Esercitazione RLC

FONDAMENTI DI INFORMATICA B

DIDATTICA A DISTANZA

Università degli studi di Parma
Dipartimento di Ingegneria dell'Informazione

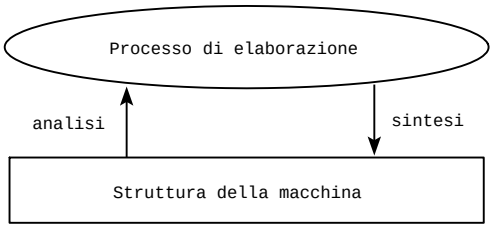
tutore: Ing. A. Tibaldi

17 maggio 2002

Indice

1 Sistemi digitali

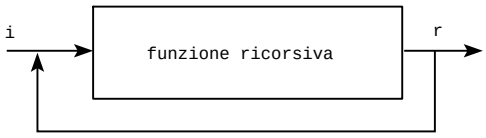
Sono macchine che attuano processi di elaborazione su grandezze dotate di finiti valori.



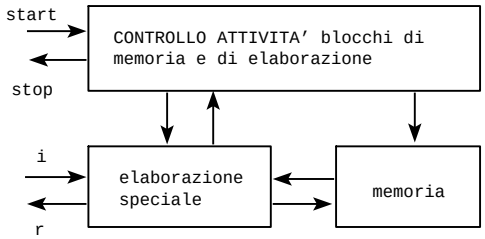
Il processo produce uscite partendo da ingressi e può essere descritto:

- tabularmente
- con formule
- con algoritmi (richiesta meno memoria del caso tabulare)

Si possono anche presentare funzioni ricorsive, che dovranno acquisire tra gli argomenti in ingresso anche valori calcolati da esse.

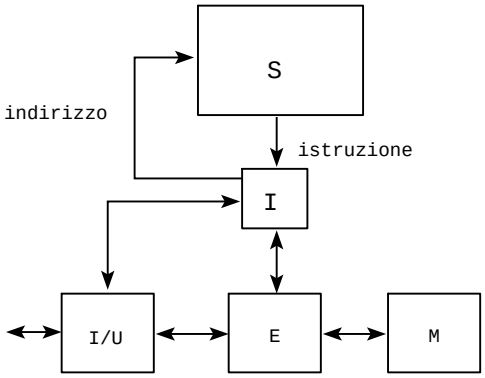


Macchine Special Purpose



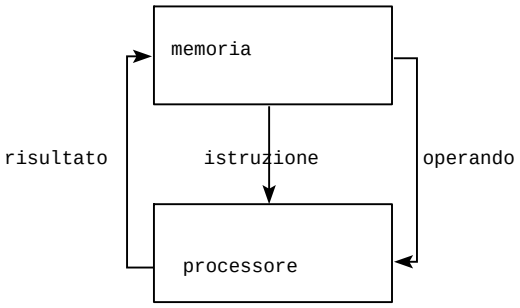
Macchine General Purpose

Offrono la possibilità di modificare facilmente la regola temporale di comportamento del blocco di controllo



Tale macchina contiene un programma memorizzato ed interpretato dall'interprete "I" che ottiene le istruzioni dalla memoria indirizzabile S, in cui l'utente dovrà inserire le istruzioni chiamate anche programma.

Von Neumann progettò il primo calcolatore a programma memorizzato, che effettua una elaborazione seriale.



Gli elaboratori elettronici basano il loro funzionamento su segnali binari (tali cioè da poter assumere esattamente due valori logici distinti). La commutazione caratterizzante questi segnali permette di classificarli come:

1. ASINCRONI: l'asse dei tempi non necessita di un'unità di misura in quanto sorgente e destinazione non attribuiscono alcun significato al permanere di una particolare situazione di valori.

2. SINCRONI: tra sorgente e destinazione è concordata un'unità di misura del tempo, più grande dei ritardi possibili sul mezzo trasmissivo. La sorgente può introdurre variazioni solo ad istanti discreti e distanti un numero intero di unità di misura e la destinazione campiona i segnali al termine di ogni intervallo elementare.

1.1 Modello matematico per lo studio dei circuiti digitali

M={I, U, S, F, G} macchina sequenziale o automa a stati finiti

I=$\{i_1, \ldots, i_n\}$ alfabeto di ingresso

U=$\{u_1,\ldots,u_m\}$ alfabeto di uscita
S=$\{s_1,\ldots,s_k\}$ insieme degli stati
funzione di uscita F:SxI $\longrightarrow$ U
funzione di stato G:SxI $\longrightarrow$ S
$u^n = F(s^n, i^n)$
$s^{n+1} = G(s^n, i^n)$

La macchina si definisce *combinatoria* se F dipende solamente dalla configurazione corrente di simboli in ingresso.

Una rete logica è una particolare macchina M che opera su segnali binari.

2 Sintesi di reti

Algebra di commutazione come insieme di:

1. SIMBOLI {0; 1}
2. OPERAZIONI
   - somma logica ($+$)
   - prodotto logico ($\cdot$)
   - complementazione ($\overline{\ }$)
3. POSTULATI

   $0+0=0$

   $0+1=1$

   $1+0=1$

   $1+1=1$

   $0\cdot 0=0$

   $0\cdot 1=0$

   $1\cdot 0=0$

   $1\cdot 1=1$

   $\overline{0}=1$

   $\overline{1}=0$

Dalla composizione di simboli, operazioni e variabili si ottengono espressioni che descrivono ciascuna un'unica funzione completa di tante variabili quante ne contiene l'espressione.

2.1 Algebra booleana

Consiste di:

1. un insieme K di elementi
2. due funzioni $\{+,\cdot\}$ da $K \times K \rightarrow K$
3. la funzione negazione $\{^-\}$

Si basa sui seguenti

ASSIOMI

1. Card(K)≥ 2
2. K è chiuso, cioè $\forall a,b \subset K : a+b \subset K \bigwedge a \cdot b \subset K$

Opera su variabili che possono solo assumere valore 0 o 1. Una funzione logica: $z = f(x_1, x_2, ..., x_n)$ è una legge che fa corrispondere ad ogni combinazione dei valori degli n ingressi (variabili indipendenti) uno ed un solo valore *binario* della variabile z.

PROPRIETÀ

- Esistenza degli elementi identità 0 ed 1 tali che
  1. $A+0=A$
  2. $A \cdot 1 = A$
- Proprietà commutativa:
  1. $A+B=B+A$
  2. $A \cdot B = B \cdot A$
- Proprietà distributiva:
  1. $A \cdot (B+C) = A \cdot B + A \cdot C$
  2. $A+(B \cdot C) = (A+B) \cdot (A+C)$
- Esistenza dell'inverso: per ogni elemento A dell'insieme binario $K = \{0;\ 1\}$ esiste un elemento $\overline{A}$ tale che:
  1. $A \cdot \overline{A} = 0$
  2. $A+\overline{A} = 1$
- Proprietà associativa:

1. $A+(B+C)=(A+B)+C$
2. $A(BC)=(AB)C$

- Proprietà dell'idempotenza:
  1. $A+A=A$
  2. $A\cdot A=A$
- Leggi di DeMorgan:
  1. $\overline{A+B}=\overline{A}\cdot\overline{B}$
  2. $\overline{A\cdot B}=\overline{A}+\overline{B}$
- Involuzione: $\overline{\overline{A}}=A$

Principio di DUALITÀ

Le proprietà precedenti sono espresse a coppie ed ognuna è derivabile dall'altra scambiando:

1. $0\leftrightarrow 1$
2. $+\leftrightarrow\cdot$

3 Reti logiche combinatorie

La sintesi di una rete logica combinatoria è soggetta a criteri di ottimalità; noi prenderemo in considerazione tali criteri inizialmente con degli esempi di reti a due livelli realizzanti funzioni che possono essere espresse da *somme di prodotti* (SP) o *prodotti di somme* (PS).

Esempio: espressioni a due livelli e schema logico di una funzione f definita tabularmente da:

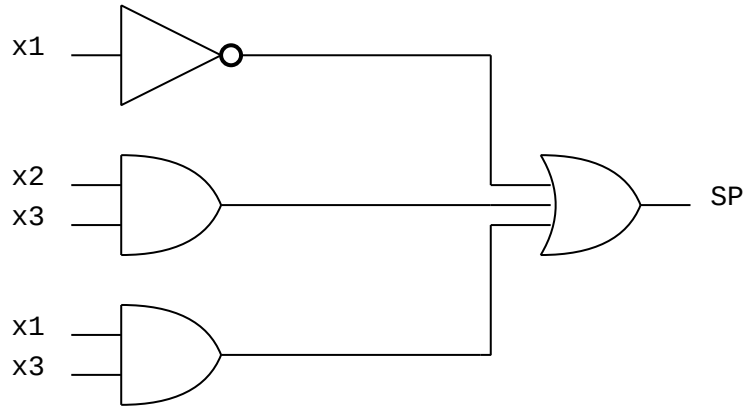
x_1	x_2	x_3	$f(x_1,x_2,x_3)$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

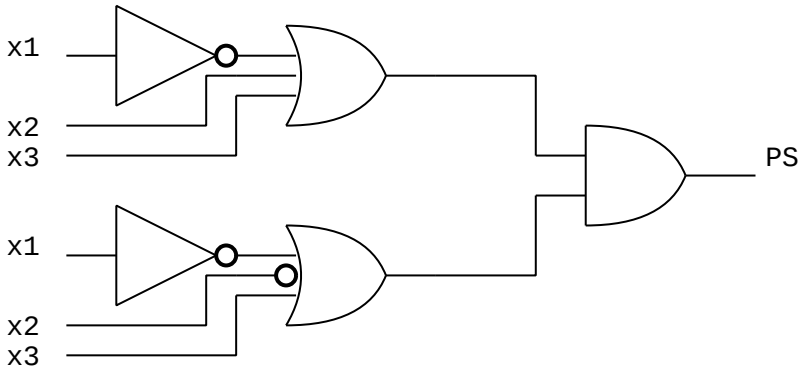
Per una stessa funzione possono esistere più espressioni che valutate restituiscano come codominio uno uguale in tutto e per tutto alla funzione data; tra le tante espressioni che realizzano f definita tabularmente

in precedenza scegliamo di rappresentarla con le seguenti espressioni a “due livelli”. È possibile effettuare una verifica esaustiva per controllare l’equivalenza alla funzione data.

$$f_{SP}(x_1,x_2,x_3) = \overline{x_1} + x_2 \cdot x_3 + x_1 \cdot x_3$$

$$f_{PS}(x_1,x_2,x_3) = (\overline{x_1} + x_2 + x_3) \cdot (\overline{x_1} + \overline{x_2} + x_3)$$





Una espressione tuttavia determina univocamente lo schema a porte logiche della rete, ma non è detto che tale schema sia quello ottimo.

3.1 Costo di una RLC

Nel caso di RLC a due livelli SP (PS) l’espressione equivalente è minima se

- non esiste altra espressione equivalente con minor numero di prodotti (somme) indicato con N_b (numero di blocchi)
- non esiste altra espressione equivalente con lo stesso numero di prodotti (somme) ma minor numero di letterali N_m

L'ottimo potrebbe essere ricercato minimizzando la funzione di costo $C_{[min]} = N_b \cdot N_m$.
Nell'esempio precedente la funzione combinatoria a due livelli aveva costo C rispettivamente

- SP: $N_b = 3+1$, $N_m = 5+3$, $C = 32$
- PS: $N_b = 2+1$, $N_m = 6+2$, $C = 24$

Si osserva che non sono stati considerati i blocchi NOT per il calcolo del costo.

Nel caso di esempio $C_{[min]}$ assume per la realizzazione SP il valore 32 e per la realizzazione PS il valore 24: si dovrà preferire dunque la forma PS.

Vale la pena osservare però che in questo caso la presenza dei *gate* di negazione innalza da due a tre il numero di livelli da attraversare da parte del segnale, se si ipotizzano i ritardi di attraversamento dei NOT dello stesso ordine di grandezza dei ritardi di attraversamento delle altre porte.

3.2 Minimizzazione algebrica

La *semplificazione di espressioni* attraverso le regole dell'algebra booleana è un primo passo verso la realizzazione di una funzione "minima".

Si indica per ora con *mintermine* il simbolo $m(N_{10})$ avente un solo argomento numerico decimale, il prodotto logico di n letterali ordinati di peso binario via via decrescente fino all'unità, con $n = \lceil log_2 N_{10} \rceil$, espressi in forma affermata o negata in modo tale che l'i–esimo letterale al quale è stato attribuito il peso 2^i sommi se affermato il proprio peso a quello degli altri per ottenere in totale il valore N_{10}.

Consideriamo la funzione di $n = 3$ variabili ordinate e di peso binario decrescente, espressa simbolicamente con una simbologia generalizzante il concetto appena introdotto. Tale funzione avrà una tabella di verità formata di soli zeri all'infuori delle righe corrispondenti ai mintermini espressi dai valori decimali tra parentesi.

$$f(a,b,c) = \sum_3 m(0,1,4,6)$$

$$= \overline{a}\overline{b}\overline{c} + \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c}$$

che può essere espressa per la proprietà distributiva

$$\overline{a}\overline{b}(\overline{c}+c) + a\overline{c}(b+\overline{b})$$

quindi per l'esistenza dell'inverso si potrà scrivere

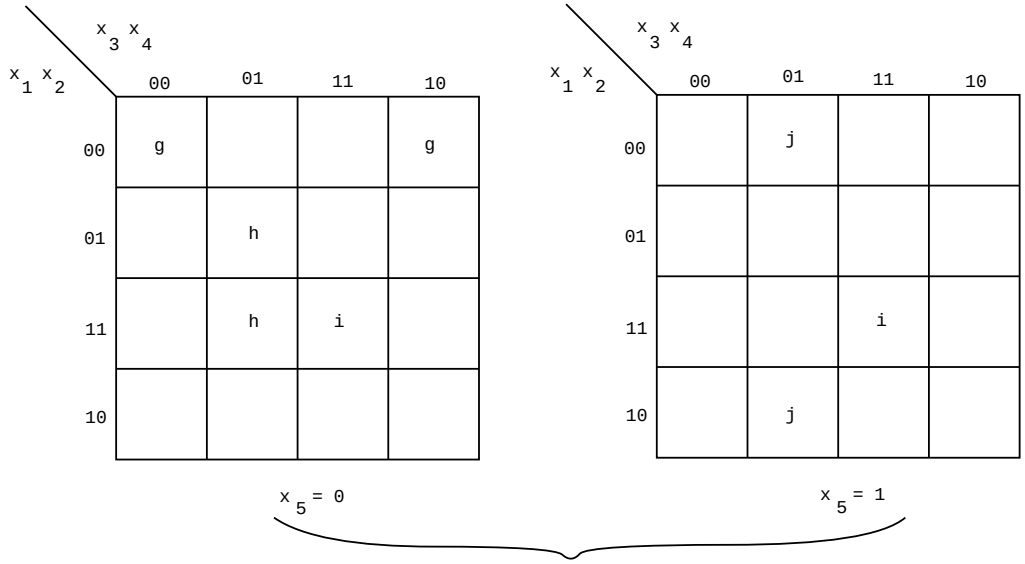
$$\overline{a}\overline{b} + a\overline{c}$$

Questo tipo di "raccoglimenti" sui letterali rispecchia le operazioni che si svolgono con la sintesi a mappe che viene presentata di seguito, nell'ipotesi che non vari il numero di livelli logici attraversati dal segnale.

4 Sintesi su due livelli con mappe di Karnaugh

Le mappe di Karnaugh sono utilizzate per la sintesi di reti rispondenti ad un assegnato comportamento ai morsetti; servono a specificare funzioni logiche in maniera alternativa alla tabella di definizione e permettono di identificare agevolmente termini SP o PS per la copertura degli "uni" o degli "zeri" di una funzione fino ad un numero di variabili pari a 5. Le mappe presentano configurazioni le seguendo un *codice ciclico*, variando cioè il valore ad una sola variabile tra una configurazione e quella "adiacente" che segue, che differirà quindi da questa per un solo bit.

Nelle mappe di Karnaugh sono adiacenti celle aventi un lato in comune o celle poste alle estremità di una stessa riga o poste all'estremità di una stessa colonna o anche celle poste nella stessa posizione su sottomappe adiacenti.



`Mintermine`: funzione logica per cui un solo elemento del codominio ha valore 1; questa presenta il prodotto di tutte le variabili in ingresso della funzione che vogliamo descrivere, in forma naturale se nella configurazione in ingresso relativa al termine da ricoprire sono presenti in forma affermata, in forma complementata se vi compaiono con valore 0.

`Maxtermine`: funzione logica duale, contenente un unico 0 all'interno del proprio codominio. Contiene tutte le variabili in ingresso sommate tra loro, in forma naturale (cioè affermata) se le variabili assumono valore zero nella configurazione in cui la funzione da ricoprire contiene l'unico 0 del codominio del M-termine, in forma complementata altrimenti.

`Implicante`: una funzione prodotto p è detta implicante di una funzione f se $f = 1$ almeno in tutti i vertici del sottocubo (rappresentazione topologica del dominio di f) in cui $p = 1$ ($f \rightarrow p$, f implica p) e si dirà che f copre p; p è un termine prodotto costituito da un numero di variabili inferiore a quelle su cui è definita la funzione. Sia k il numero di variabili presenti nell'implicante, allora questo identifica nella

mappa di Karnaugh un raggruppamento di 1 adiacenti, secondo le regole di adiacenza, di dimensioni $2^{(n-k)}$ in cui n è il numero di variabili della funzione f.

Un implicante può essere visto come un raggruppamento di mintermini o come il risultato dell'espansione di un mintermine.

Un implicante è *primo* se riducendo il numero di variabili in esso contenute, cioè espandendo la copertura degli "uni" che effettua nel codominio, non risulta più essere un implicante per f.

Un implicante è *essenziale* se è l'unico a ricoprire un vertice del dominio della funzione f in cui questa assume valore 1.

`Implicato`: una funzione somma s si dice implicato di una funzione f, se $f = 0$ almeno in tutti i vertici del sottocubo in cui $s = 0$ ($s \leftarrow f$, s è implicato da f). s contiene un numero di variabili inferiore a quelle del dominio della funzione e può essere visto come un raggruppamento di maxtermini o come l'espansione di un maxtermine riducendo il numero di letterali presenti in questo.

Di una funzione specificata tabularmente in genere si indicano gli insiemi seguenti:

- ONSet: insieme dei valori del dominio (o equivalentemente insieme di configurazioni binarie delle variabili di ingresso) tali per cui il corrispettivo valore nel codominio della funzione ha valore logico alto.
- OFFSet: insieme delle configurazioni degli ingressi della funzione per cui questa assume valore logico basso.
- DCSet: insieme delle configurazioni delle variabili di una funzione per le quali il valore logico assunto dalla funzione non è specificato.

La riduzione del costo di reti a due livelli descritte come somma di prodotti SP o prodotto di somme PS si basa sulla riduzione del numero di gate da utilizzare nel primo livello e dunque sull'identificazione di:

- tutti gli implicanti essenziali (ed il loro inserimento nella sommatoria)
- il più piccolo insieme di implicanti con il minor numero di letterali nel complesso che assieme a quelli essenziali "ricopre" interamente la funzione

Esistono algoritmi che permettono di individuare tutti gli implicanti primi di una funzione (Quine McCluskey) e la selezione del set ottimo (tabella di copertura).

4.1 Funzioni non completamente specificate

Per alcune funzioni esistono configurazioni degli ingressi non significative (condizioni di indifferenza), ma è comunque necessario selezionare una funzione che coincida con quella da realizzare nei punti in cui essa è specificata. Alle condizioni di indifferenza si assegna il valore 1 per la sintesi SP ed il valore 0 per la sintesi PS. Questo perché in tal caso è possibile individuare raggruppamenti di dimensione maggiore. Si prosegue poi come di consueto salvo che è in tal caso necessario rimuovere come prima cosa dall'insieme degli implicanti primi individuato quelli che ricoprono solamente condizioni di indifferenza.

OSS: L'espressione PS minima può avere un costo diverso dalla SP minima; risulta quindi necessario determinarle entrambe.

4.2 Reti a più di due livelli

Con le regole dell'algebra è possibile ottenere reti a minor costo ma meno veloci. Nell'esempio che segue risulta evidente come applicando dei raccoglimenti (proprietà distributiva) il costo valutato in termini di blocchi e morsetti si riduca drasticamente effettuando due soli raccoglimenti. La funzione F SP-ottima a due livelli SP ha costo $N_b = 5$ ed $N_m = 12$.

$$F_{OPT2LIV}(a,b,c,d) = b\overline{c} + a \cdot d + \overline{c} \cdot d + a \cdot b$$

per la proprietà distributiva

$$= \overline{c} \cdot (b+d) + a \cdot (b+d)$$

ed ancora distributiva

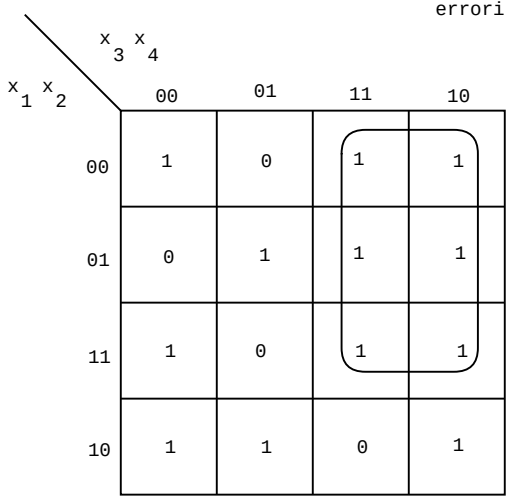
$$= (b+d) \cdot (a+\overline{c})$$

per arrivare ad avere un costo di $N_b = 3$ (due in meno dell'espressione di partenza) e $N_m = 6$ (la metà dei precedenti).

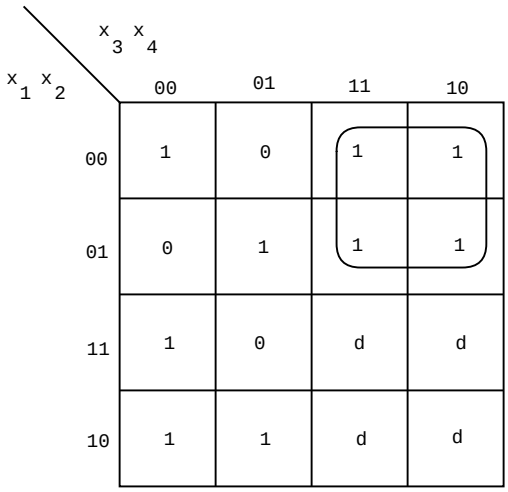
Le *keywords* risultano essere per questo tipo di ottimizzazioni {*fattorizzazione; scomposizione semplice disgiuntiva*}.

`Errori e controesempi`: non sono implicanti (o implicanti vantaggiosi) quelli della figura seguente che riporta raccoglimenti poco accorti od erronei.

errori da evitare:



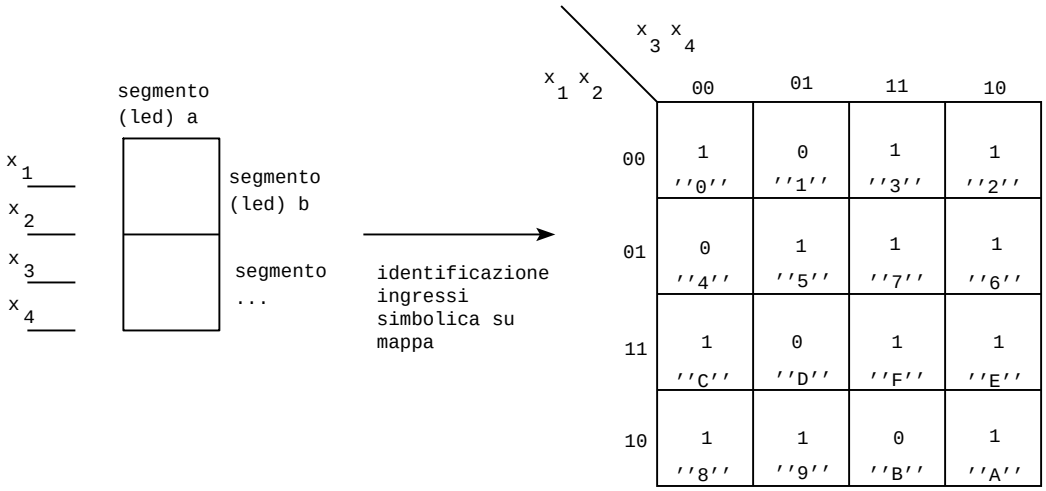
dalla definizione segue che un implicante vale 1 per un numero di vertici pari ad una potenza di due; non e' infatti possibile individuare un solo termine prodotto con un qualsiasi si voglia numero di letterali (minori di quattro in questo caso) che identifichi un siffatto ONSet



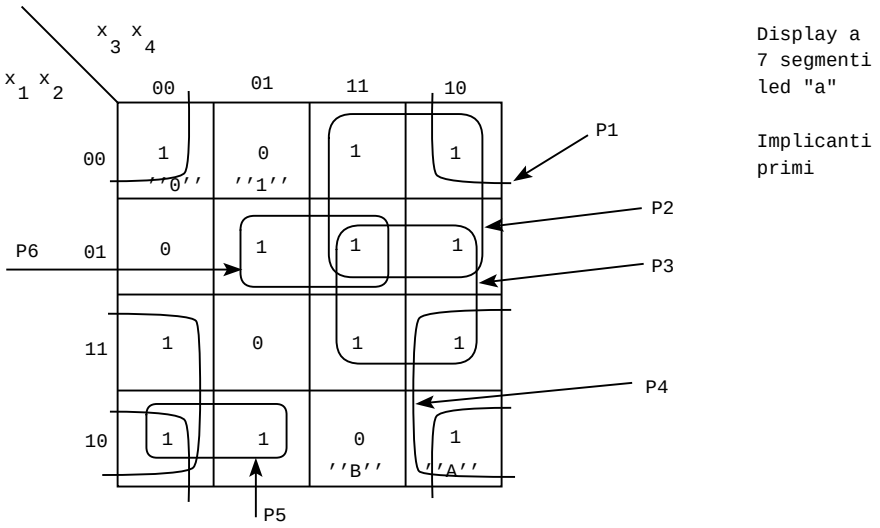
in questo caso conviene generare l'implicante primo x3 sostituendo per la ricerca della copertura ottima degli 1 ai don't care d. Infatti un solo letterale e' senz'altro preferibile ai due del termine prodotto x3*(not(x1)) evidenziato nella mappa a lato

4.3 Esercizio: il display a 7 segmenti

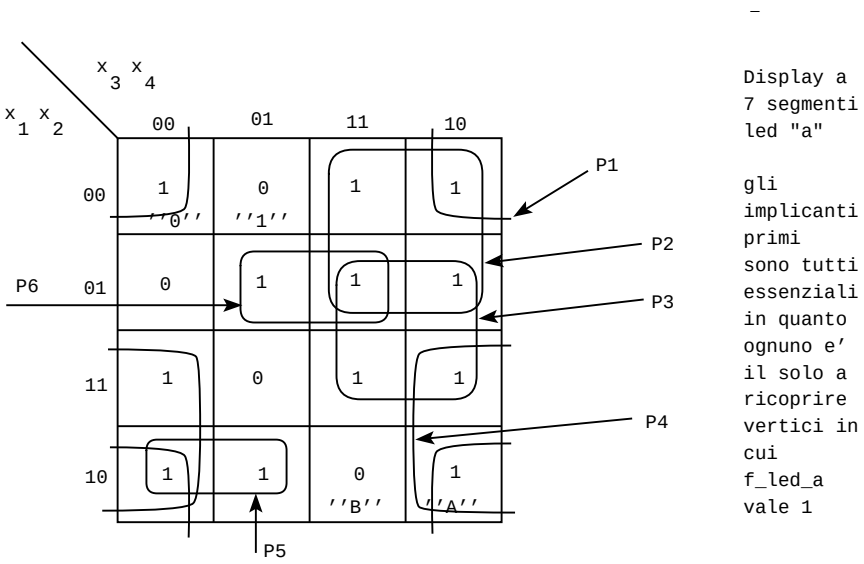
Vi sono sette funzioni di uscita (i segnali di attivazione dei segmenti) che vengono identificate ciascuna con un'etichetta diversa (ad esempio $z_1, z_2, \ldots$ ma in realtà è meglio dare loro nomi significativi come è possibile osservare nell'illustrazione seguente.



Fase uno: ricerca degli implicanti primi



Fase due: individuazione degli implicanti primi essenziali (generalmente un sottinsieme)



In questo caso i due insiemi coincidono e non è nemmeno necessario identificare un insieme di cardinalità minima di implicanti primi da aggiungere a quelli essenziali contenente il minor numero possibile di letterali in quanto gli implicanti essenziali "coprono" già tutta la regione ON della funzione led_a.

Fondamentale è riuscire ad esprimere i termini prodotto partendo dalla rappresentazione a mappe: per fare questo bisogna individuare per ogni raggruppamento (leggi implicante primo) quali sono i letterali che non variano al variare dei vertici del politopo in questo.

P1=$\overline{x_2} \cdot \overline{x_4}$

P2=$\overline{x_1} \cdot x_3$

P3=$x_2 \cdot x_3$

P4=$x_1 \cdot \overline{x_4}$

P5=$x_1 \cdot \overline{x_2} \cdot \overline{x_3}$

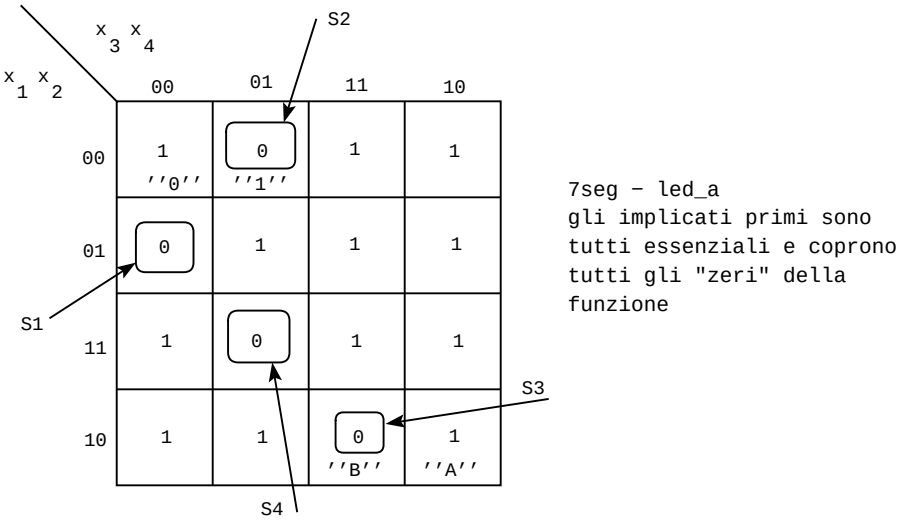
P6=$\overline{x_1} \cdot x_2 \cdot x_4$

OSS: in questo esempio $n = 4$ variabili di ingresso ed i primi 4 implicanti contengono $k = 2$ letterali, per cui avranno cardinalità $2^{(n-k)} = 2^{(4-2)} = 4$ corrispondente al ricoprimento di 4 vertici adiacenti del politopo.

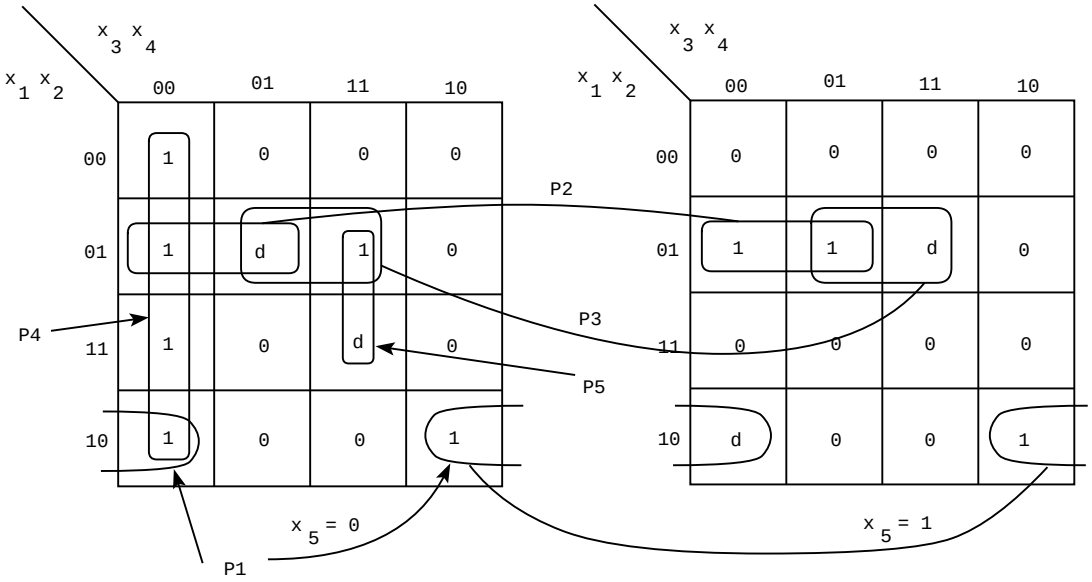
f_led_a=$\overline{x_2}\cdot\overline{x_4}+\overline{x_1}\cdot x_3+x_2\cdot x_3+x_1\cdot\overline{x_4}+x_1\cdot\overline{x_2}\cdot\overline{x_3}+\overline{x_1}\cdot x_2\cdot x_4$

Analizzando la copertura degli zeri si ha che in questo caso gli implicati primi sono tutti essenziali e ricoprono completamente i valori off della funzione, per cui si avrà

f_led_a = $(x_1+x_2+x_3+\overline{x_4})\cdot(x_1+\overline{x_2}+x_3+x_4)\cdot(\overline{x_1}+\overline{x_2}+x_3+\overline{x_4})\cdot(\overline{x_1}+x_2+\overline{x_3}+\overline{x_4})$



4.4 Esempio: una funzione di 5 variabili definita per ONSet, OFFSet, DCSet



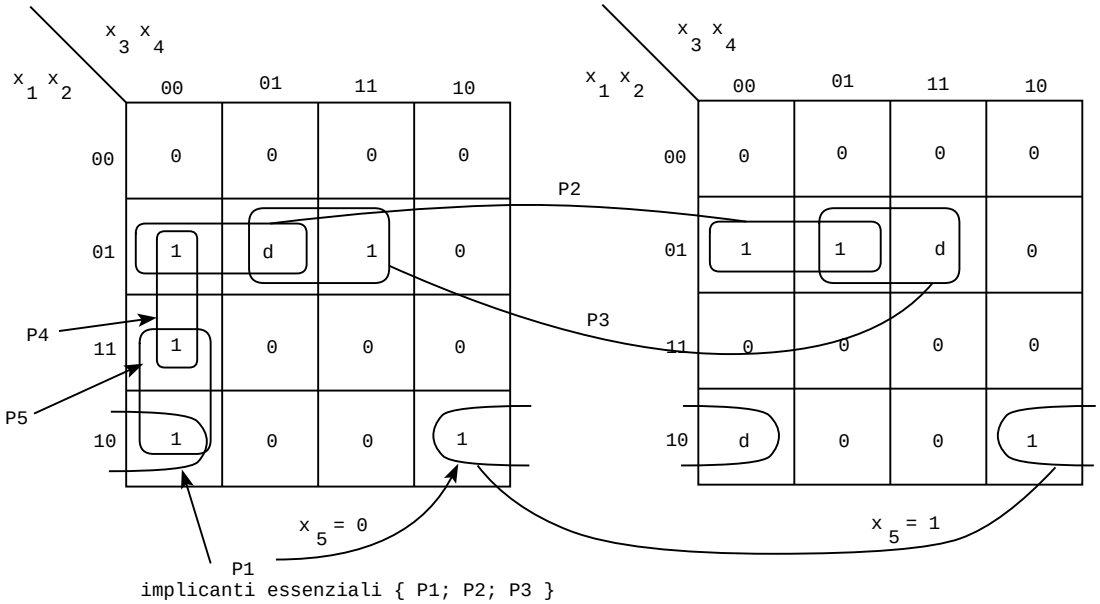
P5 e P3 sono implicanti primi non essenziali.
Implicanti essenziali { P1; P2; P4}; non ricoprono tutto l'ONSet ma è necessario effettuare una scelta tra P5 e P3 che ricadrà su P3.

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

Nel caso analizzato tutti gli implicanti sono essenziali ad eccezione di P5 e P3, mentre nella seconda variante dell'esercizio notiamo che è necessario effettuare una scelta per la copertura, ma tra opzioni equivalenti in termini di costo.

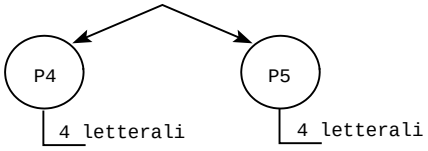
In questo caso l'espressione risulta essere:

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

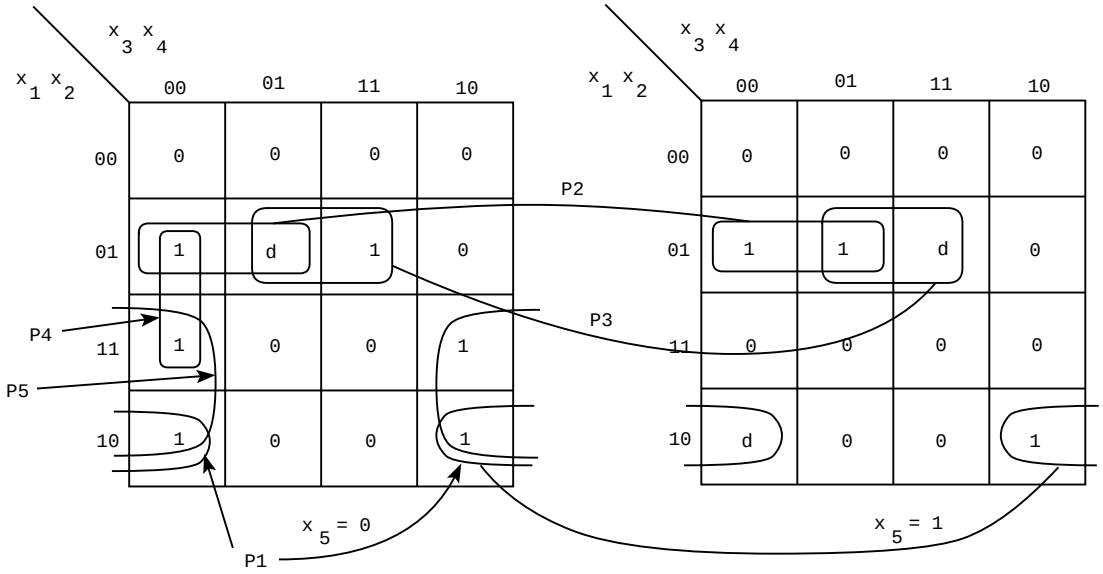


implicanti essenziali { P1; P2; P3 }
per copertura minima a minor costo e' possibile aggiungere
equivalentemente P4 o P5

Branch&Bound per la valutazione dei costi in termini di letterali
(non permette di identificare l'ottimo (costi equivalenti a 6 letterali)



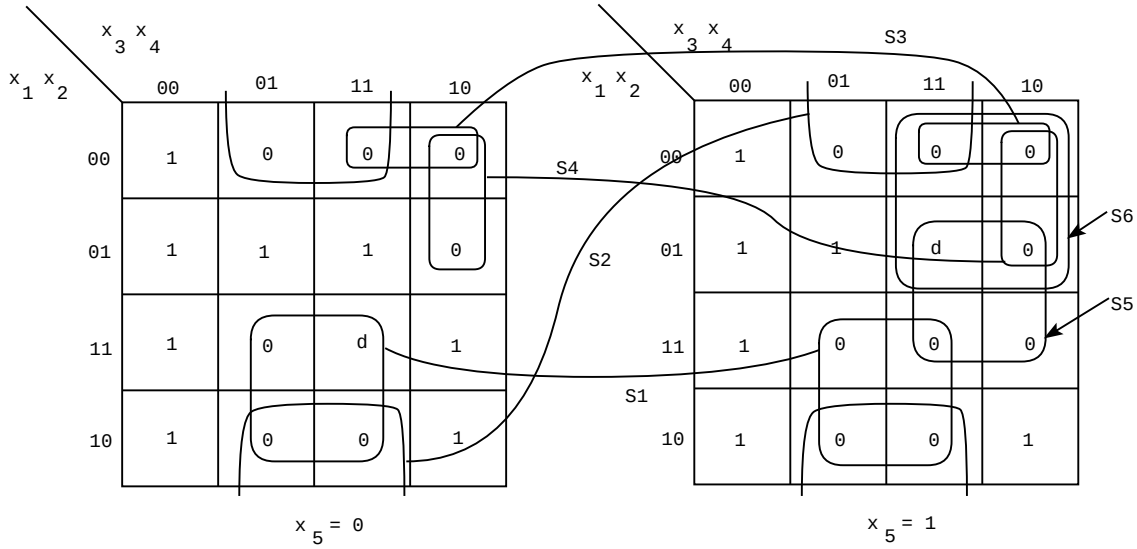
Variante terza SP:
$f = \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_4 + x_1 \cdot \overline{x_4} \cdot \overline{x_5} + x_1 \cdot \overline{x_2} \cdot \overline{x_4}$



implicanti essenziali { P1; P2; P3; P5 }; questi realizzano una copertura della funzione e non e' necessario utilizzare P4 che risulta essere semplicemente un implicante primo escluso dall'insieme degli implicanti scelti per la copertura ottima

Ricopertura a prodotto di somme in cinque variabili:

$f = (\overline{x_1} + \overline{x_4}) \cdot (x_2 + \overline{x_4}) \cdot (x_1 + \overline{x_3} + x_4) \cdot (\overline{x_2} + \overline{x_3} + \overline{x_5})$



implicati essenziali { S1; S2; S4; S5} coprono tutto l'OFFSet: non e' necessaria la ricerca della copertura ottima

4.5 Operatori NAND e NOR

Proposizione {AND; NOT} e {OR; NOT} sono insiemi funzionalmente completi (FC).

L'insieme di operatori logici {AND; NOT} risulta essere funzionalmente completo in quanto è possibile realizzare con soli elementi dell'insieme anche l'operatore OR (involuzione e De Morgan):

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

e lo è anche {OR; NOT} in quanto la realizzazione di un AND si effettua come riportato di seguito per le stesse proprietà:

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

{NAND $\uparrow$ } è FC, infatti {NAND}$\rightarrow${AND; NOT}

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

{NOR $\downarrow$ } è FC infatti {NOR}$\rightarrow${OR; NOT}

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

4.6 Regole di sintesi NAND a due livelli

Nel caso di reti a due livelli per le quali si abbia a disposizione un'espressione SP è possibile applicare le seguenti regole per pervenire ad un'espressione a due livelli in cui compaiano solamente operatori logici NAND:

- partire da un'espressione di tipo SP, SPS, SPSP...
- introdurre tutte le parentesi non indicate esplicitamente
- sostituire il simbolo del NAND ad ogni simbolo $\cdot$ di prodotto logico
- sostituire il simbolo del NAND ad ogni simbolo $+$ di somma logica e *complementare variabili e costanti* che tale simbolo affianca senza interposizione di parentesi

Questo metodo algoritmico è giustificabile grazie all'applicazione di alcune regole dell'algebra quali la proprietà dell' *involuzione* e la legge di De Morgan $\overline{A+B}=\overline{A}\cdot\overline{B}$

$$z_{SP}=(x_1\cdot x_2)+(x_3\cdot x_4)$$

involuzione

$$z_{SP}=\overline{\overline{(x_1\cdot x_2)+(x_3\cdot x_4)}}$$

De Morgan

$$z=\overline{\overline{(x_1\cdot x_2)}\cdot\overline{(x_3\cdot x_4)}}$$

e per la definizione di NAND si ha infine

$$z_{NAND} = (x_1 \uparrow x_2) \uparrow (x_3 \uparrow x_4)$$

Ogni costante o variabile che compaia a livello *uno* del prodotto logico finale deve essere complementata in quanto il simbolo può essere visto ad esempio per la proprietà dell'idempotenza come un prodotto logico con se stesso.

In tal caso questo prodotto logico dovrebbe esere convertito in un NAND e dalla tabella di verità o dalle dimostrazioni di completezza presentate in precedenza si osserva l'equivalenza con un NOT, da cui deriva la necessità di effettuare l'operazione di complemento.

4.7 ESERCIZIO: il sommatore

Progettare utilizzando porte NAND un circuito logico combinatorio con funzione di sommatore ad ingressi:

- $A(a_1, a_0)$
- $B(b_1, b_0)$
- $C(c)$

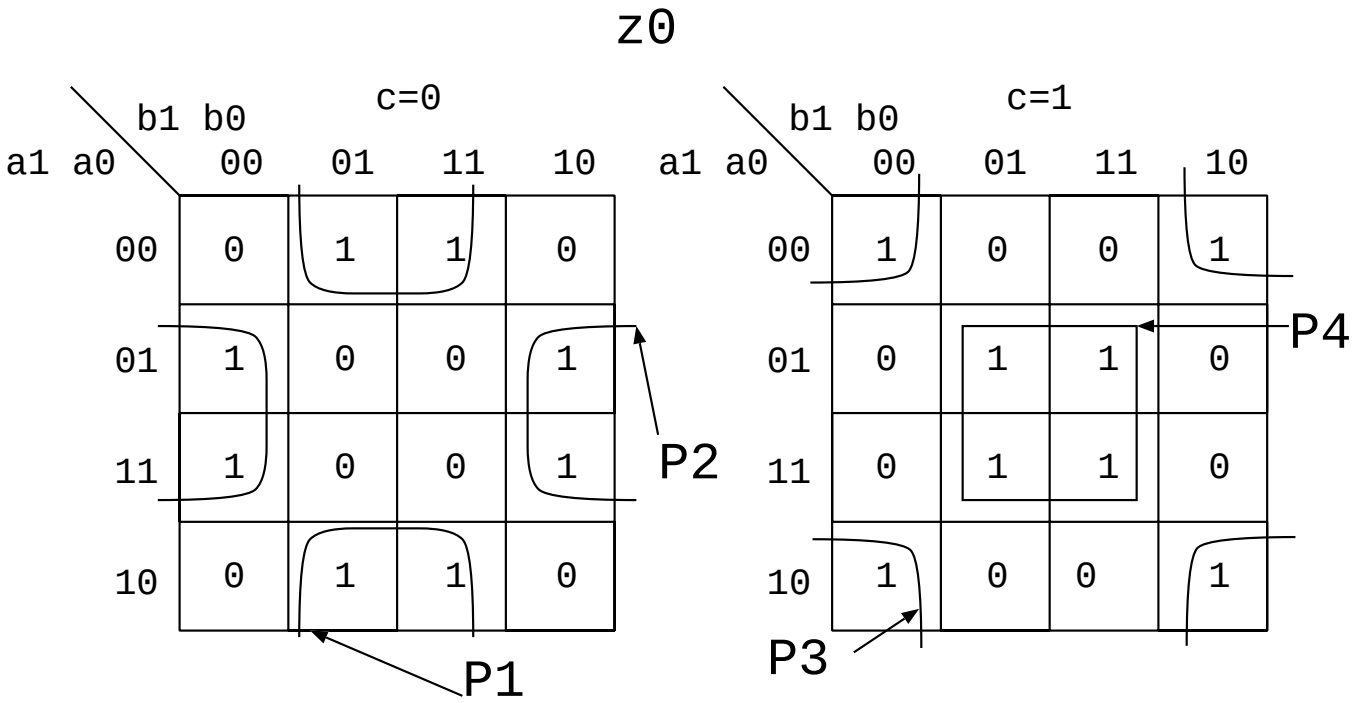
Ingressi ed uscite rappresentano numeri binari e la funzione di somma realizzata è la seguente:

$$Z = A + B + C$$

Soluzione:
Il diverso peso dei bit degli addendi determina il valore assunto dai bit dell'uscita Z. I numeri considerati sono interi positivi ed il valore minimo che l'uscita assumerà sarà 0 per una configurazione di ingresso $A(a_1, a_0) = B(b_1, b_0) = (0,0)$; $C(c) = (0)$. Il valore massimo assunto si verificherà per la configurazione di ingresso di tutti "1":
$A(a_1, a_0) = B(b_1, b_0) = (1,1); C(c) = (1)$. Sarà $A_{10} = B_{10} = 3$; $C_{10} = 1$ e dunque $Z_{10} = 7$. Il numero di segnali binari necessari a codificare tale valore risulta essere $\lceil log_2 7 \rceil = 3$. Si presenta di seguito la tabella di definizione dei tre segnali di uscita.

a1	a0	b1	b0	c	z2	z1	z0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1
0	0	1	1	0	0	1	1
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	0	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	0	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	0	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	0	1	0	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	0
1	1	0	0	0	0	1	1
1	1	0	0	1	1	0	0
1	1	0	1	0	1	0	0
1	1	0	1	1	1	0	1
1	1	1	0	0	1	0	1
1	1	1	0	1	1	1	0
1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	1

Le mappe di Karnaugh del bit meno significativo dell'uscita (z_0) è riportata di seguito



I termini prodotto sono rispettivamente

- $P1 = \overline{a_0}\, b_0\, \overline{c}$
- $P2 = a_0\, \overline{b_0}\, \overline{c}$
- $P3 = \overline{a_0}\, \overline{b_0}\, c$
- $P4 = a_0\, b_0\, c$

e il bit meno significativo della somma di uscita vale

$$z_0 = P1 + P2 + P3 + P4$$

sostituendo i termini delle espressioni prodotto introducendo le parentesi ed i prodotti non indicati esplicitamente si perviene a:

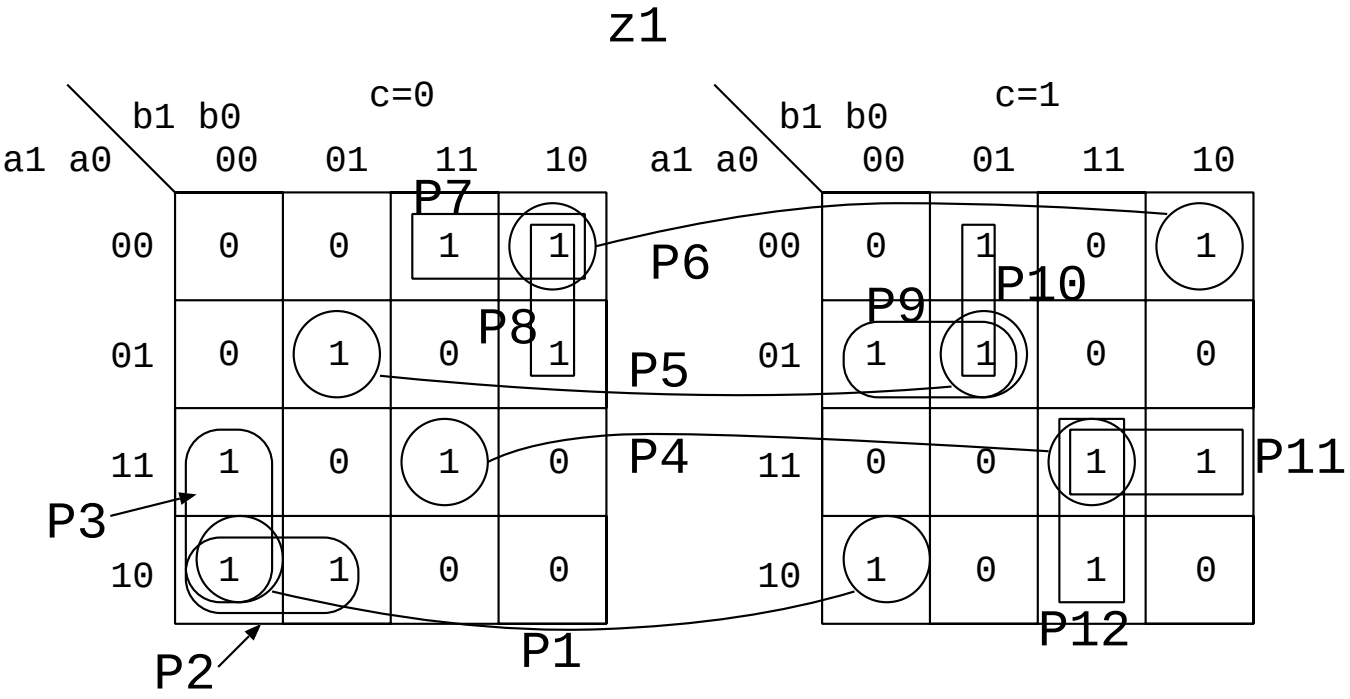
$$z_0 = (\overline{a_0} \cdot \overline{b_0} \cdot \overline{c}) + (a_0 \cdot \overline{b_0} \cdot \overline{c}) + (\overline{a_0} \cdot \overline{b_0} \cdot c) + (a_0 \cdot b_0 \cdot c)$$

proseguendo si sostituisce il simbolo del nand ($\uparrow$) ad ogni simbolo di prodotto logico ed ad ogni simbolo di somma logica a livello uno, effettuando i relativi complementi delle variabili isolate, si giunge all'espressione di z_0 a NAND:

$$z_0 = (\overline{a_0} \uparrow \overline{b_0} \uparrow \overline{c}) \uparrow (a_0 \uparrow \overline{b_0} \uparrow \overline{c}) \uparrow (\overline{a_0} \uparrow \overline{b_0} \uparrow c) \uparrow (a_0 \uparrow b_0 \uparrow c)$$

Autore: Amos Tibaldi e-mail address: tibaldi@CE.UniPr.IT

Si procede dunque con la trasposizione su mappa dei valori del bit di posizione uno (peso decimale due) nella seguente immagine.



Si può notare che tutti gli implicanti sono essenziali e verranno inseriti tutti nell'espressione del secondo bit della funzione di uscita. In seguito, se non diversamente specificato, i termini somma *Sn* ed i termini prodotto *Pn* saranno elencati nelle espressioni delle funzioni nell'ordine numerico che li caratterizza sulle mappe.

$$z_1 = (a_1 \cdot \overline{a_0} \cdot \overline{b_1} \cdot \overline{b_0}) + (a_1 \cdot \overline{a_0} \cdot \overline{b_1} \cdot \overline{c}) + (a_1 \cdot \overline{b_1} \cdot \overline{b_0} \cdot \overline{c}) + (a_1 \cdot a_0 \cdot b_1 \cdot b_0) +$$

$$(\overline{a_1} \cdot a_0 \cdot \overline{b_1} \cdot b_0) + (\overline{a_1} \cdot \overline{a_0} \cdot b_1 \cdot \overline{b_0}) + (\overline{a_1} \cdot \overline{a_0} \cdot b_1 \cdot \overline{c}) + (\overline{a_1} \cdot b_1 \cdot \overline{b_0} \cdot \overline{c}) +$$

$$(\overline{a_1} \cdot a_0 \cdot \overline{b_1} \cdot c) + (\overline{a_1} \cdot \overline{b_1} \cdot b_0 \cdot c) + (a_1 \cdot a_0 \cdot b_1 \cdot c) + (a_1 \cdot b_1 \cdot b_0 \cdot c)$$

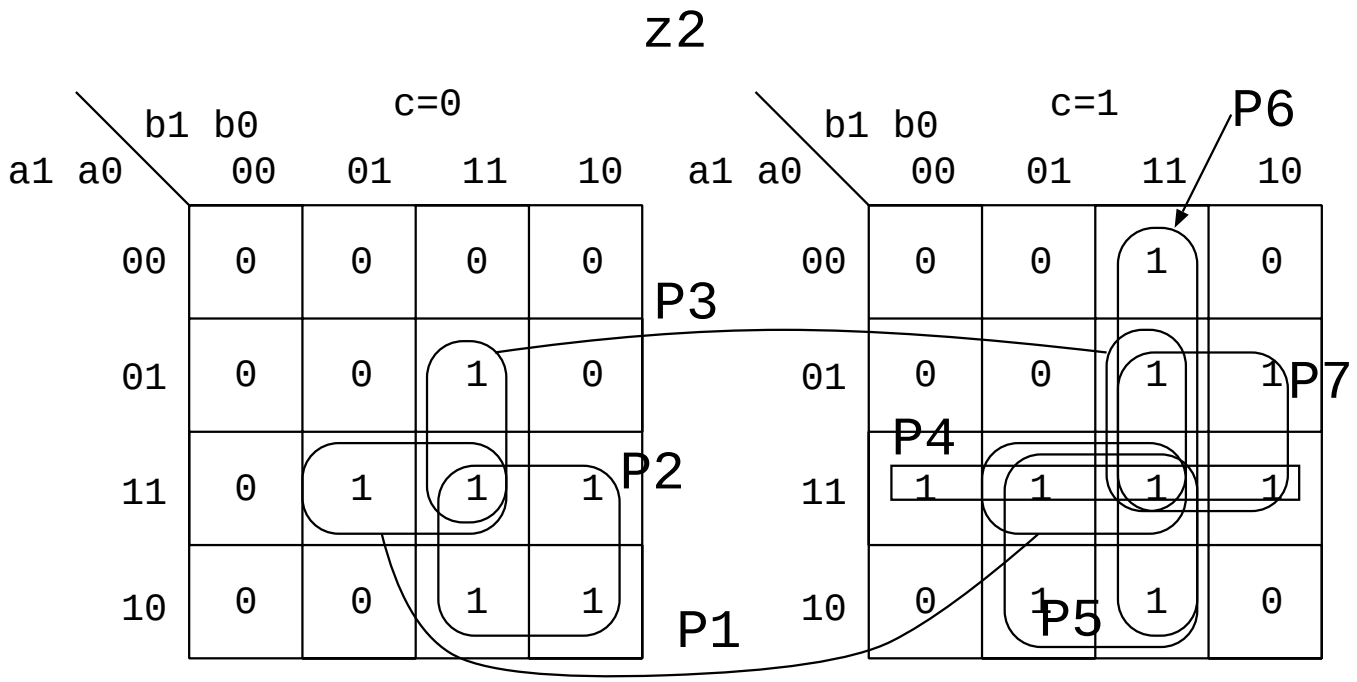
Applicando due regole di conversione si ottiene l'espressione a NAND di z_1:

$$z_1 = (a_1 \uparrow \overline{a_0} \uparrow \overline{b_1} \uparrow \overline{b_0}) \uparrow (a_1 \uparrow \overline{a_0} \uparrow \overline{b_1} \uparrow \overline{c}) \uparrow (a_1 \uparrow \overline{b_1} \uparrow \overline{b_0} \uparrow \overline{c}) \uparrow (a_1 \uparrow a_0 \uparrow b_1 \uparrow b_0) \uparrow$$

$$(\overline{a_1} \uparrow a_0 \uparrow \overline{b_1} \uparrow b_0) \uparrow (\overline{a_1} \uparrow \overline{a_0} \uparrow b_1 \uparrow \overline{b_0}) \uparrow (\overline{a_1} \uparrow \overline{a_0} \uparrow b_1 \uparrow \overline{c}) \uparrow (\overline{a_1} \uparrow b_1 \uparrow \overline{b_0} \uparrow \overline{c}) \uparrow$$

$$(\overline{a_1} \uparrow a_0 \uparrow \overline{b_1} \uparrow c) \uparrow (\overline{a_1} \uparrow \overline{b_1} \uparrow b_0 \uparrow c) \uparrow (a_1 \uparrow a_0 \uparrow b_1 \uparrow c) \uparrow (a_1 \uparrow b_1 \uparrow b_0 \uparrow c)$$

Per la realizzazione a NAND della terza componente dell'uscita si è evidenziato direttamente sulla mappa il più piccolo insieme di implicanti primi irridondanti, contenenti ovviamente tutti gli implicanti primi essenziali.



Non sono stati evidenziati implicanti non essenziali e ridondanti

L'espressione SP ottima a due livelli di z_2 è

$$z_2 = (a_1 \cdot a_0 \cdot b_0) + (a_1 \cdot b_1 \cdot \overline{c}) + (a_0 \cdot b_1 \cdot b_0) + (a_1 \cdot a_0 \cdot c) + (a_1 \cdot b_0 \cdot c) + (b_1 \cdot b_0 \cdot c) + (a_0 \cdot b_1 \cdot c)$$

e l'espressione a NAND risulta dunque:

$$z_2 = (a_1 \uparrow a_0 \uparrow b_0) \uparrow (a_1 \uparrow b_1 \uparrow \overline{c}) \uparrow (a_0 \uparrow b_1 \uparrow b_0) \uparrow$$
$$(a_1 \uparrow a_0 \uparrow c) \uparrow (a_1 \uparrow b_0 \uparrow c) \uparrow (b_1 \uparrow b_0 \uparrow c) \uparrow (a_0 \uparrow b_1 \uparrow c)$$

4.8 ESERCIZIO: algebra booleana

Verificare la seguente equivalenza:

$$Y\overline{Z} + \overline{X}YZ + XYZ + X\overline{Y}Z = Y + XZ$$

Soluzione:
Per la proprietà dell'idempotenza vale che $A = A + A$ ed anche dunque $XYZ = XYZ + XYZ$. Applicando questo al membro sinistro dell'equazione si ottiene:

$$Y\overline{Z} + \overline{X}YZ + XYZ + XYZ + X\overline{Y}Z = Y + XZ$$

Per la proprietà distributiva vale

$$A \cdot B + A \cdot C = A \cdot (B + C)$$

dunque se $A = YZ$ e $B = X$ si ottiene che

$$\overline{X}YZ + XYZ = YZ \cdot (X + \overline{X})$$

e per la proprietà dell'inverso risulta

$$YZ \cdot (X + \overline{X}) = YZ \cdot (1)$$

infine per l'esistenza dell'elemento identità per l'operazione di prodotto logico si avrà

$$YZ \cdot (1) = YZ$$

Sostituendo nell'espressione iniziale

$$Y\overline{Z} + YZ + XYZ + X\overline{Y}Z = Y + XZ$$

applicando a questa la proprietà distributiva nella versione corretta si perviene a

$$Y \cdot (Z + \overline{Z}) + XYZ + X\overline{Y}Z = Y + XZ$$

inverso

$$Y \cdot (1) + XYZ + X\overline{Y}Z = Y + XZ$$

elemento identità

$$Y + XYZ + X\overline{Y}Z = Y + XZ$$

commutativa

$$Y + ZXY + ZX\overline{Y} = Y + ZX$$

distributiva

$$Y + ZX(Y + \overline{Y}) = Y + ZX$$

inverso

$$Y + ZX \cdot (1) = Y + ZX$$

ed applicando per ultima la proprietà dell'identità si verifica l'uguaglianza delle due espressioni:

$$Y + Z \cdot X = Y + Z \cdot X$$

4.9 ESERCIZIO: sintesi ottima e coperture

Data la seguente funzione booleana di 5 variabili $f(x_4, x_3, x_2, x_1, x_0)$ effettuare una sintesi a due livelli prossima a quella ottima, con progetto a mappe ed espressioni a NAND e a NOR.

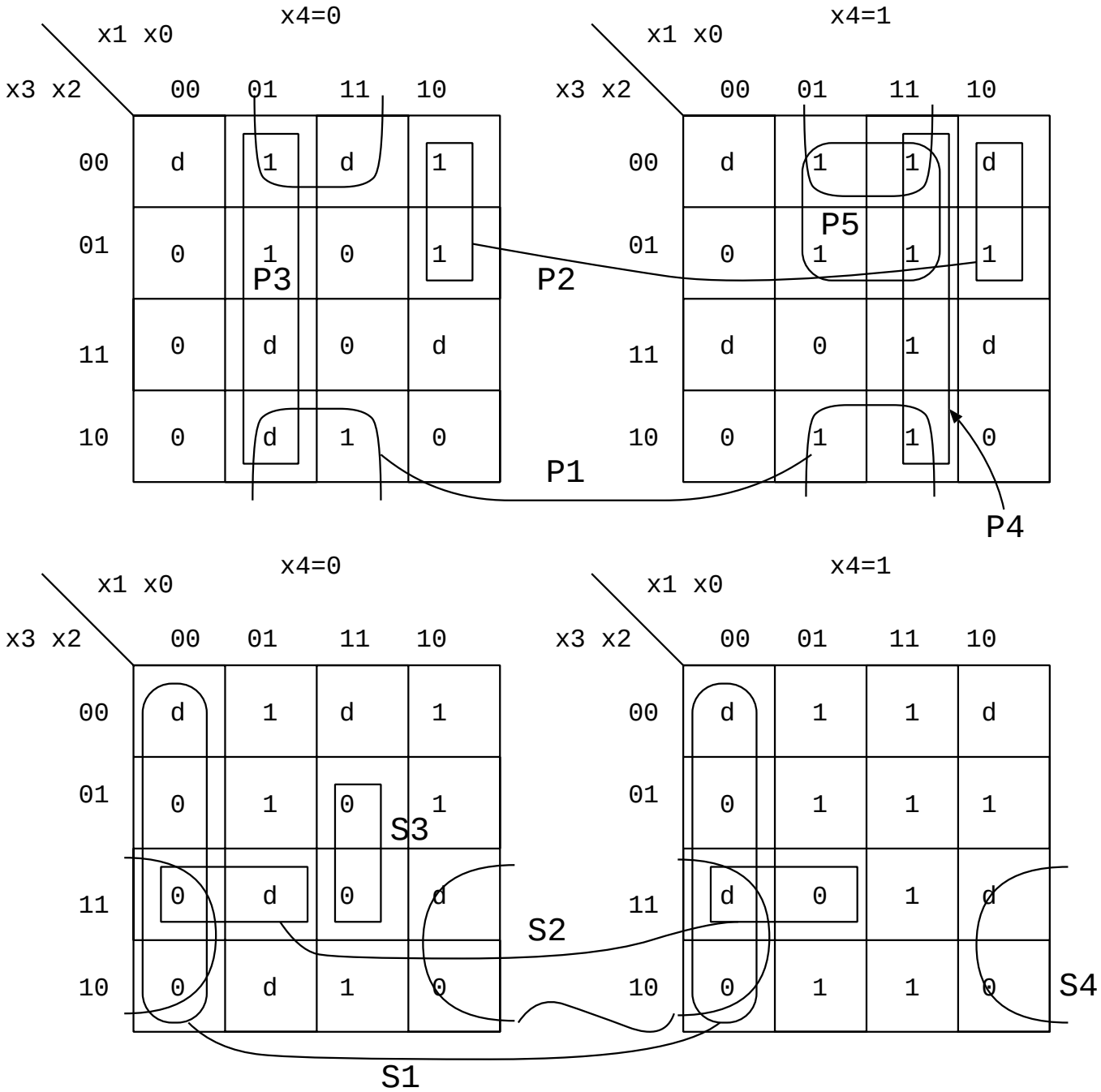
Si procede alla mappa per la copertura dell'ONSet per giungere a delle espressioni somma, alle quali applicare poi le regole per la sintesi di reti a NAND. La mappa di copertura dell'OFFSet invece permette di ottenere espressioni prodotto dalle quali è possibile pervenire ad espressioni della funzione contenenti solamente porte NOR.

Autore: Amos Tibaldi e-mail address: tibaldi@CE.UniPr.IT

È possibile ottenere coperture subottime anche con algoritmi ad esclusione: se possibile si individuano inizialmente elementi che debbano essere necessariamente coperti da raggruppamenti essenziali e si cerca di estendere il più possibile il raggruppamento ricoprente. Tra gli elementi non ancora coperti si prosegue iterando quanto sopra fino a che non esistano più elementi che possano individuare tali ricoperture essenziali. Rimangono elementi che possono appartenere a più raggruppamenti primi e si completa la copertura *irridondante* con scelte che tengano conto del costo della rete.

Nell'esercizio ad esempio si evidenzia come alcuni implicanti primi ed anche implicati primi non siano stati evidenziati su mappa.

Si osserva per esempio che nella copertura dell'OFFSet in questo caso non si è scelto l'implicato primo $(\overline{x_3}+\overline{x_2}+x_4)$ in quanto S3 è essenziale ed altrettanto è S2 che copre già gli zeri rimanenti della terza riga nel sottopolitopo $x_4 = 0$.



$$z_{SP} = (\overline{x_2} \cdot x_0) + (\overline{x_3} \cdot x_1 \cdot \overline{x_0}) +$$
$$(\overline{x_1} \cdot x_0 \cdot \overline{x_4}) + (x_1 \cdot x_0 \cdot x_4) + (\overline{x_3} \cdot x_0 \cdot x_4)$$

Dall'espressione SP è possibile pervenire all'espressione a NAND:

$$z_{NAND} = (\overline{x_2} \uparrow x_0) \uparrow (\overline{x_3} \uparrow x_1 \uparrow \overline{x_0}) \uparrow$$

$$(\overline{x_1} \uparrow x_0 \uparrow \overline{x_4}) \uparrow (x_1 \uparrow x_0 \uparrow x_4) \uparrow (\overline{x_3} \uparrow x_0 \uparrow x_4)$$

L'espressione della funzione come prodotto di somme è la seguente:

$$z_{PS} = (x_1 + x_0) \cdot (\overline{x_3} + \overline{x_2} + x_1) \cdot (\overline{x_2} + \overline{x_1} + \overline{x_0} + x_4) \cdot (\overline{x_3} + x_0)$$

Le regole per passare da una qualsiasi espressione PS contenente tutte le parentesi e gli operatori anche in genere non specificati ad una espressione NOR a due livelli sono le seguenti:

- sostituire ad ogni simbolo di somma logica $+$ il simbolo di NOR ($\downarrow$)
- sostituire ad ogni simbolo di prodotto il simbolo di NOR complementando costanti o variabili che si presentano singolarmente in ingresso a questo senza parentesi

Nel nostro caso nessun termine isolato si presenta come ingresso singolo al NOR di livello logico uno:

$$z_{NOR} = (x_1 \downarrow x_0) \downarrow (\overline{x_3} \downarrow \overline{x_2} \downarrow x_1) \downarrow (\overline{x_2} \downarrow \overline{x_1} \downarrow \overline{x_0} \downarrow x_4) \downarrow (\overline{x_3} \downarrow x_0)$$

Si osserva inoltre che la scelta delle assegnazioni del valore ai *don't care* (0 od 1) non è sempre banale quando non si procede con l'algoritmo automatico di settaggio o reset di insieme, e con le indicazioni fornite risulta talvolta inutile cercare di assegnare un valore ad un riferimento d del politopo.

Conviene:

- partire da un elemento isolato che debba essere racchiuso in un raggruppamento (meglio se essenziale)
- cercare di espandere il più possibile il raggruppamento che lo ricopre utilizzando gli elementi di indifferenza
- escludere dalla ricerca futura elementi già coperti da implicanti o implicati
- cercare di utilizzare le alternative migliori in termini di dimensioni del raggruppamento, quelle cioè con un minor numero di letterali
- iterare fino ad aver coperto tutte le configurazioni specificate.

Segue esercitazione su reti sequenziali [1]

[1] La corrente versione di questa esercitazione non è ancora definitiva; per informazioni o per riportare errori individuati: tibaldi@ce.unipr.it