MININIZAREA funcțiilor logice

Vasile Airinei

Algoritmul Quinne - McCluskey, despre care vom vorbi în următoarele pagini, este un algoritm performant de minimizare a funcțiilor logice. Autorul a implementat acest algoritm folosind limbajul C++. Sursele programului pot fi obținute prin e-mail de la autor, la adresa airineiv@hotmail.com.

Funcții logice, fundamente teoretice

Minimizarea funcțiilor logice a constituit mereu un subiect atractiv pentru cei care lucrează în știința calculatoarelor.

Componentele electronice ale calculatorului suportă tensiuni electrice pe două niveluri: nivelul 0 logic corespunde tensiunilor din jurul valorii de 0V și nivelul 1 logic corespunde tensiunilor din jurul valorii de 5V. În cele ce urmează nivelurile de 0 logic si 1 logic le vom numi simplu 0 și 1.

Definiție

Se numește ***funcție logică*** o funcție de *n* variabile, notată:

$$f: \{0,1\}^n \rightarrow \{0, 1\} \qquad (1)$$

Ne putem imagina o astfel de funcție ca fiind un circuit logic cu *n* intrări (cele *n* variabile) și o singură ieșire (valoarea funcției *f*). Deoarece circuitul are *n* intrări, și fiecare intrare poate avea două valori (0 sau 1) rezultă că putem avea 2^n configurații ale vectorului de intrare $[x_1, x_2, \ldots, x_n]$. Pentru fiecare configurație a vectorului de intrare vom avea o valoare a funcției *f*, iar această valoare va fi un element al mulțimii {0, 1}.

În expresia unei funcții logice pot apărea mai multe sume și produse de variabile. O sumă va fi numită *disjuncție* logică, iar un produs va fi numit *conjuncție* logică.

Așadar, funcția *f* se poate scrie ca o sumă de produse, fiecare produs corespunzând unei configurații a vectorului de intrare. Această formă de scriere a funcției *f* se numește forma normală disjunctivă. Fiecare produs din această sumă se numește ***minterm***.

Funcția e complet definită dacă se cunosc toate valorile sale și incomplet definită dacă unele nu sunt cunoscute.

Pentru simplitate vom trata mai întâi cazul în care funcția este complet definită. Vom numi ***literal*** o variabilă logică sau negația acesteia.

Apare, evident, interesul ca expresia funcției logice să fie cât mai simplă. Cu cât expresia funcției logice este mai simplă, cu atât costul componentelor hardware (porți logice, *buffer*-e) pentru implementarea funcției va fi mai scăzut.

S-au descoperit câteva metode de minimizare a funcțiilor logice (metoda ***Quinne - McCluskey***, metoda hărții ***Veitch - Karnaugh***). În continuare vom prezenta prima dintre aceste metode.

Metoda Quinne - McCluskey

Metoda Quinne - McCluskey este folosită pentru a reduce forma normală disjunctivă a funcției cu scopul obținerii unei sume cu număr minim de termeni. Procedura constă în doi pași principali:

1. Aplicarea sistematică a teoremei $XY + XY' = X$ (unde Y' reprezintă valoarea negată a lui Y) pentru a elimina *literalii* redundanți. Termenii rezultați sunt numiți ***implicanți primi***.
2. Se folosește o *hartă a implicanților primi* pentru a selecta o mulțime minimală de *implicanți primi* reprezentând funcția simplificată, care conține un număr minim de *literali*.

Determinarea implicanților primi

Pentru a determina expresia minimă a sumei de produse cu ajutorul metodei *Quinne - McCluskey*, funcția trebuie dată ca o sumă de *mintermi*. La primul pas al algoritmului sunt determinați toți *implicanții primi* ai funcției prin combinarea *mintermilor*. *Mintermii* sunt reprezentați în notație binară și sunt combinați folosind formula $XY + XY' = X$, unde X reprezintă un produs de *literali* și Y este un singur *literal*. Doi *mintermi* pot fi combinați numai dacă un singur literal din componența lor este diferit, restul fiind identici. Exemplele prezentate mai jos arată atât notațiile binare, cât și echivalentul algebric.

Exemple

AB'CD' + *AB'CD* = *AB'C*

1 0 1 0 1 0 1 1 = 1 0 1 _

(liniuța indică lipsa unei variabile).

Dacă în expresia de mai sus se înlocuiește produsul de *literali AB'C* cu *X*, și *literalul D* cu *Y* se obține formula *XY* + *XY'* = *X*.

A'BC'D + *A'BCD'* (nu pot fi combinați)

0 1 0 1 0 1 1 0 (nu pot fi combinați)

Pentru a găsi toți *implicanții primi*, vor fi luate în considerare toate perechile de *mintermi* și dacă este posibil, vor fi combinate. Pentru reducerea numărului de comparări, *mintermii* binari sunt sortați în grupe, în funcție de numărul biților de 1 din reprezentarea binară a fiecărui *minterm*.

În cele ce urmează vom scrie o funcție logică sub forma unei liste de numere zecimale. În listă vor apărea toți echivalenții zecimali ai acelor valori binare de la intrare pentru care funcția ia valoarea 1. Astfel, dacă avem o funcție logică de patru variabile, care pentru un vector de intrare [*a*, *b*, *c*, *d*] = 1011 scoate la ieșire valoarea 1, atunci în lista de numere zecimale care definește funcția va apărea echivalentul zecimal al numărului binar 1011, adică 11.

De exemplu, funcția logică:

$f(a, b, c, d) = \varphi m(0, 1, 2, 5, 6, 7, 8, 9, 10, 14)$ **(2)**

este reprezentată de următoarea listă de *mintermi*:

grupul 0: 0 0000
grupul 1: 1 0001
2 0010
8 1000
grupul 2: 5 0101
6 0110
9 1001
10 1010
grupul 3: 7 0111
14 1110

În lista de mai sus, termenii din grupa 0 nu au nici un bit cu valoarea 1, termenii din grupa 1 au câte un bit cu valoarea 1, cei din grupa 2 au câte doi biți cu valoarea 1, iar cei din grupa 3 au câte trei biți cu valoarea 1.

Așa cum am afirmat anterior, doi *mintermi* pot fi combinați dacă ei diferă în exact o variabilă.

Vom spune că două grupe sunt adiacente dacă numerele care reprezintă grupa diferă printr-o unitate. De exemplu, grupele 0 și 1 sunt adiacente; la fel, grupele 3 și 4. Grupele 0 și 2 sunt neadiacente

Compararea termenilor din grupe neadiacente este inutilă, deoarece acei termeni vor diferi întotdeauna în cel puțin două variabile și nu pot fi combinate folosind relația *XY* + *XY'* = *X*. Similar, compararea termenilor din aceeași grupă este inutilă, deoarece doi termeni cu același număr de biți având valoarea 1 vor diferi în cel puțin două variabile. Așadar, numai termenii din grupe adiacente vor fi comparați.

Întâi vom compara termenii din grupa 0 cu termenii din grupa 1. Termenii 0000 și 0001 pot fi combinați pentru a elimina variabila *d*, rezultând termenul 000-. Analog termenii 0000 și 0100 pot fi combinați și se obține termenul 00-0 (se elimină variabila *c*). Termenii rezultați sunt prezentați în coloana a doua a tabelului 1.

Numerele zecimale corespunzătoare unor termeni care pot fi combinați diferă printr-o putere a lui 2 (1, 2, 4, 8, 16 etc.). Această afirmație este adevărată deoarece, dacă reprezentările binare diferă în exact o coloană, atunci diferența numerelor corespunzătoare conține un singur bit cu valoarea 1 pe poziția în care există diferențe. Un număr binar în a cărui reprezentare binară există un singur bit cu valoarea 1 este o putere a lui 2.

Deoarece compararea membrilor grupului 0 cu membrii grupurilor 2 și 3 este inutilă, trecem la compararea termenilor din grupurile 1 și 2. Comparând *mintermul* 0001 cu toți termii din grupul 2, observăm că poate fi combinat cu termii 0101 și 1001, dar nu și cu termii 0110 și 1010. Similar, termul 0010 poate fi combinat numai cu termii 0110 și 1010. Termul 1000 poate fi combinat doar cu termii 1001 și 1010. Termii obținuți apar în coloana a doua. De fiecare dată când combinăm un term cu un altul, acesta este "bifat". Un term poate fi folosit de mai multe ori deoarece *X* + *X* = *X*. Chiar dacă doi termi au fost deja comparați cu alți termi, ei trebuie comparați și combinați între ei dacă e posibil. Această operație este necesară deoarece termul rezultat poate fi necesar pentru formarea soluției. În această fază, putem genera termi redundanți, dar aceștia vor fi eliminați ulterior.

Încheiem prima coloană prin compararea termilor din grupele 2 și 3. Noi termeni sunt formați prin compararea termilor 0101 și 0111, 0110 și 0111, sau 1010 și 1110.

	coloana 1	coloana 2	coloana 3
grupa 0	0 0000 *	0, 1 000- *	0, 1, 8, 9 -00-
		0, 2 00-0 *	0, 2, 8, 10 -0-0
grupa 1	1 0001 *	0, 8 -000 *	~~0, 8, 1, 9 -00-~~
	2 0010 *		~~0, 8, 2, 10 -0-0~~
	8 1000 *	1, 5 0-01 *	
		1, 9 -001 *	2, 6, 10, 14 --10
grupa 2	5 0101 *	2, 6 0-10 *	~~2, 10, 6, 14 --10~~
	6 0110 *	2, 10 -010 *	
	9 1001 *	8, 9 100- *	
	10 1010 *	8, 10 10-0 *	
grupa 3	7 0111 *	5, 7 01-1	
	14 1110 *	5, 7 011- *	
		6, 14 -110 *	
		10, 14 -110 *	

Tabelul 1: Determinarea implicanților primi

Observăm că termenii din coloana a doua sunt împărțiți în grupe în funcție de numărul de biți cu valoarea 1 din reprezentarea binară a fiecărui term. Aplicăm formula *XY* + *XY'* = *X* pentru a combina perechi de termeni în coloana a doua. Pentru a combina doi termi ei trebuie să aibă aceleași variabile termenii trebuie să difere în exact una dintre

aceste variabile. Așadar, este necesară doar compararea termenilor care au liniuța pe aceeași poziție și al căror număr de biți cu valoarea 1 diferă exact cu 1. *Termii* aflați în prima grupă în a doua coloană vor fi comparați numai cu *termii* din grupa a doua care au liniuța pe aceeași poziție. *Termul* 000- (0,1) poate fi combinat numai cu *termul* 100- (8, 9) și se obține -00-. Acesta e echivalentul algebric al relației $a'b'c' + ab'c' = b'c'$. *Termul* rezultat este listat în coloana a treia prin descrierea 0, 1, 8, 9. *Termul* (0, 2) se poate combina numai cu (8, 10) și *termul* (0, 8) se combină numai cu (1, 9) și (2, 10). Și în acest caz, *termii* care au fost deja combinați sunt bifați. Comparând *termii* din a doua și a treia grupă în a doua coloană observăm că (2, 6) poate fi combinat cu (10, 14), iar (2, 10) poate fi combinat cu (6, 14).

Observăm că sunt trei perechi de *termi* duplicate în coloana a treia. Acești *termi* duplicați au fost formați în fiecare caz prin combinarea aceluiași set de patru *mintermi* în ordine diferită. După eliminarea *termilor* duplicați, comparăm *termii* din cele două grupe din a treia coloană. Întrucât nici o combinație nu mai este posibilă, procesul se termină. În general, vom păstra *termii* comparați și vom forma noi grupe de *termi* și noi coloane până când nu se mai pot combina *termi*.

Termii care nu mai pot fi combinați se numesc *implicanți primi*. Deoarece fiecare *minterm* este inclus în cel puțin unul din *implicanții primi*, funcția este egală cu suma *implicanților primi*. În acest exemplu, avem:

$$f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd' \quad (3)$$

(1,5) (5,7) (6,7) (0,1,8,9) (0,2,8,10) (2,6,10,14)

În expresia de mai sus, fiecare *term* are număr minim de *literali*, dar numărul de *termi* nu este minim. Folosind teorema pentru eliminarea *termilor* redundanți obținem:

$$f = a'bd + b'c' + cd' \quad (4),$$

care este expresia minimă a lui *f* scrisă ca sumă de produse. În secțiunea următoare vom prezenta o metodă mai performantă pentru eliminarea *implicanților primi* redundanți, folosind o hartă a implicanților primi.

În continuare, vom defini noțiunile de ***implicant*** și ***implicant prim***.

Definiție:

Pentru o funcție *F* de *n* variabile, un produs de *termi P* este un ***implicant*** al lui *F* dacă pentru fiecare combinație de valori de *n* variabile pentru care *P* = 1, *F* are, de asemenea, valoarea 1.

Cu alte cuvinte, dacă există anumite combinații de valori pentru variabile, *P* = 1 și *F* = 0, atunci *P* nu este ***implicant*** al lui *F*. De exemplu, considerăm funcția :

$$F(a, b, c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac \quad (5)$$

Dacă $a'b'c' = 1$, atunci *F* = 1, dacă $ac = 1$, atunci *F* = 1 etc. Astfel, *termii* $a'b'c'$, *ac*, sunt *implicanți* ai lui *F*. În acest exemplu, *bc* nu este un *implicant* al lui F deoarece, dacă *a* = 0 și *b* = *c* = 1, atunci *bc* = 1 și *F* = 0. În general, dacă *F* este scrisă sub forma sumei de produse, fiecare *term* este un *implicant*. Fiecare *minterm* al lui *F* este, de asemenea, un *implicant* al lui *F* și, prin urmare, orice *term* format prin combinarea a doi sau mai mulți *mintermi* va fi un *implicant* al lui *F*. De exemplu, în tabelul 1, toți *termii* sunt listați în fiecare coloană în care sunt implicanți ai funcției dată de ecuația (2).

Definiție

Un ***implicant prim*** al funcției *F* este un *implicant* din care, dacă prin eliminarea unui *literal* din componența acestuia nu se obține alt *implicant*.

În ecuația (5), *implicantul* $a'b'c'$ nu este *implicant prim* deoarece *a'* poate fi eliminat și *termenul* rezultat ($b'c'$) este un *implicant* al lui *F*. *Implicanții* $b'c'$ și *ac* sunt *implicanți primi* deoarece, dacă ștergem un *literal* din fiecare *term*, *termul* rezultat nu va mai fi *implicant* al lui *F*. Fiecare *implicant prim* al funcției are un număr minim de *literali* în sensul că nu mai pot fi eliminați *literali* prin combinarea cu alți *termi*.

Expresia minimă sumă de produse pentru o funcție constă dintr-o sumă de anumiți implicanți primi ai funcției. Dacă o expresie sumă de produse conține un term care nu e implicant prim, atunci acea expresie nu este minimizată, deoarece acel term poate fi înlocuit cu un implicant prim. Această afirmație este adevărată deoarece termii neprimi nu conțin un număr minim de *literali* - ei pot fi combinați cu *mintermi* adiționali pentru a forma un implicant prim care are mai puțini *literali* decât *termii* neprimi. Un term neprim într-o sumă de produse poate fi deci înlocuit cu un implicant prim, care reduce numărul de *literali* și simplifică expresia.

Harta implicanților primi

A doua parte a procedurii *Quinne - McCluskey* dezvoltă o hartă pentru a selecta setul minim de implicanți primi. Un implicant prim este egal cu o sumă de mintermi și se spune că implicantul prim acoperă acei mintermi. Dacă un implicant prim acoperă un minterm dat, un × este plasat la intersecția liniei și coloanei corespunzătoare. Tabelul 2 prezintă harta implicanților primi dedusă din tabelul 1. Toți implicanții primi (termi care nu au fost eliminați din tabelul 1) sunt listați în partea stângă.

i.p. \ m	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	×	×					×	⊗		
(0, 2, 8, 10)	×		×				×		×	
(2, 6, 10, 14)			×		×				×	⊗
(1, 5)		×		×						
(5, 7)				×		×				
(6, 7)					×	×				

m - mintermi
i.p. - implicanți primi

Tabelul 2: Harta implicanților primi

În prima linie, ×-urile sunt plasate în coloanele 0, 1, 8 și 9 deoarece implicantul $b'c'$ a fost format din suma de

mintermi 0, 1, 8, 9. Similar, ×-urile sunt plasate în coloanele 0, 2, 8, 10 corespunzător implicantului prim *b'd'* ș.a.m.d.

Dacă un *minterm* este acoperit numai de un implicant prim, atunci cel din urmă este numit *implicant prim esențial* și trebuie inclus în suma minimă de produse. Implicanții primi esențiali sunt ușor de găsit folosind harta implicanților primi. Dacă o coloană dată conține numai un singur ×, atunci linia pe care se află ×-ul corespunde unui implicant prim esențial. În tabelul 2, coloanele 9 și 14 conțin fiecare câte un ×, așadar implicanții primi *b'c'* și *cd'* sunt esențiali.

De fiecare dată când un implicant prim este selectat pentru includerea în suma minimă, linia corespunzătoare este eliminată. După realizarea acestei operații, coloanele care corespund tuturor *mintermilor* acoperiți de acest implicant prim vor fi eliminate.

Tabelul 3 ilustrează harta rezultată când sunt eliminați implicanții primi esențiali, împreună cu liniile și coloanele corespunzătoare din tabelul 2. În continuare, trebuie ales un set minim de implicanți primi pentru a acoperi coloanele rămase. În acest exemplu, *a'bd* acoperă cele două coloane rămase, deci poate fi implicantul prim ales. Suma minimă de produse care rezultă este:

$$f = b'c' + cd' + a'bd,$$

care este identică cu cea din ecuația (**4**).

Observăm că, chiar dacă termul *a'bd* este inclus în suma minimă de produse, *a'bd* nu este un implicant prim esențial. El reprezintă suma de *mintermi* m_5, m_7; m_5 este acoperit și de *a'c'd*, iar m_7 este acoperit și de *a'bc*.

i.p. \ m	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	×	×					×	⊗		
(0, 2, 8, 10)	×		×				×		×	
(2, 6, 10, 14)			×	×	×				⊗	⊗
(1, 5)		×		×						
(5, 7)				×		×				
(6, 7)					×	×				

Tabelul 3

După construirea hărții implicanților primi, anumiți *mintermi* pot fi acoperiți de numai un singur implicant prim, în timp ce alți *mintermi* pot fi acoperiți de doi sau mai mulți implicanți primi. Un implicant prim este esențial (sau necesar) dacă el conține un *minterm* care nu e acoperit de alt implicant al lui *f*. Implicanții primi esențiali sunt aleși la început deoarece toți implicanții primi esențiali trebuie incluși în fiecare sumă minimă. După ce au fost aleși implicanții primi esențiali, *mintermii* pe care îi acoperă pot fi eliminați din harta implicanților primi traversând coloanele corespunzătoare. Dacă implicanții primi esențiali nu acoperă toți *mintermii*, atunci sunt necesari implicanți primi neesențiali adiționali. În cazuri simple, implicanții primi neesențiali necesari pentru formarea soluției minime pot fi selectați prin probe. Pentru hărți mai mari, trebuie folosite proceduri pentru reducerea hărții care vor fi prezentate ulterior. Anumite funcții au două sau mai multe expresii minime sume de produse, fiecare având același număr de *termi* și *literali*. Următorul exemplu ilustrează o asemenea funcție.

Exemplu cu hartă ciclică de implicanți primi

O hartă a implicanților care are două sau mai multe ×-uri în fiecare coloană este numită ***hartă ciclică***. Următoarea funcție are o asemenea hartă:

$$f = \varphi m(0, 1, 2, 5, 6, 7) \qquad (6)$$

Formarea implicanților primi se face astfel:

0	000 *	0,1	00-
1	001 *	0,2	0-0
2	010 *	1,5	-01
5	101 *	2,6	-10
6	110 *	5,7	1-1
7	111 *	6,7	11-

Tabelul 4

Tabelul 5 ilustrează harta implicanților primi care rezultă.

	0	1	2	5	6	7
① → (0, 1)	×	×				
(0, 2)	×		×			
(1, 5)		×		×		
② → (2, 6)			×		×	
③ → (5, 7)				×		×
(6, 7)					×	×

Tabelul 5

Toate coloanele au două ×-uri, așadar trebuie efectuate probe. (0, 1) și (0, 2) acoperă coloana 0; vom încerca (0, 1). După traversarea liniei (0, 1) și coloanele 0 și 1, vom examina coloana 2 care e acoperită de (0, 2) și (2, 6). Cea mai bună alegere este (2, 6) deoarece acoperă două din coloanele rămase, în timp ce (0, 2) acoperă numai una din coloanele rămase. După traversarea liniei (2, 6) și coloanele 2 și 6, observăm că (5, 7) acoperă coloanele rămase și completează soluția. Așadar o soluție este:

$$F = a'b' + bc' + ac.$$

Totuși, nu avem garanția că această soluție este minimă. Trebuie să reluăm algoritmul, folosind un alt implicant care acoperă coloana 0. Rezultă tabelul 6.

		0	1	2	5	6	7
P1	(0, 1)	×	×				
P2	(0, 2)	×		×			
P3	(1, 5)		×		×		
P4	(2, 6)			×		×	
P5	(5, 7)				×		×
P6	(6, 7)					×	×

Tabelul 6

În sfârșit, obținem soluția:

$$F = a'c' + b'c + ab$$

Întrucât această soluție are același număr de termeni și același număr de literali ca expresia lui F derivată din tabelul 5, avem 2 soluții sume de produs minime pentru această problemă.

Metoda Petrick

Aceasta este o metodă pentru determinarea tuturor soluțiilor sumă de produs din harta implicanților primi. Exemplul corespunzător tabelelor 5 și 6 are două soluții minime. Creșterea numărului de variabile duce la creșterea semnificativă a numărului de implicanți primi și a complexității hărții implicanților primi. În asemenea cazuri, pentru obținerea soluției minime este necesară efectuarea unui număr mare de probe.

Metoda Petrick este un mod mai sistematic de găsire a tuturor soluțiilor minime din harta implicanților primi. Înaintea aplicării *metodei Petrick*, toți implicanții primi esențiali și mintermii care îi acoperă trebuie șterși de pe hartă.

Vom ilustra *metoda Petrick* folosind tabelul 6.

Întâi vom eticheta liniile P_1, P_2, P_3 etc. Vom construi o funcție logică P, care e adevărată când toți mintermii în hartă au fost acoperiți. Fie P_1 o variabilă logică care este adevărată când implicantul prim din linia P_1 este inclus în soluție; fie P_2 o variabilă logică care e adevărată când implicantul prim din linia P_2 este inclus în soluție etc. Deoarece coloana 0 conține ×-uri în liniile P_1 și P_2, trebuie să alegem una din liniile P_1 sau P_2 pentru a acoperi mintermul 0. Expresia $(P_1 + P_2)$ trebuie să fie adevărată. Pentru a acoperi mintermul 1, trebuie să alegem liniile P_1 sau P_3; atunci $(P_1 + P_3)$ trebuie să fie adevărată. Pentru a acoperi mintermul 2, $(P_2 + P_4)$ trebuie să fie adevărată. Similar, pentru a acoperi mintermii 5, 6 și 7, expresiile $(P_3 + P_5)$, $(P_4 + P_6)$ și $(P_5 + P_6)$ trebuie să fie adevărate. Obținem expresia:

$$P = (P_1 + P_2) \cdot (P_1 + P_3) \cdot (P_2 + P_4) \cdot (P_3 + P_5) \cdot \\ \cdot (P_4 + P_6) \cdot (P_5 + P_6) = 1.$$

Această expresie indică faptul că trebuie să alegem linia P_1 sau P_2 și linia P_1 sau P_3 și linia P_2 sau P_4 etc.

Următorul pas este de a reduce P la o sumă minimă de produse. Operația este mai ușor de realizat deoarece nu există complemente. Folosim relația $(X + Y) \cdot (X + Z) = X+YZ$ și legea distributivității:

$$P = (P_1 + P_2P_3) \cdot (P_4 + P_2P_6) \cdot (P_5 + P_3P_6) = \\ = (P_1P_4 + P_1P_2P_6 + P_2P_3P_4 + P_2P_3P_6) \cdot (P_5 + P_3P_6) = \\ = P_1P_4P_5 + P_1P_2P_5P_6 + P_2P_3P_4P_5 + P_2P_3P_5P_6 + \\ + P_1P_3P_4P_6 + P_1P_2P_3P_6 + P_2P_3P_4P_6 + P_2P_3P_6.$$

În continuare, vom folosi relația $X + XY = X$ pentru a elimina termenii redundanți din P și obținem:

$$P = P_1P_4P_5 + P_1P_2P_5P_6 + P_2P_3P_4P_5 + P_1P_3P_4P_6 + P_2P_3P_6.$$

Pentru a acoperi toți mintermii trebuie să fie satisfăcută relația $P = 1$. Așadar, vom alege fie liniile P_1, P_4 și P_5, fie liniile P_1, P_2, P_5 și P_6, fie liniile P_2, P_3, P_4 și P_5, fie liniile P_1, P_3, P_4 și P_6, fie liniile P_2, P_3 și P_6. Sunt cinci soluții posibile, iar numai două dintre ele au un număr minim de linii. În concluzie, cele două soluții cu număr minim de implicanți primi sunt: liniile P_1, P_4 și P_5 sau liniile P_2, P_3 și P_6. Prima alegere conduce la relația:

$$F = a'b' + bc' + ac.$$

Alegând a doua opțiune, obținem:

$$F = a'c' + b'c' + ab.$$

Pe scurt, *metoda Petrick* poate fi descrisă astfel:

- Se reduce harta implicanților primi prin eliminarea liniilor implicanților primi esențiali și coloanele corespunzătoare;
- Se etichetează liniile hărții reduse de implicanții primi P_1, P_2, P_3 etc;
- Se determină o funcție logică P care este adevărată când toate coloanele sunt acoperite. P este, de fapt, un produs de sume de termi, fiecare sumă având forma $(P_{i_0} + P_{i_1} + ...)$, unde P_{i_0}, P_{i_1}, ... reprezintă liniile care acoperă coloana i;
- Se reduce P la o sumă minimă de produse prin înmulțire și aplicarea legii $X + XY = X$;
- Fiecare term din rezultat reprezintă o soluție, care este un set de linii care acoperă toți mintermii în tabel. Pentru a determina soluțiile minime, găsim acei termi care conțin un număr minim de variabile. Fiecare din acești termi reprezintă o soluție cu un număr minim de implicanți primi;
- Pentru fiecare din termii găsiți la pasul 5, numărăm literalii din fiecare implicant prim și reținem numărul de literali. Alegem termul sau termii care corespund unui număr minim de literali, și avem astfel sumele corespunzătoare de implicanți primi.

Aplicarea *metodei Petrick* este foarte utilă pentru hărți "incomode" și e relativ ușor de implementat.

Simplificarea funcțiilor incomplet definite

Termenii indiferenți corespund acelor vectori de intrare pentru care nu se cunoaște valoarea funcției logice de la ieșirea circuitului. Lista termenilor indiferenți va fi separată de lista mintermilor în expresia funcției logice. De exemplu, dacă avem o funcție logică de patru variabile a cărei valoare nu se cunoaște pentru vectorul de intrare [a, b, c, d] = 1000, atunci echivalentul zecimal al valorii binare 1000, adică 8, va face parte din lista termenilor indiferenți.

În cazul în care avem o funcție incomplet definită, trebuie să luăm în considerare și valorile termenilor indiferenți pentru a obține forma minimă a funcției. În această secțiune vom arăta cum se modifică procedura *Quinne - McCluskey* pentru a obține soluția minimă când sunt prezenți termeni indiferenți. În scopul găsirii implicanților primi, vom trata termenii indiferenți ca și cum ar fi mintermi. Astfel, ei pot fi combinați cu alți mintermi pentru a elimina cât mai mulți literali. Dacă se formează implicanți primi din termeni indiferenți, acei implicanți nu vor fi luați în considerare. În partea de sus a hărții implicanților primi nu vom trece termenii indiferenți. Când harta implicanților primi este rezolvată toți mintermii necesari vor fi aco-

periți de unul sau mai mulți implicanți primi selectați. Termenii indiferenți nu sunt incluși în soluția finală fără să fie implicați în procesul de formare a unuia dintre implicanții primi selectați. Următorul exemplu va clarifica procedura.

Exemplu de simplificare a funcțiilor incomplet definite

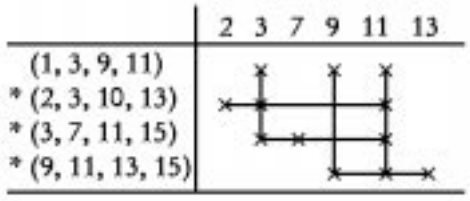
$F(A, B, C, D) = \sum m(2, 3, 7, 9, 13) + \sum d(1, 10, 15)$
(mintermii care urmează după *d* sunt indiferenți)

Termenii indiferenți sunt tratați ca mintermi necesari când găsim implicanții primi:

1 0001 *	(1, 3) 00-1 *	(1, 3, 9, 11) -0-1
2 0010 *	(1, 9) -001 *	(2, 3, 10, 11) -01-
3 0011 *	(2, 3) 001- *	(3, 7, 11, 15) --11
9 1001 *	(2, 10) -010 *	(9, 11, 13, 15) -0-1
10 1010 *	(3, 7) 0-11 *	
7 0111 *	(3, 11) -011 *	
11 1011 *	(9, 11) 10-1 *	
13 1101 *	(9, 13) 1-01 *	
15 1111 *	(10, 11) 101- *	
	(7, 15) -111 *	
	(11, 15) 1-11 *	
	(13, 15) 11-1 *	

Tabelul 7

Coloanele indiferente sunt omise când formăm harta implicanților primi, așa cum se observă în tabelul 8:



* - indică un implicant prim esențial
$F = B'C + CD + AD$

Tabelul 8

Funcția originală a fost incomplet definită; expresia finală simplificată pentru *F* este definită pentru toate combinațiile de valori pentru *A*, *B*, *C*, *D* și, prin urmare, este complet definită. În procesul simplificării am luat în considerare valorile indiferente în tabela de adevăr a lui *F*. Dacă înlocuim fiecare term în expresia finală a lui *F* prin suma de mintermi corespunzătoare, rezultatul este:

$F = (m_2 + m_3 + m_{10} + m_{11}) + (m_3 + m_7 + m_{11} + m_{15}) + (m_9 + m_{11} + m_{13} + m_{15})$,

unde termenii duplicați sunt evidențiați.

Deoarece m_{10} și m_{15} apar în expresie, în timp ce m_1 nu apare, rezultă că termenii indiferenți în tabela de adevăr a lui *F* au atribuite valorile:

- pentru *ABCD* = 0001, *F* = 0;
- pentru *ABCD* = 1010, *F* = 1;
- pentru *ABCD* = 1111, *F* = 1.

Observații privitoare la codul sursă al programului care implementează algoritmul

Structurile de date necesare implementării algoritmului au fost construite pe baza bibliotecii de clase ***MFC*** (***M**icrosoft **F**oundation **C**lasses*). Aplicația are facilități deosebite: calculează timpul de execuție - care pentru un număr de variabile mai mare ca 21 ia valori considerabile - și afișează memoria consumată la executarea algoritmului. Aceste facilități au fost realizate folosind funcții din programarea aplicațiilor *Windows*. Funcția logică, dată ca o listă de numere zecimale, este preluată dintr-un fișier de intrare (care are extensia `.IN`), iar expresia minimizată a funcției logice este afișată într-un fișier de ieșire (care are extensia `.OUT`).

Concluzii

Pornind de la implementarea acestui algoritm se pot realiza aplicații deosebite pentru cei care lucrează în domeniul *hardware*.

Bibliografie:

Edward J. Mc. Cluskey, *Logic design. Principles*, Prentice Hall Inc., Englewood Cliffs, N.J., 1986

Dl. Airinei Vasile este absolvent al Facultății de Automatică și Calculatoare din Iași. În prezent este inginer programator la S.C. Hidroconstrucția S.A., sucursala Sebeș. Poate fi contactat prin e-mail la airineiv@yahoo.com.

Definiții...

Hardware: componentele sistemului care pot fi lovite cu piciorul;
Software: componentele sistemului care nu funcționează;
Memorie: componenta sistemului care este întotdeauna insuficientă;
Imprimantă: componenta care se blochează doar atunci când nu este supravegheată de utilizator;
Procesor: componenta sistemului care este depășită din punct de vedere tehnologic;
Cablu: componenta sistemului care este întotdeauna prea scurtă;
BackUp: operație care nu este efectuată niciodată la timp;
Restaurare: operație care funcționează perfect până în momentul în care este nevoie de ea;