

Sintesi Combinatoria

Sintesi di reti combinatorie a più livelli: Introduzione

Motivazioni e Introduzione

Modello per reti combinatorie a più livelli

Trasformazioni e Algoritmi

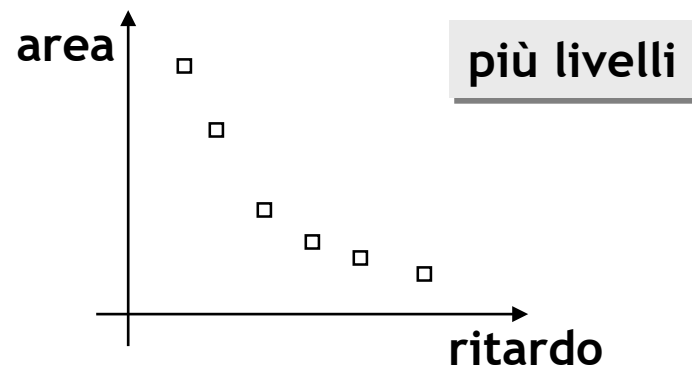
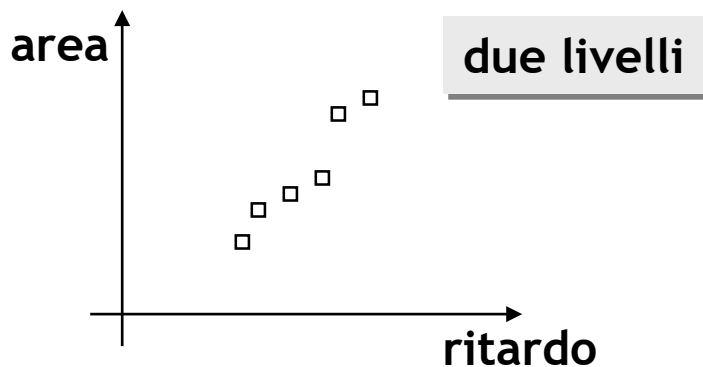
versione del 2/11/2020



Sintesi di reti combinatorie a più livelli:

Introduzione

- Obiettivo della sintesi logica: **ottimizzazione delle cifre di merito area e prestazioni**
 - Reti combinatorie a **due livelli**: area e ritardo sono ridotti contemporaneamente.
 - Reti combinatorie a **più livelli**: area e ritardo non procedono nella stessa direzione
- Le **reti a più livelli** portano in generale a **soluzioni più efficienti in termini di area/prestazioni** e consentono un utilizzo migliore delle librerie





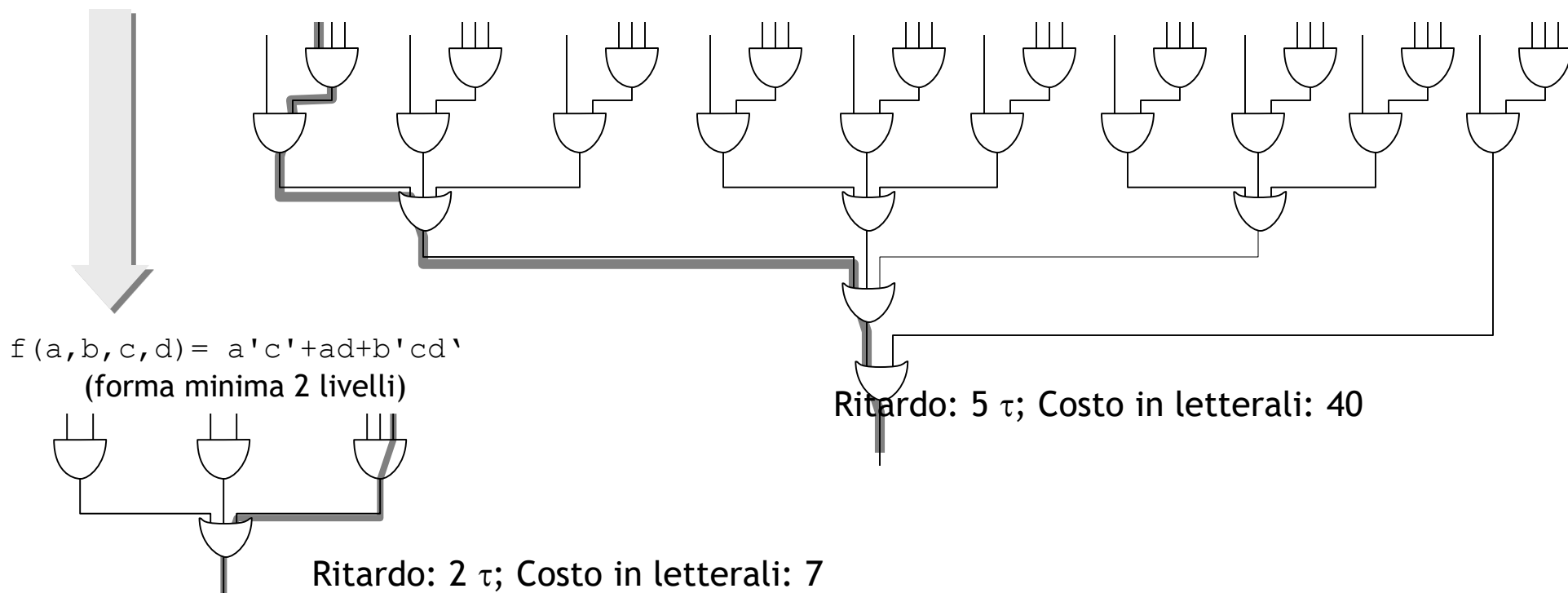
Sintesi di reti combinatorie a più livelli:

Introduzione

- **Esempio** – (Reti combinatorie a due livelli: Area e tempo sono ridotti contemporaneamente)

- Ipotesi: porte con un massimo di 3 ingressi (ritardo uniforme: τ)

$$f(a, b, c, d) = a'b'c'd' + a'b'c'd + a'b'cd' + a'bc'd' + a'bc'd + ab'c'd + ab'cd' + ab'cd + abc'd + abcd$$





Sintesi di reti combinatorie a più livelli:

Introduzione

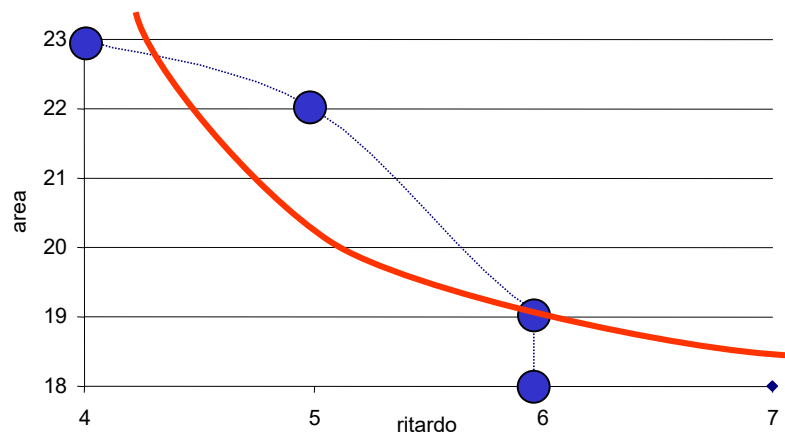
- **Esempio** – (Reti combinatorie a più livelli: trade-off area/prestazioni)
 - Ipotesi: porte con un massimo di 3 ingressi (ritardo uniforme: τ)

$f = 1' + c' * g * h' + a * b' * k' + g * k' + a' * b' * c' * d' * e' + a * d' * e' * f' + e' * g' * i' + e' * j' ;$ Ritardo: 4τ ; Costo: 23

$f = 1' + c' * g * h' + k' (a * b' + g) + a' * b' * c' * d' * e' + a * d' * e' * f' + e' * g' * i' + e' * j' ;$ Ritardo: 5τ ; Costo: 22

$f = 1' + c' * g * h' + k' (a * b' + g) + e' * (a' * b' * c' * d' + a * d' * f' + g' * i' + j') ;$ Ritardo: 6τ ; Costo: 19

$f = 1' + c' * g * h' + k' * (a * b' + g) + e' * (d' * (a' * b' * c' + a * f') + g' * i' + j') ;$ Ritardo: 6τ ; Costo: 18





Sintesi di reti combinatorie a più livelli:

Introduzione

- Nella realizzazione di **reti combinatorie multi-livello**, più che ricercare un ottimo (l'ottimo non è sempre definibile in maniera univoca), si cerca una **soluzione ragionevole in termini di area e prestazioni**.
 - Sarebbe più corretto parlare di **sintesi** invece che di ottimizzazione. La sintesi può prevedere:
 - Minimizzazione dell'area (con vincolo sul ritardo)
 - Minimizzazione del ritardo (con vincolo sull'area)
 - Le **operazioni e trasformazioni** definite per la sintesi multi-livello hanno come scopo base quello di manipolare l'espressione logica della rete combinatoria in modo da **individuare ed estrarre sotto-espressioni logiche comuni** nell'espressione di partenza
 - questo consente, in generale, di avere realizzazioni più efficienti (con riuso) in termini di porte utilizzate, rispetto all'ottimizzazione a due livelli, con tempi di propagazione peggiori
-



Sintesi di reti combinatorie a più livelli:

Introduzione

- ❑ Ottimizzazione a più livelli:
 - Vantaggi:
 - Più efficiente in termini di area e prestazioni.
 - Permette di utilizzare elementi di libreria.
 - Svantaggi:
 - Maggiore complessità della ottimizzazione.
- ❑ Metodi di ottimizzazione:
 - Esatti
 - Complessità computazionale estremamente elevata: inaccettabili.
 - **Euristici**
 - Definizione di euristica: *“procedimento non rigoroso (approssimativo, intuitivo) che permette di conseguire un risultato la cui qualità è paragonabile a quella ottenuta con metodi rigorosi”*



Sintesi di reti combinatorie a più livelli:

Introduzione

- Euristica del problema di ottimizzazione - due passi:
 - a) Si produce una soluzione ottimale ignorando i vincoli di realizzazione
 - fan_in, fan_out, elementi di libreria...

La soluzione è ottenuta tramite sequenze di **trasformazioni** applicate in **modo iterativo**. Le trasformazioni sono basate anche sulle **proprietà algebriche** delle espressioni booleane. La rete è definita **ottima** rispetto ad un insieme di trasformazioni, quando un'ulteriore applicazione di queste **non** può più **migliorare la funzione di costo**.
 - b) Si raffina il risultato considerando i vincoli strutturali
 - b) *library mapping* (o *library binding*).

Risultato dell'ottimizzazione è di inferiore qualità rispetto ad una ottimizzazione che considera contemporaneamente i punti a) e b) ma risulta computazionalmente più semplice.
 - In questa sezione si analizza solo il punto relativo **all'identificazione della soluzione ottimale** (punto a).
-



Sintesi di reti combinatorie a più livelli:

Modello della rete (1)

- Nella sintesi multilivello, il **modello** utilizzato per rappresentare un circuito combinatorio è un **grafo orientato aciclico**
 - DAG - Direct Acyclic Graph

- **Grafo per reti combinatorie**
 - È un grafo orientato $G(V,E)$ aciclico
 - V : insieme dei nodi
 - E : insieme degli archi

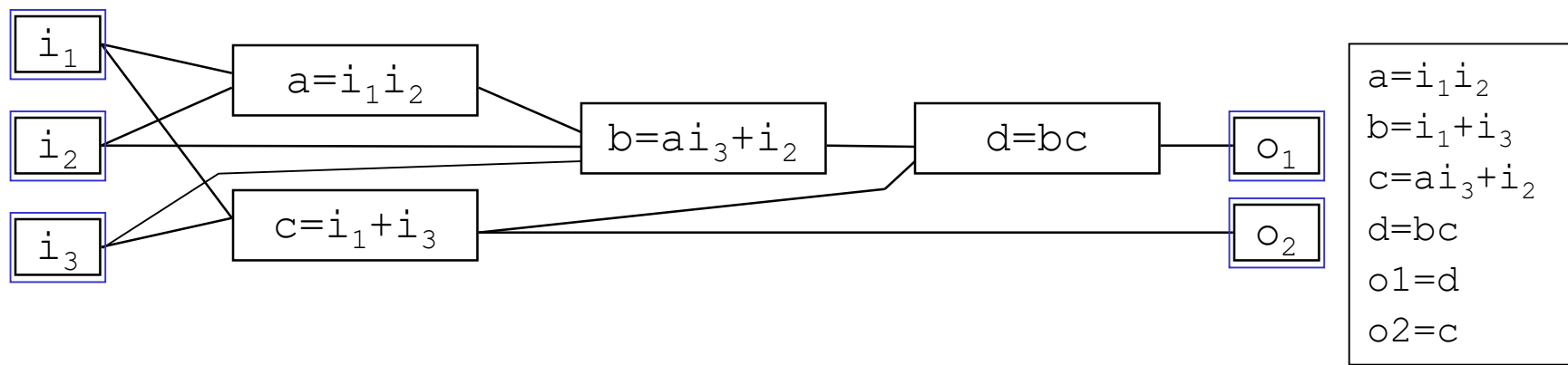
- V è partizionato negli insiemi:
 - nodi di ingresso V_I (Primary Inputs - PI)
 - nodi di uscita V_O (Primary Outputs - PO)
 - nodi interni V_G : Sono moduli della rete combinatoria a cui è associata una funzione combinatoria scalare (una sola uscita)



Sintesi di reti combinatorie a più livelli:

Modello della rete (2)

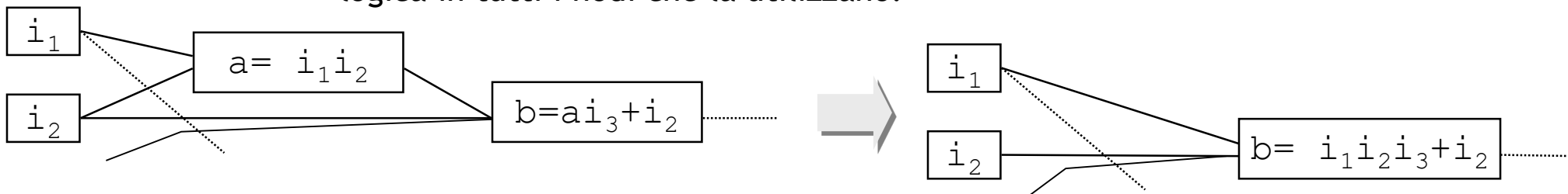
- E' un modello comportamentale/strutturale
 - **Strutturale**: connessioni
 - **Comportamentale**: ad ogni nodo è associata una funzione
 - Nel modello considerato, ogni **funzione è a due livelli** con una sola uscita
- Il modello è bipolare e non gerarchico
 - Bipolare: Ogni arco può assumere valore 0 o 1





Sintesi di reti combinatorie a più livelli: *Trasformazioni per reti logiche (1)*

- Metodi euristici
 - Realizzano un miglioramento iterativo della rete logica mediante **trasformazioni** logiche che **conservano il comportamento di I/O** del grafo
- Rispetto al grafo che rappresenta la rete combinatoria, sono possibili **due tipi di trasformazioni**:
 - **Locali**: modificano localmente (la funzione di) un nodo non toccando la struttura della rete.
 - Esempio: la fattorizzazione di un nodo
 - **Globali**: modificano anche la struttura della rete
 - Esempio: l'eliminazione di un nodo nella rete sostituendo la sua espressione logica in tutti i nodi che la utilizzano.





Sintesi di reti combinatorie a più livelli: *Trasformazioni per reti logiche (2)*

- Le trasformazioni logiche modificano sia l'**area** sia le **prestazioni** poiché agiscono:
 - Sulle funzioni locali;
 - sul numero dei letterali (**area**);
 - Sulle connessioni
 - variazione del n° di nodi (**area**) e del n° nodi del cammino critico (**prestazioni**: n° nodi attraversati, usato come stima per il ritardo di propagazione)
 - Sono usate cifre di merito per valutare le trasformazioni
 - Trasformazioni non convenienti sono rifiutate
 - Le trasformazioni sono applicate in modo iterativo
 - La rete è considerata ottimale quando, rispetto ad un insieme di operatori, nessuno di questi la migliora
-



Sintesi di reti combinatorie a più livelli: *Approcci alla ottimizzazione multi-livello*

- L'**approccio** tipicamente utilizzato è quello **algoritmico**
 - Ogni trasformazione è associata ad un algoritmo
 - L'algoritmo:
 - determina dove può essere applicata la trasformazione;
 - applica la trasformazione e la mantiene se porta benefici;
 - termina quando nessuna trasformazione di quel tipo è ulteriormente applicabile.
 - Il maggior vantaggio dell'approccio algoritmico è che trasformazioni di un dato tipo sono sistematicamente applicate alla rete.
 - Algoritmi legati a differenti trasformazioni sono applicati in sequenza.
 - Sfortunatamente **differenti sequenze** possono portare a **soluzioni diverse**.
 - Soluzione: uso di sequenze derivate da **sperimentazioni**.
-

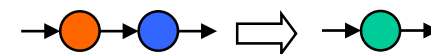


Sintesi di reti combinatorie a più livelli:

Trasformazioni base - 1

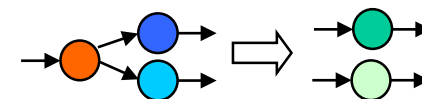
□ Sostituzione di nodi a 1 variabile (*sweep*)

- sostituisce la variabile assegnata nel nodo a monte in tutti i nodi a valle (globale)



□ Eliminazione (*eliminate*)

- sostituisce l'espressione di un nodo in uno o più nodi a valle e elimina il nodo originale, +prestazioni temporali (globale, diminuisce il percorso di I/O), -area



$$\updownarrow = n(l-1) - l$$

$$\updownarrow \text{lett}_{\max}$$

□ Semplificazione (*simplify*)

- manipola l'espressione di un nodo per portarla su due livelli (successiva all'eliminazione) - locale

□ Fattorizzazione (*factor*)

- fattorizza l'espressione di un nodo, ottenendo un'espressione su più livelli. Cerca un'espressione da portare poi a fattore comune anche per altri nodi.

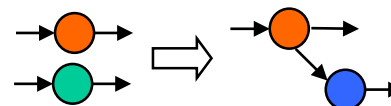


Sintesi di reti combinatorie a più livelli:

Trasformazioni base - 2

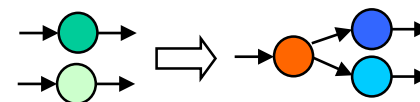
□ Sostituzione (*substitute*)

- Utilizza un nodo già presente nella rete per semplificare un altro nodo, sostituendo una sotto-espressione (diminuisce il n° di letterali nel secondo nodo) (globale, aumenta il percorso di I/O)



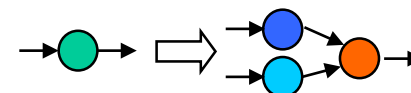
□ Estrazione di una sotto-espressione (*extract*)

- Simile alla sostituzione ma più generale: il nodo da estrarre non deve già esistere (si cerca un divisore comune a più nodi)



□ Decomposizione di una espressione (*decompose*)

- Applica il teorema di espansione di Shannon: estra da un nodo 2^k nodi (globale, aumenta il percorso di I/O)





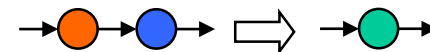
Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: eliminazione

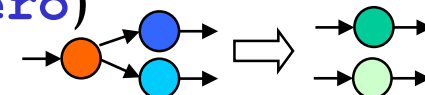
- **Eliminazione**: globale, riduce la lunghezza del percorso I/O

La lunghezza è calcolata in numero di nodi attraversati.

- **Eliminazione** nella rete di tutti i **vertici con un solo ingresso** e di quelli relativi a funzioni costanti (**Sweep**)



- **Riduzione vincolata** (**Eliminate** opzione **Val-Intero**)
eliminate 5



- L'eliminazione di un vertice è accettata se incrementa l'area di una quantità inferiore a **Val-Intero**.

- Ad esempio, l'incremento di area può venire calcolato come

$\Delta = n(l-1) - l$, dove l è numero di letterali del nodo eliminato mentre n è il numero di nodi che lo assorbono

- **Riduzione non vincolata**

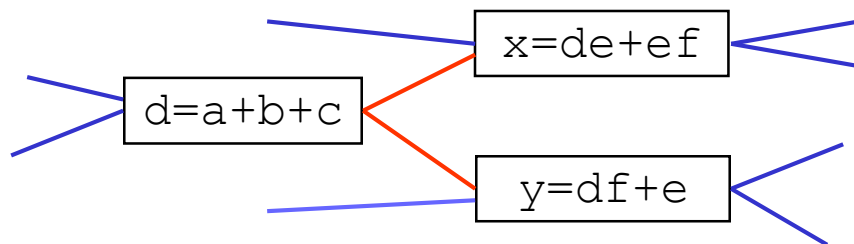
- tutti i nodi vengono ridotti ad un solo nodo; si ottiene una rete a due livelli.



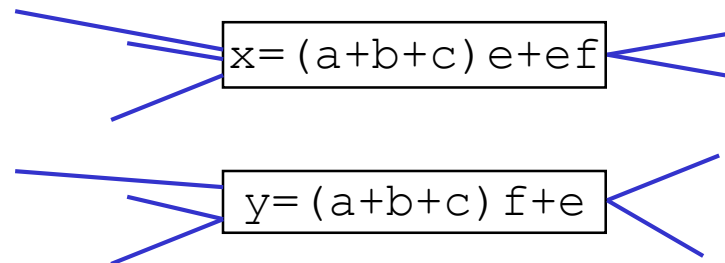
Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: eliminazione

□ Esempio di eliminate 2:



Costo: $3+4+3=10$



Costo: $6+5=11$

incremento di costo: $2*3-2-3 = 1$ (accettato)

□ eliminate -1

		n						
		1	2	3	4	5	6	7
1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	0	1	2	3	4	5	
3	-1	1	3	5	7	9	11	
4	-1	2	5	8	11	14	17	
5	-1	3	7	11	15	19	23	
6	-1	4	9	14	19	24	29	
7	-1	5	11	17	23	29	35	
8	-1	6	13	20	27	34	41	
9	-1	7	15	23	31	39	47	

Osservando i dati riportati in tabella, relativi al calcolo di $n \cdot l - n - l$ al variare di n e l , si può constatare che l'effetto di eliminate -1 (con $l=1$) è quello di eliminare tutti i nodi composti da un solo letterale (*sweep*).



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi: semplificazione*

□ Semplificazione: trasformazione locale

- **Semplificazione a due livelli** di ogni nodo (**simplify**)
 - Metodo esatto (Quine-McCluskey) o euristico.
- **Fattorizzazione** di un nodo (**factor**)
 - All'interno di un nodo, raccoglie a fattore comune alcuni termini >> da due a più livelli
 - Esempio: (ipotesi: porte a 3 ingressi)

$$f = l' + c'gh' + ab'k' + gk' + a'b'c'd'e' + ad'e'f' + e'g'i' + e'j';$$

Ritardo: 4τ ; Costo: 23



$$f = l' + c'gh' + k'(ab' + g) + e'(d'(a'b'c' + af') + g'i' + j');$$

Ritardo: 7τ ; Costo: 18



Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: fattorizzazione

□ *Fattorizzazione*

- L'espressione logica fattorizzata può essere ottenuta utilizzando una euristica.
 - Politica della euristica: si **pesano i letterali dell'espressione di partenza** con ordinamento lessico-grafico a parità di peso
 - Elemento più a destra per primo
- L'**insieme dei termini prodotto viene ricorsivamente partizionato** (*blocco della partizione e blocco residuo*) utilizzando come termine di riferimento il letterale che compare con più frequenza.
 - **Ottimizzazione**: tutti i **letterali** che hanno la **stessa cardinalità della partizione** vengono **raccolti contemporaneamente**
- Ad ogni passo della ricorsione le **partizioni** sono in **OR** fra loro mentre i **termini a fattor comune** sono in **AND**.



Sintesi di reti combinatorie a più livelli:

Fattorizzazione - esempi

□ Esempio 1:

$$f = acd + a'bc' + a'bd' + b'cd$$

Ritardo: 3τ
costo: 12

	a	a'	b	b'	c	c'	d	d'
acd	1	0	0	0	1	0	1	0
a'bc'	0	1	1	0	0	1	0	0
a'bd'	0	1	1	0	0	0	0	1
b'cd	0	0	0	1	1	0	1	0
	1	2	2	1	2	1	2	1



Sintesi di reti combinatorie a più livelli: *Fattorizzazione - esempi*

□ Esempio 1:

$$f = acd + a'bc' + a'bd' + b'cd$$

Ritardo: 3τ
costo: 12

Blocco della partizione
indotta dal fattore
comune dc

	a	a'	b	b'	c	c'	d	d'
acd	1	0	0	0	1	0	1	0
a'bc'	0	1	1	0	0	1	0	0
a'bd'	0	1	1	0	0	0	0	1
b'cd	0	0	0	1	1	0	1	0
	1	2	2	1	2	1	2	1

Blocco residuo della
partizione

+

	a	a'	b	b'
a	1	0	0	0
b'	0	0	0	1
	1	0	0	1

b'

+

a



Sintesi di reti combinatorie a più livelli:

Fattorizzazione - esempi

□ Esempio 1:

$$f = acd + a'bc' + a'bd' + b'cd$$

Ritardo: 3τ
costo: 12

	a	a'	b	b'	c	c'	d	d'
acd	1	0	0	0	1	0	1	0
a'bc'	0	1	1	0	0	1	0	0
a'bd'	0	1	1	0	0	0	0	1
b'cd	0	0	0	1	1	0	1	0
	1	2	2	1	2	1	2	1

Blocco della partizione
indotta dal fattore
comune dc

	a	a'	b	b'
a	1	0	0	0
b'	0	0	0	1
	1	0	0	1

+

Blocco residuo della
partizione

	a	a'	b	b'	c	c'	d	d'
a'bc'	0	1	1	0	0	1	0	0
a'bd'	0	1	1	0	0	0	0	1
	0	2	2	0	0	1	0	1

Fattore comune a'b

	c	c'	d	d'
c'	0	1	0	0
d'	0	0	0	1
	1	0	0	1

b' + a

+

a

Ritardo: 3τ
costo: 8

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



Sintesi di reti combinatorie a più livelli: *Fattorizzazione - esempi*

□ Esempio 2: (forma 2 livelli non ottimizzata)

$$f = abcd + ab'c'd + a'b'cd + a'b'c'd$$

Ritardo: 4τ
costo: 12

	a	a'	b	b'	c	c'	d	d'
abcd	1	0	1	0	1	0	1	0
ab'c'd	1	0	0	1	0	1	1	0
a'b'cd	0	1	0	1	1	0	1	0
a'b'c'd	0	1	0	1	0	1	1	0
	2	2	1	3	2	2	4	0



Sintesi di reti combinatorie a più livelli:

Fattorizzazione - esempi

□ Esempio 2: (forma 2 livelli non ottimizzata)

Ritardo: 4τ
costo: 12

$$f = abcd + ab'c'd + a'b'cd + a'b'c'd$$

	a	a'	b	b'	c	c'	d	d'
abcd	1	0	1	0	1	0	1	0
ab'c'd	1	0	0	1	0	1	1	0
a'b'cd	0	1	0	1	1	0	1	0
a'b'c'd	0	1	0	1	0	1	1	0
	2	2	1	3	2	2	4	0

Fattore comune d

Blocco della
partizione
indotta dal
fattore comune
 b'

	a	a'	c	c'
ac'	1	0	0	1
a'c	0	1	1	0
a'c'	0	1	0	1
	1	2	1	2

	a	a'	b	b'	c	c'
abc	1	0	1	0	1	0
ab'c'	1	0	0	1	0	1
a'b'c	0	1	0	1	1	0
a'b'c'	0	1	0	1	0	1
	2	2	1	3	2	2

+

abc

Blocco residuo
della
partizione



Sintesi di reti combinatorie a più livelli: Fattorizzazione - esempi

□ Esempio 2: (forma 2 livelli non ottimizzata)

$$f = abcd + ab'c'd + a'b'cd + a'b'c'd$$

Ritardo: 4τ
costo: 12

	a	a'	b	b'	c	c'	d	d'
abcd	1	0	1	0	1	0	1	0
ab'c'd	1	0	0	1	0	1	1	0
a'b'cd	0	1	0	1	1	0	1	0
a'b'c'd	0	1	0	1	0	1	1	0
	2	2	1	3	2	2	4	0

Fattore comune d

Blocco della partizione
indotta dal
fattore comune
 b'

Blocco della partizione
indotta dal fattore
comune c'

$a + a'$

	a	a'
a	1	0
a'	0	1
	1	1

	a	a'	b	b'	c	c'
abc	1	0	1	0	1	0
ab'c'	1	0	0	1	0	1
a'b'c	0	1	0	1	1	0
a'b'c'	0	1	0	1	0	1
	2	2	1	3	2	2

Blocco residuo
della
partizione

+

abc

+

$a'c$

Ritardo: 5τ
costo: 10

$$f = d(abc + b'(a'c + c'(a + a')))$$



Sintesi di reti combinatorie a più livelli: Fattorizzazione - esempi

□ Esempio 3: $f = abd' + a'bd + a'b'd' + a'cd + b'cd'$

Ritardo: 3τ
costo: 15

Blocco della
partizione indotta dal
fattore comune d'

	a	a'	b	b'	c	c'
ab	1	0	1	0	0	0
a'b'	0	1	0	1	0	0
b'c	0	0	0	1	1	0
	1	1	1	2	1	0

Blocco della partizione
indotta dal fattore
comune b'

	a	a'	c	c'
a'	0	1	0	0
c	0	0	1	0
	0	1	1	0

$a' + c$

	a	a'	b	b'	c	c'	d	d'
a'bd	0	1	1	0	0	0	1	0
a'cd	0	1	0	0	1	0	1	0
abd'	1	0	1	0	0	0	0	1
a'b'd'	0	1	0	1	0	0	0	1
b'cd'	0	0	0	1	1	0	0	1
	1	3	2	2	2	0	2	3

Blocco residuo
della partizione

	a	a'	b	b'	c	c'	d	d'
a'bd	0	1	1	0	0	0	1	0
a'cd	0	1	0	0	1	0	1	0
	0	2	1	0	1	0	2	0

Fattore comune $a'd$

	b	b'	c	c'
b	1	0	0	0
c	0	0	1	0
	1	0	1	0

$b + c$

Ritardo: 5τ
costo: 10

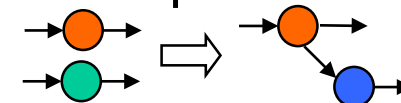
$$f = d' (ab + b' (a' + c)) + a'd (b + c)$$



Sintesi di reti combinatorie a più livelli:

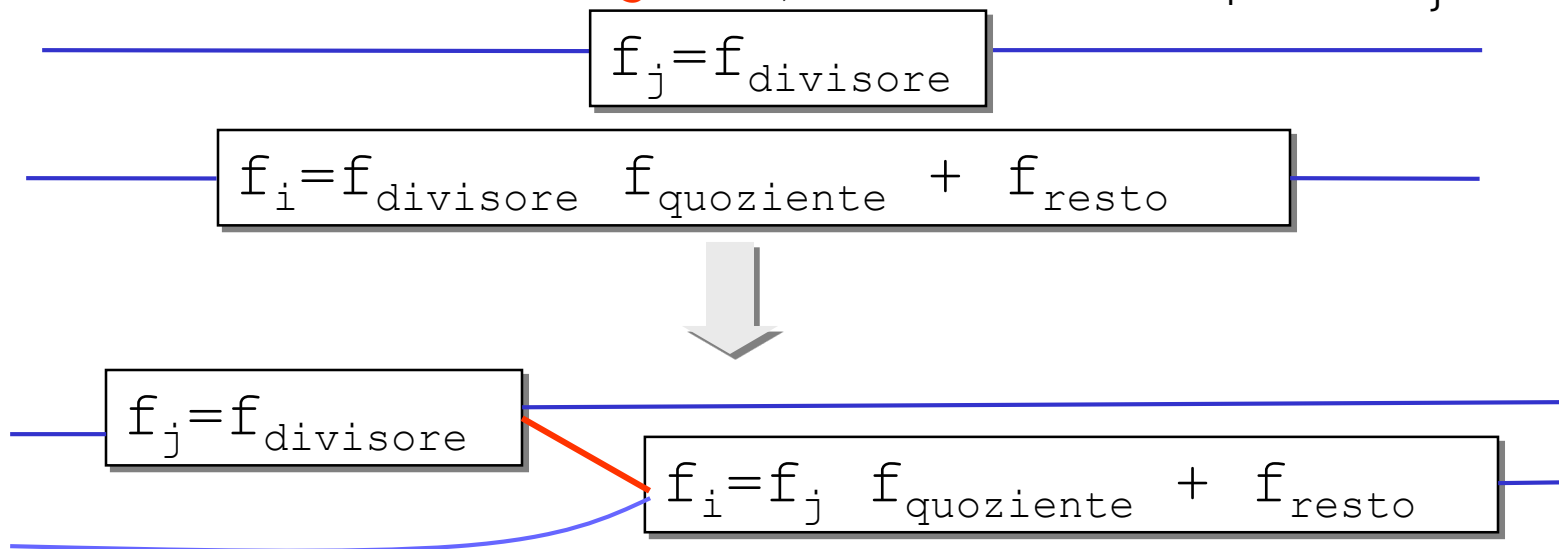
Trasformazioni e algoritmi: sostituzione

- ❑ **Sostituzione** (**substitute**): globale, aumenta la lunghezza del percorso I/O



- ❑ Sostituzione di una sotto-espressione mediante una variabile (nodo) già presente nella rete. In generale, ogni sostituzione è accettata se produce guadagno nel numero di letterali.

- Fa uso della **divisione algebrica**; si cerca di ridurre f_i usando f_j

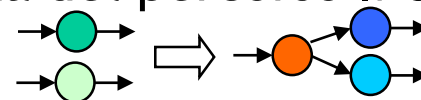




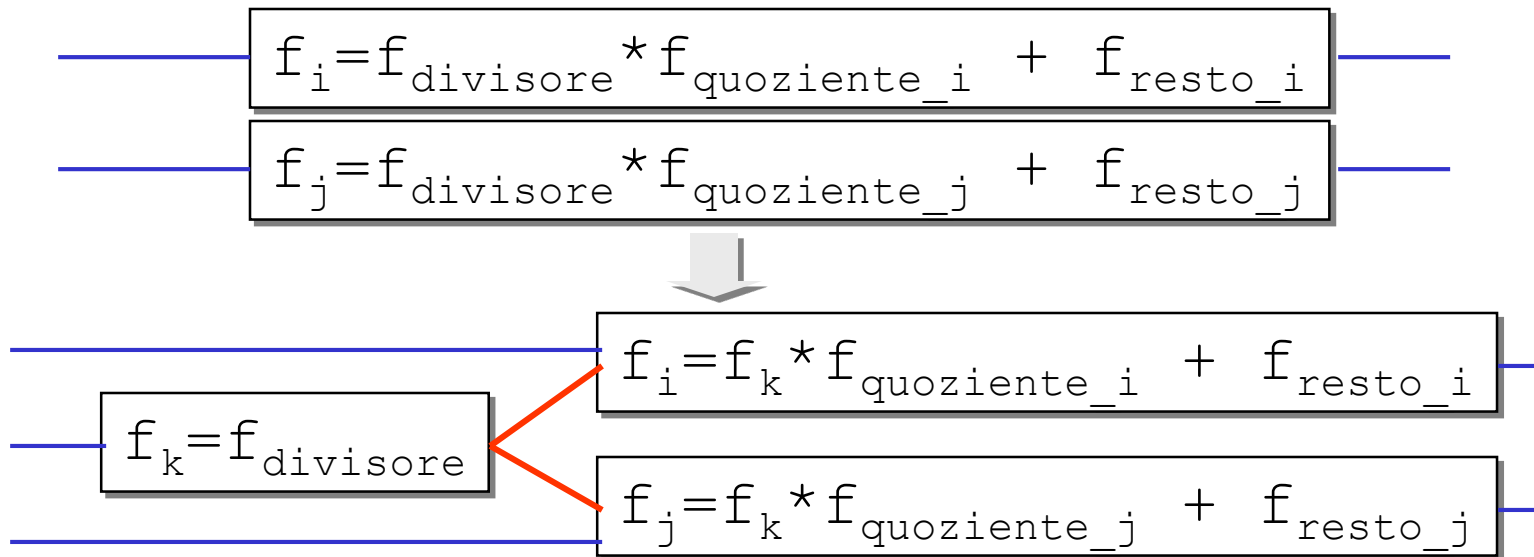
Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: estrazione

- **Estrazione** (**extract**) globale, aumenta la lunghezza del percorso I/O



- Estrae una espressione da gruppi di nodi. L'estrazione viene fatta fino a che è possibile.
 - Identificazione un divisore comune a due o più espressioni.
 - Il divisore costituisce un nuovo nodo della rete ed ha per successori i nodi da cui è stato estratto.





Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: decomposizione algebrica

- **Decomposizione algebrica** (**decompose**) : globale, aumenta la lunghezza del percorso I/O 

- Riduce le dimensioni di una espressione per:
 - Rendere più semplice l'operazione di *library mapping*.
 - Aumentare la probabilità di successo della sostituzione
- La decomposizione può essere applicata ricorsivamente al divisore, quoziente e resto.

$$f_i = f_d (f_{dq} f_{qq} + f_{rq}) + (f_{dr} f_{qr} + f_{rr})$$

$$f_k = f_{dq}$$

$$f_l = f_{dr}$$

$$f_j = f_d$$

$$f_i = f_j (f_k f_{qq} + f_{rq}) + f_l f_{qr} + f_{rr}$$



Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: divisori

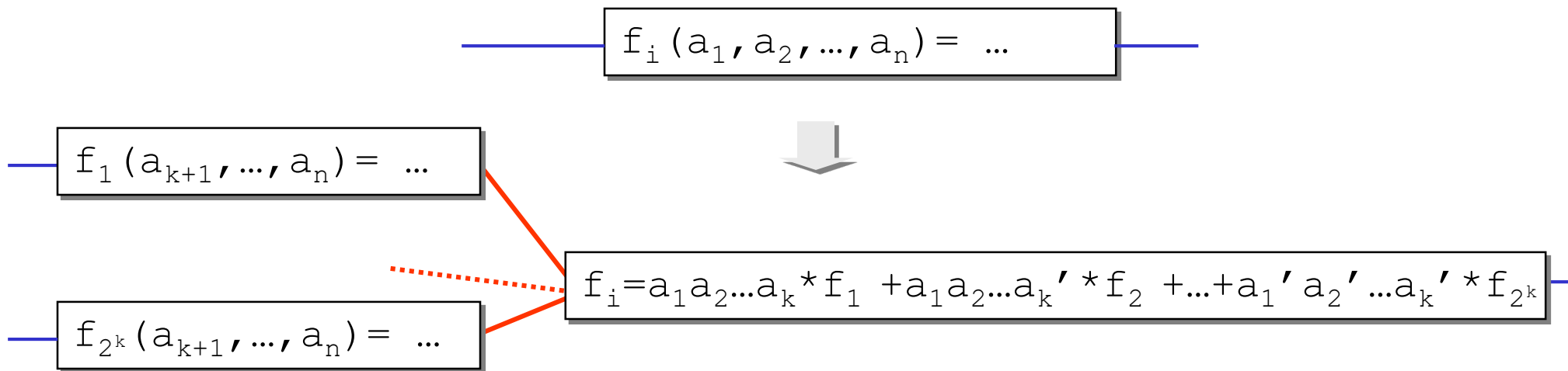
- **Decomposizione algebrica, estrazione e sostituzione:** come si trovano i divisori?
 - **Modello algebrico:** le espressioni booleane vengono viste come **espressioni algebriche**, cioè come **polinomi di primo grado**, nelle **variabili naturali e complementate**, con **coefficienti unitari**
 - Lavorando con il modello algebrico valgono le **proprietà algebriche** mentre quelle dell'algebra booleana non sono valide
 - È definita la **divisione algebrica**: $f_{divisore}$ è un divisore algebrico di $f_{dividendo}$ se
 - $f_{dividendo} = f_{divisore} \cdot f_{quoziente} + f_{resto}$ e
 - $f_{quoziente} \cdot f_{divisore} \neq 0$ e
 - il supporto di $f_{divisore}$ e di $f_{quoziente}$ è disgiunto
 - Esistono algoritmi diversi per calcolare i divisori di una espressione algebrica



Sintesi di reti combinatorie a più livelli:

Trasformazioni e algoritmi: decomposizione disgiuntiva

- ❑ **Decomposizione disgiuntiva semplice** (**decompose**) globale, aumenta la lunghezza del percorso I/O
 - Riduce le dimensioni di una espressione (v. decomposizione algebrica)
 - La decomposizione disgiuntiva semplice può essere applicata ricorsivamente.





□ Decomposizione disgiuntiva (cont.)

- Deriva dalla applicazione del teorema di *espansione di Shannon*:

$$f(a_1, a_2, \dots, a_n) = a_1 * f_{a_1} + a_1' * f_{a_1'}$$

- Il risultato, in termini di costo, dipende fortemente dalla decomposizione che viene effettuata sulle variabili di supporto della funzione.
 - Con n variabili il numero di possibili scomposizioni è $2^n - 2$

□ Teorema di espansione di Shannon

- se $f: B^n \rightarrow B$ è una *funzione booleana* si ha

$$f(x_1, x_2, \dots, x_n) = x_1' * f_{x_1=0} + x_1 * f_{x_1=1}$$

per ogni (x_1, x_2, \dots, x_n) in B^n .

Ad esempio, $f(a, b, c) = a' * f(0, b, c) + a * f(1, b, c)$



Sintesi di reti combinatorie a più livelli:

Decomposizione disgiuntiva - esempi

□ Esempio 1:

- Esempio: scomposizione disgiuntiva di f rispetto a b

$$f = a'b + ab' + c'd' + cd$$

Costo: 8

$$f = bf_b + b'f_{b'}$$



Sintesi di reti combinatorie a più livelli:

Decomposizione disgiuntiva - esempi

□ Esempio 1:

- Esempio: scomposizione disgiuntiva di f rispetto a b

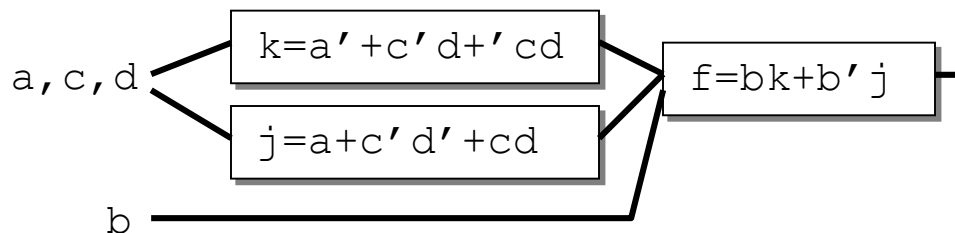
$$f = a'b + ab' + c'd' + cd$$

Costo: 8

$$f = bf_b + b'f_{b'}$$

$$f_b = f(a, 1, c, d) = a'1 + a1' + c'd' + cd = a' + c'd' + cd$$

$$f_{b'} = f(a, 0, c, d) = a'0 + a0' + c'd' + cd = a + c'd' + cd$$



Costo: 14



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio 1:

- Esempio: scomposizione disgiuntiva di f rispetto ad ab

$$\boxed{f = a'b + ab' + c'd' + cd} \quad \text{Costo: 8}$$

$$f = a(bf_{ab} + b'f_{ab'}) + a'(bf_{a'b} + b'f_{a'b'}) = abf_{ab} + ab'f_{ab'} + a'b f_{a'b} + a'b' f_{a'b'}$$



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio 1:

- Esempio: scomposizione disgiuntiva di f rispetto ad ab

$$f = a'b + ab' + c'd' + cd$$

Costo: 8

$$f = a(bf_{ab} + b'f_{ab'}) + a'(bf_{a'b} + b'f_{a'b'}) = abf_{ab} + ab'f_{ab'} + a'b f_{a'b} + a'b' f_{a'b'}$$

$$f_{ab} = f(1, 1, c, d) = c'd' + cd$$

$$f_{ab'} = f(1, 0, c, d) = 1 + c'd' + cd = 1$$

$$f_{a'b} = f(0, 1, c, d) = 1 + c'd' + cd = 1$$

$$f_{a'b'} = f(0, 0, c, d) = c'd' + cd = c'd' + cd$$

$$c, d \rightarrow k = c'd' + cd$$

$$f = abk + a'b'k + a'b + ab'$$

Semplificazione a due livelli
sul nodo f

$$c, d \rightarrow k = c'd' + cd$$

$$f = k + a'b + ab'$$

Costo: 9

a, b \ k	0	1
	00	01
0	0	1
1	1	1
2	0	1
3	1	1



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio 2 (xor):

- scomposizione disgiuntiva di f rispetto ad ab

$$f = a'b'c'd' + a'b'cd + a'bc'd + a'bcd' + abc'd' + abcd + ab'c'd + ab'cd'$$

Costo: 32

$$f_{ab} = c'd' + cd$$

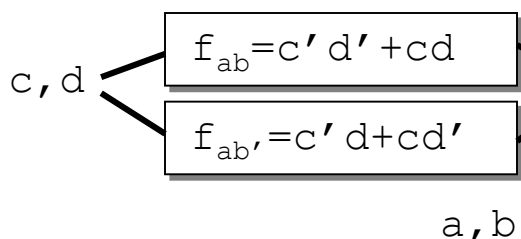
$$f_{ab'} = c'd + cd'$$

$$f_{a'b} = c'd + cd'$$

$$f_{a'b'} = c'd' + cd$$

$$f_{ab} = f_{a'} = c'd' + cd$$

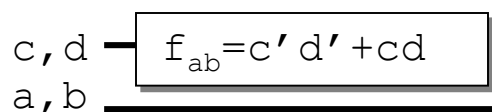
$$f_{a'b} = f_{a'b'} = c'd + cd'$$



$$f = (ab + a'b')f_{ab} + (a'b + ab')f_{ab'}$$

Costo: 18

Trasformazione booleana: $f_{ab} = f'_{ab'}$



$$f = (ab + a'b')f_{ab} + (a'b + ab')f'_{ab}$$

Costo: 14



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio3: — $f = a * !b * c + !a * !b * d + !a * c * d + !c * !d$ — Costo: 11

scomposizione disgiuntiva di f rispetto ad ab



Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio3:

$$f = a * !b * c + !a * !b * d + !a * c * d + !c * !d$$

Costo: 11

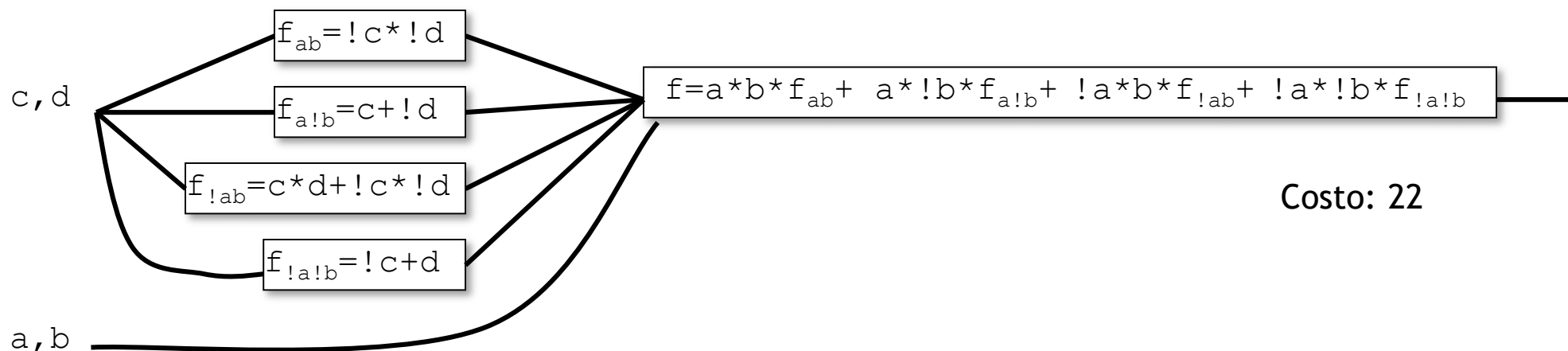
scomposizione disgiuntiva di f rispetto ad ab

$$f_{ab} = 1 * 0 * c + 0 * 0 * d + 0 * c * d + !c * !d = !c * !d$$

$$f_{a!b} = 1 * 1 * c + 0 * 1 * d + 0 * c * d + !c * !d = c + !c * !d \Rightarrow c + !d$$

$$f_{!ab} = 0 * 0 * c + 1 * 0 * d + 1 * c * d + !c * !d = c * d + !c * !d$$

$$f_{!a!b} = 0 * 1 * c + 1 * 1 * d + 1 * c * d + !c * !d = d + c * d + !c * !d \Rightarrow !c + d$$





Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio 3:

$$f = ab'c + a'b'd + a'cd + c'd'$$

Costo: 11

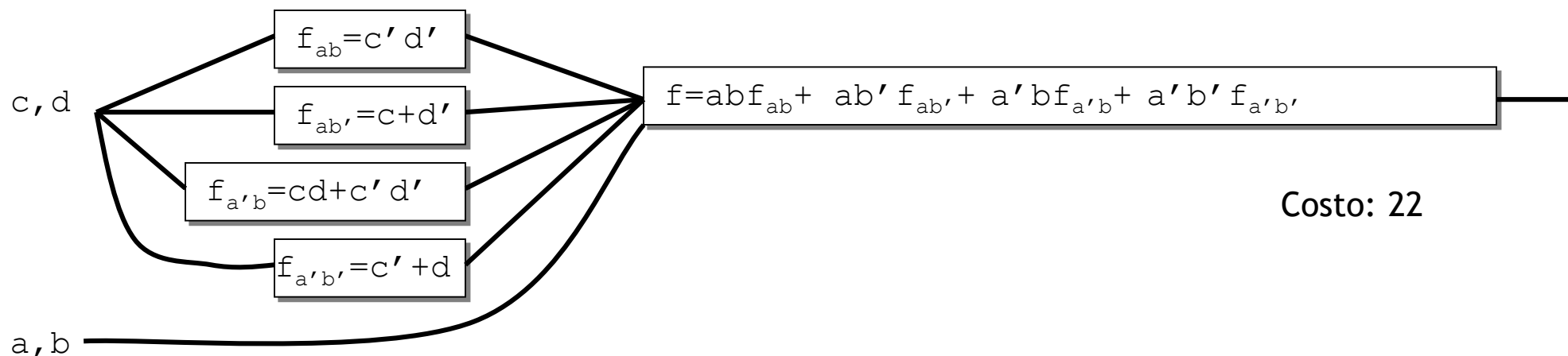
scomposizione disgiuntiva di f rispetto ad ab

$$f_{ab} = c'd'$$

$$f_{ab'} = c + c'd' \Rightarrow c + d'$$

$$f_{a'b} = cd + c'd'$$

$$f_{a'b'} = d + cd + c'd' \Rightarrow c' + d$$



Costo: 22



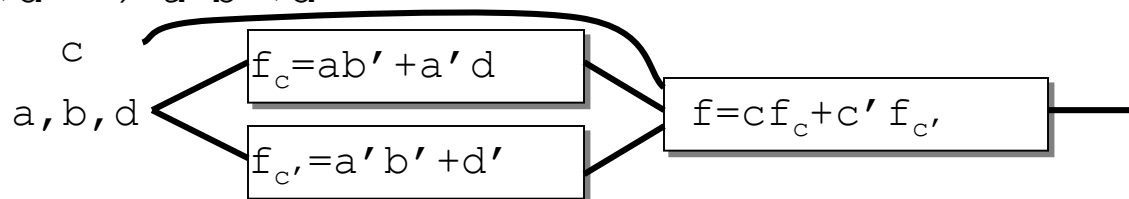
Sintesi di reti combinatorie a più livelli: *Trasformazioni e algoritmi*

□ Esempio 3 (cont.): — $f = ab'c + a'b'd + a'cd + c'd'$ — Costo: 11

scomposizione disgiuntiva di f rispetto ad c

$$f_c = ab' + a'b'd + a'd \Rightarrow ab' + a'd$$

$$f_{c'} = a'b'd + d' \Rightarrow a'b' + d'$$



Costo: 11

scomposizione disgiuntiva di f rispetto ad a

Costo: 14

$$f_a = b'c + c'd'$$

$$f_{a'} = b'd + cd + c'd'$$

scomposizione disgiuntiva di f rispetto ad b

Costo: 15

$$f_b = a'cd + c'd'$$

$$f_{b'} = ac + a'd + a'cd + c'd' \Rightarrow ac + a'd + c'd'$$

scomposizione disgiuntiva di f rispetto ad d

Costo: 13

$$f_d = ab'c + a'b' + a'c \Rightarrow ab' + a'b' + a'c$$

$$f_{d'} = ab'c + c' \Rightarrow ab' + c'$$

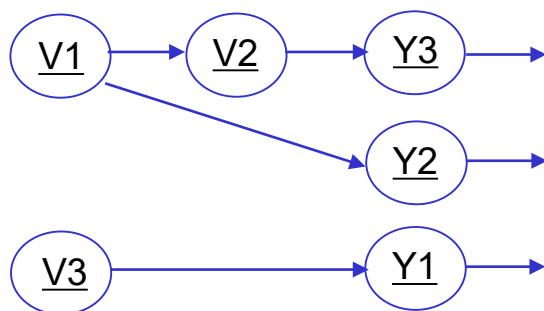


```
sweep; eliminate -1; simplify -m nocomp; eliminate -1  
sweep; eliminate 5; simplify -m nocomp  
resub -a; fx; resub -a  
sweep; eliminate -1  
sweep; full_simplify -m nocomp
```



Sintesi multilivello di reti combinatorie - *esempio (1)*

- Data una rete combinatoria con ingressi (a, b, c, d, e, f), uscite (Y1, Y2, Y3) e rappresentata dal seguente modello (*rete multilivello*)



Equazioni dei nodi

$$V_1 = \bar{a}b\bar{d} + ef$$

$$V_2 = a\bar{V}_1b + aV_1c + V_1cd + \bar{V}_1bd$$

$$V_3 = \bar{b}de + ab\bar{c}e + b\bar{d}e + \bar{a}b\bar{c}e + \bar{b}def$$

$$Y_1 = V_3 + a\bar{f}$$

$$Y_2 = \bar{V}_1ab + \bar{a}bcd\bar{e} + \bar{b}cde + \bar{V}_1a\bar{b}f + \bar{a}bcde$$

$$Y_3 = a\bar{b}\bar{f} + d\bar{b}\bar{f} + ae + a\bar{V}_2 + de + d\bar{V}_2$$



Sintesi multilivello di reti combinatorie - *esempio (2)*

Applicare in sequenza alla rete multi livello le trasformazioni sotto indicate

- a) **COST()**: Calcolo del numero di letterali. La funzione COST() calcola il costo in letterali indipendentemente dalla forma (SOP o multi livello) delle espressioni algebriche dei nodi.
 - b) **SIMPLIFY(Y_2)**: Minimizzazione a due livelli di Y_2 .
 - c) **SIMPLIFY(V_3)**: Minimizzazione a due livelli di V_3 .
 - d) **ELIMINATE(V_3 , +3)**: Eliminazione del nodo V_3 : la trasformazione viene accettata solo se l'incremento di area, dovuto all'eliminazione è inferiore o uguale alla soglia data (+3).
 - e) **FACTOR(V_2)**: Fattorizzazione del nodo V_2 .
 - f) **COST()**: Calcolo del numero di letterali.
 - g) **[V_4] = EXTRACT(V_2 , Y_3)**: Estrazione di un fattore comune a V_2 e Y_3 . Il nodo V_4 derivato dall'estrazione può essere un nuovo nodo o un nodo già presente nella rete.
 - h) **COST()**: Calcolo del numero di letterali.
-



Sintesi di reti combinatorie a più livelli:

Esercizi

□ Esercizi & Soluzioni di fattorizzazione:

$$f = abcd' + ab'c' + a'bc' + b'cd = c(abd' + b'd) + c'(ab' + a'b)$$

$$f = abcd' + abc'd + ab'c'd' + a'bc'd' + a'b'd + a'cd + b'cd = d'(abc + c'(ab' + a'b)) + d(abc' + c(b' + a') + a'b')$$

$$f = ac'd + a'bcd + a'c'd' + b'c'd = a'bcd + c'(d(b' + a) + a'd')$$

$$f = abc' + abd' + ab'cd + ac'd' + a'bcd + bc'd' = a(b'cd + c'd') + b(a'cd + d'(c' + a) + a'c')$$

$$f = ab'cd + a'bcd + a'b'c' + a'b'd' + b'c'd' = a'bcd + b'(acd + d'(c' + a') + a'c')$$