Reti combinatorie Ottimizzazione con l'ausilio di SIS

A cura di:
Ph.D., Ing. **Alessandra De Benedictis**, alessandra.debenedictis@unina.it

Metodi esatti

Metodo di Quine Mc Cluskey per funzioni a una uscita

Esercizio 1

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

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

  ONSet={1,4,5,6,7,9,11,14,15}; DCSet=∅

Soluzione:

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

ONSet={0001,0100,0101,0110,0111,1001,1011,1110,1111}

- che dà origine alla seguente partizione:

ONSet={{0001,0100}{0101,0110,1001}{0111,1011,1110}{1111}}

I Fase:ricerca degli implicanti primi

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	✓

$\{m_1..m_n\}$	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	–	✓

$\{m_1..m_n\}$	x	y	z	v	
4, 5, 6, 7	0	1	–	–	*E*
6, 7, 14, 15	–	1	1	–	*F*

0-01 A

-001 B

10-1 C

1-11 D

01- - E

-11- F

II Fase:copertura (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

C(F)= {E,F}

	m1	m9	m11
A	X		
B	X	X	
C		X	X
D			X

II Fase:copertura (2/2)

	m1	m9	m11
~~A~~	~~X~~		
B	X	X	
C		X	X
~~D~~			~~X~~

	m1	m9	m11
B	X	X	
C		X	X

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

F=E+F+B+C=!xy+yz+!y!zv+x!yv

Soluzione con SIS(1/3)

- Creare il file monof.blif contenente la definizione della funzione da minimizzare:

.model esempio1

.inputs x y z v

.outputs f

Specifico il nome della funzione, i suoi input e i suoi output

.names x y z v f

0001 1

0100 1

0101 1

0110 1

0111 1

1001 1

1011 1

1110 1

1111 1

.end

Con la parola chiave **.names** specifico i mintermini della funzione con le relative uscite

Soluzione con SIS(2/3)

```
c:\sis-1.2\ESERCI~1>sis
UC Berkeley, SIS 1.3 (compiled Jun 11 2003)
sis> read_blif monof.blif
sis> write_blif
.model esempio1
.inputs x y z v
.outputs f
.names x y z v f
1111 1
0111 1
1011 1
0101 1
1001 1
0001 1
1110 1
0110 1
0100 1
.end
sis> write_eqn
INORDER = x y z v;
OUTORDER = f;
f = !x*y*!z*!v + !x*y*z*!v + x*y*z*!v + !x*!y*!z*v + x*!y*!z*v + !x*y*!z*v + x*!

y*z*v + !x*y*z*v + x*y*z*v;
sis> print_stats
esempio1        pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=  36
```

Col comando
>**read_blif** ***nome_file***
è possibile caricare una descrizione di un circuito e con

>**write_blif**

è possibile visualizzare l'ultimo circuito caricato

Il comando **write_eqn** visualizza le equazioni del componente

Il comando **print_stats** visualizza alcune informazioni sul componente, tra cui il costo in termini di letterali

Il modello ha un costo di 36 letterali poiché è descritto da 9 mintermini di 4 variabili ognuno

Soluzione con SIS(3/3)

```
sis> simplify
sis> write_eqn
INORDER = x y z v;
OUTORDER = f;
f = !y*!z*v + x*z*v + y*z + !x*y;
sis> print_stats
esempio1        pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=  10
sis> write_blif
.model esempio1
.inputs x y z v
.outputs f
.names x y z v f
01-- 1
-11- 1
1-11 1
-001 1
.end
sis>
```

Il comando **simplify** effettua la minimizzazione di una rete combinatoria sfruttando un'implementazione ottimizzata del metodo di McCluskey (algoritmo ESPRESSO)

La funzione è ora descritta con 4 prodotti il cui costo è sceso a 10 letterali.

Gli implicanti determinati da simplify concordano quasi completamente con il risultato ottenuto manualmente (**F=!y!zv+x!yv+yz+!xy**): la differenza riguarda l'implicante primo essenziale D (1-11) che è stato selezionato invece dell'implicante C (10-1). Il loro costo in termini di letterali è però identico quindi la loro scelta in fase di copertura è indifferente.

Esercizio 2

(tratto da “Reti Logiche” di Bolchini,Brandolese,Salice,Sciuto)

- 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}}

I Fase:ricerca degli implicanti primi

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)

01-- A

--11 B

-1-1 C

-11- D

1-1- E

II Fase:copertura

	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

Soluzione con SIS(1/2)

- Creare il file monofDC.blif contenente la definizione della funzione da minimizzare:

.model esempio1DC
.inputs x y z v
.outputs f
.names x y z v f
0100 1
1010 1
1011 1
1101 1
1110 1
1111 1
…..

….
.exdc
.inputs x y z v
.outputs f
.names x y z v f
0011 1
0101 1
0110 1
0111 1

.end

Il DC-set viene descritto come l'ON-set ma viene identificato perché posto dopo la parola chiave **.exdc**

Soluzione con SIS(2/2)

```
sis> read_blif monofDC.blif
sis> write_eqn
INORDER = x y z v;
OUTORDER = f;
f = !x*y*!z*!v + x*!y*z*!v + x*y*z*!v + x*y*!z*v + x*!y*z*v + x*y*z*v;

Don't care:
INORDER = x y z v;
OUTORDER = f;
f = !x*y*z*!v + !x*y*!z*v + !x*!y*z*v + !x*y*z*v;

sis> print_stats
esempio1DC      pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=  24
sis> simplify
sis> write_eqn
INORDER = x y z v;
OUTORDER = f;
f = !x*y*!z*!v + x*y*v + x*z;

Don't care:
INORDER = x y z v;
OUTORDER = f;
f = !x*y*z*!v + !x*y*!z*v + !x*!y*z*v + !x*y*z*v;

sis> print_stats
esempio1DC      pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=   9
sis> full_simplify
sis> write_eqn
INORDER = x y z v;
OUTORDER = f;
f = y*v + x*z + !x*y;

Don't care:
INORDER = x y z v;
OUTORDER = f;
f = !x*y*z*!v + !x*y*!z*v + !x*!y*z*v + !x*y*z*v;

sis> print_stats
esempio1DC      pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=   6
```

Esercizio 3

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

- 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=Ø

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}}

I Fase:ricerca degli implicanti primi

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.

II Fase:copertura (1/2)

	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	

	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	

C(f)={P1,P3}

II Fase:copertura (2/2)

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

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

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

Soluzione con SIS(1/2)

- Creare il file monof2.blif contenente la definizione della funzione da minimizzare:

```
.model esempio1
.inputs a b c d
.outputs f
.names a b c d f
0000 1
0010 1
0100 1
1000 1
0101 1
0110 1
1001 1
0111 1
1101 1
1111 1
.end
```

Soluzione con SIS(2/2)

```
C:\Users\alessandra>sis
UC Berkeley, SIS 1.3 (compiled Jun 11 2003)
sis> read_blif c:\monof2.blif
sis> write_eqn
INORDER = a b c d;
OUTORDER = f;
f = !a*!b*!c*!d + a*!b*!c*!d + !a*b*!c*!d + !a*!b*c*!d + !a*b*c*!d + a*!b*!c*d
 + !a*b*!c*d + a*b*!c*d + !a*b*c*d + a*b*c*d;
sis> print_stats
esempio1        pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=  40
sis> simplify
sis> write_eqn
INORDER = a b c d;
OUTORDER = f;
f = a*!b*!c + !a*!d + b*d;
sis> print_stats
esempio1        pi= 4   po= 1   nodes=  1       latches= 0
lits(sop)=   7
sis>
```

General-purpose Commands

- **help *command***: fornisce informazioni su come utilizzare uno specifico comando.
- **source *scriptfile***: esegue i comandi presenti nel file *scriptfile*
- **history**: fornisce una lista dei comandi dati precedentemente. Con **%n** si ripete l’ n-esimo comando della lista.
- **undo**: annulla l’effetto dell’ultimo comando che ha modificato il circuito
- **quit (o exit)**: per uscire dalla sessione corrente

Metodi esatti

Metodo di Quine Mc Cluskey per funzioni a più uscite

Esercizio 1

(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 ∪ ONSet2 ∪ ONSet3=

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

Soluzione

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	0	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

Soluzione

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	

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

Copertura

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 f0
- Pongo il costo di P0 a 0

C(f0)=∅ C(f1)={P0,P8} C(f2)={P8}

Copertura

	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

Copertura

	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

Copertura

	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

	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}

Copertura

	f0		
	0	2	
P0	X		0
P1		X	**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}

Rappresentazione in .blif e minimizzazione

La descrizione di una funzione multil-uscita in .blif è analoga al caso mono uscita

Si consideri il file **multif.blif** contenente la descrizione della funzione

.model multif1

.inputs x y z v

.outputs f1 f2 f3

.names x y z v f1

0000 1

0010 1

0100 1

0101 1

0110 1

0111 1

.names x y z v f2

0000 1

0100 1

0101 1

0111 1

1010 1

1011 1

1110 1

1111 1

.names x y z v f3

0010 1

0110 1

1010 1

1011 1

1110 1

1111 1

.end

Utilizzando il comando **simplify** o **full_simplify** si ottiene una minimizzazione esatta in cui però le uscite vengono minimizzate separatamente, senza valutare possibili sovrapposizioni di porte. La descrizione del circuito in uscita inoltre è tipicamente su più livelli.

Soluzione con Espresso(1/4)

- La suite di algoritmi Espresso e la sua implementazione in SIS consentono di ottenere una forma minima esatta su 2 livelli di una rete multi-uscita, tenendo conto delle sovrapposizioni.

- Espresso opera su reti logiche rappresentate in formato PLA; un file .blif può essere convertito in formato .pla con i seguenti comandi:

> read_blif <nome_file_blif>

> write_pla <nome_file_pla>

Soluzione con Espresso(2/4)

- dal file multif.blif si ricava la seguente descrizione PLA (file *multif_pla.pla*):

```
.i 4
.o 3
.ilb x y z v
.ob f1 f2 f3
.p 20

0111 100
0101 100
0110 100
0010 100
0100 100
0000 100
1111 010
0111 010
1011 010
0101 010
1110 010
1010 010
0100 010
0000 010
1111 001
1011 001
1110 001
0110 001
1010 001
0010 001
.e
```

Soluzione con Espresso(3/4)

- per eseguire Espresso sulla rete appena definita uscire dal programa SIS nel prompt dei comandi e lanciare il programma **espresso** localizzato nella directory c:\sis-1.2\sis\bin:

```
C:\sis-1.2\ESERCI~1>espresso -s multif_pla.pla
# c:/sis-1.2/sis/bin/espresso.exe -s multif_pla.pla
# UC Berkeley, Espresso Version #2.3, Release date 01/31/88
# PLA is multif_pla.pla with 4 inputs and 3 outputs
# ON-set cost is  c=20(20) in=80 out=20 tot=100
# OFF-set cost is c=7(6) in=14 out=11 tot=25
# DC-set cost is  c=0(0) in=0 out=0 tot=0
# ESPRESSO      Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
.i 4
.o 3
.ilb x y z v
.ob f1 f2 f3
.p 4
01-1 110   P3
0-10 101   P1
0-00 110   P0
1-1- 011   P8
.e
```

Costi in termini di letterali rispettivamente dell'ON-set, OFF-set e DC-set

Costi della soluzione minimizzata

La soluzione trovata da Espresso è la stessa determinata con la procedura manuale:

C(f1)={P3,P1,P0}   f1=!xyv+!xz!v+!x!z!v

C(f2)={P3,P0,P8}   f2=!xyv!x!z!v+xz

C(f3)={P1,P8}   f3= !xz!v+xz

NOTA: Espresso minimizza su 2 livelli e, nel caso di funzioni multiuscita, cerca di trarre vantaggio da sovrapposizioni di implicanti fra le diverse uscite

Soluzione con Espresso(4/4)

```
C:\sis-1.2\ESERCI~1>espresso -t -x multif_pla.pla
# c:/sis-1.2/sis/bin/espresso.exe -t -x multif_pla.pla
# UC Berkeley, Espresso Version #2.3, Release date 01/31/88
# READ          Time was 0.00 sec, cost is c=20(20) in=80 out=20 tot=100
# COMPL         Time was 0.00 sec, cost is c=7(6) in=14 out=11 tot=25
# PLA is multif_pla.pla with 4 inputs and 3 outputs
# ON-set cost is  c=20(20) in=80 out=20 tot=100
# OFF-set cost is c=7(6) in=14 out=11 tot=25
# DC-set cost is  c=0(0) in=0 out=0 tot=0
# SETUP         Time was 0.00 sec, cost is c=20(20) in=80 out=20 tot=100
# EXPAND        Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
# IRRED         Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
# ESSEN         Time was 0.00 sec, cost is c=2(0) in=5 out=4 tot=9
# REDUCE        Time was 0.00 sec, cost is c=2(0) in=6 out=4 tot=10
# EXPAND        Time was 0.00 sec, cost is c=2(0) in=6 out=4 tot=10
# IRRED         Time was 0.00 sec, cost is c=2(0) in=6 out=4 tot=10
# REDUCE_GASP   Time was 0.00 sec, cost is c=2(0) in=6 out=4 tot=10
# EXPAND_GASP   Time was 0.00 sec, cost is c=0(0) in=0 out=0 tot=0
# IRRED_GASP    Time was 0.00 sec, cost is c=2(0) in=6 out=4 tot=10
# ADJUST        Cost is c=4(0) in=11 out=8 tot=19
# MV_REDUCE     Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
# VERIFY        Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
# READ           1 call(s) for 0.00 sec ( 0.0%)
# COMPL          1 call(s) for 0.00 sec ( 0.0%)
# ESSEN          1 call(s) for 0.00 sec ( 0.0%)
# EXPAND         2 call(s) for 0.00 sec ( 0.0%)
# IRRED          2 call(s) for 0.00 sec ( 0.0%)
# REDUCE         1 call(s) for 0.00 sec ( 0.0%)
# EXPAND_GASP    1 call(s) for 0.00 sec ( 0.0%)
# IRRED_GASP     1 call(s) for 0.00 sec ( 0.0%)
# REDUCE_GASP    1 call(s) for 0.00 sec ( 0.0%)
# MV_REDUCE      1 call(s) for 0.00 sec ( 0.0%)
# VERIFY         1 call(s) for 0.00 sec ( 0.0%)
# ESPRESSO      Time was 0.00 sec, cost is c=4(0) in=11 out=8 tot=19
```

Esercizio 2

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

  ONSet1={1,2,5,10}; DCSet1={3,6,7,14}

  ONSet2={1,4,5,6,8,10}; DCSet2={0,13,14}

Soluzione:

- L'insieme di tutti i mintermini è:

  $\text{ONSet} = \text{ONSet1} \cup \text{ONSet2} \cup \text{DCSet1} \cup \text{DCSet2} =$

  $= \{0,1,2,3,4,5,6,7,8,10,13,14\}$

Soluzione

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

mi	x	y	z	v	f1	f2
0	0	0	0	0	0	-
1	0	0	0	1	1	1
2	0	0	1	0	1	0
4	0	1	0	0	0	1
8	1	0	0	0	0	1
3	0	0	1	1	-	0
5	0	1	0	1	1	1
6	0	1	1	0	-	1
10	1	0	1	0	1	1
7	0	1	1	1	-	0
13	1	1	0	1	0	-
14	1	1	1	0	-	-

Tabella di partenza in cui sono presenti solo i mintermini per cui almeno una delle funzioni vale 1 o è un don't care

Soluzione

mi	x	y	z	v	f1	f2	
0	0	0	0	0	0	-	X
1	0	0	0	1	1	1	X
2	0	0	1	0	1	0	X
4	0	1	0	0	0	1	X
8	1	0	0	0	0	1	X
3	0	0	1	1	-	0	X
5	0	1	0	1	1	1	X
6	0	1	1	0	-	1	X
10	1	0	1	0	1	1	X
7	0	1	1	1	-	0	X
13	1	1	0	1	0	-	X
14	1	1	1	0	-	-	X

mi	x	y	z	v	f1	f2
0,1	0	0	0	-	0	1
0,4	0	-	0	0	0	1
0,8	-	0	0	0	0	1
1,3	0	0	-	1	1	0
1,5	0	-	0	1	1	1
2,3	0	0	1	-	1	0
2,6	0	-	1	0	1	0
2,10	-	0	1	0	1	0
4,5	0	1	0	-	0	1
4,6	0	1	-	0	0	1
8,10	1	0	-	0	0	1
3,7	0	-	1	1	1	0
5,7	0	1	-	1	1	0
5,13	-	1	0	1	0	1
6,7	0	1	1	-	1	0
6,14	-	1	1	0	1	1
10,14	1	-	1	0	1	1

mi	x	y	z	v	f1	f2	
0,1	0	0	0	-	0	1	X
0,4	0	-	0	0	0	1	X
0,8	-	0	0	0	0	1	P0
1,3	0	0	-	1	1	0	X
1,5	0	-	0	1	1	1	P1
2,3	0	0	1	-	1	0	X
2,6	0	-	1	0	1	0	X
2,10	-	0	1	0	1	0	X
4,5	0	1	0	-	0	1	X
4,6	0	1	-	0	0	1	P2
8,10	1	0	-	0	0	1	P3
3,7	0	-	1	1	1	0	X
5,7	0	1	-	1	1	0	X
5,13	-	1	0	1	0	1	P4
6,7	0	1	1	-	1	0	X
6,14	-	1	1	0	1	1	P5
10,14	1	-	1	0	1	1	P6

mi	x	y	z	v	f1	f2	
0,1,4,5	0	-	0	-	0	1	**P7**
0,4,1,5	0	-	0	-	0	1	
1,3,5,7	0	-	-	1	1	0	P8
1,5,3,7	0	-	-	1	1	0	
2,3,6,7	0	-	1	-	1	0	P9
2,6,3,7	0	-	1	-	1	0	
2,6,10,14	-	-	1	0	1	0	P10
2,10,6,14	-	-	1	0	1	0	

IMPLICANTI:

P0=-000 =y’z’v’ *0,8* 01 -> f2

P1=0-01 =x’z’v *1,5* 11 -> f1,f2

P2=01-0 =x’yv’ *4,6* 01 -> f2

P3=10-0 =xy’v’ 8,10 01 -> f2

P4=-101 =yz’v 5,*13* 01 -> f2

P5=-110 =yzv’ *6,14* 11 -> f1,f2

P6=1-10 =xzv’ *10,14* 11 -> f1,f2

P7=0-0- =x’z’ 0,1,4,5 01 -> f2

P8=0--1 =x’v 1,3,5,7 10 -> f1

P9=0-1- =x’z 2,3,6,7 10 -> f1

P10=--10 =zv’ 2,6,10,14 10 -> f1

Copertura

All'inizio ad ogni implicante viene associato un costo pari al numero di letterali contenuti

	f1				f2						
	1	2	5	10	1	4	5	6	8	10	
P0									X		3
P1	X		X		X		X				3
P2						X		X			3
P3									X	X	3
P4							X				3
P5								X			3
P6				X						X	3
P7					X	X	X				2
P8	X		X								2
P9		X									2
P10		X		X							2

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

- P2 domina P5
- P3 domina P0
- P1 e P7 dominano P4
- P10 domina P9

Copertura

	f1				f2						
	1	2	5	10	1	4	5	6	8	10	
P1	X		X		X		X				3
P2						X		X			3
P3									X	X	3
P6				X						X	3
P7					X	X	X				2
P8	X		X								2
P10		X		X							2

- P2 e P3 sono essenziali solo per f2 mentre P10 per f1:
  - cancello le colonne relative ai mintermini coperti
  - **cancello le righe P2 e P3** perché sono essenziali per f2 e non coprono mintermini di f1;
  - **cancello la riga P10** perché è essenziale per f1 e non copre mintermini di f2;
- P3 ∪ P10 domina P6

C(f1)={P10} C(f2)={P2,P3}

Copertura

	f1		f2		
	1	5	1	5	
P1	X	X	X	X	3
P7			X	X	2
P8	X	X			2

- Scelgo P7 e P8 per la copertura poiché P1 ha costo 3 mentre P7 e P8 hanno costo 2

C(f1)={P8,P10}

C(f2)={P2,P3,P7}

f1 = x’z’ + x’v

f2 = x’yv’ + xy’v’ + x’z’

Soluzione con Sis(1/2)

- Creare il file es2.blif contenente la definizione della funzione da minimizzare:

```
.model es2
.inputs a b c d
.outputs f1 f2
.names a b c d f1
0001 1
0010 1
0101 1
1010 1
.names a b c d f2
0001 1
0100 1
0101 1
0110 1
1000 1
1010 1
```

```
.exdc
.inputs a b c d
.outputs f1 f2
.names a b c d f1
0011 1
0110 1
0111 1
1110 1
.names a b c d f2
0000 1
1101 1
1110 1
.end
```

Soluzione con Sis(2/2)

```
sis> read_blif es2.blif
sis> print_stats
es2                    pi= 4    po= 2    nodes=   2         latches= 0
lits(sop)=  40
sis> full_simplify -m nocomp
sis> write_eqn
INORDER = a b c d;
OUTORDER = f1 f2;
f1 = c*!d + !a*d;
f2 = a*!b*!d + !a*b*!d + !a*!c;

Don't care:
INORDER = a b c d;
OUTORDER = f1 f2;
f1 = !a*b*c*!d + a*b*c*!d + !a*!b*c*d + !a*b*c*d;
f2 = !a*!b*!c*!d + a*b*c*!d + a*b*!c*d;

sis> print_stats
es2                    pi= 4    po= 2    nodes=   2         latches= 0
lits(sop)=  12
```

Metodi euristici
Applicazione delle trasformazioni

Trasformazioni

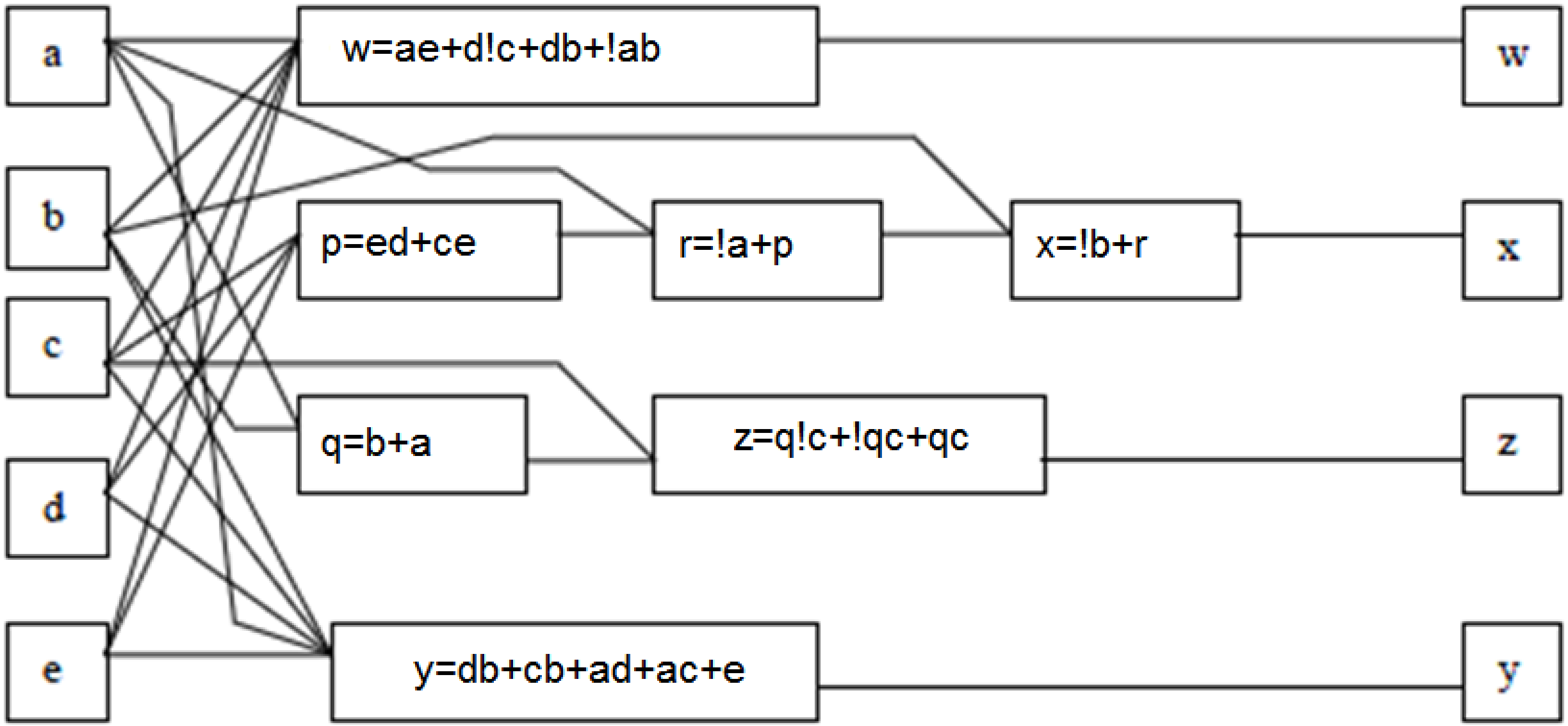
SIS mette a disposizione dei comandi per l’esecuzione delle trasformazioni euristiche:

- **Sweep**: *sweep*
- **Eliminazione**: *eliminate* [-l limit] n (effettua l’eliminazione rimuovendo i nodi tali che la loro rimozione non aumenti il numero di letterali di una quantità superiore a n; per eliminare i nodi che compaiono solo una volta usare il valore -1). *Limit* indica il numero massimo di cubi in un nodo (1000 per default).
- **Decomposizione**: *resub* lista (Esegue l’operazione di scomposizione dei nodi indicati nella lista. Se la lista non viene specificata, la sostituzione viene eseguita per tutti i nodi della rete. I nodi nella lista devono essere specificati con il nome della loro uscita e vanno intervallati tra loro da uno spazio)
- **Fattorizzazione**: *factor* [-gq] node-list
- **Estrazione**: *fx*
- **Semplificazione**: *simplify* e *full_simplify*

Altri comandi utili..

- **Source script**: carica lo script ed esegue tutti i comandi contenuti al suo interno; lo script che generalmente fornisce i risultati migliori è lo *script.rugged*
- **Set autoexec** *comando*: stampa automaticamente il risultato del comando specificato dopo l'esecuzione di un qualunque altro comando

Esercizio



Esercizio

La rete può essere descritta in formato blif come segue (file *multilivello.blif)*

```
.model multilivello
.inputs a b c d e
.outputs w x z y

.names c e d p
11- 1
-11 1
.names a b q
1- 1
-1 1
.names p a r
1- 1
-0 1
.names r b s
1- 1
-0 1
```

```
.names a d b c e v
01--- 1
-11-- 1
-1-0- 1
1---1 1
.names a c d b e t
11--- 1
1-1-- 1
-1-1- 1
--11- 1
----1 1
.names q c u
01 1
10 1
11 1
```

Nodi intermedi

```
.names v w
1 1
.names s x
1 1
.names t y
1 1
.names u z
1 1
```

uscite

```
.end
```

Esercizio

Caricare il file multilivello.blif: con write_eqn è possibile verificare che le equazioni siano quelle della rete da minimizzare

```
sis> read_blif c:\multilivello.blif
sis> write_eqn
INORDER = a b c d e;
OUTORDER = w x z y;
p = e*d + c*e;
q = b + a;
r = !a + p;
x = !b + r;
w = a*e + d*!c + d*b + !a*d;
y = d*b + c*b + a*d + a*c + e;
z = q*!c + !q*c + q*c;
sis> print_stats
multilivello     pi= 5    po= 4    nodes=  7        latches= 0
lits(sop)=  33
sis>
```

Esercizio

Applicare la sequenza di trasformazioni definita da **script-rugged**:

sis> **set autoexec print_stats**

multilivello pi= 5 po= 4 nodes= 7 latches= 0

lits(sop)= 33

sis> **source -x script.rugged**

sweep multilivello pi= 5 po= 4 nodes= 7 latches= 0

lits(sop)= 33

elimina tutti i nodi non usati e sostituisce le costanti ed i nodi con un solo ingresso all’interno dei nodi che li usano;in questo caso non ci sono modifiche

eliminate -1multilivello pi= 5 po= 4 nodes= 5 latches= 0

lits(sop)= 31

Elimina tutti i nodi della rete che permettono un guadagno superiore o uguale alla soglia; Il guadagno è dato dalla differenza in letterali tra la rete in presenza del nodo e la rete in cui il nodo è stato sostituito in tutti i suoi fanout. Se un nodo è usato una sola volta il suo guadagno è -1. In questo caso vengono quindi eliminati tutti i nodi che sono usati una sola volta.

Esercizio

simplify -m nocompmultilivello pi= 5 po= 4 nodes= 5 latches= 0

lits(sop)= 23

semplifica tutti i nodi della rete usando una minimizzazione alla ESPRESSO (opzione –m nocomp) basata anche sul calcolo dei don't care set.

eliminate -1multilivello pi= 5 po= 4 nodes= 5 latches= 0

lits(sop)= 23

sweep multilivello pi= 5 po= 4 nodes= 5 latches= 0

lits(sop)= 23

eliminate 5multilivello pi= 5 po= 4 nodes= 4 latches= 0

lits(sop)= 26

simplify -m nocompmultilivello pi= 5 po= 4 nodes= 4 latches= 0

lits(sop)= 26

resub -**a**multilivello pi= 5 po= 4 nodes= 4 latches= 0

lits(sop)= 26

sostituisce ogni nodo della rete negli altri, applicando una divisione algebrica (opzione –a), finché non si verificano ulteriori cambiamenti.

Esercizio

fxmultilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

ricerca in maniera greedy il cubo singolo e doppio che sia miglior divisore di tutti gli altri cubi.

resub -a multilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

sweep multilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

eliminate -1 multilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

sweep multilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

full_simplify -m nocompmultilivello

multilivello pi= 5 po= 4 nodes= 6 latches= 0

lits(sop)= 21

minimizza tutti i nodi della rete alla ESPRESSO usando i don't care locali di ingresso e uscita calcolati per ogni nodo.

Esercizio

```
sis> write_eqn
INORDER = a b c d e;
OUTORDER = w x z y;
x = e*[11] + !b + !a;
y = [11]*[12] + e;
z = [12] + c;
w = a*e + d*!c + d*b + !a*d;
[11] = d + c;
[12] = b + a;
```