

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Master: Circuite și Sisteme Integrate, An II, 2020-2021

Minimizarea euristica folosind algoritmul ESSPRESO

Student: Cibu Oana

**Introducere**

Este normal ca implementarea cu porţi logice a unei funcţii binare să se facă pornind de la o exprexie booleană cât mai simplă, care se numeşte **forma minimă** a funcţiei. Există mai multe metode de minimizare care pot fi utilizate de proiectant: **minimizarea algebrică**, **minimizarea cu diagrame Veitch-Karnaugh**, **metoda Quine-McCluskey**, **metoda consensurilor** sau **metoda ESPRESSO**.

Minimizarea manuală a funcțiilor booleane folosind diagramele clasice Karnaugh este un proces laborios, obositor și predispus la erori. Nu este potrivit pentru mai mult de șase variabile de intrare și practic numai pentru până la patru variabile, în timp ce distribuirea termenului de produs pentru mai multe funcții de ieșire este chiar mai dificil de realizat. Mai mult, această metodă nu se pretează să fie automatizată sub forma unui program de calculator. Deoarece funcțiile logice moderne nu sunt în general limitate la un număr atât de mic de variabile, în timp ce costul, precum și riscul de a comite erori este prohibitiv pentru implementarea manuală a funcțiilor logice, utilizarea computerelor a devenit indispensabilă.

Prima metodă alternativă care a devenit populară a fost metoda tabulară dezvoltată de Willard Quine și Edward McCluskey. Aceastǎ metodǎ este utilizabilǎ, în principiu, pentru minimizarea funcţiilor scalare ori vectoriale cu un numǎr arbitrar de variabile. Metoda se poate utiliza pentru programarea unui calcul automat de minimizare. Minimizarea are loc în doi paşi: (1) stabilirea implicanţilor primi, şi (2) calculul acoperirii minimale pentru funcţia respectivǎ. Metoda utilizeazǎ scrierea binarǎ a mintermilor funcţiei şi printr-o ordonare judicioasǎ faciliteazǎ gǎsirea implicanţilor adiacenţi dar şi calculul implicanţilor de ordin superior rezultaţi. Într-o primǎ abordare vor fi prezentate principalele repere ale metodei aplicate funcţiilor scalare. Acestea vor facilita mult înţelegerea metodei în cazul general, funcţiilor booleene vectoriale.

Deși acest algoritm Quine-McCluskey este foarte potrivit pentru a fi implementat într-un program de calculator, rezultatul este încă departe de a fi eficient în ceea ce privește timpul de procesare și utilizarea memoriei. Adăugarea unei variabile la funcție va dubla aproximativ ambele, deoarece lungimea tabelului de adevăr crește exponențial cu numărul de variabile. O problemă similară apare la creșterea numărului de funcții de ieșire ale unui bloc funcțional combinațional. Drept urmare, metoda Quine-McCluskey este practică numai pentru funcțiile cu un număr limitat de variabile de intrare și funcții de ieșire.

**Metoda ESPRESSO** este folosită mai ales în cazul funcţiilor cu număr foarte mare de variabile. Ea foloseşte o strategie euristică bazată pe aproximaţie, care evită construcţia listelor de termeni şi minimizează timpul de calcul

"Dacă timpul în care este găsită o soluţie optimă nu contează, atunci există întotdeauna o cale de a o obţine. Dar, aproape întotdeauna, timpul contează. Degeaba ştim să facem un lucru dacă nu ajungem la soluţie în timp util. Acesta este motivul pentru care uneori ne mulţumim şi cu soluţii numai parţial optimizate" .

Metoda ESPRESSO se aplică repetitiv şi generează de fiecare dată expresii boolene disjunctive mai simple decât precedentele lor. Fără a garanta forma minimă absolută, metoda oferă rezultate comparabile cu cele obţinute prin metoda Quine-McCluskey.

Algoritmul folosit pentru minimizarea funcţiei *F* presupune parcurgerea următoarelor etape:

1. se aplică procedura EXPAND funcţiilor *F* şi *F* , rezultatul fiind expresia redusă a funcţiei *F* , notată cu *F* \* .
2. se aplică procedura IRREDUNDANT expresiilor *F* \* şi Ø, rezultând o nouă expresie redusă notată tot cu *F* \* .
3. Se aplică procedura ESSENTIAL\_PRIMES expresiei *F* \* , iar rezultatul este expresia *E*.
4. Rezultatul procedurii anterioare se notează cu *D* şi se elimină din expresia *F* \* , iar rezultatul este notat tot cu *F* \* .
5. Se aplică procedura REDUCE expresiei notate cu *F* \* şi expresiei *D*, iar rezultatul este notat cu *F* \* .
6. Se aplică procedura EXPAND funcţiilor *F* \* şi *F* , iar rezultatul se notează cu *F* \* .
7. Se aplică procedura IRREDUNDANT expresiilor *F* \* şi *D*, rezultând o nouă expresie redusă notată cu *F* \* .
8. Dacă expresia redusă *F* \* este mai simplă decât cea folosită la pasul 5, atunci se reia de la pasul 5; în caz contrar, se introduce o "perturbaţie" pentru a scoate procesul dintr-un eventual minim local. Dacă se reuşeşte acest lucru, atunci se reia de la pasul 5, in caz contrar, forma minimă a funcţiei estef= *F* \* + *E* .

Minimizatorul logic Espresso este un program de calculator care utilizează algoritmi euristici și specifici pentru reducerea eficientă a complexității circuitelor cu porți logice digitale. Espresso a fost dezvoltat la IBM de Robert K. Brayton. Richard Rudell a publicat ulterior varianta Espresso-MV în 1986, sub titlul "Multiple-Valued Logic Minimization for PLA Synthesis". Espresso a inspirat multe derivate.

Algoritmul Espresso s-a dovedit atât de reușit încât a fost încorporat ca o etapă de minimizare a funcției logice standard în practic orice instrument de sinteză logică contemporană. Pentru implementarea unei funcții în logică pe mai multe niveluri, rezultatul minimizării este optimizat prin factorizare și mapat pe celulele logice de bază disponibile din tehnologia țintă, indiferent dacă este vorba despre un tablou de poartă programabil pe câmp (| FPGA) sau un circuit integrat specific aplicației (ASIC).

Espresso

Programul original Espresso este disponibil ca cod sursă C de la site-ul web al Universității din California, Berkeley. Ultima versiune a fost versiunea 2.3 din 1988. [9] Programul Espresso-ab și eqntott (ecuație la tabelul adevărului), o versiune actualizată a Espresso pentru sisteme moderne POSIX, este disponibil în formatul de fișier distribuție Linux (.deb) Debian Linux, precum și codul sursă C. Ultima versiune a fost versiunea 9.0 din 2008. [10]

Algoritmul espresso consta in 3 pasi de baza:

* Expandare
* Iredundant
* Reducere

**Expand,** acest operator calculează o acoperire primă şi minimală în raport cu conţinerea într-un singur implicant. Implicanţii care formează acoperirea sunt procesaţi rând pe rând. Fiecare implicant care nu este prim este expandat întrun implicant prim, adică este înlocuit printr- -un implicant prim care-l conţine.

**Consecutiv,** toţi implicanţii din acoperire (cei încă ne-procesaţi) care sunt conţinuţi în acest implicant prim sunt îndepărtaţi.

Irredundant, are drept acţiune calculul unei acoperiri iredundante. Operatorul alege un subset al implicanţilor astfel încât nici un implicant al acestui subset nu este acoperit de ceilalţi implicanţi din subset.

**Reduce,** este un operator care realizează transformarea unei acoperiri oarecare, într-o acoperire ne-primă dar de aceeaşi cardinalitate cu cea iniţială. Implicanţii sunt procesaţi unul câte unul. Acest operator încearcă să înlocuiască fiecare implicant printr-un altul care este conţinut în acesta, cu condiţia că implicanţii reduşi împreună cu cei rămaşi continuă să acopere funcţia.

Algoritmul ESSPRESO este bazat pe notatia cubica utilizata pentru reprezentarea functiilor buleene.Considerand o functie boleana cu n intrari.Functis poate fi considerata ca exista in spatiul n dimensional cu cate o axa pentru fiecare din cele n intrari.

Valorile functiei boleene pot fi considerate ca ocupa puncte in spatiu corespunzatoare valorilor 0 si 1 pentru fiecare din intrari.

O expresie boleana cu o variabiala poate ocupa oricare din punctele 0 sa 1 dintr-un spatiu unidimensional.O funtie boleana cu 2 intrari poate ocupa oricare din cele 4 puncte dupa cum se poate vedea in figura b). In concluzie, o expresie boleana cu n intrari poate ocupa oricare din cele 2^n puncte din cele n dimensiuni ale cubului.

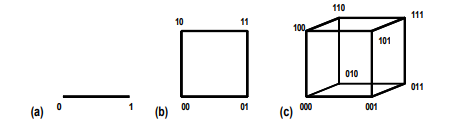
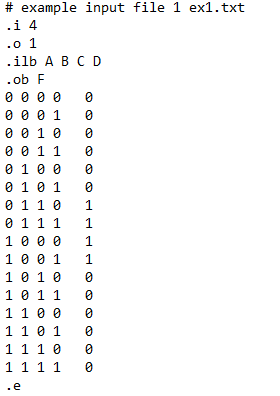


Figura 1.

Principalii operatori, care au fost enumeraţi anterior, sunt încadraţi în anumite strategii care fac ca rezultatele să fie superioare şi mai eficient calculate comparativ cu aplicarea directă a acestora. Între acestea sunt de remarcat, prin eficienţă, următoarele: o Identificarea şi extragerea cât mai timpurie a implicanţilor esenţiali. o Utilizarea complementului funcţiei pentru verificarea eficientă a expansiunii implicantului curent (aflat în expansiune) comparativ cu o expansiune anterior efectuată şi care ar putea conţine deja termenii canonici ai acestuia, invalidând adăugarea acestuia la acoperirea nou calculată. Minimizatoarele logice euristice pot fi caracterizate în baza operatorilor utilizaţi şi a ordinii în care aceştia sunt aplicaţi în procesul de calcul.

Programul ESPRESSO utilizează operatorul irredundant pentru asigurarea iredundanţei acoperirilor. Algoritmul principal din ESPRESSO calculează întâi o acoperire primă şi iredundantă prin aplicarea operatorilor expand şi irredundant. Apoi, se îmbunătăţeşte acoperirea, când este posibil, prin calculul unei secvenţe de acoperiri prime şi iredundante de cardinalităţi descrescătoare. Această suită de acoperiri se obţine iterativ prin aplicarea succesivă a operatorilor reduce, expand şi irredundant, dar şi prin comutarea euristicilor din implementările acestor operatori.

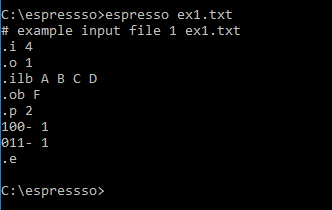
Exemplul 1.



**Explicatia ceintei:**

**.i 4** (varuabile de intrare = 4)  
**.o 1** (varuabile de iesire= 1)  
**.ilb A B C D** (Numele variabilelor de intrare)  
**.ob F** (Numele variabilei de iesire)  
**0 0 0 0   0** (tabelul de adevar, 4 intrari, o iesire)  
**.e** (Marcheaza sgarsitul fisierului)

**Rezultat:**

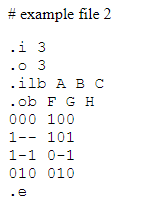


Interpretarea rezultatului:  
.p 2 (Indica faptul ca sunt doi termini in expresia de iesire)  
100- 1 ( termenul este AB'C'. notatie: B' este B negat,

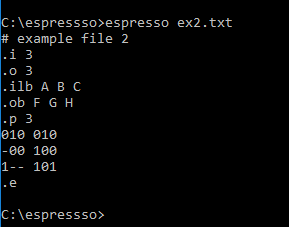
011- 1 ( termenul este A'BC)

Expresia logica este F = AB'C' + A'BC.

Exemplul 2:

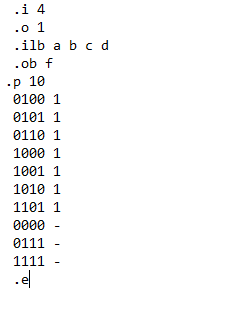


Rezultat:

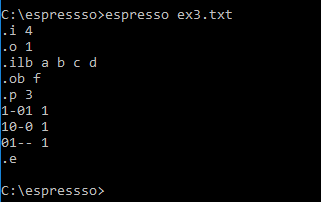


Exemplul 3:

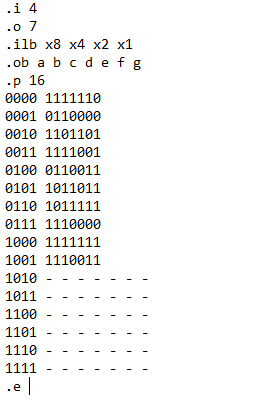
Fie funcţia scalară de patru variabile f(a,b,c,d) = m4 + m5 + m6 + m8 + m9 + m10 + m13 , având următorii termeni canonici neprecizaţi: m0 + m7 + m15.



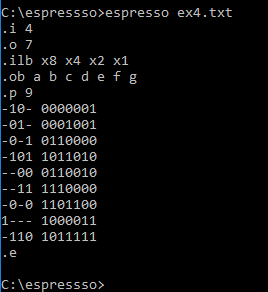
Rezultatul:\

 f(a,b,c,d) = ac’d + ab’d’ + a’b.

Exemplul 4:



Rezultatul:



Corespunzător acestor termeni produs se pot scrie expresiile fiecărei linii de ieşire:

a = x4x2’x1 + x2x1 + x4’x1’ + x8 + x4x2x1’

b = x4’x1 + x2’x1’ + x2x1 + x4’x1’

c = x4’x1 + x4x2’x1 + x2’x1’ + x2x1 + x4x2x1’

d = x4’x2 + x4x2’x1 + x4’x1’ + x4x2x1’

e = x4’x1’ + x4x2x1’

f = x4x2’x1 + x2’x1’ + x8 + x4x2x1’

g = x4x2’ + x4’x2 + x8 + + x4x2x1’

Bibliografie:

[1] <http://www.etc.ugal.ro/rpopa/assn_book/capitolul_4.pdf>

[2] <http://andrei.clubcisco.ro/cursuri/1pl/cursuri/Minimizarea%20Logica%20Euristica.pdf>

[3] <https://en.wikipedia.org/wiki/Espresso_heuristic_logic_minimizer>

[4] <http://www.physics.dcu.ie/~bl/digi/unitd17.pdf>