Analiza şi sinteza circuitelor combinaţionale

Note de curs

Dr.Ing.Mat. Ion I. Bucur

Un circuit combinaţional C, este definit prin relaţiile dintre intrări şi ieşiri :

$$f_i : \mathbf{B}^n \to \mathbf{B}, \qquad (\mathbf{B}=\{0,1\}),$$
$$z_i = f_i(x_1, x_2, \ldots, x_n),$$

unde :

- z_i , $0 \le i \le m$-1, este una dintre liniile de ieşire ale circuitului, iar
- $x_0, x_1, \ldots, x_{n-1}$ sunt liniile de intrare în circuitul combinaţional considerat (diagrama modelului circuitelor combinaţionale, este prezentată în figura 1).

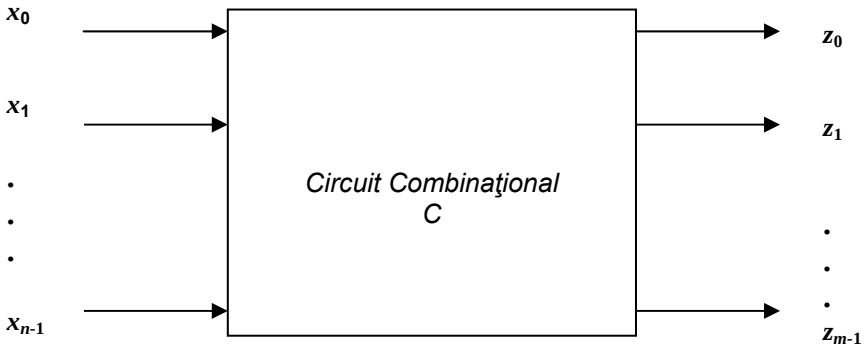

Figura 1. Reprezentarea modelului unui circuit combinaţional

1. Introducere

Se poate remarca, din relaţia care leagă liniile de intrare de liniile de ieşire, dependenţa în exclusivitate a valorilor ieşirilor de valorile aplicate intrărilor.
Ca particularitate, funcţiile Boole-ene sunt, întotdeauna, funcţii cu domeniul de definiţie finit. Aşa cum au fost prezentate mai sus funcţiile f_i au 2^n puncte, n-uple, distincte în domeniul lor de definiţie (produsul cartezian al mulţimii **B** cu aceasta însăşi de n ori).
Datorită faptului că funcţiile au un domeniu de definiţie cu 2^n n-uple distincte, iar funcţiile pot lua doar două valori, atunci numărul funcţiilor distincte, astfel considerate, este 2^{2^n} .

Exemplul 1.1
Funcţiile distincte $f_i : \mathbf{B}^2 \to \mathbf{B}$ cu două variabile sunt :

Tabelul 1.1 Mulţimea tuturor funcţiilor Boole-ene cu două variabile.

x_1	x_0	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Dintre acestea, se remarcă, pentru ilustrarea exemplului :
- funcţiile constante 0 şi 1, respectiv f_0 şi f_{15} ;
- funcţiile identic x_1 şi x_0, respectiv f_3 şi f_5 ;
- funcţiile x_1' şi x_0', respectiv f_{12} şi f_{10};
- funcţiile SAU şi ŞI, respectiv f_7 şi f_1.

□

Circuitele combinaţionale pot fi introduse prin enumerarea valorilor funcţiei corespunzătoare punctelor, în număr finit, domeniului de definiţie sau printr-o descriere comportamentală a circuitului. Cea de-a doua cale este reductibilă la prima.

Exemplul 1.2
Se consideră un circuit combinaţional care realizează suma a două numere binare *a* şi *b* fără semn reprezentate fiecare printr-un singur rang. Un astfel de circuit se numeşte, tradiţional, *semi-sumator*.

Circuitul, se poate remarca, este introdus printr-o descriere comportamentală. Acestei descrieri se poate asoca, simplu, o descriere enumerativă punct cu punct, după cum urmează :

((*a,b*) | *suma, transportul*) :{ ((0, 0) | 0, 0), ((0, 1) | 1, 0), ((1, 0) |1, 0), ((1, 1) | 0, 1) }.

Se remarcă utilizarea unui mod de scriere explicit atât al valorilor argumentelor *a* şi *b* cât şi al valorilor funcţiilor de ieşire *suma* şi *transportul*, separând prin caracterul bară verticală (|) punctul curent din domeniul de definiţie, de valorile funcţiilor în acel punct. Acest mod de scriere este mult răpândit în literatură.

Cele două funcţii sunt exprimabile separat ca formule Boole-ene utilizând fie valorile 1 (acolo unde funcţia este *asertată*), fie valorile 0 (acolo unde funcţia este *complementată*). În acest exemplu se va considera exprimarea celor două funcţii prin aserţiuni.

Pentru aceasta se construiesc formulele celor două funcţii utilizând teorema de reprezentare a algebrelor Boole-ene :

$$suma(a,b) = a'b + ab' \;\; ; \;\; transportul(a,b) = ab.$$

S-a utilizat notaţia, mult răspândită, *a'* pentru variabila *a* complementată.

Există o strânsă corespondenţă între definirea punct cu punct (numită şi definirea prin tabela de adevăr) şi scrierea prin formule a unei funcţii. În realitate, ambele cuprind aceeaşi informaţie. Se poate remarca, din forma canonică disjunctivă, că produsele variabilelor funcţiilor sunt calculate acolo unde funcţia ia valoarea 1. Produsele respective se numesc, tradiţional, *termeni produs* (din cauza analogiei, curent practicate, dintre funcţia ŞI şi Multiplicarea numerelor reale) şi se calculează astfel :
- valoare 0 a variabilei, variabila apare în produs complementată,
- valoare 1 a variabilei, variabila apare în produs asertată.

Pentru termenii produs se mai utilizează, alternativ, denumirea de mintermi. Mintermii sunt indiciaţi, curent, cu valorile zecimale corespunzătoare scrierii binare a mintermului. Astfel formulele pentru cele două funcţii pot fi scrise, în aceeaşi ordine, astfel :

$$suma(a,b) = m_1 + m_2 \;\; ; \;\; transportul(a,b) = m_3.$$

Construcţia formulelor prin punctele unde funcţiile sunt complementate se poate deduce similar sau se poate calcula simplu utilizând relaţiile De Morgan aplicate formulei deduse pentru punctele unde funcţiile sunt asertate. Este un foarte bun exerciţiu.

□

1.1 Dualitatea şi Legile DeMorgan

Dualitatea este o proprietate foarte utilă a algebrelor booleene. Expresia dual㠹 a unei expresii Booleene se deduce prin:

- înlocuirea operatorului ŞI (·), prin operatorul SAU (+) şi reciproc;
- înlocuirea constantei 0, prin constanta 1 şi reciproc;
- în timp ce, variabilele expresiei rămân neschimbate.

Orice teoremă ori propoziţie demonstrată din algebra booleană ca fiind adevărată, are întodeauna o dual㠹, deasemenea adevărată. Dualitatea este, în esenţă, o *meta-teoremă*, cu alte cuvinte o teoremă despre teoreme.

Cu toate că dualitatea nu cuprinde, în sine, o modalitate directă de simplificare a expresiilor booleene, aceasta oferă posibilitatea deducerii unor noi teoreme din cele deja cunoscute ajuntând astfel în procesul de simplificare al expresiilor.

Astfel, teorema de unificare $x \cdot y + x \cdot y' = x$, are duala formulată astfel $(x + y) \cdot (x + y') = y$.

O demonstraţie a dualei teoremei de unificare decurge, succesiv, în două etape astfel:
(1) aplicând legea de distributivitate se poate transcrie expresia membrului stâng al dualei teoremei de unificare:

$$(x + y) \cdot (x + y') = x \cdot (x + y') + y \cdot (x + y'),$$

(2) în continuare, expresia obţinută devine:

$$x \cdot (x + y') + y \cdot (x + y') = x + y \cdot x = x \cdot (y + 1) = x,$$

ceea ce trebuia demonstrat.

Se consideră expresia: $f = abc + a'(b + c)$, pentru care se calculează duala. O cale simplă de calcul a dualei poate fi imaginată prin divizarea expresiei date în sub-expresii mai mici pentru care efortul de calcul poate fi mai uşor controlat: $f = e_1 + e_2$, unde $e_1 = abc$, iar $e_2 = a'(b + c)$.
Se notează, tradiţional, duala expresiei f prin f_D, rezultând că:

$$f_D = e_{1D} \cdot e_{2D}.$$

Aplicând principiul dualităţii sub-expresiei e_{1D}, rezultă:

$$e_{1D} = a + b + c.$$

Similar, se calculează sub-expresia:

$$e_{2D} = a' + bc.$$

Rezultatul final arată astfel :

$$f_D = (a + b + c)(a' + bc).$$

Legea DeMorgan oferă o modalitate teoretică de complementare a unei funcţii, de complexitate modică. Expresia complementară a unei expresii date se formează pornind de la expresia originală prin înlocuirile:

- oricare literal, prin complementul său (x prin x' şi reciproc),
- oricare constantă, prin complementara sa (0 se substituie prin 1 şi reciproc),
- operatorul ŞI se substituie prin operatorul SAU şi reciproc.

Această teoremă, aplicată chiar operatorilor ŞI şi SAU arată relaţiile cu operatorii complementari SAU-NU respectiv ŞI-NU:

$$(x + y)' = x' \cdot y',$$
$$(x \cdot y)' = x' + y'.$$

Relaţiile anterioare pot fi interpretate astfel:

Operatorul SAU-NU aplicat unor variabile, este identic cu operatorul ŞI aplicat variabilelor respective dar complementate, în timp ce operatorul ŞI-NU aplicat unor variabile este identic cu operatorul SAU aplicat variabilelor respective dar complementate.

Se consideră expresia booleană de trei variabile $E(a,b,c) = a'b'c + a'bc + ab'c + abc'$. Complementara acesteia se calculează, pas cu pas astfel:

$$(E(a,b,c))' = (a'b'c + a'bc + ab'c + abc')',$$
$$(E(a,b,c))' = (a'b'c)' \cdot (a'bc)' \cdot (ab'c)' \cdot (abc')',$$
$$(E(a,b,c))' = (a + b + c') \cdot (a + b' + c') \cdot (a' + b + c') \cdot (a' + b' + c),$$
$$(E(a,b,c))' = (a + ab' + ac' + ab + bc' + b'c' + c') \cdot (a' + a'b' + a'c + a'b + bc + a'c' + b'c'),$$
$$(E(a,b,c))' = (a + bc' + b'c' + c') \cdot (a' + bc + b'c'),$$
$$(E(a,b,c))' = (a + c')(a' + bc + b'c'),$$
$$(E(a,b,c))' = abc + ab'c' + a'c' + b'c',$$
$$(E(a,b,c))' = abc + b'c' + a'c'.$$

O metodă, puţin mai simplă, pentru calculul formal al complementului expresiei unei funcţii constă în calculul dualei expresiei funcţiei urmat de complementarea fiecărui literal.

Astfel, complementul expresiei booleene de trei variabile din exemplul precedent ar putea fi calculat după cum urmează:

$$E_D(a,b,c) = (a' + b' + c) \cdot (a' + b + c) \cdot (a + b' + c) \cdot (a + b + c'),$$

Complementând fiecare literal din expresia dualei rezultă:

$$(E(a,b,c))' = (a + b + c') \cdot (a + b' + c') \cdot (a' + b + c') \cdot (a' + b' + c).$$

De remarcat faptul că duala unei expresii şi legea DeMorgan aplicată aceleiaşi expresii nu sunt unul şi acelaşi lucru. Procedeul de obţinere al dualei este similar cu menţiunea că *literalii nu sunt complementaţi* pe durata procesului de calcul. Astfel, duala funcţiei SAU-NU este funcţia ŞI-NU şi reciproc, iar duala funcţiei ŞI este funcţia SAU şi reciproc. Atunci când se aplică, unei funcţii, teorema dualităţii se obţine o cu totul altă funcţie. Prin aplicarea legii DeMorgan unei funcţii anumite se obţine complementara respectivei funcţii.

2. Optimizarea circuitelor logice combinaţionale

În continuare se vor aborda principiile optimizării circuitelor logice combinaţionale modelate prin expresii formate din sume de produse (SDP) în două niveluri sau, echivalent, prin forme tabelare (FT) cum ar fi, spre exemplu, tabelele de implicanţi. Translatarea între cele două forme este imediată, în ambele sensuri. Dat fiind faptul că translatarea între cele doua forme este fără echivoc, pentru fixarea ideilor, se poate considera în continuare că orice circuit este prezentat sub prima formă, suma de produse.

Minimizarea exactă unei funcţii scalare Booleene are drept principal obiectiv, pentru respectiva funcţie, stabilirea unei expresii algebrice echivalente, prin sume de produse, minime ca număr de termeni şi având, eventual, un număr minim de literali.
Astfel de exprimare se mai numeşte, tradiţional, acoperire prin sumă de produse. Sunt utilizate, în acest scop, toate punctele din domeniul de definiţie în care funcţia este definită prin valoarea 1 şi punctele în care valoarea funcţiei nu este precizată, dacă astfel de puncte există în specificaţia funcţiei.
Găsirea unei forme în produs de sume este similară şi simplu de dedus din forma în sumă de produse.

Înainte să se considere aspecte atât teoretice cât şi de natură tehnologică, pragmatice, este util să se aibă în vedere un exemplu, de complexitate redusă dar care să contureze aspectele definitorii ale acestei probleme, esenţiale în proiectarea logică.

Exemplul 2.1. Pentru funcţia specificată prin suma de mintermi:
$f = m_5 + m_6 + m_9 + m_{10} + m_{13} + m_{14}$,
se doreşte stabilirea unor posibile implementări, cât mai simple
Numărul (minim) de variabile pentru această funcţie este patru, şi aceste variabile vor fi notate prin: x_8, x_4, x_2 şi x_1.

Exprimarea mintermilor în format binar este prezentată în tabelul 4:

Tabelul 2.1
Funcţia din exemplul 2.1

Mintermii funcţiei *f*	Variabilele funcţiei x_8	x_4	x_2	x_1
m_5	0	1	0	1
m_6	0	1	1	0
m_9	1	0	0	1
m_{10}	1	0	1	0
m_{13}	1	1	0	1
m_{14}	1	1	1	0

Translatarea literală, în ordine, a mintermilor funcţiei conduce la această expresie, în sume de produse canonice, a funcţiei considerate:

$$f = x_8'x_4x_2'x_1 + x_8'x_4x_2x_1' + x_8x_4'x_2'x_1 + x_8x_4'x_2x_1' + x_8x_4x_2'x_1 + x_8x_4x_2x_1', \quad (1)$$

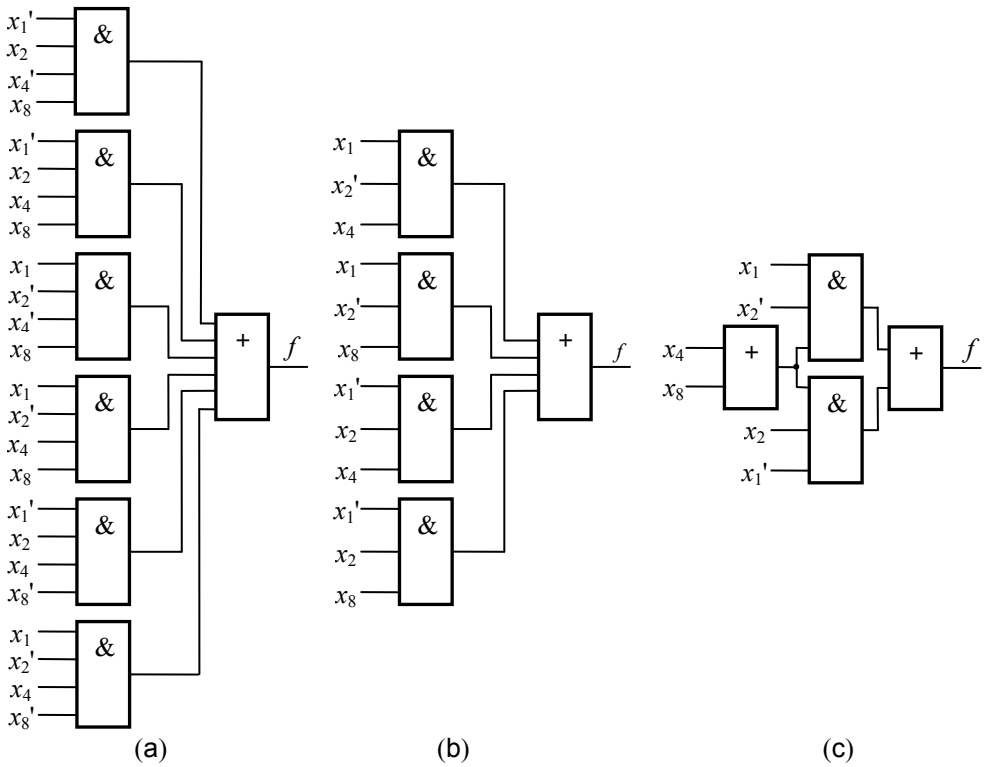

Figura 2.1. Realizări posibile ale funcţiei din exemplul 2.1.

Ţinând cont de comutativitatea operatorului + şi de faptul că în algebrele Boole-ene are loc identitatea $a + a = a$, expresia funcţiei *f* se poate scrie, după o prealabilă grupare convenabilă a termenilor produs, astfel:

$$f = (x_8'x_4x_2'x_1 + x_8x_4x_2'x_1\ ) + (x_8x_4'x_2'x_1 + x_8x_4x_2'x_1) +$$

$$(x_8'x_4x_2x_1' + x_8x_4x_2x_1') + (x_8x_4'x_2x_1' + x_8x_4x_2x_1'), \quad (2)$$

În expresia anterioară a funcţiei *f,* din fiecare paranteză se pot factoriza produse de câte trei variabile, astfel:

$$f = x_4x_2'x_1(x_8' + x_8) + x_8x_2'x_1(x_4 + x_4') + x_4x_2x_1'(x_8' + x_8) + x_8x_2x_1'(x_4' + x_4), \quad (3)$$

Considerând identităţile Boole-ene $a + a' = 1$ şi $a \cdot 1 = a$, expresia funcţiei *f* se rescrie astfel:

$$f = x_4x_2'x_1 + x_8x_2'x_1 + x_4x_2x_1' + x_8x_2x_1'. \quad (4)$$

Se poate remarca, în expresia anterioară a funcţiei *f*, următoarea factorizare:

$$f = (x_4 + x_8)x_2'x_1 + (x_4 + x_8)x_2x_1'. \quad (5)$$

Expresia iniţială a funcţiei, expresia (1), forma minimizată (4) şi forma factorizată (5) sunt reprezentate schematic, simbolic, în figura 2 (a), (b) şi respectiv (c).

Cercetând circuitele din figura 2.1 se poate observa cu uşurinţă că dintre cele trei realizări ale funcţiei *f*, cea mai complexă este cea din figura 2.1.(a).
Comparând, în continuare, circuitele 2.1.(b) şi 2.1.(c) se poate spune că în cazul circuitului 2.1(c) sunt utilizate mai puţine componente dar, în acelaşi timp se observă că, spre deosebire de celelalte două circuite are mai multe niveluri. Un număr crescut de niveluri poate implica, depinzând de tehnologie, o viteză de lucru mai scăzută pentru un circuit combinaţional.

Alegerea între un circuit mai economicos, dar mai lent, şi unul mai costisitor, dar mai rapid, este tranşată prin considerente care ţin seama de contextul în care este plasat circuitul aflat în discuţie. Este de reţinut că toate considerentele ce vor fi făcute în continuare se vor referi exclusiv la circuitele cu două niveluri.
De remarcat faptul că în cazul circuitelor combinaţionale cu două niveluri, aşa cum se poate vedea în figura 2.1(b), fiecare produs din suma de produse reprezintă o poartă (ŞI) iar fiecare literal dintr-un produs reprezintă o intrare într-o poartă. Întreaga sumă de produse corespunde unei porţi (SAU). Similar se întâmplă în cazul implementării circuitelor combinaţionale prin produse de sume: fiecare sumă reprezintă o poartă (SAU) iar întregul circuit este realizat printr-o poartă (ŞI). Ca şi în cazul sumelor de produse, un literal dintr-o sumă este o linie de intrare într-o poartă SAU. Trebuie remarcat, pe scurt, că numărul de linii de intrare într-o poartă este, în general, limitat iar numărul respectiv depinde de tehnologia în care este construită poarta în cauză.

Porţile au costuri după cum şi liniile de intrare în porţi au costurile lor. Raportul dintre costul unei linii de intrare într-o poartă şi costul acelei porţi este dependent de tipul porţii, de tehnologia în care se realizează poarta etc. Se poate aprecia, în general, că atunci când se pune problema unei porţi suplimentare, costul acesteia este de câteva ori mai mare decât costul unei linii de intrare într-o poartă deja existentă în circuitul respectiv. Din acest punct de vedere, micşorarea numărului de porţi este un criteriu important în găsirea, în general, a unei forme mai simple pentru o funcţie Boole-eană dată.

2.1 Principiile optimizării logice.

Obiectivul minimizării logice în două niveluri este reducerea mărimii reprezentării funcţiilor Booleene în oricare din formele de reprezentare *sume de produse* ori *produse de sume*.
Se poate remarca faptul că oricare din cele două forme ale funcţiilor Booleene poate fi dedusă din cealaltă cu ajutorul legilor De Morgan iar această transformare păstrează numărul de termeni şi de literali. Prin urmare, se poate concentra abordarea doar asupra minimizării unei singure forme şi anume suma de produse, fără să se piardă din generalitate.

Obiectivul detaliat al minimizării logice în două niveluri poate varia puţin în funcţie de stilurile de implementare. Circuitele PLA sunt un astfel de stil de implementare. Prima ţintă a minimizării logice, pentru acest stil de implementare este reducerea numărului de produse şi ţinta secundară este reducerea literalilor din produse.

Alte obiective de optimizare sunt relevante atunci când funcţiile modelate prin forme în două niveluri sunt implementate altfel decât prin PLA-uri. Reprezentările logice în doua niveluri pentru funcţii scalare, spre exemplu, pot fi implementate prin porţi complexe a căror mărime este corelată cu numărul de literali din forma factorizată a acelei reprezentări. În astfel de situaţii obiectivul major este minimizarea numărului de literali.

Minimizarea logică pentru o funcţie scalară sau o funcţie vectorială se face după aceleaşi principii, dar cazul vectorial este mult mai complex. Minimizarea disjunctă a componentelor scalare ale unei funcţii vectoriale poate conduce la rezultate suboptimale deoarece optimizarea nu poate exploata comunitatea unor termeni produs. Un rezultat important în optimizarea logică în două niveluri este echivalenţa funcţiilor vectoriale binare de variabile binare cu funcţiile binare scalare de variabile multi-valorice. Din astfel de raţiuni, pentru început, abordarea se va concentra asupra tehnicilor de optimizare ale funcţiilor scalare de variabile bi-valorice.

2.2 Definiţii

Se vor considera funcţii binare (Boole-ene) de variabile binare incomplet specificate, de forma:

$$\boldsymbol{f}: \mathbf{B}^n \rightarrow \{0, 1, *\}^m,$$

deoarece funcţiile complet specificate sunt un caz particular al funcţiilor cu puncte nedefinite, nespecificate ("*don't care*").

Pentru fiecare linie de ieşire f_i $(1 \leq i \leq m)$ se definesc trei mulţimi de puncte disjuncte, partiţii ale domeniului de definiţie:

- mulţimea punctelor în care funcţia f_i are valoarea 1,
- mulţimea punctelor în care funcţia f_i are valoarea 0 şi
- mulţimea punctelor în care funcţia f_i are valoarea * sau mulţimea punctelor nedefinite.

Aceste partiţii sunt notate respectiv prin *mp*1, *mp*0, respectiv *mp** (submulţimi ale domeniului de definiţie $\mathbf{B}^n$).
Prin această partiţionare a domeniului de definiţie, fiecare funcţie scalară incomplet specificată poate fi privită ca un triplet de funcţii complet specificate.

Funcţiile sunt reprezentate (prin sume de produse ori tabelar) ca liste de implicanţi. Conceptul implicantului *multi-ieşire* (*multi-funcţie*) este general ca natură, deoarece combină vectorul valorilor liniilor de intrare cu vectorul valorilor corespunzătoare liniilor de ieşire.
În consideraţiile care urmează se va restricţiona noţiunea de implicant doar la valoarea 1 sau * (neprecizată) a unei funcţii.

În consecinţă, partea de ieşire a unui implicant multi-ieşire este bivalorică pentru o componentă scalară a respectivei funcţii vectoriale, având următoarea semnificaţie:
- o valoare 1 implică fie o valoare 1, fie o valoare * a componentei scalare respective, în timp ce,
- o valoare 0 nu implică atribuirea unei valori a respectivei componente scalare; pur şi simplu acea componentă scalară este ignorată în punctul corespunzător părţii de intrare.

În astfel de situaţii partea de intrare a implicantului, cu alte cuvinte, se referă la unul sau mai multe puncte din domeniul de definiţie, unde anumite componente scalare ale funcţiei vectoriale iau valoarea 1 ori * şi anume acele componente pentru care partea de ieşire a implicantului multi-valoric este nenulă.

> Definiţia 2.1 Un *implicant multi-ieşire* (*multi-funcţie*) al unei funcţii vectoriale
> $\boldsymbol{f}: \mathbf{B}^n \rightarrow \{0, 1, *\}^m$
> este o pereche de vectori linie de dimensiune *n* şi *m* numiţi *partea de intrare* şi respectiv *partea de ieşire*.

Partea de intrare are componente cu valori din mulţimea {0, 1, *} şi reprezintă un produs de literali. Partea de ieşire are componente cu valori din mulţimea {0, 1}. Pentru fiecare componenta a ieşirii o valoare nenulă implică o valoare 1 sau * a respectivei funcţii scalare în punctul (punctele) corespunzător (corespunzătoare) părţii de intrare.

Mintermii multi-ieşire sunt implicanţi supuşi unor restricţii deosebite.

> Definiţia 2.2 Un *minterm multi-ieşire* al unei funcţii precum cea definită anterior este un implicant multi-ieşire a cărui parte de intrare are componente cu valori din mulţimea {0, 1} (exact *n* literali) şi care implică valoare nenulă pentru una şi numai pentru una dintre componentele (funcţiile scalare) de ieşire.

Exemplul care urmează ilustrează printr-o funcţie vectorială simplă, cu două componente scalare, aspectele definitorii ale implicanţilor canonici, mintermii, în cazul funcţiilor vectoriale.

> Exemplul 2.2. $\boldsymbol{f} = (f_1, f_2)$;
> $f_1 = a'b'c' + a'b'c + ab'c + abc + abc'$,
> $f_2 = a'b'c + ab'c$
>
> Un implicant multi-ieşire al acestei funcţii vectoriale $\boldsymbol{f}$ este (*01|11) .
>
> Acest implicant multi-ieşire reprezintă patru mintermi multi-ieşire:
>
> (001|10), (101|10), (001|01) şi (101|01).

Datorită faptului că un implicant, necanonic, poate acoperi mai mulţi implicanţi canonici (mintermi) multi-ieşire, se poate concepe reprezentarea unei funcţii printr-o mulţime de implicanţi necanonici. Iar numărul implicanţilor necanonici ai respectivei mulţimi să fie, posibil, mai mic decât numărul implicanţilor canonici ai funcţiei. O astfel de mulţime cu implicanţi necanonici, asociată unei funcţii date, este definită în continuare.

> Definiţia 2.3 O *acoperire* a unei funcţii vectoriale binare este o mulţime (o listă) de implicanţi care conţin (acoperă) mintermii acelei funcţii.

Se notează prin $\boldsymbol{F}$ acoperirea funcţiei $\boldsymbol{f}$. Mulţimile *mp*1, *mp*0, *mp** pot fi modelate prin acoperiri, unde implicanţii şi mintermii sunt corespunzători funcţiilor complet specificate respective. Acoperirile *mp*1, *mp*0 şi *mp** sunt notate, tradiţional, prin F^{ON}, F^{OFF} şi respectiv F^{DC}. O acoperire $\boldsymbol{F}$ a unei funcţii $\boldsymbol{f}$ satisface inegalitatea: $\boldsymbol{F}^{ON} \subseteq \boldsymbol{F} \subseteq \boldsymbol{F}^{ON} \cup \boldsymbol{F}^{DC}$. Atunci când o funcţie este complet definită, $\boldsymbol{F}^{ON}$ şi $\boldsymbol{F}$ sunt identice (deoarece $\boldsymbol{F}^{DC}$ este mulţimea vidă).

Mărimea sau cardinalitatea unei acoperiri este numărul de implicanţi ai acelei acoperiri.

> Definiţia 2.4. O *acoperire minimă* este o acoperire de cardinalitate minimă.

În cele ce urmează se va considera că obiectivul minimizării logice combinaţionale exacte în două niveluri este *determinarea unei acoperiri minime*.
Este, adesea, util să se determine acoperiri minimale, locale, numite *acoperiri minimale*, deoarece calculul acestora poate fi atins cu resurse (memorie şi timp) mai reduse.
Obiectivul minimizării euristice logice combinaţionale în două niveluri este determinarea unei acoperiri minimale. Astfel de acoperiri sunt adesea foarte apropiate în cardinalitate de acoperirile minime şi pot astfel oferi soluţii eficiente pentru problemele practice. În mod normal minimalitatea locală se defineşte în termeni de conţinere, includere.

> Definiţia 2.5 O acoperire *iredundantă* a unei funcţii este o acoperire care nu este un superset propriu al niciunei alte acoperiri pentru aceeaşi funcţie.

Altfel spus, îndepărtarea oricărui implicant dintr-o acoperire minimă nu mai acoperă, nu mai conţine, funcţia. Echivalent, nici un implicant nu este conţinut în orice subset de implicanţi ai acoperirii. O proprietate de minimalitate mai slabă este minimalitatea în raport cu conţinerea unui singur implicant, numită adeseori, conţinerea singulară.

> Definiţia 2.6 O acoperire este *minimală* în *raport cu conţinerea singulară* dacă nici un implicant nu este conţinut în orice alt implicant al acoperirii.

O acoperire iredundantă este deasemenea minimală în raport cu conţinerea singulară, dar nu şi reciproc. Raţiunea acestei afirmaţii este aceea că un implicant poate fi redundant deoarece este conţinut într-un subset de implicanţi ai acoperirii şi nu de un singur implicant. Din acest motiv, redundanţa este o proprietate mai puternică.

> Exemplul 2.3. Se reia funcţia $\boldsymbol{f} = (f_1, f_2)$ din exemplul 2.2.
> O acoperire minimală de cardinalitate trei este specificată prin:
>
> (00*|10)
> (*01|11)
> (11*|10)
>
> O acoperire iredundantă de cardinalitate patru este definită prin:
>
> (00*|10)
> (*01|01)
> (1*1|10)
> (11*|10)
>
> O acoperire redundantă de cardinalitate cinci, dar care este minimală în raport cu conţinerea singulară este exemplificată prin:
>
> (00*|10)
> (*01|01)
> → (*01|10)
> (1*1|10)
> (11*|10)
>
> De reţinut că cel de-al treilea implicant (cel indicat printr-o săgeată orizontală) este conţinut în reuniunea dintre primul şi cel de-al patrulea implicant.

O altă proprietate a implicanţilor şi a acoperirilor este aceea de a fi primi, respectiv prime.

> Definiţia 2.7. Un implicant este *prim* dacă nu este conţinut de nici un alt implicant al funcţiei. O acoperire este *primă* dacă toţi implicanţii săi sunt primi.

În raport cu definiţia 2.7 se cuvin făcute câteva remarci complementare:
- definiţia calităţii de implicant prim a unui implicant este legată de toţi implicanţii posibili ai funcţiei şi nu doar de cei ai acoperirii considerate;
- pentru funcţiile scalare, un implicant prim corespunde unui produs de literali, unde nici un literal nu poate fi eliminat fără să se piardă proprietatea de implicant.

În termeni generici, un implicant prim corespunde unui implicant cu dimensiune maximală. Dimensiunea maximală înseamnă cea mai mare dimensiune ce poate fi atinsă fără să se intersecteze $\boldsymbol{F}^{OFF}$ sau, echivalent, $mp0$.

Pentru funcţiile vectoriale, un implicant prim conţine şi un număr maxim de componente scalare (funcţii scalare) ale funcţiei vectoriale reprezentate în partea de ieşire a implicantului. Implicanţii

primi sunt adesea numiţi primi, pentru o exprimare mai scurtă, mai simplă, fără să existe riscul de echivoc.

Exemplul 2.4. Se reia funcţia vectorială de variabile binare din exemplele precedente.
Lista tuturor implicanţilor primi arată astfel:

(00*|10)
(*01|11)
(1*1|10)
(11*|10)

Prin definiţie nici un implicant prim nu este conţinut în alt implicant prim al funcţiei. Implicanţii anteriori reprezintă o acoperire primă, dar care nu este minimă. In adevăr, al treilea implicant prim este conţinut în reuniunea dintre al doilea şi al patrulea implicant prim. Prin definiţia unui implicant prim, dacă acoperirea este primă, atunci aceasta este minimală în raport cu conţinerea singulară.
Înlăturând cel de-al treilea implicant prim se obţine o acoperire minimă care este necesar iredundantă.
Se presupune, acum, că funcţia este incomplet specificată şi mulţimea *mp** (sau F^{DC}) pentru ambele ieşiri este specificată, simplu, prin implicantul cu partea de intrare 100.

Atunci, implicanţii primi ar fi:

(*0*|10),
(*01|11),
(1**|10).

Aceşti implicanţi primi formează o acoperire minimă.

Anumiţi implicanţi primi au proprietatea specială că trebuie să fie incluşi în orice acoperire a funcţiei. Aceşti implicanţi sunt numiţi implicanţi primi esenţiali.

Definiţia 2.8. Un implicant prim este *esenţial* dacă există un minterm al funcţiei acoperit (conţinut) în exclusivitate (faţă de mulţimea tuturor implicanţilor primi ai funcţiei) de acest implicant prim.

Exemplul următor abordează câteva proprietăţi importante ale implicanţilor primi.

Exemplul 2.5. Funcţia considerată în exemplele anterioare are asociaţi aceşti implicanţi primi:

(00*|10)
(*01|11)
(1*1|10)
(11*|10)

Pentru prima componentă scalară se observă că primul implicant prim este esenţial deoarece acoperă în exclusivitate mintermul (000|10), iar al patrulea implicant prim este deasemenea esenţial pentru că acoperă în exclusivitate mintermul (110 |10).

A doua funcţie scalară este implicată de numai un singur implicant, mai exact de al doilea implicant. Acest implicant este, în consecinţă, esenţial.
Dintre toţi implicanţii primi, doar al treilea nu este esenţial.

Minimizarea logică exactă

Minimizarea logică exactă vizează rezolvarea găsirii unei acoperiri minimale. Este considerată o problema clasică în teoria funcţiilor binare de variabile discrete şi primul rezultat remarcabil în găsirea unei acoperiri minime a fost formulat de Quine şi McCluskey. Soluţia problemei, aşa cum a fost

formulată de autori, se bazează pe teorema lui Quine, teoremă care delimitează spaţiul căutărilor soluţiei optime.

Teorema Quine

Există o acoperire minimă care este primă.

Demonstraţie Se consideră o acoperire minimă care nu este alcătuită din implicanţi primi. Fiecare implicant care nu este prim poate fi înlocuit printr-un implicant prim care-l conţine. Astfel, mulţimea rezultată de implicanţi este o acoperire şi are aceeaşi cardinalitate ca şi acoperirea iniţială. În consecinţă, există o acoperire minimă care este alcătuită doar din implicanţi primi.

Teorema Quine permite limitarea căutării unei acoperiri minimale la acele acoperiri care constau exclusiv din implicanţi primi. Se remarcă faptul ca teorema se poate generaliza pentru a manevra definiţii mai largi ale acoperirii minime, unde costul unui implicant este întotdeauna mai mic sau egal cu costul unui implicant pe care-l conţine. Teorema se aplică, spre exemplu, cazului de minimizare al numărului literalilor pentru funcţiile scalare (cu o singură ieşire).

E. McCluskey a formulat căutarea unei acoperiri minime ca o problema de acoperire într-un *tabel de implicanţi primi*. Se va aborda aceasta formulare considerând funcţii scalare complet definite.
Un tabel de implicanţi primi este, în fapt, o matrice binară $\boldsymbol{A}$ ale cărei coloane sunt în corespondenţă biunivocă cu implicanţii primi ai funcţiei *f*, iar rândurile sunt în corespondenţă biunivocă cu mintermii funcţiei. Un element $a_{i,j} \in \boldsymbol{A}$ este 1 dacă şi numai dacă cel de-al *j*-lea prim acoperă (conţine) cel de-al *i*-lea minterm. O acoperire minimă este o mulţime minimă de coloane care acoperă toate liniile, sau echivalent: o mulţime minimă de primi ce acoperă (conţin) toţi mintermii funcţiei.
Se poate observa că:
Problema acoperirii poate fi văzută ca fiind problema găsirii unui vector binar $\boldsymbol{x}$ reprezentând o mulţime de implicanţi primi cu cardinalitate ($|\boldsymbol{x}|$) minimă astfel încât :

$$\boldsymbol{Ax} >= 1 \qquad (1)$$

Matricea $\boldsymbol{A}$ poate fi privită ca matricea de incidenţă a unui hipergraf ale cărui noduri corespund mintermilor, iar arcele corespund implicanţilor primi. Într-o astfel de modelare, problema acoperirii corespunde unei acoperiri cu arce ale hipergrafului.

Exemplul 2.6. Fie funcţia scalară de trei variabile *a, b* şi *c:*

$$f(a, b, c) = a'b'c' + a'b'c + ab'c + abc + abc'$$

Se poate verifica cu uşurinţă faptul că implicanţii primi ai acestei funcţii sunt aceştia:

(p)	0	0	*	|1
(q)	*	0	1	|1
(r)	1	*	1	|1
(s)	1	1	*	|1

Ţinând seama de implicanţii primi *p, q, r* şi s, matricea $\boldsymbol{A}$ arată astfel:

	p	*q*	*r*	*s*
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

Vectorul $\boldsymbol{x} = [1101]^T$ reprezintă o acoperire, pentru că $\boldsymbol{Ax} >= 1$.
Se poate afirma că vectorul $\boldsymbol{x}$ selectează implicanţii primi *p*,*q* şi *s*.

Minimizarea exactă poate fi rezolvată calculând întâi tabelul de implicanţi primi şi apoi soluţionând problema de acoperire rezultată. De remarcat faptul că problema de acoperire în acest caz este unată, deoarece toate clauzele de acoperire pot fi exprimate ca o disjuncţie de implicanţi. Dificultatea abordării constă din intractabilitatea problemei de acoperire şi din mărimea tabelului de implicanţi.

O funcţie scalară binară cu n variabile binare poate avea $3^n/n$ primi şi 2^{n-1} mintermi. De aceea, un algoritm exponenţial al unei probleme de mărime exponenţială este probabil să necesite un timp lung de calcul şi un volum mare de memorie. Rezultatele actuale arată că multe probleme practice de minimizarea logică ale unor funcţii dificile pot fi rezolvate exact prin algoritmi performanţi care exploatează natura problemei şi structuri de date eficiente.

Tabelul de implicanţi poate fi redus. Se pot extrage coloanele esenţiale corespunzătoare implicanţilor primi esenţiali, deoarece aceştia oricum trebuie să aparţină oricărei soluţii. Se pot înlătura liniile dominante şi coloanele dominate.
Extracţia implicanţilor primi esenţiali şi înlăturarea coloanelor dominate şi a liniilor dominante se poate face iterativ. Se obţine astfel, în final, tabelul redus al implicanţilor primi.

Dacă tabelul redus se întâmplă sa fie vid, atunci s-a găsit soluţia deja, prin implicanţii primi esenţiali anterior extraşi.
Altfel, tabelul redus, numit uneori şi tabel ciclic, modelează aşa-zisul *miez ciclic al problemei*. Metoda originală propusă de E. McCluskey revine la branşări, adică se aleg diferite combinaţii de coloane (implicanţi primi) şi se evaluează costul corespunzător.

Chiar dacă alegerea unei coloane (un prim implicant) poate conduce la simplificări bazate pe regulile de dominanţă şi de esenţiali, procesul este exponenţial (în cel mai defavorabil caz) în raport cu mărimea tabelei reduse.

Exemplul 2.7. Se consideră matricea ***A*** din exemplul 2.6.

	p	*q*	*r*	*s*
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

Se remarcă imediat că implicanţii *p* şi *s* sunt esenţiali. Aceasta revine la a spune că implicanţii primi *p* şi *s* aparţin oricărei acoperiri.
Din acest motiv, coloanele corespunzătoare pot fi şterse la fel ca şi liniile incidente lor. După procesul de reducere al matricei, matricea astfel obţinută are o singură linie şi două coloane, arătând astfel :

	q	*r*
101	1	1

În acest caz matricea ilustrează faptul că oricare dintre cei doi implicanţi *q* şi *r* poate fi folosit pentru a completa o acoperire minimă. Matricea redusă nu este ciclică şi nu este necesar, în consecinţă, un proces de branşare.

În concluzie, există două soluţii ale problemei acoperirii prime pentru această funcţie:

{*p, q, s*} şi {*p, r, s*}.

O altă abordare clasică numită, adesea, metoda lui Petrick constă în scrierea clauzelor de acoperire ale tabelului (redus) de implicanţi în forma unui produs de sume. Fiecare clauză (sau, echivalent, fiecare sumă din acest produs) corespunde unui minterm şi aceasta reprezintă disjuncţia implicanţilor primi

care acoperă respectivul minterm. Produsul de sume este apoi transformat într-o suma de produse ce este satisfăcută ori de câte ori un termen al său ia valoarea 1. În acest caz, termenii produs reprezintă implicanţii primi care au fost aleşi. Costul unei acoperiri este legat de numărul de literali din produs. Ca rezultat, o acoperire minimă este identificată prin orice termen produs al SDP având cei mai puţini literali.

Exemplul 2.8. Se aplică metoda lui Petrick matricei ***A*** (tabelului cu incidenţa implicanţilor primi, cum mai este numit) din exemplul 2.6.

	p	*q*	*r*	*s*
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

Clauza care stabileşte acoperirea primului minterm este *p*; clauza relativă la cel de-al doilea minterm este *p* +*q*, etc. Produsul de sume arată astfel:

$$(p)(p + q)(q + r)(r + s)(s) = 1$$

Calculând produsele se obţine forma sumă de produse (SDP):

$$pqs + prs = 1$$

ceea ce exprimă faptul că exista două acoperiri minime având aceeaşi cardinalitate, 3.

Prima acoperire este alcătuită din această submulţime {*p*, *q*, *s*} a implicanţilor primi, iar a doua reprezintă submulţimea {*p*, *r*, *s*} a implicanţilor primi.

De remarcat că metoda lui Petrick s-ar fi putut aplica, încă mai eficient, tabelului redus de implicanţi primi şi ar fi condus la o singura clauză:

$$\beta+\gamma =1$$

Astfel, ori implicantul prim *q* ori implicantul prim *r,* împreună cu implicanţii primi esenţiali {*p*, *s*} determină o acoperire minimă cu implicanţi primi ai funcţiei considerate.

Chiar dacă exprimarea produsului de sume şi alegerea termenului produs din suma de produse sunt imediate, transformarea produsului de sume într-o sumă de produse (SDP) implică un număr exponenţial de operaţii. Acest fapt limitează metoda Petrick la tabele cu dimensiuni relativ mici.

Algoritmul Quine-McCluskey poate fi extins la funcţii vectoriale prin calculul implicanţilor primi multi-ieşire şi a tabelei corespunzătoare. Extensii pentru ca să opereze cu funcţii incomplet specificate sunt, de asemenea, relativ simplu de alcătuit şi de aplicat.