Strumento CAD per la sintesi ottima di reti a 2 livelli

Approfondimento del corso di reti logiche

M. Favalli

Engineering Department in Ferrara

Sommario

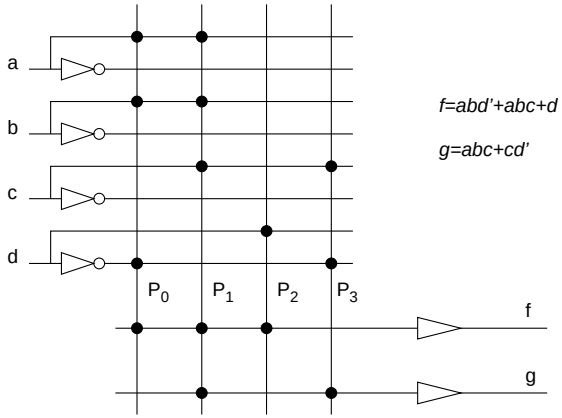
Motivazioni

- I metodi grafici (Karnaugh) hanno esclusivamente una valenza come strumento di ragionamento
- Il metodo di Quine-McCluskey puó essere esteso a reti a piú uscite, ma ha delle pesanti limitazioni computazionali nel caso di funzioni di grandi dimensioni
- Il largo utilizzo di PLA (’80) ha richiesto lo sviluppo di diversi strumenti CAD per la sintesi ottima di reti a due livelli
- Fra questi il piú noto é ESPRESSO, lo utilizzeremo per introdurre l’utilizzo di strumenti CAD e perché le sue procedure servono anche nell’ambito della sintesi multilivello

PLA (Programmable Logic Array)

- Rete logica programmabile a due livelli
- Mette a disposizione un piano AND con la possibilitá di programmare un numero limitato di implicanti e un piano OR con la possibilitá di connetterli ai termini somma in uscita
- La programmazione puó avvenire sia mediante fusibili p mediante la deposizione di un interconnessione

Esempio di PLA



Espresso

- É stato sviluppato a Berkeley e ha rappresentato un significativo miglioramento nelle prestazioni degli strumenti di sintesi allora utilizzati
- Riceve in ingresso un file di testo con una rete a due livelli descritta nel formato standard di Berkeley e produce in uscita la rete ottimizzata
- Consente di scegliere fra l'utilizzo di un algoritmo esatto e un euristico che nel caso di PLA molto grandi é computazionalmente piú efficiente

Formato di ingresso

Si tratta di un file di testo

.i [d] numero di variabili di ingresso

.o [d] numero di funzioni di uscita

.ilb [s1] [s2] .... [sn] lista dei nomi delle variabili di ingresso

.ob [s1] [s2] .... [sn] lista dei nomi delle funzioni di uscita

.p [d] numero di termini prodotto

.e fine della descrizione

Esistono ulteriori opzioni che sono consultabili sul manuale

Descrizione della PLA - I

- Il punto di partenza é una funzione da ottimizzare con n ingressi ed m uscite:

$$f : \{0,1\}^n \rightarrow \{0,1,-\}^m$$

- Come si vede la funzione puó anche essere non completamente specificata per la presenza di don't care
- Le funzioni vengono descritte mediante termini prodotto (e quindi anche mintermini) rappresentati come cubi
- Il formato di ESPRESSO tenta di usare una matrice compatta di caratteri (con una riga per ogni termine prodotto) per rappresentare tale funzione
- Il significato di tali caratteri dipende dal tipo di descrizione della funzione specificato da **.type**

Descrizione della PLA – II

- Ciascun elemento f^i di f puó essere descritto in vari modi che utilizzano:
  1. f^i_{ON} (ON-set) dove la funzione vale 1;
  2. f^i_{OFF} (OFF-set) dove la funzione vale 0;
  3. f^i_{DC} (DC-set) dove la funzione ha un valore indifferente;
- Si ricorda che devono valere:

$$f^i_{ON} + f^i_{OFF} + f^i_{DC} = 1 \quad (1)$$

$$f^i_{ON} \cdot f^i_{OFF} = f^i_{ON} \cdot f^i_{DC} = 0 \quad (2)$$

Nota

La funzione f^i_{ON} é una funzione da $\{0,1\}^n$ a $\{0,1\}$ che vale 1 dove f^i vale 1, f^i_{OFF} vale 1 dove $f^i = 0$ e $f^i_{DC} = 1$ dove $f^i = -$

Descrizione della PLA - III

1. dando solo f^i_{ON} (ON-set) ovvero quei mintermini dove le funzioni valgono 1 (**.type f**): $f^i_{OFF} = {f^i_{ON}}'$ e $f^i_{DC} = \emptyset$
2. dando f^i_{ON} (ON-set) e f^i_{DC} (DC-set) ovvero quei mintermini dove le funzioni valgono 1 e quelli dove le funzioni hanno un valore indifferente (**.type fd**): $f^i_{OFF} = (f^i_{ON} + f^i_{DC})'$
3. dando f^i_{ON} (ON-set) e f^i_{OFF} (OFF-set) ovvero quei mintermini dove le funzioni valgono 1 e 0 (**.type fr**). In questo caso il DC-set viene calcolato come: $f^i_{DC} = (f^i_{ON} + f^i_{OFF})'$
4. dando f^i_{ON} (ON-set), f^i_{DC} (DC-set) e f^i_{OFF} (OFF-set) ovvero quei mintermini dove le funzioni valgono 1, - e 0 (**.type fdr**) ESPRESSO non verifica in modo chiaro se le relazioni $f^i_{ON} + f^i_{OFF} + f^i_{DC} = 1$ e $f^i_{ON} \cdot f^i_{OFF} = f^i_{ON} \cdot f^i_{DC} = 0$ sono verificate

Matrice di caratteri (piano AND)

- La matrice ha una riga per ogni cubo e ciascuna riga é divisa in due stringhe (separate da uno spazio): la prima contiene la descrizione del cubo, la seconda le informazioni su come tale cubo é usato nelle funzioni di uscita
- La matrice é divisa in due parti che in analogia al layout elettrico della PLA vengono definite AND plane e OR plane
- Il cubo viene descritto in maniera simile a quanto visto nell'algoritmo di Quine-McCluskey:
  - si ha una stringa (corrispondenti per posizione alle variabili definite in **.ilb**) che valgono 1 se la variabile compare nel termine prodotto con un letterale in forma vera, 0 se compare in forma negata, e - se non compare
  - ad esempio se *abcd* sono le variabili, 1- -0 equivale ad ad'

Piano OR - I

- L'interpretazione della stringa nel piano OR dipende dal tipo di descrizione **.type**
- Ciascun carattere c^i di tale stringa corrisponde in maniera ordinata alle funzioni dichiarate con **.olb**.
- Si possono descrivere sia funzioni completamente specificate che funzioni non completamente specificate
- Si noti che possiamo avere termini prodotto che non interessano una certa uscita, mentre altri ne interessano piú di una

Piano OR - II

.type f se c^i vale '1' il termine prodotto appartiene all'ON-set di tale funzione, se vale '0' o '-' vuole dire che non ha alcun significato per tale funzione

.type fd (default) se c^i vale '1' il termine prodotto appartiene all'ON-set di tale funzione, se vale '0' vuole dire che non ha alcun significato per tale funzione, se vale '-' vuole dire che appartiene al DC-set

.type fr se c^i vale '1' il termine prodotto appartiene all'ON-set di tale funzione, se vale '0' appartiene all'OFF-set se vale '-' vuole dire che non ha alcun significato per tale funzione

.type fdr se c^i vale '1' il termine prodotto appartiene all'ON-set di tale funzione, se vale '0' vuole dire che appartiene all'OFF-set, se vale '-' vuole dire che appartiene al DC-set Infine una '~' significa che il termine prodotto non ha significato per la PLA

Esempio

Si consideri una funzione $f : \{0,1\}^3 \rightarrow \{0,1,-\}^2$ definita sul supporto a, b, c con x, y come funzioni di uscita

abc	x	x_{ON}	x_{OFF}	x_{DC}
000	0	0	1	0
001	1	1	0	0
010	1	1	0	0
011	0	0	1	0
100	1	1	0	0
101	-	0	0	1
110	0	0	1	0
111	-	0	0	1

abc	y	y_{ON}	y_{OFF}	y_{DC}
000	0	0	1	0
001	1	1	0	0
010	-	0	0	1
011	-	0	0	1
100	1	1	0	0
101	-	0	0	1
110	1	1	0	0
111	0	0	1	0

Rappresentazione per Espresso

```
.i 3
.o 2
.ilb a b c
.ob x y
.p 8
.type fd
000 00
001 11
010 1-
011 0-
100 11
101 --
110 01
111 -0
.e
```

Nota che **.type f** in questo caso non ha significato in quanto le due funzioni hanno delle indifferenze. Inoltre, le matrici di tipo **.type fr** e **.type fdr** sono uguali a **.type fd** (che é il valore di default) in quanto abbiamo tutti i mintermini e quindi Espresso non compie inferenze.

Esempio: .type fd

Per apprezzare le differenze fra i diversi tipi di descrizione conviene considerare una PLA in cui compaiono termini prodotto che non sono dei mintermini

$x_{ON} = ab'$ e $x_{DC} = a'b'c + abc$, mentre $y_{ON} = ab' + bc + abc$ e $y_{DC} = a'b'c$. Si nota che questa descrizione implica l'aggiunta all'ON-set di un termine ridondante (abc)

```
.i 3
.o 2
.ilb a b c
.ob x y
.p 4
.type fd
10- 11
-11 01
001 --
111 -1
.e
```

Esempio: .type FR

Vediamo ora la stessa PLA descritta mediante **.type fr**

$x_{ON} = ab'$ e $x_{OFF} = bc' + a'b + a'c'$, mentre $y_{ON} = ab' + bc$ e $y_{OFF} = bc' + a'c'$. Si noti il diverso uso del simbolo -.

```
.i 3
.o 2
.ilb a b c
.ob x y
.p 5
.type fr
10- 11
-10 00
01- 0-
0-0 00
-11 -1
.e
```


Esempio: .type fdr

Vediamo di nuovo la stessa PLA descritta mediante **.type fdr**

$x_{ON} = ab'$, $x_{OFF} = bc' + a'c' + a'b$ e $x_{DC} = a'b'c + abc$, mentre $y_{ON} = ab' + bc + abc$, $y_{OFF} = b'c + a'c'$ e $y_{DC} = a'b'c$.

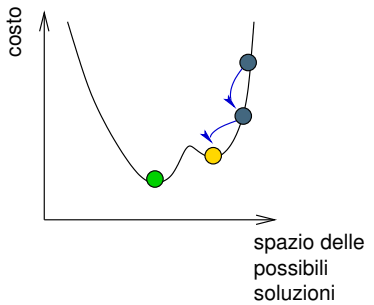
```
.i 3
.o 2
.ilb a b c
.ob x y
.p 6
.type fdr
10- 11
-10 00
-11 ~1
001 --
0-0 00
01- 0~
.e
```


Algoritmo

- Espresso dispone di un algoritmo esatto e di un euristico
- L'algoritmo esatto consente di determinare la soluzione di costo minimo (esatta) fino a circa 25 variabili (simile a Quine - McCluskey)
- L'algoritmo di tipo euristico consente di trattare PLA piú grandi ottenendo una soluzione che puó non essere quella di costo minimo
- Si tratta di un algoritmo iterattivo che a partire da una soluzione iniziale cerca di convergere verso una soluzione di costo minimo
- In ciascuna iterazione vengono applicate delle trasformazioni alla rete

Problemi degli euristici

- Consideriamo uno spazio di possibili soluzioni S ciascuna delle quali s é caratterizzata da un costo $C(s)$
- Un euristico itera (secondo qualche regola) cercando il minimo di C ($dC/ds = 0$)
- Le regole semplici (gradiente) possono dare luogo facilmente a soluzioni corrispondenti a minimi locali
- Sono necessari approcci che consentano di uscire dai minimi locali



Il problema e la strategia di Espresso

- Spazio di ricerca: tutte le possibili espressioni di tipo SP
- Spazio delle soluzioni: insieme delle possibili espressioni di tipo SP (a piú uscite) equivalenti alla PLA di partenza
- Costo: numero di letterali
- Strategia di ricerca: utilizzo di trasformazioni (EXPAND, REDUCE) per un numero limitato di termini prodotto
- Metodi di ottimizzazione (deterministici): (ESSENTIAL, IRREDUNDANT)

Algoritmo - I

1. Ottimizza la PLA esistente utilizzando EXPAND, IRREDUNDANT e ESSENTIAL (tutte trasformazioni che riducono C)
2. Prova a uscire da un minimo locale utilizzando REDUCE
3. Prosegue fino a raggiungere la soluzione
4. Nota: fino a circa 20 variabili l'algoritmo raggiunge spesso il minimo assoluto

Algoritmo - II

$F_{OFF} = complement(F_{ON} \cup F_{DC})$;
$F = expand(F_{ON}, F_{OFF})$;
$F = irredundant(F, F_{DC})$;
$E = essential(F, F_{DC})$;
$F = F - E$;
$F_{DC} = F_{DC} \cup E$;

repeat
  $c = cost(F)$;
  repeat
    $c' = |F|$
    $F = reduce(F, F_{DC})$;
    $F = expand(F, F_{OFF})$;
    $F = irredundant(F, F_{DC})$;
  until $(|F| \geq c)$;
  $F = last\_gasp(F, F_{DC}, F_{OFF})$;
until $(cost(F) \geq c)$
$F = F \cup E$; $F_{DC} = F_{DC} - E$;

EXPAND

- Elimina un letterale da un cubo giá esistente
- Verifica se l'espansione é corretta (non coinvolge degli 0). Si noti che questo implica il calcolo di f_{OFF}
- Prosegue fino ad ottenere un implicante primo
- Esempio

IRREDUNDANT

- Elimina dalla copertura corrente quegli implicnti che coprono mintermini giá coperti da altri implicanti
- Esempio

ESSENTIAL

- Determina se un implicante primo é essenziale
- In tale caso lo aggiunge (temporaneamente) al DC-set

REDUCE

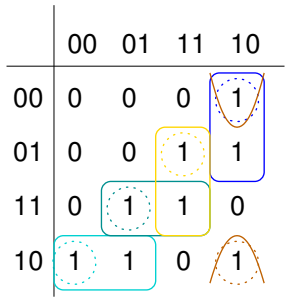
- Dato un implicante aggiunge un letterale fra quelli non specificati
- Verifica se l'operazione é valida (si potrebbe avere un implicante che copre esclusivamente delle condizioni di indifferenza)
- Esempio:

PLA iniziale

	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	0	1	1	0
10	1	1	0	1

```
.type fc
0010 1
0111 1
0110 1
1101 1
1111 1
1000 1
1001 1
1010 1
```

Iteration 1 - EXPAND

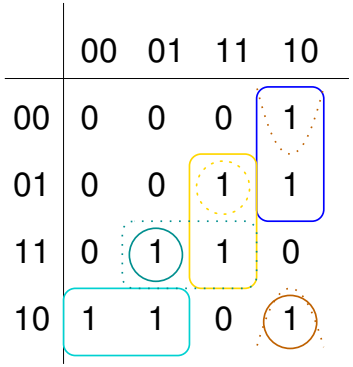



```
EXPAND 1000 -> 100- (covered 1)
EXPAND 0010 -> 0-10 (covered 1)
EXPAND 1101 -> 11-1 (covered 1)
EXPAND 0111 -> -111 (covered 1)
EXPAND 1010 -> -010 (covered  1

.type fd
100- 1
0-10 1
11-1 1
-111 1
-010 1
```

La copertura non é ridondante, quindi IRREDUNDANT non cambia nulla. Lo stesso vale per ESSENTIAL che non trova implicanti essenziali.

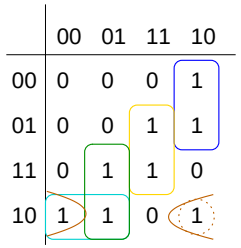
Iteration 1 - REDUCE



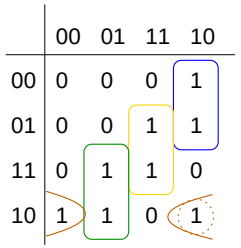

```
REDUCE 11-1 -> 1101
REDUCE -010 -> 1010

.type fd
100- 1
0-10 1
1101 1
-111 1
1010 1
```


ITERATION 2 - EXPAND & IRREDUNDANT




```
.type fd
100- 1
0-10 1
1-01 1
-111 1
10-0 1
```




```
.type fd
1-01 1
-111 1
10-0 1
0-10 1
```

```
EXPAND 1101 -> 1-01
EXPAND 1010 -> 10-0
```

IRREDUNDANT

Si osserva come la sequenza di operazioni svolte dall'euristico abbia eliminato un minimo locale

Modello di costo

Il modello di Espresso é basato sul numero totale di ingressi di gate, questo puó essere verificato con il comando **-s**

Comandi di Espresso

I comandi di Espresso possono essere passati sulla linea di comando, qui verranno riporati solo i piú significativi

Comandi di Espresso: verifica

-Dcheck verifica che ON-set, OFF-set e DC-set costituiscano una partizione dello spazio delle variabili di ingresso

-Dequiv verifica se se 2 uscite sono equivalenti

-Dverify verifica se 2 PLA sono equivalenti (le PLA stanno in due file i cui nomi vengono passati sulla linea di comando)

-DPLAverify verifica se 2 PLA sono equivalenti permettendo una permutazione dell'ordine delle uscite

Comandi di espresso: ottimizzazione

-Dexact algoritmo di ottimizzazione esatto

-Dopo determina se alcune uscite possono essere complementate per ridurre il costo della PLA

-Dpair invece di utilizzare un letterale e il suo complemento, utilizza decoder a 2 bit

-Dso minimizza la funzione considerando le singole uscite separatamente

-efast non itera sulla soluzione

Comandi di Espresso: vari

-Dmap disegna la mappa di Karnaugh

-Dstats produce alcune statistiche sulle dimensioni della funzione

-s stampa in uscita alcune statistiche sul costo della funzione di partenza e quello della funzione ottenuta

-o[type] consente di selezionare il formato di uscita

Riferimenti

- Distribuzione di Berkeley di Espresso (puó essere facilmente compilata sotto Linux):
  `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm`
- Espresso precompilato per windows:
  `http://web.cecs.pdx.edu/~alanmi/research/soft/softPorts.htm`
- R. Brayton, G. Hachtel and A. Sangiovanni-Vincentelli, "Logic minimization algorithms for VLSI synthesis", Kluwer Academic Publisher, 1984

Progetti

Sono disponibili alcuni progetti basati su Espresso
Regole

- Numero massimo di persone del gruppo = 2
- Ciascun progetto puó essere eseguito da un solo gruppo che deve "registrarsi" a lezione o a ricevimento

Esempi di progetti

1. Differenza di costo fra reti a 2 livelli e piú uscite sintetizzate ottimizzando le singole uscite (e sfruttando il fan-out nel caso in cui occasionalmente due o piú funzioni abbiano in comune lo stesso implicante)
2. Utilizzo della decodifica degli ingressi a coppie (in pratica é una sostituzione di variabili)
3. ....

I dati sono da valutare su un insieme di benchmark che verrá reso disponibile