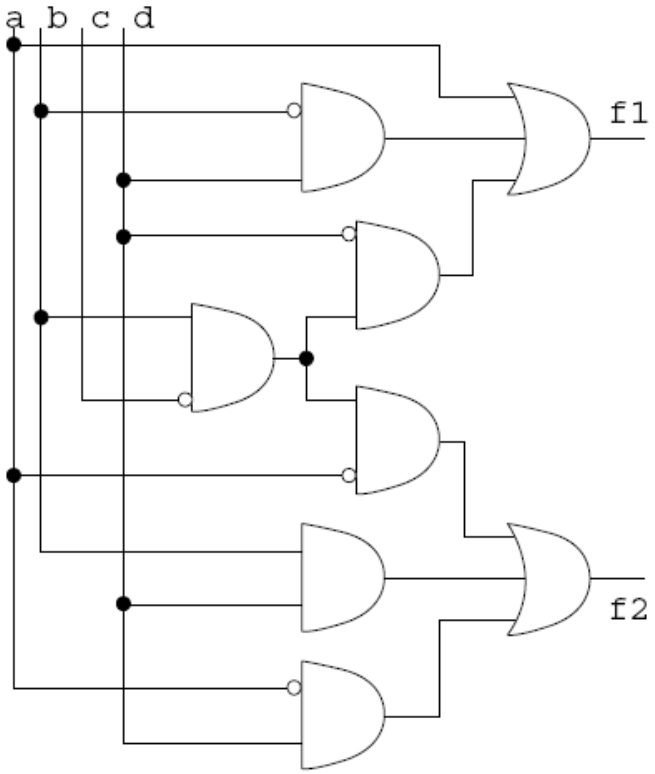
Minimizzazione di funzioni a più uscite

- Esempio:



Forma ottima senza condivisione

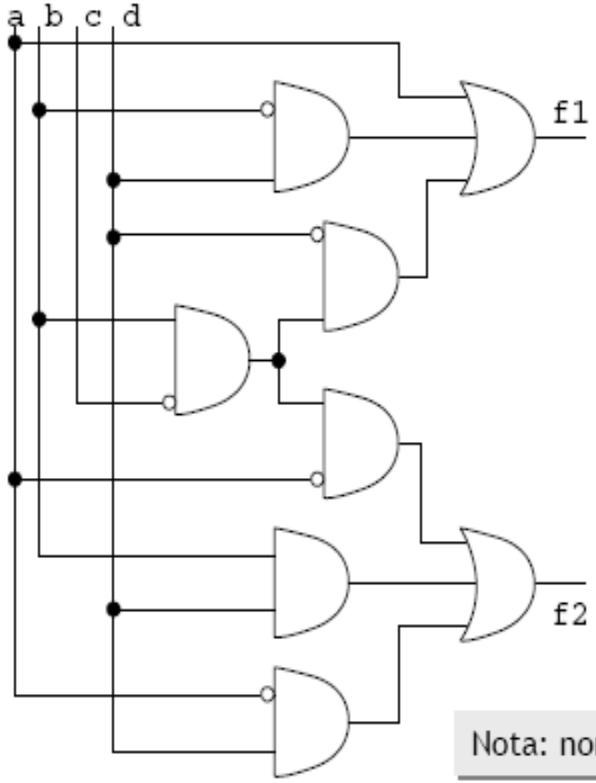
In queste condizioni, il massimo che posso pensare di fare è applicare la seguente condivisione: `b!c!d` di `f1` e `!ab!c` di `f2` possono condividere il termine `b!c`.



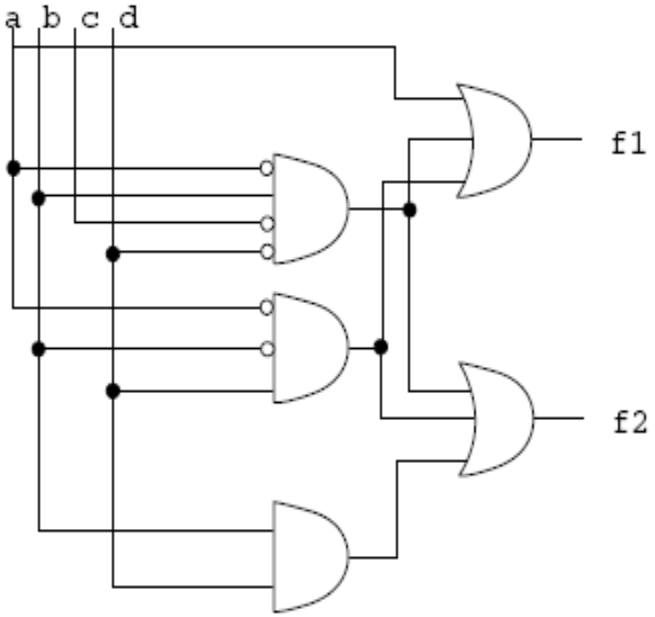
Forma sub-ottima con condivisione

Minimizzazione di funzioni a più uscite

- Esempio:



?



Nota: non si dimentichi il vincolo dei due livelli deve permanere

Forma sub-ottima con condivisione

Forma ottima con condivisione

Il metodo di Quine-McCluskey per funzioni a più uscite – I Fase (1/5)

1. Si considerano i mintermini appartenenti all'ON-Set e al DC-Set di ciascuna uscita, espressi mediante i valori dei corrispondenti letterali, e li si ordina in senso crescente in base al numero di "1" contenuti, dividendoli in classi.

   A ciascun mintermine si associa un ulteriore identificatore costituito da tanti bit quante sono le funzioni considerate:

   - il bit assume valore 1 se e solo se la funzione che ad esso corrisponde contiene tale mintermine

     se !abc→011 appartiene a f1 e f3 ma non a f2 allora l'identificatore ad esso associato è 101

Il metodo di Quine-McCluskey per funzioni a più uscite – I Fase (2/5)

2. Si cercano i mintermini adiacenti confrontando ciascun elemento di una classe con tutti quelli della classe successiva secondo le modalità viste in precedenza;

   L'identificatore di ogni nuovo implicante generato viene ottenuto come l'AND bit a bit dei corrispondenti identificatori di partenza.

   Possono verificarsi 4 casi, mostrati di seguito

Il metodo di Quine-McCluskey per funzioni a più uscite – I Fase (3/5) Generazione della maschera

- Quattro casi possibili – esempi:
  1. l'identificatore di appartenenza risultante è 000…000
     - ✓ La configurazione ottenuta non corrisponde a nessuna espansione valida poiché non appartiene a nessuna delle funzioni

     0000 → 1100

     AND ➡ 0000 nessun implicante introdotto, nessuna marcatura

     0001 → 0011

  2. L'identificatore di appartenenza risultante non coincide con nessuno degli identificatori di partenza
     - ✓ La configurazione ottenuta corrisponde a un espansione valida ma non coinvolge tutte le funzioni né del primo né del secondo implicante coinvolto

     1100 → 1110

     AND ➡ 110- → 0110 Il nuovo implicante viene introdotto ma nessuno degli implicanti di partenza viene marcato

     1101 → 0111

Il metodo di Quine-McCluskey per funzioni a più uscite – I Fase (4/5) Generazione della maschera

- Quattro casi possibili – esempi:
  3. L'identificatore di appartenenza risultante coincide con uno solo dei due identificatori di partenza
     - ✓ La configurazione ottenuta corrisponde a una espansione valida che coinvolge tutte le funzioni del solo implicante coinvolto

     0000 → 1100 X

     AND ➡ 000- → 1100

     0001 → 1101

     Il nuovo implicante viene introdotto e viene marcato solo l'implicante di partenza il cui identificatore coincide con quello risultante dalla AND

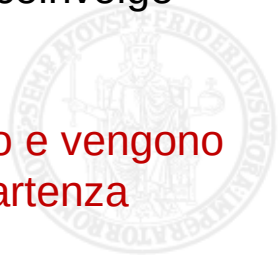
  4. L'identificatore di appartenenza risultante coincide con entrambi gli identificatori di partenza
     - ✓ La configurazione ottenuta corrisponde a una espansione valida che coinvolge tutte le funzioni di entrambi gli implicante coinvolti

     1100 → 1100 X

     AND ➡ 110- → 1100

     1101 → 1100 X

     Il nuovo implicante viene introdotto e vengono marcati entrambi gli implicanti di partenza



Il metodo di Quine-McCluskey per funzioni a più uscite – I Fase (5/5)

4. Il procedimento viene ripetuto finché non è più possibile determinare consensi; gli implicanti che risulteranno *non marcati* alla fine della prima fase sono gli implicanti primi della funzione vettoriale.

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (1/6)

1. Si costruisce la tabella di copertura della funzione vettoriale, ottenuta per giustapposizione delle tabelle di copertura relative a ciascuna uscita
2. Si individuano gli implicanti primi essenziali e si effettuano le semplificazioni viste per il caso monofunzione
3. Si applicano i criteri di dominanza ed essenzialità secondaria per ridurre ulteriormente la tabella, fino a che non si è assicurata la copertura di tutti i mintermini

NB: i criteri di essenzialità primaria e secondaria e di dominanza vengono modificati come mostrato di seguito

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (2/6) *Essenzialità*

Si applica in maniera simile al caso di singola uscita con le seguenti differenze:

- ✓ se l'implicante in oggetto è essenziale per tutte le funzioni coinvolte la riga viene eliminata (e l'implicante scelto per la copertura delle relative funzioni) così come tutte le colonne coperte
- ✓ se l'implicante in oggetto non è essenziale per tutte le funzioni coinvolte (una o più funzioni hanno tale implicante non essenziale) la riga viene mantenuta.

  Come per il caso monofunzione l'implicante viene inserito nella copertura delle funzioni per cui è essenziale e vengono cancellate le colonne relative ai mintermini in esse coperti.

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (3/6) *Essenzialità*

- Se un implicante è essenziale solo per una parte delle funzioni di uscita esso viene scelto per la copertura delle funzioni suddette ma la riga ad esso relativa non può essere cancellata dalla tabella.
- A causa di questo modo di procedere potrebbe succedere che l'implicante in questione venga scelto successivamente per una delle restanti funzioni; in tal caso bisogna valutare l'impatto sul costo complessivo della copertura a seguito dell'introduzione di tale implicante.

Per tener traccia di ciò si associa all'implicante in questione una funzione di costo (es. numero di implicanti o di letterali)

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (4/6) *Essenzialità – criteri di costo*

- Costo in termini di numero di porte: a ciascun implicante è associato un costo iniziale unitario, che viene posto a zero quando l'implicante è selezionato per essenzialità primaria o secondaria per un sottoinsieme delle uscite e rimane nella tabella (la porta è già stata inserita nella copertura, aggiungerla comporta un costo nullo)

- Costo in termini di numero di letterali: a ciascun implicante è associato un costo iniziale pari al numero di letterali; tale costo viene ridotto a 1 quando l'implicante è selezionato per essenzialità primaria o secondaria per un sottoinsieme delle uscite e rimane nella tabella (l'implicante è già stato inserito nella copertura, aggiungerlo comporta un costo unitario dovuto al pin di collegamento dell'uscita della porta corrispondente all'implicante con l'ingresso di una successiva porta)

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (5/6) *Dominanza*

I concetti di dominanza di riga e di colonna vengono modificati come segue:

- **Dominanza di riga**

La *riga i domina la riga j* se l'implicante Pi copre tutti i mintermini che copre l'implicante Pj più almeno uno e il costo associato a Pi è minore o uguale al costo associato a Pj

- **Dominanza di colonna**

La dominanza di colonna ha la stessa definizione del caso di funzioni a una sola uscita ma si applica solo tra colonne della stessa funzione

Il metodo di Quine-McCluskey per funzioni a più uscite – II Fase (6/6)

Dominanza

Come per il caso monofunzione la tabella di copertura può essere ridotta eliminando le righe e le colonne dominate, secondo le definizioni appena viste.

Cardinalità e costo: osservazioni

Come visto, l'identificazione della copertura ottima può considerare, oltre alla cardinalità della copertura, anche il costo di ogni implicante.

L'aggiunta del costo di ogni implicante potrebbe aumentare il livello di precisione nella ricerca della soluzione. Comunque, oltre ad aumentare la complessità algoritmica, tale livello di precisione potrebbe essere assolutamente inutile se si considera che il collegamento alla libreria tecnologica (library binding) cambia la struttura del circuito e, come conseguenza, il costo della realizzazione.

In media due soluzioni che differiscono nel costo stimato del 10%-20% sono da considerarsi equivalenti.

Esercizio (1/9)

(tratto da "Reti Logiche" di Bolchini,Brandolese,Salice,Sciuto

- Minimizzare con il metodo di Quine-McCluskey, la rete vettoriale specificata come segue:

  ONSet1={0,2,4,5,6,7}; DCSet1=∅

  ONSet2={0,4,5,7,10,11,14,15}; DCSet2=∅

  ONSet3={2,6,10,11,14,15}; DCSet3=∅

Soluzione:

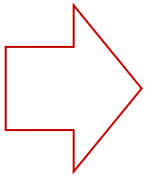
- L'insieme di tutti i mintermini è:

  $ONSet = ONSet1 \cup ONSet2 \cup ONSet3 =$

  $= \{0,2,4,5,6,7,10,11,14,15\}$

Esercizio – I Fase (2/9)

mi	x	y	z	v	f1	f2	f3
0	0	0	0	0	1	1	0
1	0	0	0	1	0	0	0
2	0	0	1	0	1	0	1
3	0	0	1	1	0	0	0
4	0	1	0	0	1	1	0
5	0	1	0	1	1	1	0
6	0	1	1	0	1	0	1
7	0	1	1	1	1	1	0
8	1	0	0	0	0	0	0
9	1	0	0	1	0	0	0
10	1	0	1	0	0	1	1
11	1	0	1	1	0	1	1
12	1	1	0	0	0	0	0
13	1	1	0	1	0	0	0
14	1	1	1	0	0	1	1
15	1	1	1	1	0	1	1



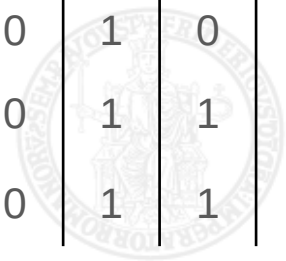
mi	x	y	z	v	f1	f2	f3
0	0	0	0	0	1	1	0
2	0	0	1	0	1	0	1
4	0	1	0	0	1	1	0
5	0	1	0	1	1	1	0
6	0	1	1	0	1	0	1
10	1	0	1	0	0	1	1
7	0	1	1	1	1	1	0
11	1	0	1	1	0	1	1
14	1	1	1	0	0	1	1
15	1	1	1	1	0	1	1

Tabella di partenza in cui sono presenti solo i mintermini per cui almeno una delle funzioni vale 1 divisi in classi

Esercizio – I Fase (3/9)

mi	x	y	z	v	f1	f2	f3	
0	0	0	0	0	1	1	0	x
2	0	0	1	0	1	0	1	x
4	0	1	0	0	1	1	0	x
5	0	1	0	1	1	1	0	x
6	0	1	1	0	1	0	1	x
10	1	0	1	0	0	1	1	x
7	0	1	1	1	1	1	0	x
11	1	0	1	1	0	1	1	x
14	1	1	1	0	0	1	1	x
15	1	1	1	1	0	1	1	x

mi	x	y	z	v	f1	f2	f3
0,2	0	0	-	0	1	0	0
0,4	0	-	0	0	1	1	0
2,6	0	-	1	0	1	0	1
2,10	-	0	1	0	0	0	1
4,5	0	1	0	-	1	1	0
4,6	0	1	-	0	1	0	0
5,7	0	1	-	1	1	1	0
6,7	0	1	1	-	1	0	0
6,14	-	1	1	0	0	0	1
10,11	1	0	1	-	0	1	1
10,14	1	-	1	0	0	1	1
7,15	-	1	1	1	0	1	0
11,15	1	-	1	1	0	1	1
14,15	1	1	1	-	0	1	1



Esercizio – I Fase (4/9)

mi	x	y	z	v	f1	f2	f3	
0,2	0	0	-	0	1	0	0	X
0,4	0	-	0	0	1	1	0	**P0**
2,6	0	-	1	0	1	0	1	**P1**
2,10	-	0	1	0	0	0	1	X
4,5	0	1	0	-	1	1	0	**P2**
4,6	0	1	-	0	1	0	0	X
5,7	0	1	-	1	1	1	0	**P3**
6,7	0	1	1	-	1	0	0	X
6,14	-	1	1	0	0	0	1	X
10,11	1	0	1	-	0	1	1	X
10,14	1	-	1	0	0	1	1	X
7,15	-	1	1	1	0	1	0	**P4**
11,15	1	-	1	1	0	1	1	X
14,15	1	1	1	-	0	1	1	X

mi	x	y	z	v	f1	f2	f3	
0,2,4,6	0	-	-	0	1	0	0	**P5**
2,6,10,14	-	-	1	0	0	0	1	P6
4,5,6,7	0	1	-	-	1	0	0	P7
10,11,14,15	1	-	1	-	0	1	1	P8

IMPLICANTI:

P0=0-00 =x'z'v' *0,4* 110-> f0,f1

P1=0-10 =x'zv' *2,6* 101->f0,f2

P2=010- =x'yz' *4,5* 110->f0,f1

P3=01-1 =x'yv *5,7* 110->f0,f1

P4=-111 = yzv *7,15* 010 ->f1

P5=0--0 =x'v' *0,2,4,6* 100->f1

P6=--10 = zv' *2,6,10,14* 001->f2

P7=01-- = x'y *4,5,6,7* 100->f0

P8=1-1- = xz *10,11,14,15* 011->f1,f2

Esercizio – II Fase (5/9)

Consideriamo il costo in termini di porte: all'inizio ad ogni implicante viene associato costo 1

	f0						f1								f2						
	0	2	4	5	6	7	0	4	5	7	10	11	14	15	2	6	10	11	14	15	
P0	X		X				X	X													1
P1		X			X										X	X					1
P2			X	X				X	X												1
P3				X		X			X	X											1
P4										X				X							1
P5	X	X	X		X																1
P6															X	X	X		X		1
P7			X	X	X	X															1
P8											X	X	X	X			X	X	X	X	1

P0 è essenziale solo per f1 e P8 per f1 e f2:

- cancello le colonne relative ai mintermini coperti
- **cancello la riga P8** perché è essenziale per f1 e f2 e non copre mintermini di f0;
- **non cancello la riga P0** perché è essenziale solo per f1 ma copre anche dei mintermini di f2

Pongo il costo di P0 a 0

C(f0)=∅ C(f1)={P0,P8} C(f2)={P8}

Esercizio – II Fase (6/9)

	f0						f1		f2		
	0	2	4	5	6	7	5	7	2	6	
P0	X		X								0
P1		X			X				X	X	1
P2			X	X			X				1
P3				X		X	X	X			1
P4								X			1
P5	X	X	X		X						1
P6									X	X	1
P7			X	X	X	X					1

La nuova tabella non presenta essenzialità: si procede con i criteri di dominanza:

- P3 domina P4 poiché copre tutti i mintermini di P4 più almeno uno e in più C(P3)<=C(P4) => Elimino P4
- P1 domina P6 => Elimino P6

NB: Secondo la definizione usuale si potrebbe pensare che P5 domina P0: ciò non è vero poiché C(P5) è maggiore di C(P0)

	f0						f1		f2		
	0	2	4	5	6	7	5	7	2	6	
P0	X		X								0
P1		X			X				X	X	1
P2			X	X			X				1
P3				X		X	X	X			1
P5	X	X	X		X						1
P7			X	X	X	X					1

Esercizio – II Fase (7/9)

	f0						f1		f2		
	0	2	4	5	6	7	5	7	2	6	
P0	X		X								0
P1		X			X				X	X	1
P2			X	X			X				1
P3				X		X	X	X			1
P5	X	X	X		X						1
P7			X	X	X	X					1

Emergono delle relazioni di essenzialità:

- P3 è essenziale per f1 ma poiché copre anche f0, per cui non è essenziale, non lo elimino dalla tabella e pongo il suo costo a 0
- P1 è essenziale per f2 ma poiché copre anche f0, per cui non è essenziale, non lo elimino dalla tabella e pongo il suo costo a 0

C(f0)=∅ C(f1)={P0,P8,P3} C(f2)={P8,P1}

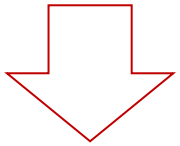
	f0						
	0	2	4	5	6	7	
P0	X		X				0
P1		X			X		**0**
P2			X	X			1
P3				X		X	**0**
P5	X	X	X		X		1
P7			X	X	X	X	1

Esercizio – II Fase (8/9)

	f0						
	0	2	4	5	6	7	
P0	X		X				0
P1		X			X		**0**
P2			X	X			1
P3				X		X	**0**
P5	X	X	X		X		1
P7			X	X	X	X	1

Non si evidenziano essenzialità:applico i criteri di dominanza:

- P7 domina P2: cancello P2 perché C(P7)<=C(P2)
- 0 domina 4 => Elimino 4
- 2 domina 6 => Elimino 6
- 7 domina 5 => Elimino 5



	f0			
	0	2	7	
P0	X			0
P1		X		**0**
P3			X	**0**
P5	X	X		1
P7			X	1

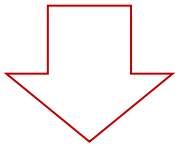
- P3 domina P7 poiché C(P3)<=C(P7)
  => Elimino P7

Esercizio – II Fase (9/9)

	f0			
	0	2	7	
P0	X			0
P1		X		**0**
P3			X	**0**
P5	X	X		1

P3 è essenziale:

C(f0)={P3} C(f1)={P0,P8,P3}
C(f2)={P8,P1}



	f0		
	0	2	
P0	X		0
P1		X	**0**
P3			**0**
P5	X	X	1

Scelgo P0 e P1 per la copertura poiché P5 ha costo 1 mentre P0 e P1 hanno costo 0

C(f0)={P3,P0,P1}
C(f1)={P0,P8,P3}
C(f2)={P8,P1}

Esercizio – variante fase copertura (1/6)

Consideriamo come criterio di costo il numero di letterali di ciascun implicante.

La tabella di copertura dell'esempio svolto si modifica come segue:

Ad ogni implicante viene associato un costo pari al numero di letterali contenuti

	f0						f1								f2						
	0	2	4	5	6	7	0	4	5	7	10	11	14	15	2	6	10	11	14	15	
P0	X		X				X	X													3
P1		X			X										X	X					3
P2			X	X				X	X												3
P3				X		X			X	X											3
P4										X				X							3
P5	X	X	X		X																2
P6															X	X	X		X		2
P7			X	X	X	X															2
P8											X	X	X	X			X	X	X	X	2

Esercizio – variante fase copertura (2/6)

	f0						f1								f2						
	0	2	4	5	6	7	0	4	5	7	10	11	14	15	2	6	10	11	14	15	
P0	X		X				X	X													3
P1		X			X										X	X					3
P2			X	X				X	X												3
P3				X		X			X	X											3
P4										X				X							3
P5	X	X	X		X																2
P6															X	X	X		X		2
P7			X	X	X	X															2
P8											X	X	X	X			X	X	X	X	2

P0 è essenziale solo per f1 e P8 per f1 e f2:

- cancello le colonne relative ai mintermini coperti
- **cancello la riga P8** perché è essenziale per f1 e f2 e non copre mintermini di f0;
- **non cancello la riga P0** perché è essenziale solo per f1 ma copre anche dei mintermini di f2

Pongo il costo di P0 a **1**

C(f0)=∅ C(f1)={P0,P8} C(f2)={P8}

Esercizio – variante fase copertura (3/6)

	f0						f1		f2		
	0	2	4	5	6	7	5	7	2	6	
P0	X		X								1
P1		X			X				X	X	3
P2			X	X			X				3
P3				X		X	X	X			3
P4								X			3
P5	X	X	X		X						2
P6									X	X	2
P7			X	X	X	X					2

La nuova tabella non presenta essenzialità: si procede con i criteri di dominanza:

- P3 domina P4 poiché copre tutti i mintermini di P4 più almeno uno e in più C(P3)<=C(P4) => Elimino P4
- P1 **NON** domina P6 poiché C(P1)>C(P6)
- Le colonne 2 e 6 in f2 si dominano reciprocamente; elimino per esempio la colonna 6
- In f0 0 domina 4, 2 domina 6 e 7 domina 5: elimino 4,6,5

	f0			f1		f2	
	0	2	7	5	7	2	
P0	X						1
P1		X				X	3
P2				X			3
P3			X	X	X		3
P5	X	X					2
P6						X	2
P7			X				2

Esercizio – variante fase copertura (4/6)

	f0			f1		f2	
	0	2	7	5	7	2	
P0	X						1
P1		X				X	3
P2				X			3
P3			X	X	X		3
P5	X	X					2
P6						X	2
P7			X				2

- P3 è essenziale per f1 ma poiché copre anche f0, per cui non è essenziale, non lo elimino dalla tabella e pongo il suo costo a **1**

C(f0)=∅ C(f1)={P0,P8,P3} C(f2)={P8}

	f0			f2	
	0	2	7	2	
P0	X				1
P1		X		X	3
P2					3
P3			X		1
P5	X	X			2
P6				X	2
P7			X		2

- P3 domina P7 per ragioni di costo

Esercizio – variante fase copertura (5/6)

	f0			f2	
	0	2	7	2	
P0	X				1
P1		X		X	3
P2					3
P3			X		1
P5	X	X			2
P6				X	2

P3 diventa essenziale per f0 e viene cancellato dalla tabella

C(f0)={P3} C(f1)={P0,P8,P3} C(f2)={P8}

	f0		f2	
	0	2	2	
P0	X			1
P1		X	X	3
P2				3
P5	X	X		2
P6			X	2

Non posso applicare più i criteri di dominanza per via dei costi

P6 e P1 coprono entrambi il mintermine 2 in f2: poiché P6 ha costo inferiore è possibile eliminare P1 in f2 e selezionare P6 nella copertura

C(f0)={P3}
C(f1)={P0,P8,P3}
C(f2)={P8,P6}

La riga P6 può essere cancellata poiché non copre mintermini in f0

Esercizio – variante fase copertura (6/6)

	f0		
	0	2	
P0	X		1
P1		X	3
P2			3
P5	X	X	2

- P5 domina P1
  => elimino P1

	f0		
	0	2	
P0	X		1
P2			3
P5	X	X	2

- P5 risulta essenziale per f0:

C(f0)={P3,P5}
C(f1)={P0,P8,P3}
C(f2)={P8,P6}

La nuova copertura individuata è
C(f0)={P3,P5} C(f1)={P0,P8,P3}
C(f2)={P8,P6}

Esercizio – confronto criteri di costo

Minimizzazione implicanti:

f0 = P0+P1+P3 = x'z'v'+x'zv'+x'yv

f1 = P0+P3+P8 = x'z'v'+x'yv+xz

f2 = P1+P8 = x'zv'+xz

4 implicanti selezionati

19 letterali

Minimizzazione letterali:

f0 = P3+P5 = x'yv+x'v'

f1 = P0+P3+P8 = x'z'v'+x'yv+xz

f2 = P6+P8 = zv'+xz

5 implicanti selezionati

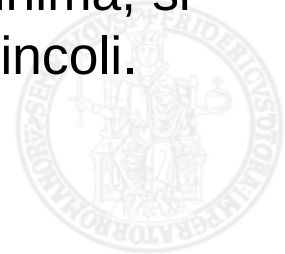
16 letterali

Metodi di supporto alla copertura per tabelle cicliche

In caso di tabelle cicliche è possibile applicare le tecniche viste per il caso di singola funzione con alcuni accorgimenti:

- Il **B&B** viene applicato considerando le singole funzioni separatamente (non viene imposto che un implicante copra mintermini di funzioni differenti).

  L'aumento della complessità è notevole a causa dell'aumento dei gradi di libertà. L'uso di un implicante per una funzione può essere applicato anche ad un'altra e quindi lo stesso implicante può comparire più volte nell'albero di copertura.
- **Petrik** viene applicato in parallelo a tutte le funzioni ed i risultati vengono messi in AND (i vincoli estratti da ogni funzione devono essere validi contemporaneamente). Scelta la copertura minima, si scelgono i termini che nelle funzioni singole rispettano tali vincoli.



Metodi di supporto alla copertura per tabelle cicliche - esempio

- Esempio

	f1					f2				
	A	B	C	D	E	F	G	H	I	L
P0	x	x					x	x		
P1		x	x		x			x	x	
P2			x	x		x			x	x
P3	x			x	x		x			x
P4	x		x		x	x		x		

$f1=(P_0+P_3+P_4)(P_0+P_1)(P_1+P_2+P_4)(P_2+P_3)(P_1+P_3+P_4)=1$

$f2=(P_2+P_4)(P_0+P_3)(P_0+P_1+P_4)(P_1+P_2)(P_2+P_3)=1$

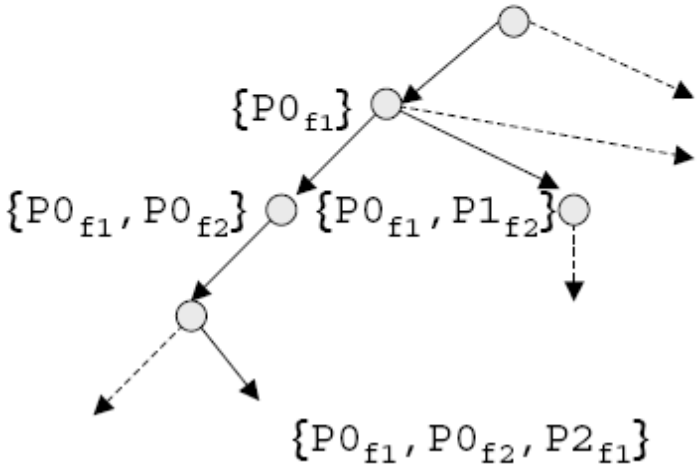
Con f1*f2=1

$f1=P_0P_1P_2+P_0P_2P_3+P_0P_2P_4+P_1P_2P_3+P_1P_2P_4+P_1P_3+P_0P_3P_4=1$

$f2=P_0P_2+P_1P_2P_3+P_2P_3P_4+P_0P_1P_3P_4+P_1P_3P_4=1$

Con f1*f2=1

$f1*f2=P_0P_1P_2+P_0P_2P_3+P_0P_2P_4+P_1P_2P_3+P_1P_3P_4=1$



$f1=P_0P_1P_2;\ f1=P_0P_2P_3;\ f1=P_0P_2P_4;\ f1=P_1P_3\ ;\ f1=P_1P_3$

$f2=P_0P_2\ ;\ f2=P_0P_2\ ;\ f2=P_0P_2\ ;\ f2=P_1P_2P_3;\ f2=P_1P_3P_4$

Minimizzazione euristica di reti a 2 livelli

Algoritmo ESPRESSO

Metodi esatti di minimizzazione a 2 livelli

- La minimizzazione esatta di reti a due livelli può rivelarsi impraticabile quando il numero di variabili è elevato.

- In molti casi, specie per problemi di elevate dimensioni, non è strettamente indispensabile trovare la soluzione ottima ma ci si accontenta di una soluzione sub-ottima, ottenuta mediante un processo iterativo.

- A partire da una soluzione iniziale attraverso iterazioni successive si costruisce una copertura della funzione “migliore” secondo un dato criterio di costo

Espresso

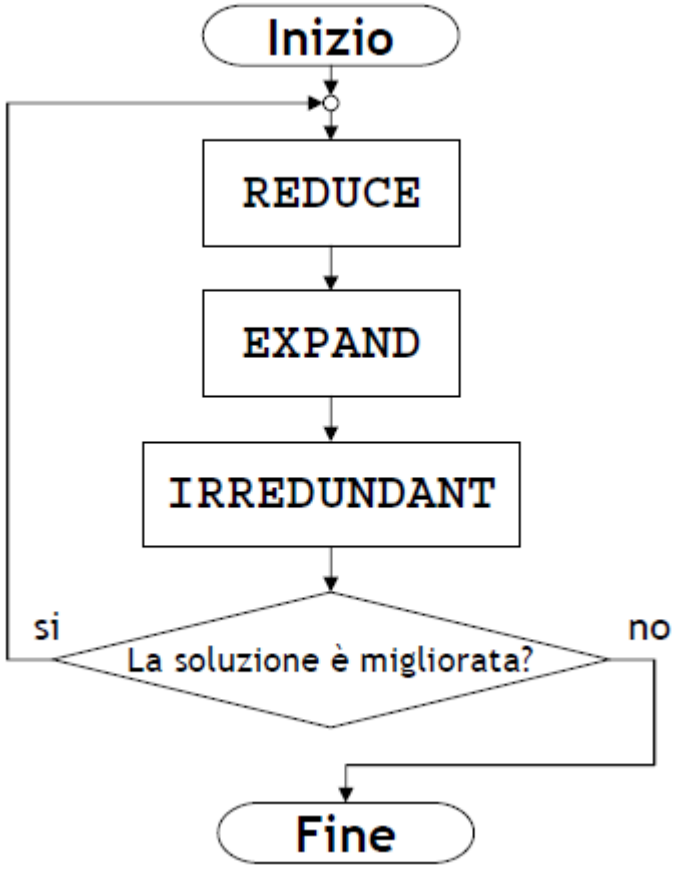
- Standard per la minimizzazione logica su 2 livelli
- Procedura di minimizzazione:
  - Ingresso: lista degli implicanti
  - Condizione iniziale: la lista degli implicanti rappresenta la copertura iniziale della funzione
  - Sviluppo: la copertura iniziale viene iterativamente manipolata da alcuni operatori
  - Termine: l'operazione si conclude quando nessun operatore migliora la copertura

Espresso: operatori

- Reduce: alcuni implicanti primi appartenenti alla copertura sono spezzati in implicanti non più primi in modo che la nuova copertura sia valida e abbia cardinalità uguale alla copertura di partenza
- Expand: alcuni implicanti non primi appartenenti alla copertura sono espansi fino a diventare primi: i nuovi implicanti devono essere validi, ovvero non devono coprire alcun elemento dell'off-set della funzione.
- Irredundant: la copertura ottenuta al passo precedente è prima per costruzione, cioè costituita da soli implicanti primi. Questa trasformazione ha lo scopo di eliminare gli implicanti completamente ridondanti, riducendo potenzialmente la cardinalità della copertura.

Algoritmo ESPRESSO

Il processo di trasformazione viene ripetuto fino a che si ha un miglioramento del costo della rete; su questa idea di base sono stati sviluppati diversi algoritmi di cui il più noto è **Espresso**.



Espansione

- Lo scopo di questa trasformazione è quello di produrre una copertura prima procedendo con l'applicazione della proprietà di assorbimento dell'algebra di Boole.

- Gli implicanti della copertura sono rielaborati uno alla volta. Ogni implicante è espanso finché non risulta primo, cioè finché non si ottiene un nuovo implicante non valido. Gli implicanti non primi coperti vengono eliminati. La copertura deve avere cardinalità invariata.

- Un implicante è *non valido* se copre qualche zero della funzione, ossia se l'intersezione tra l'insieme dei mintermini coperti dal nuovo implicante P e l'OFF-Set della funzione non è vuoto: se **Pf' = 0** allora P è valido.

Espansione - scelta degli implicanti da espandere (1/2)

Una volta terminata la fase di espansione, la trasformazione prevede l'eliminazione degli implicanti non primi completamente coperti.

La qualità della soluzione ottenuta è influenzata da due aspetti:

1. Con che ordine conviene scegliere i letterali rispetto a cui effettuare l'espansione?

   - spesso si segue semplicemente un ordine prefissato, si eseguono tutte le espansioni possibili e si seleziona a posteriori la soluzione più soddisfacente

2. Con che ordine conviene scegliere gli implicanti da espandere?

   - questo problema richiede l'uso di euristiche di scelta che tendano ad individuare gli implicanti che più probabilmente possono essere espansi senza essere coperti da altri implicanti

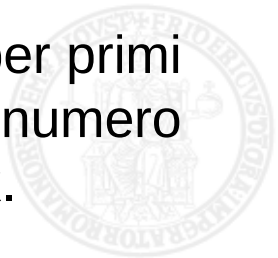
Espansione - scelta degli implicanti da espandere (2/2)

Una tecnica adottata per la scelta dell'ordine con cui espandere gli implicanti si basa sulla codifica *positional-cube* che codifica ciascun letterale di un implicante su due bit:

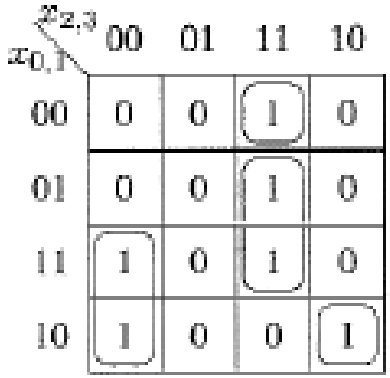
Codifica	Significato
01	1
10	0
11	–

Es.l'implicante x0 x2'x3' della funzione di 4 variabili x0 … x3 è codificato come : 01 11 10 10

- Utilizzando questa codifica si costruisce una tabella in cui sono riportati tutti gli implicanti e si sommano algebricamente i bit dei simboli codificati, colonna per colonna.
- Il peso degli implicanti è calcolato come prodotto scalare tra la matrice delle codifiche ed il vettore dei pesi totali delle colonne.
- Una volta determinati tutti i pesi degli implicanti si scelgono per primi quelli con peso minore e, in caso di ugual peso, quelli con un numero maggiore di letterali. L'ordine di scelta così stabilito è detto **rank**.



Espansione - scelta degli implicanti da espandere - esempio



Implicante	x_0	x_1	x_2	x_3
$x_0x_2'x_3'$	0 1	1 1	1 0	1 0
$x_0'x_1'x_2x_3$	1 0	1 0	0 1	0 1
$x_1x_2x_3$	1 1	0 1	0 1	0 1
$x_0x_1'x_2x_3'$	0 1	1 0	0 1	1 0
Totali:	2 3	3 2	1 3	2 2

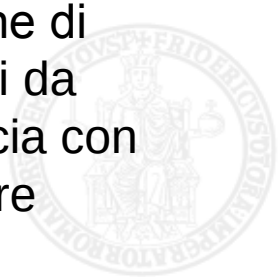
Matrice delle codifiche

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2 & 3 & 3 & 2 & 1 & 3 & 2 & 2 \end{bmatrix}^T = \begin{bmatrix} 11 \\ 10 \\ 12 \\ 11 \end{bmatrix}$$

Il prodotto scalare fra la matrice e il vettore dei pesi delle colonne fornisce il peso di ciascun implicante

Implicante	x_0	x_1	x_2	x_3	Peso	Rank
$x_0x_2'x_3'$	0 1	1 1	1 0	1 0	11	3
$x_0'x_1'x_2x_3$	1 0	1 0	0 1	0 1	10	1
$x_1x_2x_3$	1 1	0 1	0 1	0 1	12	4
$x_0x_1'x_2x_3'$	0 1	1 0	0 1	1 0	11	2
Totali:	2 3	3 2	1 3	2 2		

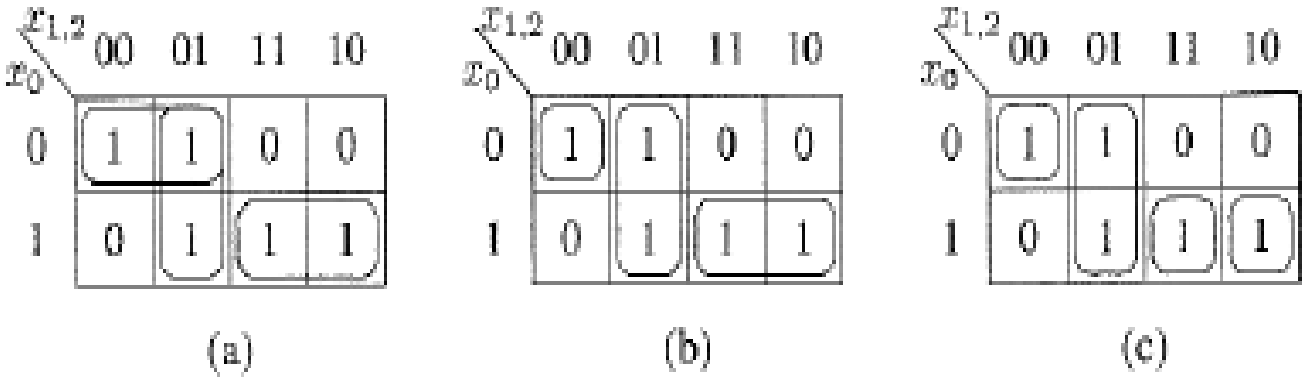
Il rank fornisce l'ordine di scelta degli implicanti da espandere: si comincia con quelli con peso minore



Riduzione (1/2)

La riduzione opera nella direzione opposta dell'espansione trasformando una copertura prima in una nuova copertura non prima con uguale cardinalità.

La riduzione di un implicante comporta l'aggiunta di un nuovo letterale in forma naturale o negata. La condizione essenziale affinchè una riduzione sia valida è che non modifichi la cardinalità della copertura: ciò è possibile solo se l'implicante selezionato per la riduzione è parzialmente sovrapposto ad un altro implicante della copertura (in questo modo viene assorbito).



La figura c corrisponde ad una riduzione non accettabile poiché ha cardinalità maggiore di quella iniziale

Riduzione (2/2)

Come per l'espansione anche in questo caso si pone il problema della scelta degli implicanti da ridurre e dei letterali rispetto a cui ridurre.

Si noti che mentre l'ordine di espansione influisce sull'efficienza computazionale dell'algoritmo, l'ordine di riduzione influisce sulla copertura non prima prodotta e quindi sulla qualità della soluzione finale.

- Esistono diverse euristiche per la scelta dell'ordine di riduzione, tra cui una basata ancora sulla codifica *positional-cube*, che si volge allo stesso modo visto per l'espansione, con la differenza che stavolta sono gli implicanti con peso maggiore ad essere i migliori candidati.

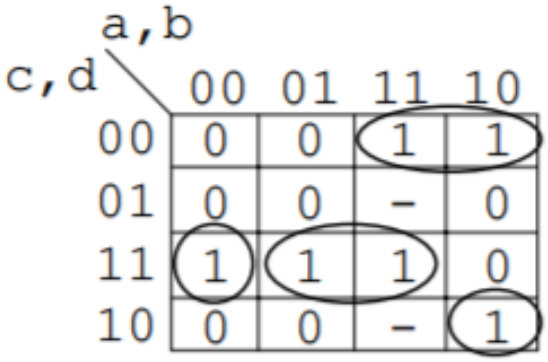
Copertura non ridondante

L’ultima operazione svolta all’interno di ogni ciclo, indicata come ***irredundant()*** consiste nel determinare una copertura non ridondante a partire dall’insieme degli implicanti primi.

Da tale insieme vengono rimossi quelli completamente ridondanti, cioè quelli che possono essere rimossi senza lasciare mintermini non coperti.

Esempio (step 1)

Condizione iniziale:
deriva direttamente dalle specifiche del problema

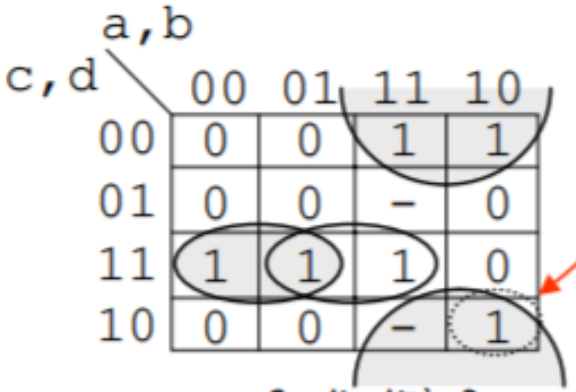


Cardinalità: 4

Costo in letterali: 14

Espansione

Mintermine eliminato poiché coperto durante l'espansione



Cardinalità: 3

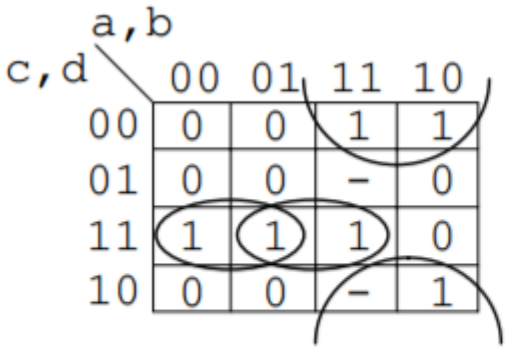
Costo in letterali: 8

Irredundant non ha effetto perché non ci sono implicanti ridondanti

Reduce non ha effetto perché modificherebbe la cardinalità

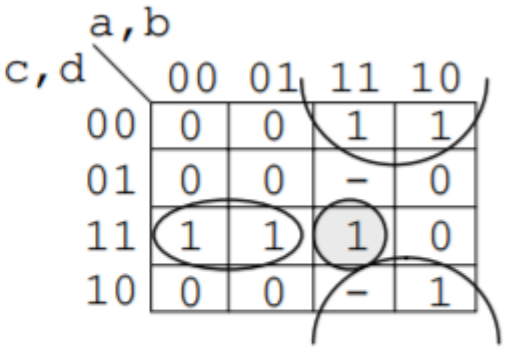
Il costo è migliorato rispetto alla condizione iniziale quindi si procede..

Esempio (step 2)



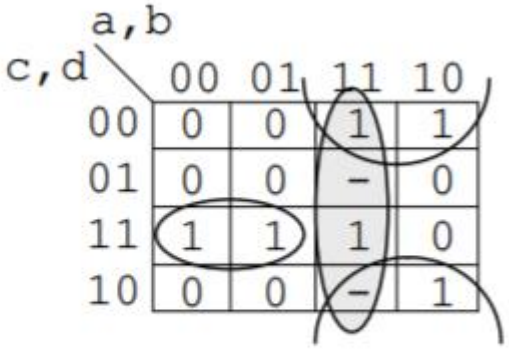
Cardinalità: 3
Costo: 8
Nota: copertura prima

Riduzione



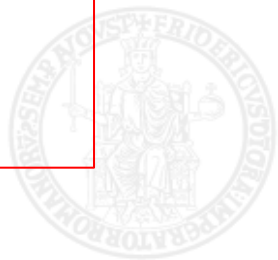
Cardinalità: 3
Costo: 9
Nota: copertura non prima

Espansione

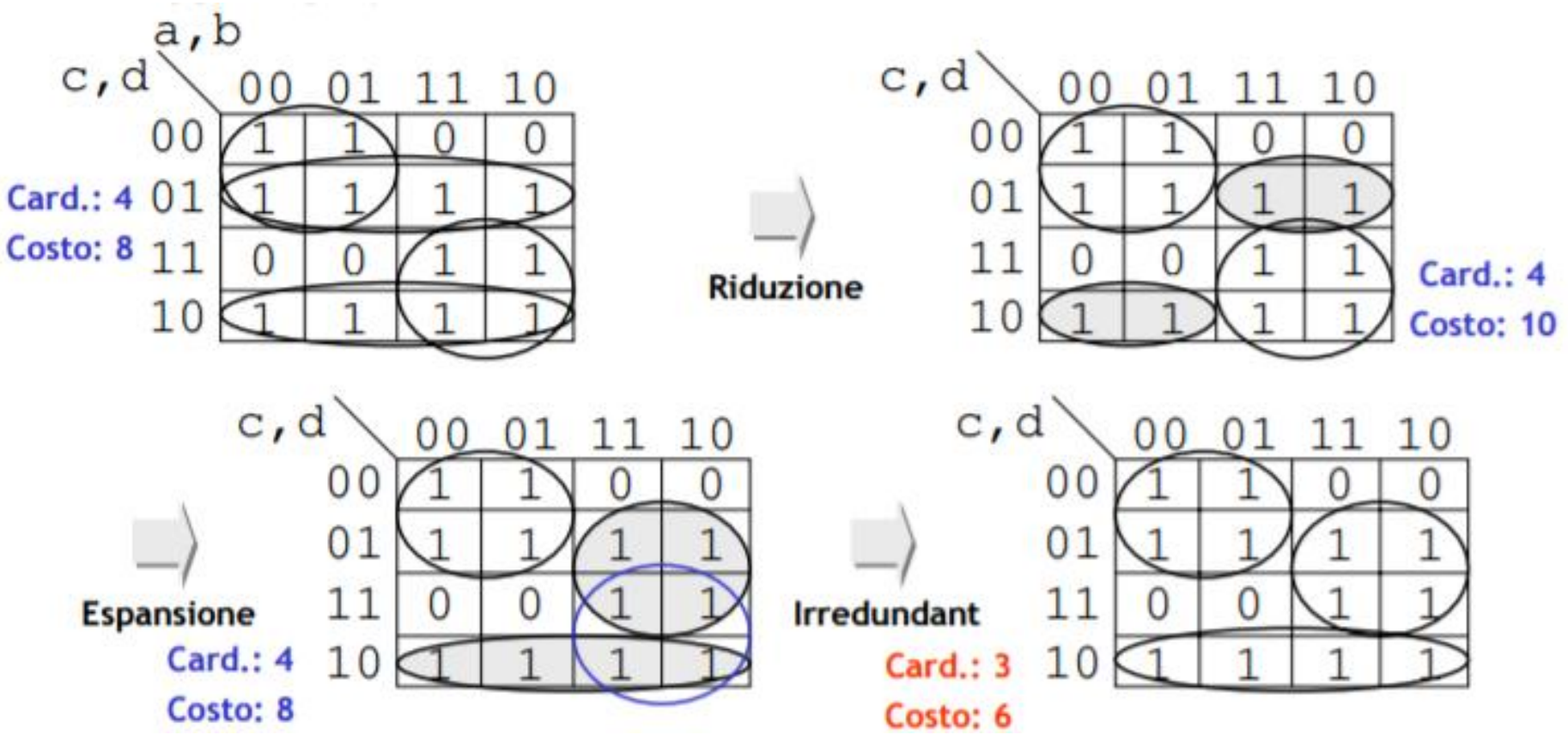


Cardinalità: 3
Costo: 7
Nota: copertura prima

La procedura termina perché una nuova applicazione della reduce porterebbe ad una soluzione precedente individuata



Esempio 2


a,b
c,d
00 01 11 10
00 1 1 0 0
01 1 1 1 1
11 0 0 1 1
10 1 1 1 1
Card.: 4
Costo: 8
Riduzione
c,d
00 01 11 10
00 1 1 0 0
01 1 1 1 1
11 0 0 1 1
10 1 1 1 1
Card.: 4
Costo: 10
Espansione
Card.: 4
Costo: 8
c,d
00 01 11 10
00 1 1 0 0
01 1 1 1 1
11 0 0 1 1
10 1 1 1 1
Irredundant
Card.: 3
Costo: 6
c,d
00 01 11 10
00 1 1 0 0
01 1 1 1 1
11 0 0 1 1
10 1 1 1 1

Minimizzazione euristica di reti su più livelli

Modello di una rete multilivello

Per la sintesi e ottimizzazione di reti su più di due livelli si adotta il seguente *modello di riferimento*:

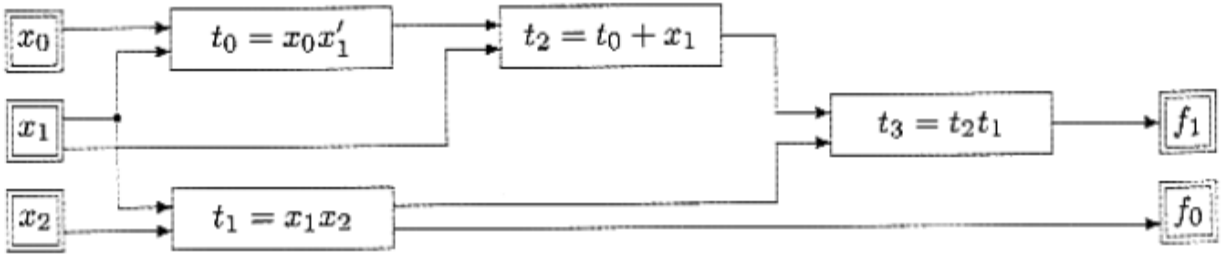
Una rete multilivello è vista come un grafo aciclico orientato (DAG) G(V,E) in cui V è l'insieme dei nodi e E l'insieme degli archi.

L'insieme V è partizionato in 3 insiemi:

***Nodi di ingresso* (V_I).** Rappresentano le variabili indipendenti della rete $x_0,\ldots x_{n_I}$

***Nodi di uscita* (Vo).** Rappresentano le variabili di uscita ossia le funioni $f_0,\ldots,f_{n_O}$

***Nodi interni* (V_G).** Rappresentano le variabili temporanee interne $t_0,\ldots,t_{n_G}$ alla rete e sono annotati come espressioni booleane su due livelli



$$t_0 = x_0 x_1' \qquad t_2 = t_0 + x_1$$
$$t_1 = x_1 x_2 \qquad t_3 = t_2 t_1$$
$$f_0 = t_1$$
$$f_1 = t_3$$

Trasformazioni

Il procedimento di ottimizzazione di una rete multilivello si basa sull'applicazione iterativa di alcune trasformazioni ben definite ad una rete descritta secondo il modello appena mostrato.

Non è possibile definire a priori un ordinamento delle trasformazioni che garantisca il risultato ottimo, per cui si ricorre ad euristiche basate su meccanismi di scelta casuali e criteri quantitativi semplici per la valutazione della qualità della soluzione trovata.

Una trasformazione può essere:

Locale: modifica solo le espressioni algebriche nei vari nodi e lascia inalterata la struttura topologica della rete

Globale: modifica sia le espressioni algebriche nei nodi che la struttura topologica della rete

Trasformazioni

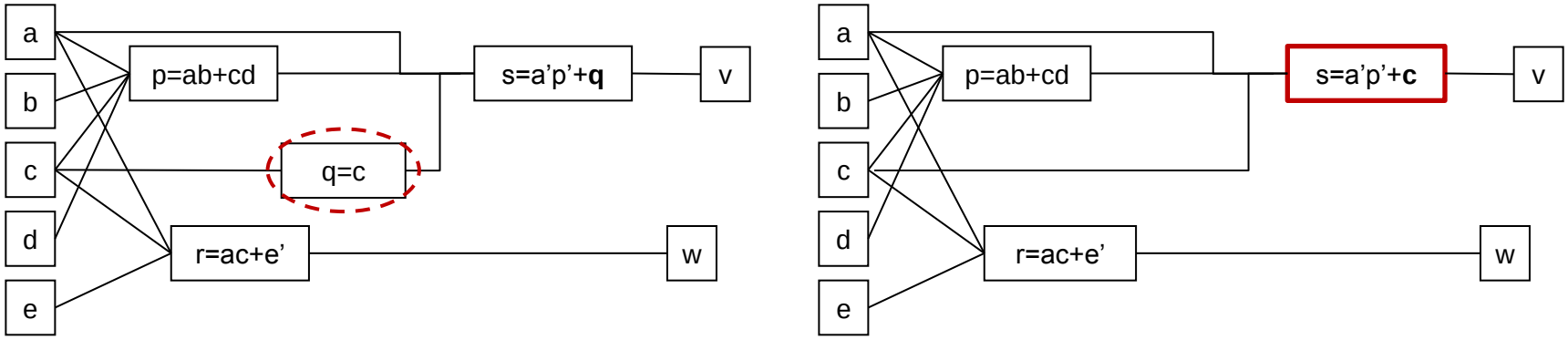
Alcune comuni trasformazioni:

- Sweep
- Eliminate
- Simplify
- Factor
- Substitute
- Extract
- Decompose

Trasformazioni - sweep

Durante il processo di applicazione delle trasformazioni ad una rete può accadere di ottenere nodi interni in cui l'espressione è semplicemente una variabile.

La trasformazione nota come *sweep* sostituisce la variabile assegnata nel nodo a monte in tutti nodi a valle (tutti i nodi connessi agli archi di uscita).

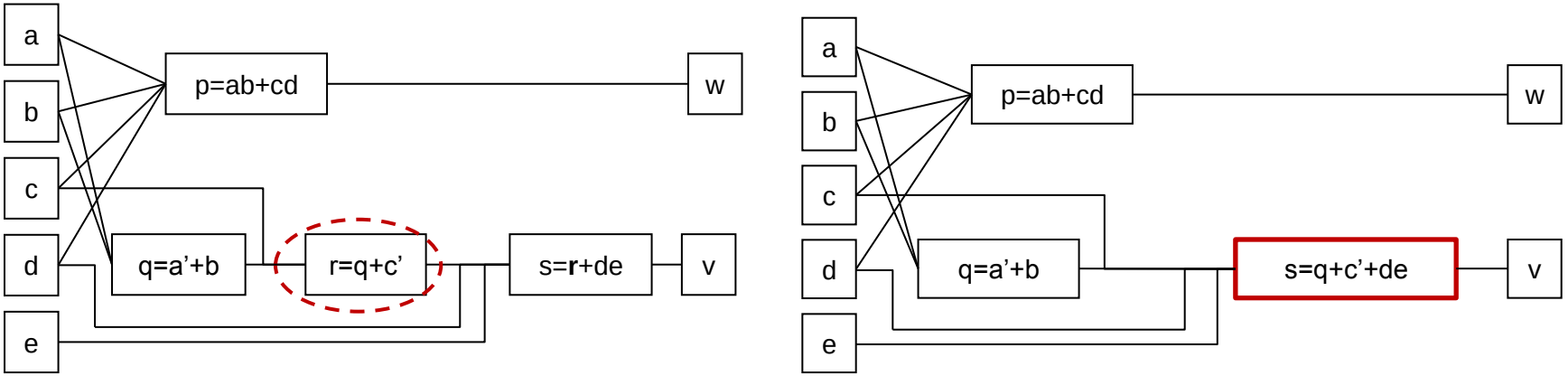


Questa trasformazione ha lo scopo di ridurre la cardinalità dell'insieme VG senza modificare le espressioni dei nodi interni.

Si noti che sweep è una trasformazione globale in quanto altera la struttura della rete.

Trasformazioni - eliminate

Tale trasformazione ha lo scopo di eliminare un nodo interno sostituendo la sua espressione in tutte le sue occorrenze nella rete logica a valle.



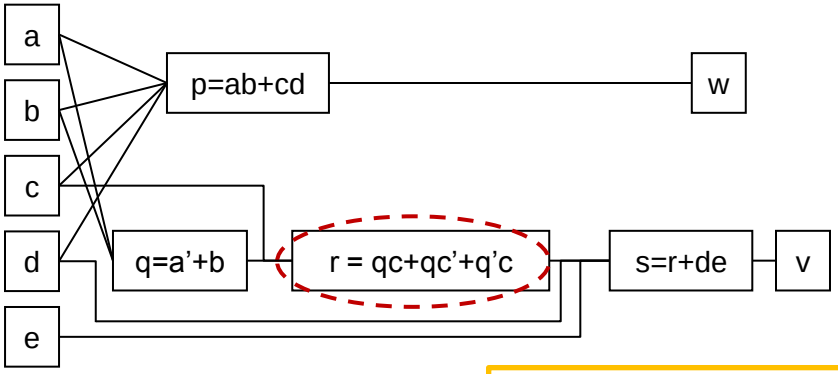
L'eliminazione di un nodo comporta in genere un aumento del numero di letterali della rete in quanto la stessa espressione è replicata più volte nei nodi a valle;

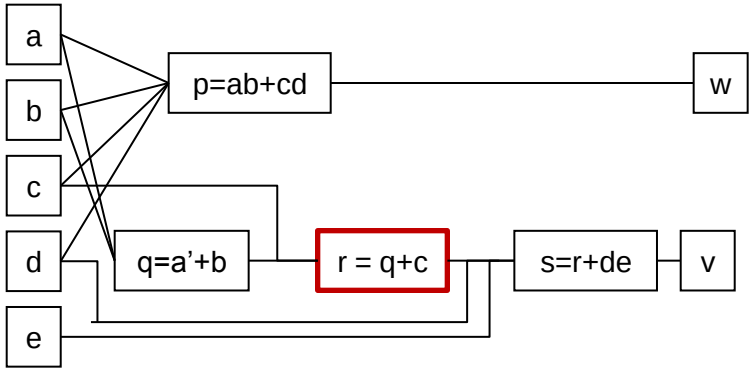
Normalmente tale trasformazione viene effettivamente eseguita solo se l'aumento di letterali Δl è al di sotto di una soglia.

La trasformazione è globale in quanto modifica la struttura della rete.

Trasformazioni - simplify

Tale trasformazione riduce la complessità della funzione rappresentativa di un nodo sfruttando le proprietà della sua rappresentazione. Se la funzione è rappresentata nella forma a due livelli allora è possibile utilizzare metodi esatti come McCluskey oppure metodi euristici.



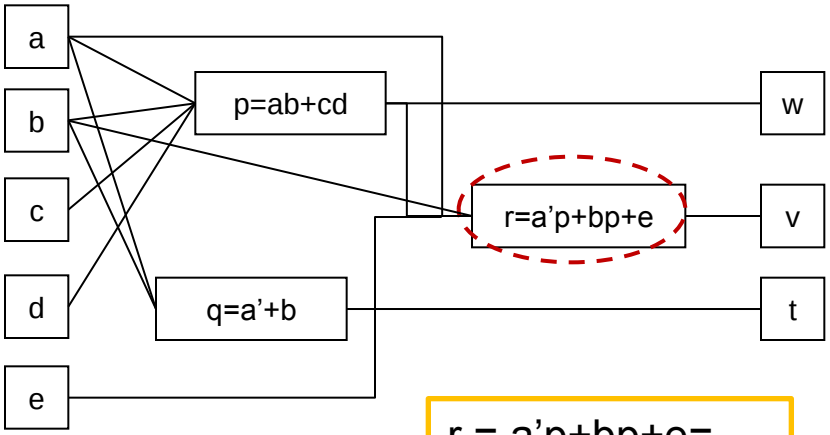


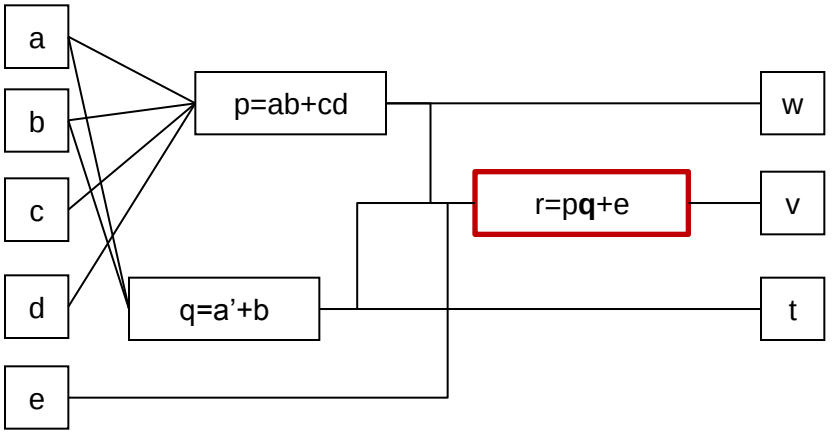
r = qc+qc'+q'c=
= qc+qc'+q'c+qc=
= q(c+c')+c(q+q')=
= q+c

Se la trasformazione lascia inalterato il supporto della funzione a cui viene applicata è locale, altrimenti è globale.

Trasformazioni - substitute

Tale trasformazione utilizza un nodo già presente nella rete per semplificare l'espressione di un altro nodo: ciò implica una modifica della topologia della rete e pertanto substitute è una trasformazione globale.





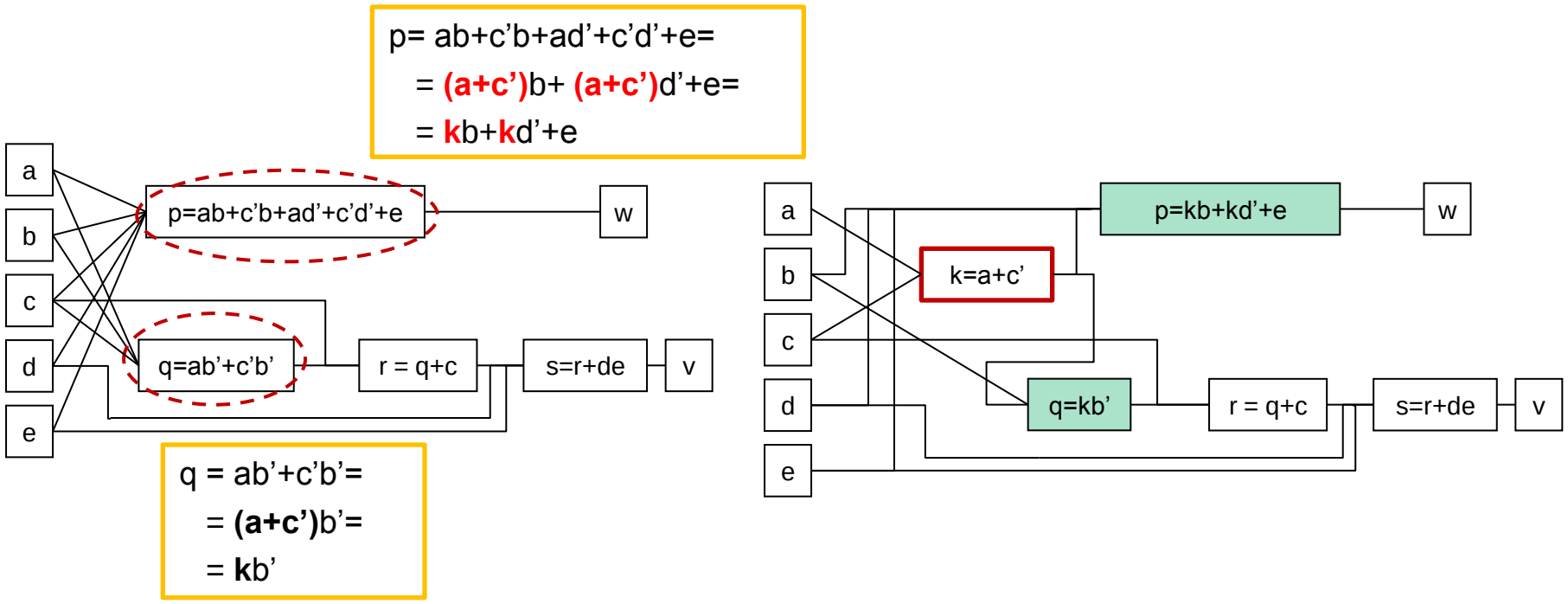
r = a'p+bp+e=
= p**(a'+b)**+e=
= p**q**+e

La trasformazione richiede la creazione di una dipendenza da un vertice pre-esistente non appartenente al supporto della funzione; essa è un'applicazione diretta della divisione algebrica.

Trasformazioni - extract

L'estrazione di un nodo è simile all'operazione svolta dalla substitute ma più generale in quanto rimuove il vincolo che richiede che il nodo da estrarre sia già presente nella rete.

Una sotto-espressione comune a due o più funzioni (vertici) può essere estratta introducendo un nuovo vertice associato alla sotto-espressione stessa (si sfrutta il fan-out del nuovo nodo).



La trasformazione è globale poichè modifica la topologia della rete.

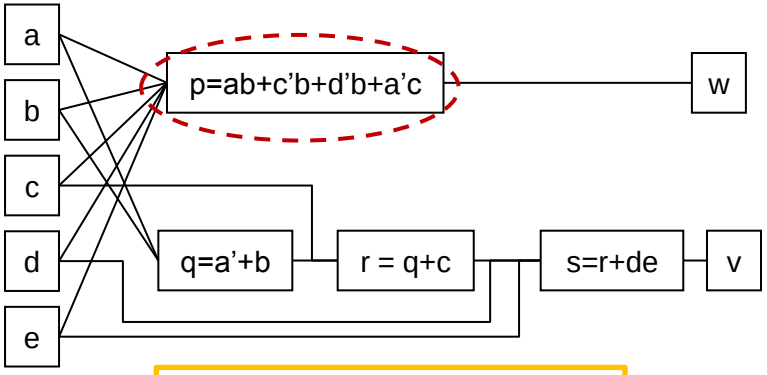
Trasformazioni - decompose

Sostituisce un nodo con due o più nodi che formano una rete equivalente. Per esempio, si può usare il teorema di Shannon (per l'elettronica digitale) per estrarre da un nodo 2^k nuovi nodi, dove k è il numero di variabili usate per l'espansione:

Data una funzione booleana f di n variabili $x_1,..,x_n$, vale l'uguaglianza:

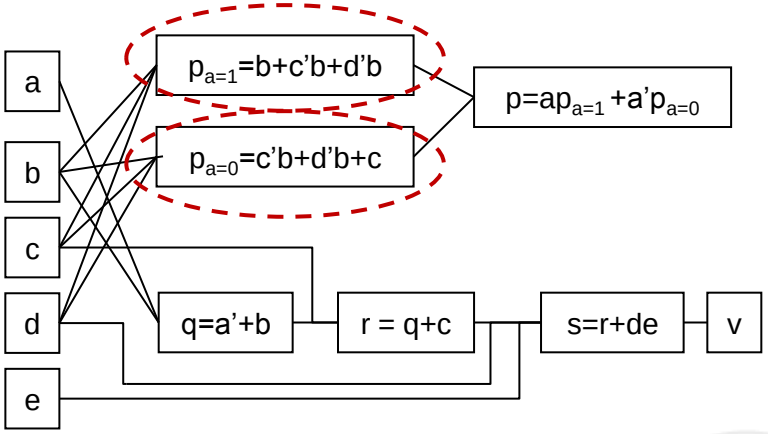
$f(x_0,\ldots,x_n) = x_0 * \underbrace{f(1,x_1,\ldots,x_n)}_{f_1} + x_0' * \underbrace{f(0,x_1,\ldots,x_n)}_{f_2}$ per k=1

È possibile inserire i nodi corrispondenti a f1 e f2 a monte di quello f, la cui espressione verrà modificata di conseguenza



p=ab+c'b+d'b+a'c=
=a(b+c'b+d'b)+a'(c'b+d'b+c)

$p_{a=1}$ $p_{a=0}$



Dopo la decomposizione potrebbe emergere che alcune delle funzioni estratte sono già presenti in altri nodi della rete: in tal caso il beneficio della trasformazione è ancora maggiore.

Trasformazioni - factor

Tale trasformazione effettua la fattorizzazione di un'espressione su due livelli, producendo come risultato un'espressione su più livelli. Consiste nel trovare una forma contenente fattori comuni (letterali, prodotti, somme) per la funzione in esame (può essere funzionale all'applicazione delle trasformazioni substitute ed extract).

- La fattorizzazione si realizza applicando le proprietà distributiva e commutativa oltre alle proprietà degli operatori;
- Su una forma fattorizzata si possono applicare ulteriori proprietà (De Morgan) per giungere a soluzioni di costo inferiore in termini di porte logiche grazie alla condivisione di porte

Il problema della fattorizzazione è generalmente affrontato sulla base di metodi euristici volti ad individuare quali variabili raccogliere e in che ordine: in generale conviene raccogliere i letterali che compaiono con maggior frequenza negli implicanti della funzione.

La trasformazione è applicata all'espressione all'interno di un nodo e pertanto non modifica la topologia della rete: è quindi locale.

Trasformazioni – factor – Esempio

Si consideri la funzione **f=abd'+a'bd+ab'd'+a'cd+b'cd'**

I letterali **a'** e **d**' compaiono con maggior frequenza negli implicanti e pertanto sono buoni candidati per la fattorizzazione. Scegliendo **d'** si ottiene:

f=d'(ab+ab'+b'c)+a'bd+a'cd =>

f_1= ab+ab'+b'c

f_2= a'bd+a'cd

Si proceda in modo analogo per f_1 e f_2 . Se si sceglie di raccogliere **b'** in f1 e **a'** in f2 , si ottiene:

f_1=ab+b'(a+c) => f_3=*a+c*

f_2=a'(bd+cd) => f_4=*bd+cd*

Applicando ancora lo stesso metodo si raccoglie **d** in f_4 ottenendo f_4 =d(b+c)

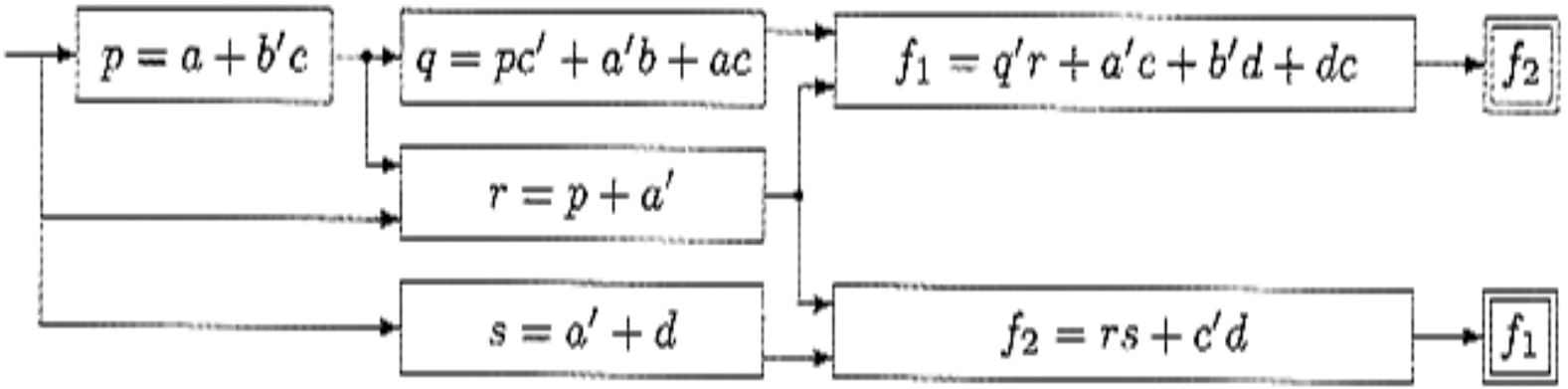
Ripercorrendo a ritroso il procedimento si ha:

F = d'f1+f2=

= d'(ab+b'f3)+a'f4 =

= ***d'(ab+b'(a+c))+a'(d(b+c))***

ApplicazioneTrasformazioni – Esempio(1/11)



Si vuole applicare alla rete in figura la sequenza di trasformazioni:

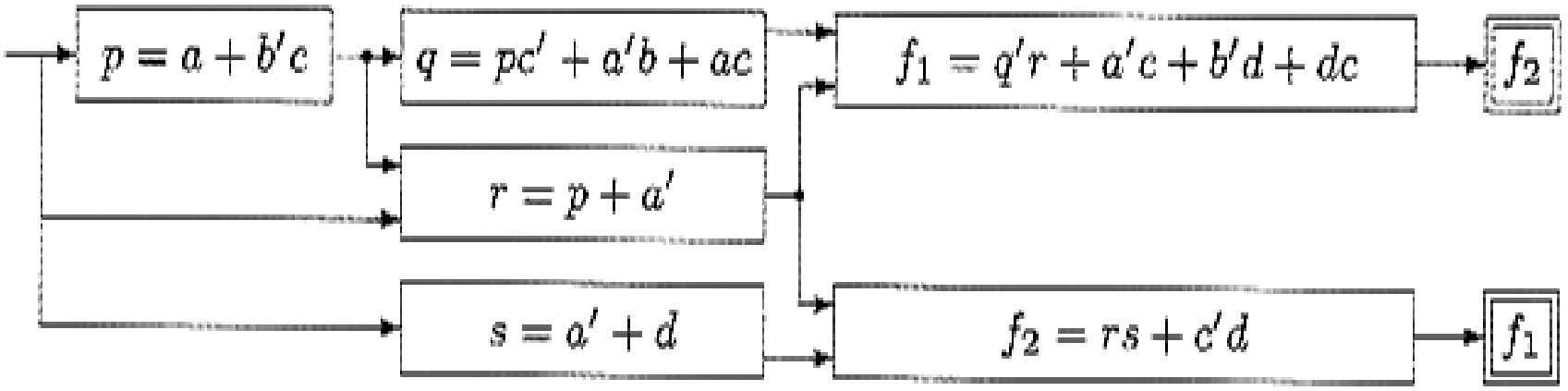
(i) *eliminate(p); simplify(q,r); cost()*

(ii) *eliminate(q,r); simplify(f_1, f_2); cost()*

(iii) *factor(f_1); substitute(f_1); cost()*

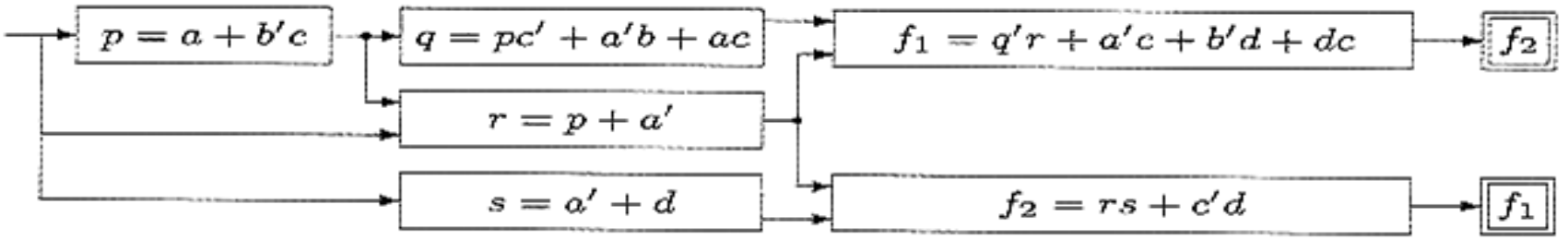
Gli argomenti delle funzioni indicano i nodi cui applicare le trasformazioni e la funzione *cost()* ritorna il costo totale in numero di letterali.

ApplicazioneTrasformazioni – Esempio(2/11)



Costo Iniziale della rete = 25 letterali

ApplicazioneTrasformazioni – Esempio(3/11)



(i) *eliminate(p); simplify(q,r); cost()*

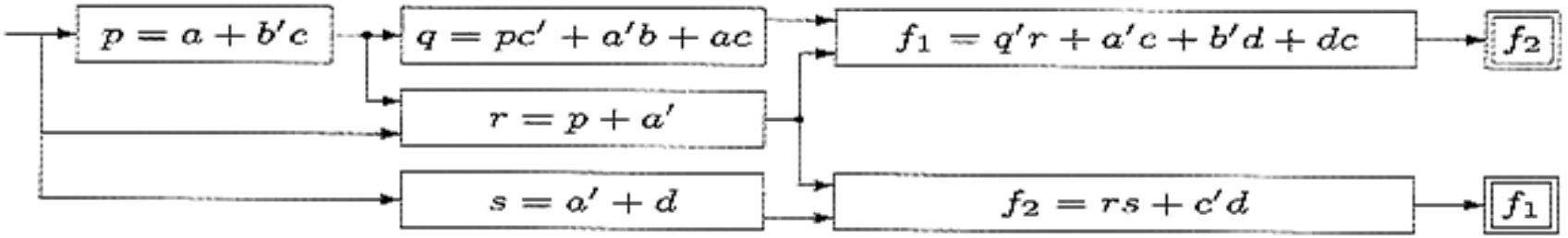
Eliminate(p) richiede la sostituzione dell'espressione del nodo ***p*** nei nodi ***q*** ed ***r*** e la conseguente eliminazione di ***p***.

Si ottengono le espressioni:

q=pc'+a'b+ac=(a+b'c)c'+a'b+ac

r=p+a'=(a+b'c)+a'

ApplicazioneTrasformazioni – Esempio(4/11)



(i) *eliminate(p); simplify(q,r); cost()*

Applicando ***Simplify(q,r)*** alle espressioni appena trovate si ottiene:

q=(a+b'c)c'+a'b+ac = ac'+a'b+ac=a+a'b=>a+b

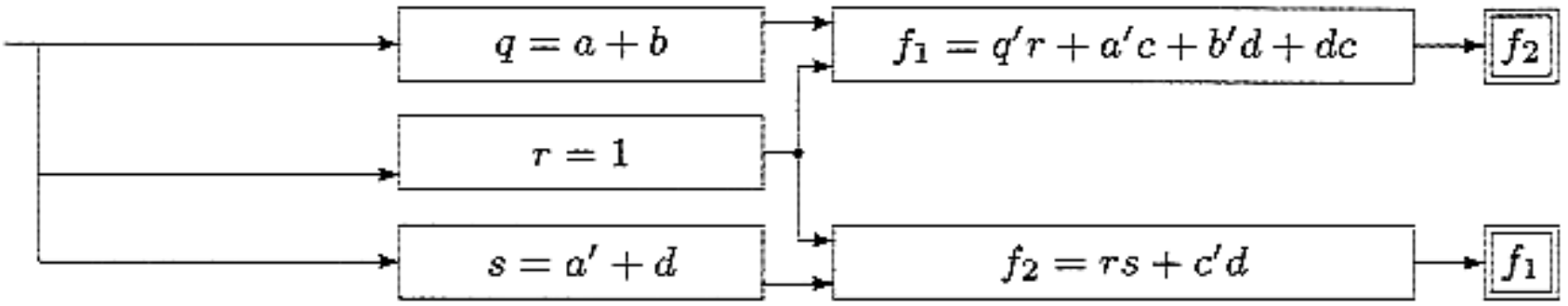
r=(a+b'c)+a' = (a+a')+b'c=1+b'c=>1

ApplicazioneTrasformazioni – Esempio(5/11)

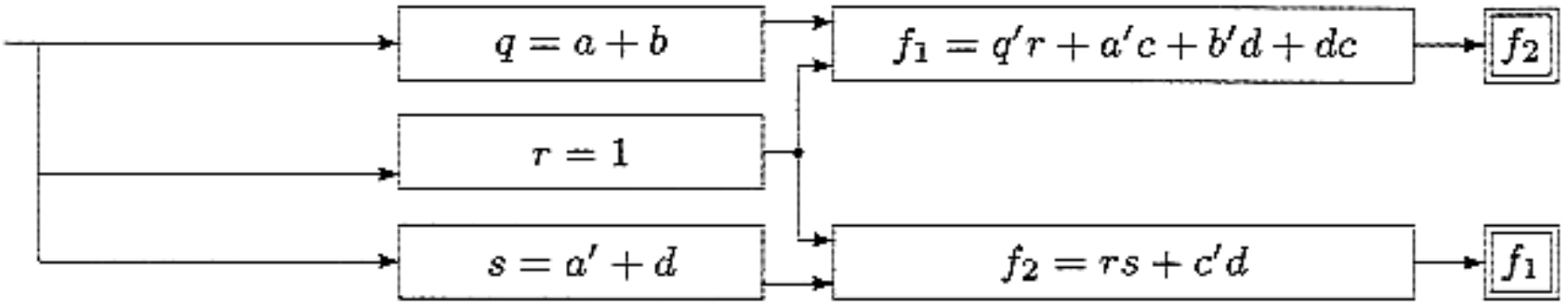
Al termine del passo:

(i) *eliminate(p); simplify(q,r); cost()*

Si ottiene la rete multilivello di costo 16 letterali:



ApplicazioneTrasformazioni – Esempio(6/11)



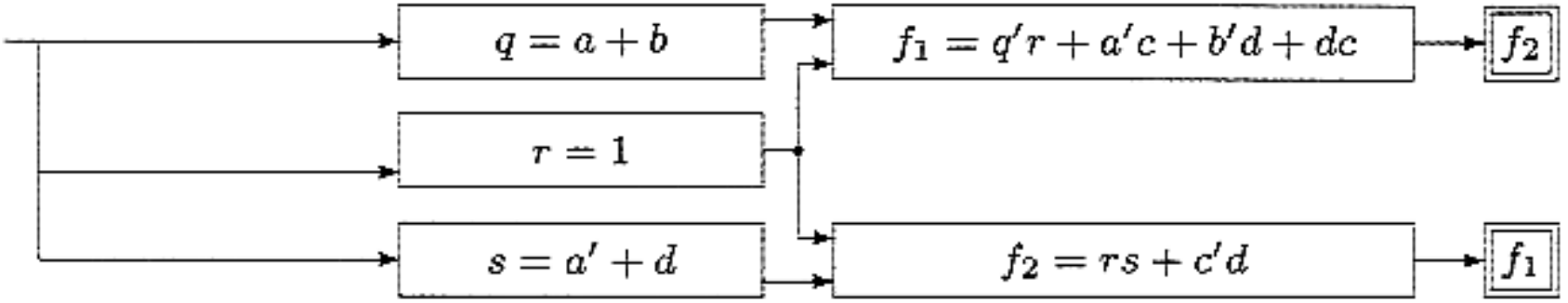
(ii) *eliminate(q,r); simplify(f_1, f_2); cost()*

Eliminate(q,r) elimina i nodi ***q*** e ***r***, modificando le espressioni di f_1 e f_2 nel modo seguente:

$$f_1 = q'r + a'c + b'd + dc = (a + b)'(1) + a'c + b'd + dc$$

$$f_2 = rs + c'd = (1)s + c'd$$

ApplicazioneTrasformazioni – Esempio(7/11)



(ii) *eliminate(q,r); simplify(f_1, f_2); cost()*

Simplify(f1,f2) riporta i nodi f_1 e f_2 in somma di prodotti:

$$f_1 = (a+b)'(1) + a'c + b'd + dc = a'b' + a'c + b'd + dc$$

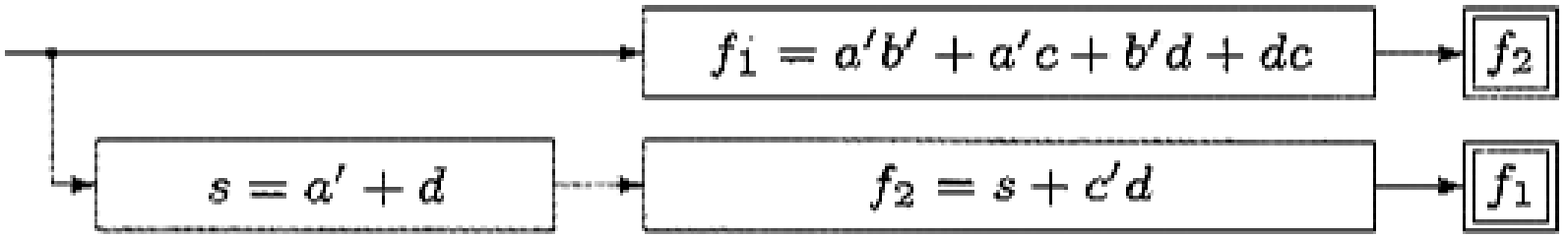
$$f_2 = (1)s + c'd = s + c'd$$

ApplicazioneTrasformazioni – Esempio(8/11)

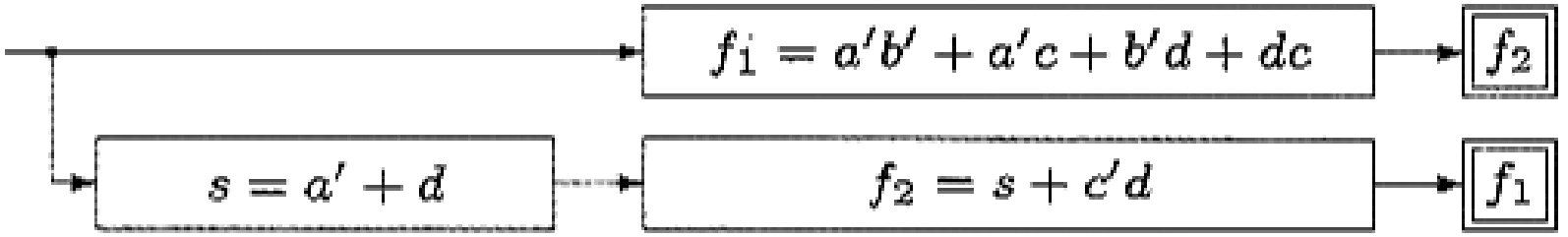
Al termine del passo:

(ii) $eliminate(q,r);\ simplify(f_1, f_2);\ cost()$

Si ottiene la rete multilivello di costo 13 letterali:



ApplicazioneTrasformazioni – Esempio(9/11)

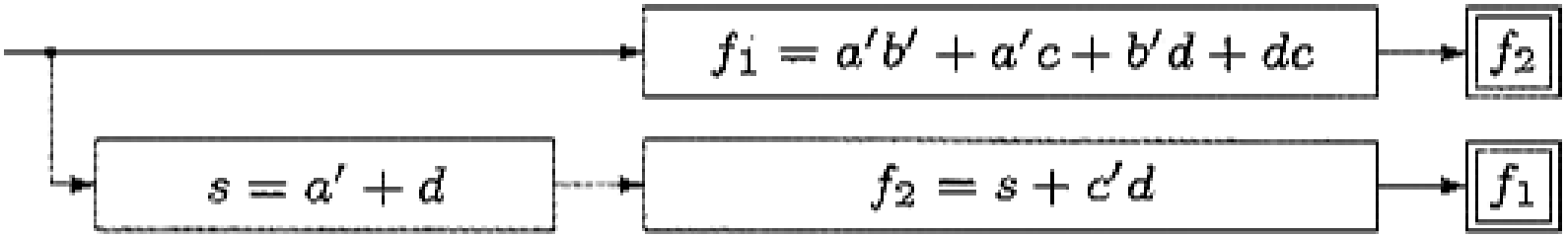


(iii) $factor(f_1);\ substitute(f_1);\ cost()$

L'applicazione di ***factor(f1)*** produce il raggruppamento a fattor comune di ***a'*** tra i primi due termini e ***d*** fra i secondi due

$$f_1 = a'b' + a'c + b'd + dc = a'(b' + c) + d(b' + c) = (a' + d)(b' + c)$$

ApplicazioneTrasformazioni – Esempio(10/11)



(iii) *factor*(f_1); *substitute*(f_1); *cost*()

$$f_1 = a'b' + a'c + b'd + dc = a'(b' + c) + d(b' + c) = (a' + d)(b' + c)$$

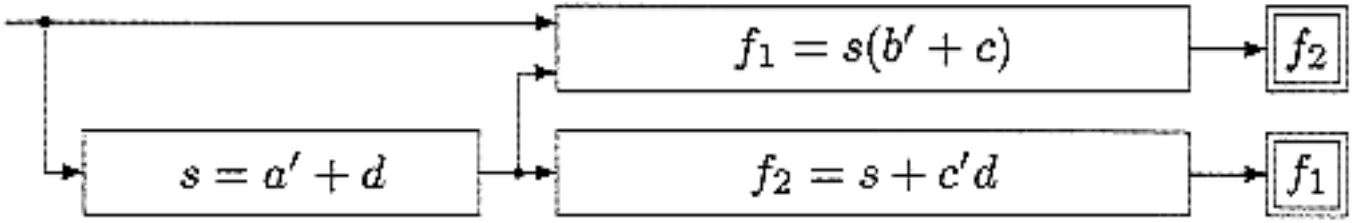
Applicando ***substitute(f1)*** si produce la sostituzione di una parte dell'espressione di f1 con un nodo già presente nella rete, s=(a'+d);

L'algoritmo di trasformazione individua la coincidenza tra il primo fattore di f1 con la funzione calcolata dal nodo s, per cui riscrive f1 come:

$$f_1 = (a' + d)(b' + c) = s(b' + c)$$

ApplicazioneTrasformazioni – Esempio(11/11)

Si ottiene la rete finale di costo pari a 8



ApplicazioneTrasformazioni – Esempio 2

Data la rete multilivello descritta dalle seguenti equazioni:

$$\begin{cases} p=ab \\ q=p+a \\ f=a'b'c+b'd+bd'+p \\ g=q+d'+ab \\ h=a'bc+bd \end{cases}$$

Si identifichino gli ingressi primari della rete, quindi si applichino le seguenti trasformazioni:

***1**)eliminate(p), **2**)simplify(q), **3**)sweep(), **4**)simplify(g), **5**)decompose(f,b), **6**)extract(h), **7**)substitute($f_{b'}$) e **8**)sweep()*

Rugged Script

Le trasformazioni viste sono alla base dei metodi euristici di minimizzazione delle funzioni su più livelli;

Non è possibile individuare alcun procedimento sistematico per determinare l'ordine in cui devono essere applicate per portare alla sintesi ottima ma è possibile individuare una "buona" sequenza, detta ***rugged script***, in grado di produrre risultati spesso vicini all'ottimo:

sweep; eliminate -1; simplify -m nocomp; eliminate -1
sweep; eliminate 5; simplify -m nocomp
resub -a; fx; resub -a
sweep; eliminate -1
sweep; full_simplify -m nocomp