Sistemi per la Progettazione Automatica

Informatica - Tiziano Villa

17 Marzo 2008

Nome e Cognome:
Matricola:
Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	14	
problema 2	4	
problema 3	6	
problema 4	6	
totale	30	

1. Sia f una funzione con copertura $F = w'x'y'z' + wx'z' + wx + w'x'yz$.

   (a) Si complementi f con il metodo di ricorsione monotona. Si mostrino i risultati ad ogni passo.

   Traccia di soluzione.

   Cofattorizzando con l'ordine w, x, y, z, si ottiene la seguente copertura del complemento $F_1' = wx'z + w'x + w'yz' + w'y'z$.

.

(b) Dati due cubi α e β, si definisca la seguente operazione $\alpha \tilde{\sharp} \beta$:

$$\alpha \tilde{\sharp} \beta = \begin{cases} a_1.b'_1 & a_2 & \dots & a_n \\ a_1.b_1 & a_2.b'_2 & \dots & a_n \\ \dots & & & \\ a_1.b_1 & a_2.b_2 & \dots & a_n.b'_n \end{cases}$$

Si analizzi l'effetto di tale operazione e la si usi per calcolare il complemento di f, mostrando i passi del calcolo.

Traccia di soluzione.

Calcolando $U \tilde{\sharp} F$ (dove U e' l'universo su x, y, z, w), si ottiene la seguente copertura disgiunta del complemento $F'_2 = wx'z + w'x + w'x'yz' + w'x'y'z$.

.

(c) Si usi la legge di De Morgan per ottenere il complemento di f. Si semplifichi tale risultato rispetto al contenimento per cubo singolo. Ci sono primi di $\overline{f}$ che mancano dalla lista cosi' ottenuta ? Questo fatto a quale congettura induce ?

Traccia di soluzione.

Si ottiene la seguente copertura del complemento $F_3' = w'x + w'yz' + w'y'z + wx'z + x'y'z$.

Tale copertura contiene tutti i primi di $\overline{f}$.

E' una proprieta' generale di questo procedimento di produrre tutti i primi del complemento.

2. (a) Si definisca la proprieta' di monotonia per una funzione Booleana a due valori.

   Si definisca la proprieta' di monotonia per una copertura di una funzione Booleana a due valori.

   Traccia di soluzione.

   Una funzione Booleana f e' monotona crescente in x_j se cambiando x_j da 0 a 1 la funzione f rimane invariata o cambia da 0 a 1.

   Una copertura F e' monotona crescente in x_j se in ogni prodotto di F la variabile x_j ha il valore 1 oppure 2.

(b) Si dimostri o si trovi un controesempio alla seguente affermazione: una funzione Booleana a due valori e' monotona se e solo se ogni sua copertura e' monotona.

Traccia di soluzione.

Se una copertura F e' monotona in x_j, la funzione corrispondente e' monotona in x_j. Infatti, sia F monotona crescente in x_j, allora essa si puo' rappresentare come somma di prodotti dove in ogni prodotto o x_j non compare (valore 2) o compare in fase positiva (valore 1). Percio', dato un mintermine, cambiando x_j da 0 a 1, il valore della funzione o rimane invariato (c'e' un prodotto contenente il mintermine con x_j cambiato da 0 a 1 in cui la variabile x_j ha il valore 2) o cambia da 0 a 1 (in ogni prodotto contenente il mintermine con x_j cambiato da 0 a 1 la variabile x_j ha il valore 1).

E' falso che ogni copertura di una funzione monotona sia monotona. Si consideri il seguente semplice contro-esempio: la copertura $F_1 = x'y'z' + x'yz' + x'z$ non e' monotona, ma esiste una copertura monotona equivalente $F_2 = x'$ per cui la funzione f di cui F_1 e F_2 sono coperture e' monotona. Similmente la copertura $G_1 = x'z' + y'z + z$ non e' monotona, ma esiste una copertura monotona equivalente $G_2 = x' + z$ per cui la funzione g di cui G_1 e G_2 sono coperture e' monotona.

Pero' se una funzione e' monotona, c'e' almeno una copertura monotona. Supponiamo che f sia monotona crescente in x_j. Dato un mintermine con $x_j = 0$ per cui la funzione vale 1, dato che cambiando il valore della coordinata x_j da 0 a 1 il valore di f non cambia oppure cambia da 0 a 1, vuol dire che si puo' definire un cubo dove la variabile x_j vale 2 (e il resto e' uguale al mintermine considerato); dato un mintermine con $x_j = 0$ per cui la funzione vale 0, mentre cambiando x_j da 0 a 1 per lo stesso mintermine la funzione vale 1, si puo' definire un cubo dove la variabile x_j vale 1 e il resto e' uguale al mintermine considerato. In questo modo per ogni mintermine dove la funzione vale 1 si puo' definire un cubo dove la variabile x_j vale 2 o 1 e in tal modo si costruisce una copertura monotona crescente in x_j.

3. (a) Si ottenga il diagramma di decisione binaria ridotto e ordinato secondo l'ordine $a < b < c < d$ (senza lati complementati) della funzione

$$f = abd' + ab'd + a'c + a'c'd.$$

Si mostrino i passi del procedimento seguito.

Traccia di soluzione.

Si applichi la procedura *robdd_build*.

.

(b) Si ottenga il diagramma di decisione binaria ridotto e ordinato della precedente funzione usando anche i lati complementati, trasformando il diagramma precedente. Si mostrino i passi del procedimento seguito.

Si adotti la convenzione che le negazioni compaiono solo sui lati *else*, per cui i lati *else* si etichettano con un cerchietto vuoto se sono regolari o con un cerchietto pieno se sono complementati (regola per verificare il diagramma della funzione: poiche' i lati *then* non sono mai complementati, il valore di una funzione per il mintermine $1, 1, \ldots, 1$ sara' 0 se e solo se il lato uscente dal nodo associato alla funzione e' complementato).

Traccia di soluzione.
Si converta il grafo ottenuto in precedenza propagando una negazione dalla radice alle foglie. Si ottiene:

```
radice f: (un solo) lato complementato punta a nodo n1 con variabile a
nodo n1: lato T punta a nodo n2 con variabile b
         lato E complementato punta a nodo n3 con variabile c
nodo n2: lato T punta a nodo n4 con variabile d
         lato E complementato punta a nodo n4 con variabile d
nodo n3: lato T punta a nodo '1'
         lato E punta a nodo n4 con variabile d
nodo n4: lato T punta a nodo '1'
         lato E complementato punta a nodo '1'
```

Si noti che il grafo con i lati complementati ha 3 nodi in meno del precedente.

.

4. Si consideri il codice VHDL incluso e se ne spieghi concisamente il funzionamento. In particolare si scriva la tavola delle operazioni del contatore c74163/b74163 e si disegni lo schema a blocchi del contatore c74163test/tester.

```
-- 74163 FULLY SYNCHRONOUS COUNTER

library BITLIB;                 -- contains int2vec and vec2int functions
use BITLIB.bit_pack.all;

entity c74163 is
   port(LdN, ClrN, P, T, CK: in bit;  D: in bit_vector(3 downto 0);
        Cout: out bit; Q: inout bit_vector(3 downto 0) );
end c74163;

architecture b74163 of c74163 is
begin
   Cout <= Q(3) and Q(2) and Q(1) and Q(0) and T;
   process
   begin
      wait until CK = '1';              -- change state on rising edge
      if ClrN = '0' then  Q <= "0000";
         elsif LdN = '0' then Q <= D;
         elsif (P and T) = '1' then
           Q <= int2vec(vec2int(Q)+1,4);
      end if;
   end process;
end b74163;

library BITLIB;
use BITLIB.bit_pack.all;

entity c74163test is
        port(ClrN,LdN,P,T1,Clk: in bit;
                Din1, Din2: in bit_vector(3 downto 0);
                Qout1, Qout2: inout bit_vector(3 downto 0);
                Carry2: out bit);
end c74163test;

architecture tester of c74163test is
        component c74163
           port(LdN, ClrN, P, T, CK: in bit;  D: in bit_vector(3 downto 0);
           Cout: out bit; Q: inout bit_vector(3 downto 0) );
        end component;
        signal Carry1: bit;
        signal Count: integer;
        signal temp: bit_vector(7 downto 0);
begin
        ct1: c74163 port map (LdN,ClrN,P,T1,Clk,Din1,Carry1,Qout1);
        ct2: c74163 port map (LdN,ClrN,P,Carry1,Clk,Din2,Carry2,Qout2);
        temp <= Qout2 & Qout1;
        Count <= vec2int(temp);
end tester;
```

.

.