Laboratorio di Architettura degli Elaboratori

Nicola Bombieri
Dipartimento di Informatica
Università di Verona
A.A. 2014/2015

Ottimizzazione di circuiti combinatori

In questa lezione vengono riassunti i concetti fondamentali dell'ottimizzazione esatta a 2 livelli. Inoltre viene mostrato come utilizzare SIS per effettuare l'ottimizzazione di un dispositivo digitale combinatorio specificato in formato blif.

Ottimizzazione esatta a 2 livelli

Tra le caratteristiche principali di un circuito digitale, una particolare importanza rivestono l'**area** (misurabile come il *numero di porte logiche a 2 ingressi* necessarie per la realizzazione del circuito) e il **ritardo di propagazione** con cui i segnali applicati in ingresso forniscono il corrispondente risultato in uscita (misurabile come il *massimo numero di porte logiche che un segnale applicato agli ingressi deve attraversare per raggiungere l'uscita*). Segue da questa affermazione che è importante studiare tecniche che permettano di trasformare un circuito digitale in uno funzionalmente equivalente avente area e/o ritardo minore (ottimizzazione); infatti è compito di una buona tecnologia produrre circuiti "piccoli" e "veloci".

Per comprendere pienamente le tecniche di ottimizzazione è necessario fornire le seguenti definizioni (il concetto di implicante è stato definito nella lezione precedente "Introduzione a SIS-Algebra di Commutazione"):

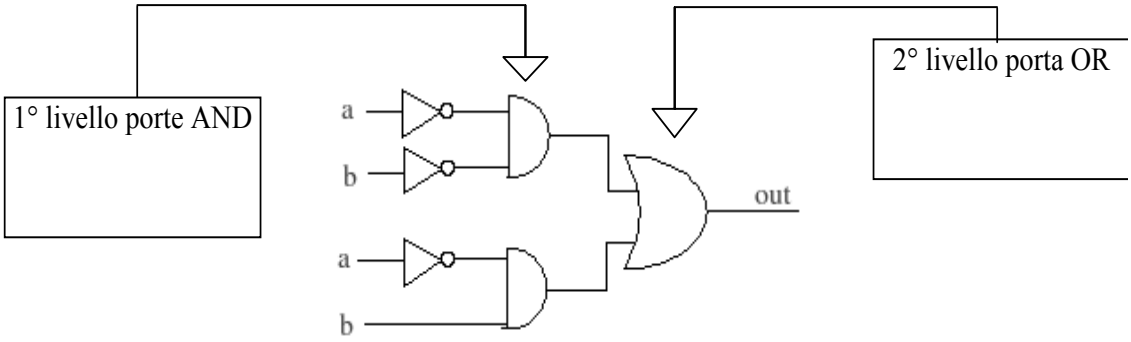
- Un ***implicante*** si dice ***primo*** se non esiste nessun altro implicante di dimensioni maggiori (ovvero formato da un minor numero di letterali) che lo contenga interamente.

- Un ***implicante*** si dice ***essenziale*** se esiste almeno un mintermine coperto dall'implicante che non è coperto da nessun altro implicante della funzione

L'idea alla base del ragionamento è trasformare un'espressione logica in una equivalente avente un numero di letterali minore.

Si definisco **circuiti a 2 livelli** i circuiti che si realizzano a partire da espressioni booleane costituite da somme di prodotti o prodotti di somme (forme canoniche); nei corrispondenti circuiti si può infatti identificare un livello di porte AND e uno di porte OR.

Esempio:



La minimizzazione di circuiti a 2 livelli avviene come segue:

1. Si identificano tutti gli implicanti primi essenziali.
2. Si identifica un insieme minimo di implicanti che coprano tutti i mintermini non coperti dagli implicanti primi essenziali.
3. La funzione di copertura ottima è data dalla somma degli implicanti trovati ai punti 1 e 2.

   La seguente tabella evidenzia i principali risultati ottenuti in termini di algoritmi di minimizzazione a 2 livelli per circuiti a una o più uscite.

	Circuiti a 2 livelli
Circuiti a una uscita	Esiste un metodo esatto per trovare gli implicanti primi essenziali Esiste un metodo esatto o approssimato (dipende dal circuito) per ottenere la funzione di copertura ottima
Circuiti a più uscite	Esiste un metodo approssimato per trovare la copertura ottima che si basa sul metodo esatto di identificazione degli implicanti primi essenziali di ogni singola uscita

Metodo pratico per trovare la copertura minima di una funzione booleana usando il metodo di Quine-McCluskey per funzioni completamente specificate

1. Estrarre dalla tabella delle verità i mintermini della funzione.
2. Ordinare i mintermini in base al numero di 1 (letterali positivi) che contengono. Raggruppare i mintermini con le stesso numero di 1.

3. Per ogni mintermine *m* il cui numero di "1" è pari a *i*, cercare tra i mintermini il cui numero di "1" è *i*+1 quelli a distanza di Hamming 1 da *m*.
   Per ogni mintermine a distanza di Hamming 1 da *m*, riportare in una nuova tabella una riga che "collassi" i letterali di *m* e di *n* riportando il simbolo di *don't care* (–) nella posizione corrispondente all'unico letterale in cui *m* e *n* differiscono.
   Ad esempio, dati i mintermini *m* = 0101 e *n* = 0111, nella nuova tabella verrà riportata la riga 01–0.

4. Considerare la tabella ottenuta al passo 3 e reiterare i passi 2 e 3 finché non sarà più possibile collassare mintermini. Durante i passi 2 e 3 marcare i mintermini (o le righe della tabella corrispondenti al collassamento di 2 mintermini) che non è stato possibile collassare.

5. Le righe non collassate delle varie tabelle create reiterando i passi 2 e 3 corrispondono agli implicanti primi della funzione.

6. Creare una tabella che riporta in riga i mintermini e in colonna gli implicanti primi. Inserire nell'elemento (*i, j*) il valore 1 se e solo se il mintermine *i* è coperto dall'implicante *j*.

7. Nella tabella del punto 6 marcare i mintermini coperti da un unico implicante primo. Tali implicanti sono implicanti primi essenziali e faranno sicuramente parte della copertura ottima.

8. Creare una nuova tabella a partire dalla tabella del punto 6 ottenuta eliminando le colonne degli implicanti essenziali e le righe corrispondenti ai mintermini coperti dagli implicanti corrispondenti a tali colonne.

9. Nella tabella creata al punto 8 eliminare le eventuali colonne dominate (ovvero le colonne in cui gli 1 sono un sottoinsieme degli 1 contenuti in un'altra colonna).

10. Ripetere i passi 7, 8 e 9 nella tabella ottenuta al punto 9 finché tutti i mintermini risultano coperti, oppure si ottiene una tabella in cui non esistono colonne dominanti.
    Nel primo caso sono stati identificati tutti gli implicanti necessari per ottenere la copertura ottima. Nel secondo caso procedere come indicato al punto 11.

11. Nella tabella ottenuta al punto 10 cercare ed eliminare le righe dominanti (ovvero le righe in cui gli 1 sono una sovra-insieme degli 1 contenuti in un'altra riga) e ripetere i passi a partire dal 9. Qualora non esistano righe dominanti si deve procedere usando una tecnica di branch and bound, ovvero si sceglie un implicante (di solito quello che copre il maggior numero di mintermini), si considera come se fosse essenziale, e si riparte dal passo 8.

Esempio di applicazione del metodo di Quine-McCluskey alla seguente tabella delle verità:

X	Y	Z	W	U
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**
0	**0**	**1**	**0**	**0**
0	**0**	**1**	**1**	**0**
0	**1**	**0**	**0**	**1**
0	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**1**
0	**1**	**1**	**1**	**1**
1	**0**	**0**	**0**	**0**
1	**0**	**0**	**1**	**1**
1	**0**	**1**	**0**	**0**
1	**0**	**1**	**1**	**1**
1	**1**	**0**	**0**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**0**	**1**
1	**1**	**1**	**1**	**1**

1. f(x,y,z,w) = {m1; m4; m5; m6; m7; m9; m11; m14; m15};
   per m1 si intende il mintermine 0001
   per m4 si intende il mintermine 0100
   per m5 si intende il mintermine 0101
   etc..

2. $S_{0,0}$ {} insieme dei mintermini con 0 1 al tempo 0;

   $S_{1,0}$ {m1, m4} insieme dei mintermini con 1 1 al tempo 0;

   $S_{2,0}$ {m5, m6, m9} insieme dei mintermini con 2 1 al tempo 0;

   $S_{3,0}$ {m7, m11, m14} insieme dei mintermini con 3 1 al tempo 0;

   $S_{4,0}$ {m15} insieme dei mintermini con 4 1 al tempo 0;

3. Confronto $S_{1,0}$ con $S_{2,0}$ ottengo:
$S_{1,1}$ { 010- m4,m5
01-0 m4,m6
0-01 m1,m5 X
-001 m1,m9 X}

Confronto $S_{2,0}$ con $S_{3,0}$ ottengo:
$S_{2,1}$ { 01-1 m5,m7
011- m6,m7
-110 m6,m14
10-1 m9,m11 **X**}

Confronto $S_{3,0}$ con $S_{4,0}$ ottengo:

$S_{3,1}$ {-111 m7,m15
1-11 m11,m15 **X**
111- m14,m15}

4. Confronto $S_{1,1}$ con $S_{2,1}$ ottengo:

$S_{1,3}$ {01-- m4,m5,m6,m7 **X**}

Confronto S 2,1 con S3,1 ottengo:

$S_{2,3}$ {-11- m6,m7,m14,m15 **X**}

I mintermini che non è stato possibile collassare sono stati marcati con **X**;

5. Gli implicanti primi della funzione sono:

P1= 0-01 ; P2=-001 ; P3=10-1 ; P4=1-11 ; P5=01-- ; P6=-11-.

6. 7. Si ottiene:

	P1	**P2**	**P3**	**P4**	**P5**	**P6**
m1	**1**	**1**				
m4					**1**	
m5	**1**				**1**	
m6					**1**	**1**
m7					**1**	**1**
m9		**1**	**1**			
m11			**1**	**1**		
m14						**1**
m15				**1**		**1**

Implicanti primi essenziali sono P5 e P6.

8. Si ottiene:

	P1	**P2**	**P3**	**P4**
m1	**1**	**1**		
m9		**1**	**1**	
m11			**1**	**1**

9. Si ottiene

	P2	**P3**
m1	**1**	
m9	**1**	**1**
m11		**1**

10. Tutti i mintermini sono coperti da P2 e P3.

Il Risultato dell'ottimizzazione è quindi:

$$U = P5 + P6 + P2 + P3;$$

cioè:

$$U = \overline{X}*Y + Y*Z + \overline{Y}*\overline{Z}*W + X*\overline{Y}*W.$$

Funzioni non completamente specificate

Nel caso di funzioni non completamente specificate, esistono alcune combinazioni delle variabili di input per cui non è specificato un corrispondente valore per le variabili di output. Le righe della tabella delle verità che corrispondono a tali combinazione costituiscono il ***don't care set*** della funzione. I valori per le variabili in output che corrispondono alle righe del ***don't care set*** possono assumere indifferentemente il valore 0 o il valore 1. A seconda di tale scelta è possibile ottenere circuiti più o meno ottimizzati.

Metodo pratico per trovare la copertura minima di una funzione booleana usando il metodo di Quine-McCluskey per funzioni non completamente specificate

Il metodo pratico per ottimizzare funzioni non completamente specificate è esattamente quello descritto precedentemente per le funzioni completamente specificate considerando le righe della tabella delle verità corrispondenti al don't care set come se fossero mintermini (ovvero appartenenti all' on-set). E' necessario tuttavia ricordare che tali "finti mintermini" non devono essere necessariamente coperti dagli implicanti che costituiranno la copertura minima. Pertanto le righe corrispondenti al don't care set devono essere marcate come "collassabili" a priori durante i passi 2 e 3 dell'algoritmo.

Riflessione sul don't care set

Si noti che l'introduzione del don't care set non ha a che vedere con la **funzionalità** del circuito, infatti quest'ultima non ne risentirebbe se nella tabella delle verità del circuito ogni simbolo di *don't care* venisse sostituito da un *1* o da uno *0*. L'**ottimizzazione** del circuito viene resa più difficile se si forzano a priori tali valori a 1 o 0 mentre viene facilitata se gli si dà valori tali da facilitare il collassamento delle righe.

Minimizzazione di circuiti combinatori a 2 livelli tramite SIS

Per minimizzare una rappresentazione blif con SIS è possibile utilizzare il comando **`full_simplify`** dopo aver caricato il modello da ottimizzare.

E' possibile osservare l'effetto della minimizzazione usando il comando **`write_eqn`** prima e dopo aver utilizzato il comando `full_simplify`. Il comando `write_eqn` visualizza l'espressione booleana corrispondente al modello rappresentato in blif. L'espressione booleana cambia dopo l'uso del comando `full_simplify`. Per

calcolare l'area e il ritardo del circuito relativo all'espressione considerata occorre disegnarlo utilizzando solo porte NOT, e AND/OR a due ingressi. L'area è data dal numero di porte utilizzate mentre per il ritardo si deve individuare il numero di porte attraversate sul cammino più lungo tra uno qualunque degli ingressi e una qualunque delle uscite (cammino critico).

Inoltre il comando **`print_stats`** visualizza le seguenti informazioni sul circuito: numero variabili in input (PI), numero variabili in output (PO), numero di letterali (LITS). Il numero di letterali è un'altra misura del grado di complessità del circuito e può diminuire dopo l'ottimizzazione.

Per descrivere il ***dont'care set*** di una funzione è necessario usare la keyword **`.exdc`**.

Ad esempio, data la seguente tabella di verità parzialmente specificata

a	b	c	z
0	0	0	0
0	0	1	1
0	1	0	-
0	1	1	0
1	0	0	1
1	0	1	-
1	1	0	-
1	1	1	1

il formato blif che rappresenta la tabella sopra illustrata è il seguente:

```
.model esempio
.inputs a b c
.outputs z

.names a b c z
001 1
100 1
111 1

.exdc
.names a b c z
010 1
101 1
110 1
.end
```

Sezione che specifica l'on-set

Sezione che specifica il don't care set

Comandi utili per l'utilizzo di sis:

- **`write_eqn`** — Stampa a video l'equazione del circuito caricato in sis, si noti che l'espressione è scritta in somma di prodotti e che il simbolo " ! " indica la negazione del letterale che lo segue;
- **`full_simplify`** — Ottimizza il circuito caricato in sis, è bene verificare l'operato di questo comando lanciano il comando `write_eqn` prima e dopo il `full_simplify`;
- **`print_stats`** — Stampa a video importanti informazioni sul circuito: il numero di segnali in input (`PI`), il numero di segnali in output (`PO`), in numeto di nodi (`nodes`), il numero di elementi di memoria (`latches`) e il numero di letterali (`lits`).

Esercizi

Esercizio 1: Scrivere la tabella delle verità e rappresentare nel formato blif il circuito digitale corrispondente alla seguente espressione booleana:

$$f(x,y,z,v) = \bar{x} * y * z + \bar{x} * \bar{y} + y * \bar{z} * v + x * v * \bar{y} + y * z$$

Eseguire l'ottimizzazione con SIS usando il comando `full_simplify`. Fornire il grado di ottimizzazione confrontando l'area (numero porte logiche), il ritardo (lunghezza cammino critico) e il numero di letterali del circuito prima e dopo l'ottimizzazione. Visualizzare l'espressione booleana corrispondente al circuito ottimizzato con il comando `write_eqn` e confrontarla con quella che si ottiene manualmente usando il metodo di Quine-McCluskey.

Esercizio 2: Descrivere in formato blif il circuito digitale che esegue la sottrazione di 2 numeri binari su 2 bit rappresentati in complemento a 2 con risultato ancora su 2 bit in complemento a 2. Il circuito corrispondente avrà quindi 4 ingressi e 2 uscite. Si parta scrivendo la tabella di verità per poi scrivere il file blif. Eseguire l'ottimizzazione con SIS. Visualizzare l'espressione booleana corrispondente al circuito prima e dopo l'ottimizzazione e fornire il grado di ottimizzazione (area, ritardo, nr. di letterali).

Esercizio 3: Descrivere nel formato blif un circuito digitale che riceve in input una sequenza di 5 bit che rappresenta la codifica binaria di una lettera dell'alfabeto italiano (21 lettere) considerando di associare in ordine crescente i numeri dallo 0 al 20 alle lettere dalla A alla Z (A=0, B=1, …, Z=20). Il circuito ha un solo bit in uscita che vale 1 se e solo se la sequenza di input corrisponde ad una vocale. La tabella delle verità risulterà parzialmente specificata dal momento che con 5 cifre binarie è possibile rappresentare i numeri dallo 0 al 31. Ottimizzare il circuito con SIS associando all'uscita quando essa non è specificata:

- il valore 0,
- il valore 1,
- il valore don't care.

Quale dei tre circuiti è maggiormente ottimizzato ?

Esercizio 4: Descrivere nel formato blif un circuito che calcola la parte intera della radice quadrata dei numeri da 0 a 9 rappresentati da numeri binari a 4 bit. L'output sarà una sequenza composta da 2 bit. La tabella delle verità del circuito risulterà parzialmente specificata dal momento che con 4 bit è possibile rappresentare i numeri dallo 0 al 15. Ottimizzare il circuito con SIS associando all'uscita quando essa non è specificata:

- il valore 0,
- il valore 1,
- il valore don't care.

Quale dei tre circuiti è maggiormente ottimizzato ?