Circuite Combinaționale
Exemple și exerciții

Costin-Gabriel Chiru

Prefață

Această culegere de probleme este destinată tuturor celor interesați de analiza și sinteza circuitelor combinaționale, tratând subiecte asociate acestora. Informațiile prezentate aici sunt utile, în primul rând, studenților din anul întâi ai Facultății de Automatică și Calculatoare din cadrul Universității Politehnica din București care urmează cursul de Proiectare Logică prin prezentarea, pentru fiecare subiect tratat, a unui scurt breviar teoretic, urmat de o serie de exemple de probleme tipice rezolvate, referitoare la acel subiect. Fiecare capitol se încheie cu o serie de probleme suplimentare propuse spre rezolvare, oferind astfel posibilitatea studiului individual în vederea pregătirii pentru examinările programate la această materie.

Subiectele prezentate în această carte sunt în concordanță cu conținutul științific predat la cursul și laboratorul de Proiectare Logică. Totuși, trebuie precizat de la bun început faptul că parcurgerea acestei cărți nu suplinește în totalitate explicațiile oferite de cadrele didactice în cadrul orelor de curs și laborator la această materie.

În plus, cartea de față acoperă doar prima jumătate a materiei de la cursul de Proiectare Logică, urmând ca în viitor să fie publicată o a doua carte dedicată circuitelor secvențiale.

Astfel, subiectele tratate în această culegere de probleme sunt asociate circuitelor combinaționale. Aceste circuite se caracterizează prin faptul că ieșirile de la un moment dat ale circuitului depind doar de intrările aplicate acestuia la momentul respectiv de timp. Cu alte cuvinte, aceste circuite nu au memorie și nu țin cont de istoria execuției, lucruri care ar duce la complicații suplimentare, ce vor fi prezentate în următoarea carte. Dar, circuitele complexe (cele secvențiale) sunt bazate pe circuitele combinaționale și fără o cunoaștere temeinică a acestora nu se poate trece la studierea și ulterior sinteza unor astfel de circuite.

În vederea realizării oricărui tip de circuit, trebuie în prealabil să fie definite funcții logice pe baza cărora se exprimă funcționalitatea acestora. Aceste funcții se numesc booleene, întrucât lucrează în logica booleană, logică bazată doar pe elementele 0 și 1, folosite în reprezentarea internă a calculatoarelor. De aceea, funcțiile booleene au o importanță deosebită pentru

circuite (fie ele combinaționale sau secvențiale) și pentru materia Proiectare Logică, în general.

Drept urmare, această carte începe cu prezentarea elementelor de bază ale algebrei booleene și cu aspecte relevante pentru calculul numeric din domeniul teoriei numerelor. În continuare, sunt prezentate pe scurt metodele de reprezentare a funcțiilor booleene.

Un accent deosebit se pune pe diagramele Karnaugh și pe metodele de minimizare a funcțiilor booleene bazate pe acestea, care sunt tratate în capitolul al patrulea.

Cum astfel de reprezentări sunt adaptate doar pentru un număr mic de variabile, metodele de minimizare nescalând corespunzător, în capitolul 5 este prezentat un algoritm de minimizare a funcțiilor booleene ce poate fi implementat pe calculator și care nu prezintă problemele ridicate de diagramele Karnaugh.

Cartea se încheie prin prezentarea diferitelor categorii de circuite combinaționale, care în timp au devenit foarte populare și se folosesc pe scară largă în industrie.

Încă o dată doresc să subliniez faptul că această carte nu își propune să fie exhaustivă, prezentând subiectele succint și insistând pe exemple rezolvate care să ajute la înțelegerea materiei predate la curs și laborator. De aceea, pentru o utilizare optimă, urmărirea acestei cărți trebuie însoțită de lecturarea și înțelegerea subiectelor predate la cursul și laboratorul de Proiectare Logică.

Apariția acestei cărți a fost facilitată de ajutorul, îndrumarea și feedbackul furnizate cu amabilitate de-a lungul anilor de către doamna profesoară Mariana Mocanu și pentru toate acestea aș dori să îi mulțumesc în mod deosebit. În plus, aș dori să le adresez mulțumiri și celor care de-a lungul timpului au făcut parte din echipa de Proiectare Logică și care au ajutat la corectarea unor greșeli apărute. Nu în ultimul rând, aș dori să le mulțumesc foștilor studenți care au urmat acest curs și care, prin feedbackul lor, au exprimat nevoia existenței unei astfel de cărți, iar prin atenția acordată subiectelor expuse au evidențiat elementele mai greu de înțeles și care astfel în carte au primit o atenție sporită, fiind abordate mai intens.

Sper ca această carte să fie utilă cititorilor și să nu dezamăgească așteptările pe care le aveți de la ea!

Cuprins

I Baze de numerație

I.1 Schimbarea bazei de numerație

I.1.1 SCHIMBAREA BAZEI DE NUMERAȚIE PENTRU NUMERELE NATURALE

Trecerea din baza 10 în baza *b* a numerelor naturale se face prin împărțiri succesive la *b*. Astfel, inițial se împarte numărul din baza 10 la *b*, obținându-se un cât și un rest, după care câtul obținut se împarte din nou la *b*, operația repetându-se până când se ajunge la câtul 0. Pentru a obține numărul în noua bază (*b*), se concatenează resturile obținute în ordine inversă obținerii lor.

Exemple:

1. $23_{(10)} = ?_{(2)}$

   23 : 2 = 11 rest 1

   11 : 2 = 5 rest 1

   5 : 2 = 2 rest 1

   2 : 2 = 1 rest 0

   1 : 2 = 0 rest 1

   ⇨ $23_{(10)} = 10111_{(2)}$

2. $47_{(10)} = ?_{(5)}$

   47 : 5 = 9 rest 2

   9 : 5 = 1 rest 4

   1 : 5 = 0 rest 1

   ⇨ $47_{(10)} = 142_{(5)}$

Procedeul invers (trecerea unui număr natural din baza *b* în baza 10) constă în însumarea produselor dintre fiecare element al numărului de convertit și baza în care era scris acesta, ridicată la puterea indicelui elementului minus o unitate.

Exemple:

1. $1001_{(2)} = ?_{(10)}$

   $1001_{(2)} = 1 * 2^0 + 0 * 2^1 + 0 * 2^2 + 1 * 2^3 = 9_{(10)}$

   ⇨ $1001_{(2)} = 9_{(10)}$

2. $234_{(7)} = ?_{(10)}$

   $234_{(7)} = 4 * 7^0 + 3 * 7^1 + 2 * 7^2 = 4 + 21 + 98 = 123_{(10)}$

   ⇨ $234_{(7)} = 123_{(10)}$

Trecerea numerelor naturale din baza *a* în baza *b* (unde *a* și *b* sunt diferite de 10) se face prin trecerea intermediară în baza 10.

Exemplu:

$1110110_{(2)} = ?_{(7)}$

(2) → (10) → (7)

$1110110_{(2)} = ?_{(10)}$

$1110110_{(2)} = 1 * 2^6 + 1 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 118_{(10)}$

⇨ $1110110_{(2)} = 118_{(10)}$

$118_{(10)} = ?_{(7)}$

118 : 7 = 16 rest 6

16 : 7 = 2 rest 2

2 : 7 = 0 rest 2

⇨ $118_{(10)} = 226_{(7)}$

⇨ $1110110_{(2)} = 226_{(7)}$

I.1.2 CAZURI PARTICULARE LA SCHIMBAREA BAZEI DE NUMERAȚIE PENTRU NUMERELE NATURALE

Există 4 cazuri particulare la schimbarea bazei de numerație: trecerea din baza 2 în baza 2^a, trecerea din baza 2^a în baza 2, trecerea din baza 2^a în baza 2^b și trecerea rapidă din baza 10 în baza 2.

I.1.2.1 TRECEREA NUMERELOR NATURALE DIN BAZA 2 ÎN BAZA 2^A

Pentru trecerea numerelor naturale din baza *2* în baza 2^a se fac grupuri de câte *a* cifre și fiecare grup se înlocuiește cu cifra corespunzătoare din baza 2^a. Gruparea pornește de la dreapta numărului (bitul cel mai puțin semnificativ) spre stânga până la sfârșitul numărului. Dacă rămân grupări incomplete, se adăuga 0-uri spre stânga până la completarea grupării.

Exemplu:

$1110110_{(2)} = ?_{(8)} = ?_{(16)}$

$\underline{001}\ \underline{110}\ \underline{110}_{(2)} = 166_{(8)}$

$\underline{0111}\ \underline{0110}_{(2)} = 76_{(16)}$

⇨ $1110110_{(2)} = 166_{(8)} = 76_{(16)}$

I.1.2.2 TRECEREA NUMERELOR NATURALE DIN BAZA 2^A ÎN BAZA 2

Pentru convertirea numerelor naturale din baza 2^a în baza *2*, fiecare cifră/literă este înlocuită cu un grup de *a* biți în baza 2.

Exemple:

1. $3FB2D_{(16)} = ?_{(2)}$

   $3FB2D_{(16)} = \underline{0011\ 1111\ 1011}\ \underline{0010}\ \underline{1101}_{(2)}$

   ⇨ $3FB2D_{(16)} = 111111101100101101_{(2)}$

2. $765_{(8)} = ?_{(2)}$

   $765_{(8)} = \underline{111}\ \underline{110}\ \underline{101}_{(2)}$

   ⇨ $765_{(8)} = 111110101_{(2)}$

I.1.2.3 TRECEREA NUMERELOR NATURALE DIN BAZA 2^A ÎN BAZA 2^B

Pentru convertirea numerelor naturale din baza 2^a în baza 2^b fiecare cifră/literă este înlocuită cu un grup de *a* biți în baza 2, după care fiecare grup de *b* biți din numărul obținut în baza 2 este înlocuit cu cifra/litera corespunzătoare din baza 2^b pentru a obține noul număr.

Exemple:

1. $A3B8F_{(16)} = ?_{(8)}$

   010|100|011|101|110|001|111

   2 4 3 5 6 1 7

   ⇨ $A3B8F_{(16)} = 2435617_{(8)}$

2. $7654_{(8)} = ?_{(4)}$

   11|11|10|10|11|00

   3 3 2 2 3 0

   ⇨ $7654_{(8)} = 332230_{(4)}$

I.1.2.4 TRECEREA RAPIDĂ A NUMERELOR NATURALE DIN BAZA 10 ÎN BAZA 2

Pentru o convertire mai rapidă a numerelor naturale din baza 10 în baza *2* se poate folosi procedeul invers (de trecere din baza 2 în baza 10). Astfel, se scrie numărul de convertit ca sumă de puteri ale lui 2 și apoi pe baza acestor puteri se determină numărul în binar astfel: dacă puterea respectivă apare în sumă, atunci se trece valoarea 1 pe poziția corespunzătoare din numărul în baza 2; altfel, pe acea poziție se va trece valoarea 0.

Exemple:

1. $23_{(10)} = ?_{(2)}$

   $23 = 16 + 7 = 2^4 + 4 + 2 + 1 = 2^4 + 2^2 + 2^1 + 2^0$

   ⇨ $23_{(10)} = 10111_{(2)}$

2. $826_{(10)} = ?_{(2)}$

   $826 = 512 + 314 = 512 + 256 + 58 = 512 + 256 + 32 + 26 = 512 + 256 + 32 + 16 + 10 = 512 + 256 + 32 + 16 + 8 + 2 = 2^9 + 2^8 + 2^5 + 2^4 + 2^3 + 2^1$

   ⇨ $826_{(10)} = 1100111010_{(2)}$

I.1.3 SCHIMBAREA BAZEI DE NUMERAȚIE PENTRU NUMERELE SUBUNITARE

Trecerea numerelor subunitare din baza 10 în baza *b* se face prin înmulțiri succesive cu *b*. Astfel, numărul inițial se înmulțește cu *b*. Apoi, partea fracționară a produsului obținut se înmulțește cu *b* și procesul se repetă până când partea fracționară devine 0 sau se atinge precizia dorită. Rezultatul în baza *b* se obține prin plasarea după virgulă a numărului obținut prin concatenarea părții întregi a produselor obținute, în ordinea obținerii lor.

Exemple:

1. $0.125_{(10)} = ?_{(2)}$

   $0.125 * 2 = \underline{0}.25$

   $0.25 * 2 = \underline{0}.5$

   $0.5 * 2 = \underline{1}.0$

   ⇨ $0.125_{(10)} = 0.001_{(2)}$

2. $0.24_{(10)} = ?_{(5)}$

   $0.24 * 5 = \underline{1}.2$

   $0.2 * 5 = \underline{1}.0$

   ⇨ $0.24_{(10)} = 0.11_{(5)}$

I.1.4 SCHIMBAREA BAZEI DE NUMERAȚIE PENTRU NUMERELE RAȚIONALE

Pentru numerele raționale, se convertesc separat partea întreagă și partea fracționară a acestora conform regulilor descrise anterior.

Exemplu:

$25.34_{(10)} = ?_{(7)}$

$25_{(10)} = ?_{(7)}$

25 : 7 = 3 rest 4

3 : 7 = 0 rest 3

⇨ $25_{(10)} = 34_{(7)}$

$0.34_{(10)} = ?_{(7)}$

0.34 * 7 = $\underline{2}$.38

0.38 * 7 = $\underline{2}$.66

0.66 * 7 = $\underline{4}$.62

0.62 * 7 = $\underline{4}$.34

⇨ $0.34_{(10)} = 0.(2244)_{(7)}$

⇨ $25.34_{(10)} = 34.(2244)_{(7)}$

I.1.5 REPREZENTAREA NUMERELOR NEGATIVE ÎN BINAR

Pentru a reprezenta un număr negativ în binar, se transformă numărul pozitiv în baza 2, după care se neagă fiecare element al acestei transformări (rezultând astfel complementul față de 1 → c1), pentru ca apoi să se adauge 1 și astfel să se obțină numărul negativ dorit (complementul față de 2 → c2).

Exemplu:

$-36_{(10)} = ?_{(2)}$

$36_{(10)} = 32_{(10)} + 4_{(10)} = 100100_{(2)}$

$c1_{(36)}$: 011011 complementul față de 1 (se neagă toți biții)

$c2_{(36)}$: 011100 complementul fata de 2 (c1 + 1)

⇨ $-36_{(10)} = 011100_{(2)}$

Observație: Pentru a determina rapid reprezentarea binară a unui număr negativ se poate utiliza următoarea regulă pentru a evita trecerea prin complementul față de 1: în reprezentarea binară a numărului pozitiv se pornește de la cel mai puțin semnificativ bit spre cel mai semnificativ și se mențin la fel toți biții până la întâlnirea primei unități inclusiv. După această primă unitate (ce rămâne la fel ca în reprezentarea inițială), toți ceilalți biți rămași se vor nega.

Exemple:

1. $-36_{(10)} = ?_{(2)}$

   $36_{(10)} = 100\underline{100}_{(2)}$

   $-36_{(10)} = 011\underline{100}_{(2)}$

   ⇨ $-36_{(10)} = 011100_{(2)}$

2. $-9_{(10)} = ?_{(2)}$

   $9_{(10)} = 8_{(10)} + 1_{(10)} = 1001_{(2)}$

   $-9_{(10)} = 0111_{(2)}$

   ⇨ $-36_{(10)} = 0111_{(2)}$

I.2 Operații în baza 2

I.2.1 ADUNAREA

Pentru a aduna două sau mai multe numere în binar, se adună bit cu bit și se ține cont de transporturile ce pot să apară.

Observație: Dacă numerele au reprezentări în binar de lungimi diferite, atunci numerele mai scurte se completează cu zerouri înaintea celui mai semnificativ bit până când toate numerele au aceeași lungime.

Exemplu:

$14_{(10)} + 18_{(10)} = ?_{(2)}$

$14_{(10)} = 8_{(10)} + 4_{(10)} + 2_{(10)} = 1110_{(2)}$

$18_{(10)} = 16_{(10)} + 2_{(10)} = 10010_{(2)}$

$14_{(10)} + 18_{(10)} = \underline{0}1110_{(2)} + 10010_{(2)}$

01110 +

10010

$100000_{(2)} = 32_{(10)}$

⇨ $14_{(10)} + 18_{(10)} = 100000_{(2)} = 32_{(10)}$

I.2.2 ÎNMULȚIREA

Înmulțirea numerelor în bază 2 este similară cu o succesiune de adunări și deplasări. Astfel, se face înmulțirea cu ultimul bit (cel mai puțin semnificativ) din înmulțitor, după care se realizează o deplasare spre stânga și apoi se face înmulțirea cu următorul bit al înmulțitorului ș.a.m.d. până se termină toți biții înmulțitorului. La final, se adună bit cu bit și se ține cont de transport pentru a obține rezultatul înmulțirii.

Exemplu:

$14_{(10)} * 9_{(10)} = ?_{(2)}$

$14_{(10)} * 9_{(10)} = 1110_{(2)} * 1001_{(2)}$

1110

1001

1110

1110

$1111110_{(2)} = 126_{(10)}$

⇨ $14_{(10)} * 9_{(10)} = 1111110_{(2)} = 126_{(10)}$

I.2.3 SCĂDEREA (ADUNARE ÎN COD COMPLEMENTAR)

Pentru a realiza scăderea a două numere în binar se scrie scăzătorul ca număr negativ și în felul acesta scăderea se transformă în sumă.

Observație: Rezultatul obținut în urma operației de scădere trebuie să fie reprezentat pe același număr de biți ca operanzii implicați în operație. Dacă se obțin mai mulți biți, atunci bitul suplimentar este bit de semn și se ignoră în calculul valorii obținute.

Exemplu:

$47_{(10)} - 36_{(10)} = ?_{(2)}$

$47_{(10)} - 36_{(10)} = 47_{(10)} + (-36)_{(10)} = ?_{(2)}$

$47_{(10)} = 32_{(10)} + 8_{(10)} + 4_{(10)} + 2_{(10)} + 1_{(10)} = 101111_{(2)}$

$-36_{(10)} = 011100_{(2)}$

101111+

011100

1001011

⇨ $47_{(10)} - 36_{(10)} = 001011_{(2)} = 11_{(10)}$

Deoarece ambele numere au fost pe 6 biți și s-a realizat o scădere, înseamnă că și rezultatul va fi tot pe 6 biți și atunci se ignoră transportul de la cel mai semnificativ bit când se calculează numărul obținut. Drept urmare, rezultatul scăderii va fi $1011_{(2)} = 11_{(10)}$.

Alte observații:

Observația 1: Deși este ignorat în calculul numărului obținut în urma operației de scădere, totuși transportul de la cel mai semnificativ bit este important, acesta stabilind dacă rezultatul este pozitiv sau negativ. Astfel, dacă avem acest transport înseamnă că s-a obținut un rezultat pozitiv și valoarea acestuia se determină în mod direct. Dacă nu avem transport, înseamnă că rezultatul este negativ, iar pentru a determina adevărata valoare a rezultatului obținut, acesta trebuie reconvertit urmând pașii prezentați în sub-capitolul I.1.5 (reprezentarea numerelor negative în binar).

Pentru a exemplifica problemele ce pot apărea dacă se ignoră transportul, voi furniza următoarele exemple care sunt simetrice, dar care implică operații diferite datorită faptului că rezultatul la unul din exemple este pozitiv, în timp ce la celălalt e negativ.

Exemple:

$15_{(10)} - 12_{(10)} = ?_{(2)}$ și $12_{(10)} - 15_{(10)} = ?_{(2)}$

$12_{(10)} = 8_{(10)} + 4_{(10)} = 1\underline{100}_{(2)}$

$-12_{(10)} = 0\underline{100}_{(2)}$

$15_{(10)} = 8_{(10)} + 4_{(10)} + 2_{(10)} + 1_{(10)} = 111\underline{1}_{(2)}$

$-15_{(10)} = 000\underline{1}_{(2)}$

1. $15_{(10)} - 12_{(10)} = 1111_{(2)} + 0100_{(2)}$

   1111+

   0100

   10011

   ⇨ $15_{(10)} - 12_{(10)} = 0011_{(2)} = 3_{(10)}$

   Având transport de la cel mai semnificativ bit, înseamnă că s-a obținut un număr pozitiv și după ignorarea acestuia se obține rezultatul corect al operației ($0011_{(2)} = 3_{(10)}$).

2. $12_{(10)} - 15_{(10)} = 1100_{(2)} + 0001_{(2)}$

   1100+

   0001

   1101

   ⇨ $12_{(10)} - 15_{(10)} = 1101_{(2)} = 13_{(10)}$ → **rezultatul este incorect**!

   Neavând transport, înseamnă că s-a obținut un număr negativ și atunci el trebuie reconvertit pentru a se obține numărul real: $1101\underline{}_{(2)}$ → $0011_{(2)}$ → rezultatul este $-3_{(10)}$.

   ⇨ $12_{(10)} - 15_{(10)} = 1101\underline{}_{(2)} = -0011_{(2)} = -3_{(10)}$ → considerând inexistența transportului, **rezultatul este corect**!

Observația 2: Pentru a realiza corect operația de scădere, trebuie să se țină cont de numărul de biți pe care sunt reprezentate numerele implicate. Astfel, înainte de reprezentarea numărului negativ, se identifică care dintre cele două numere este mai mare și apoi se determină numărul de biți pe care se va lucra. Dacă numărul negativ ce trebuie reprezentat are mai puțini biți, atunci acestuia i se adaugă 0-uri înaintea celui mai semnificativ bit și apoi are loc reprezentarea, considerând inclusiv zerourile adăugate pentru a lucra pe același număr de biți.

Exemplu:

$25_{(10)} - 9_{(10)} = ?_{(2)}$

$25_{(10)} = 16_{(10)} + 8_{(10)} + 1_{(10)} = 11001_{(2)}$

$9_{(10)} = 8_{(10)} + 1_{(10)} = 1001_{(2)}$

Exemplu negativ: Dacă nu se ține cont că operația trebuie să se realizeze pe același număr de biți (5), atunci se vor obține următoarele valori: $c1_{(9)}$: 0110, $c2_{(9)}$: 0111, iar suma $25_{(10)} + (-9)_{(10)}$:

```
 11001+
  0111
----------
100000
```

⇨ $25_{(10)} - 9_{(10)} = 00000_{(2)} = 0_{(10)}$

În concluzie, rezultatul obținut ignorând transportul este 0, ceea ce este eronat.

Exemplu pozitiv: Dacă ținem cont că operația trebuie să se realizeze pe același număr de biți (5), atunci $9_{(10)} = 01001_{(2)}$, iar la calculul lui -9 vom obține următoarele rezultate: $c1_{(9)}$: 10110, $c2_{(9)}$: 10111. Astfel, suma $25_{(10)} + (-9)_{(10)}$ va fi:

```
 11001+
 10111
----------
110000
```

⇨ $25_{(10)} - 9_{(10)} = 10000_{(2)} = 16_{(10)}$

Ignorând transportul, rezultatul obținut este $10000_{(2)} = 16_{(10)}$, rezultatul corect al scăderii.

I.3 Probleme propuse

1. Transformați din baza 5 în baza 2 numărul $1234_{(5)}$.

2. Convertiți numerele 14 și 23 în binar și apoi realizați produsul reprezentărilor obținute.

3. Convertiți numerele 13 și 26 în binar și apoi calculați suma și diferența lor.

4. Ce număr decimal are reprezentare binară $10110010101_{(2)}$?

5. Convertiți număr $F0CA_{(16)}$ în octal prin două metode diferite.

6. Realizați scăderea 19 – 25. Ce număr binar se obține?

7. Realizați scăderea 21 – 40. Ce număr binar se obține?

8. Transformați numărul $987.65_{(10)}$ în baza 2.

9. Transformați numărul $234.98_{(10)}$ în baza 7.

II Algebra booleană

Algebra booleană este algebra formată din elementele 0 și 1, două operații binare ȘI (*, Λ) și SAU (V, +) și o operație unară NOT (~, !) (se va nota cu $\bar{x}$ de acum înainte).

O funcție f: $B^n \rightarrow B$ se numește **funcție booleană** <=> B={0,1}.

II.1 Axiomele algebrei booleene

1 Comutativitate: x * y = y * x; x + y = y + x;

2 Elemente de identitate distincte (1 pentru * și 0 pentru +):
x * 1 = 1 * x = x; x + 0 = 0 + x = x;

3 Distributivitate: x * (y + z) = x * y + x * z;

4 Pentru orice x există $\bar{x}$ astfel încât:

4.1 $x * \bar{x} = 0$ (principiul contradicției);

4.2 $x + \bar{x} = 1$ (principiul terțului exclus:).

II.2 Teoremele algebrei booleene

1 Elementul $\bar{x}$ asociat lui x prin axioma 4 a algebrei booleene este unic.

2 Principiul idempotenței: x * x = x, x + x = x;

3 Proprietățile elementelor neutre: x * 0 = 0, x + 1 = 1;

4 Principiul absorbției: $x_1 * (x_1 + x_2) = x_1$; $x_1 + (x_1 * x_2) = x_1$;

5 Asociativitate: x * y * z = x * (y * z) = (x * y) * z;
x + y + z = (x + y) + z = x + (y + z);

6 principiul dublei negații : $\overline{\overline{x}} = x$;

7 $\overline{0} = 1$ și $\overline{1} = 0$.

II.3 Formulele lui De Morgan

- $\overline{x_1 x_2} = \overline{x_1} + \overline{x_2}$
- $\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$

II.4 Formulele lui De Morgan generalizate

- $\overline{x_1 x_2 ... x_n} = \overline{x_1} + \overline{x_2} + ... + \overline{x_n}$
- $\overline{x_1 + x_2 + ... + x_n} = \overline{x_1} * \overline{x_2} * ... * \overline{x_n}$

Demonstrația este imediată, prin inducție.

II.5 Probleme propuse

1. Folosind axiomele algebrei booleene, demonstrați teoremele algebrei booleene.

2. Folosind axiomele algebrei booleene, demonstrați formulele lui De Morgan.

3. Demonstrați formulele lui De Morgan generalizate.

III Metode de reprezentare a funcțiilor booleene

III.1 Metode de reprezentare

Există mai multe metode de reprezentare a funcțiilor booleene:

- *Reprezentarea algebrică* – se definește funcția pe baza unei expresii folosind teoremele algebrei booleene.

  Exemplu: $f(x, y) = y + x * \overline{y}$

- *Reprezentarea prin tabele de adevăr* – se scriu într-un tabel toate combinațiile valorilor variabilelor de intrare, împreună cu valoarea pe care trebuie să o aibă funcția la ieșire.

  Exemplu: pentru funcția f de mai sus, tabelul de adevăr este:

x	**y**	**f**
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

- *Reprezentarea grafică prin diagrame Karnaugh și scheme logice*
- *Reprezentarea zecimală* – se asociază fiecărui monom o valoare zecimală și permite automatizarea minimizării.

Primele trei metode oferă o reprezentare intuitivă pentru un număr mic de variabile, fiind inadecvate pentru funcții booleene cu mai mult de patru variabile, în timp ce ultima metodă se folosește pentru funcții booleene cu mai mult de 5 variabile, dar și pentru a realiza o minimizare automată.

III.2 Reprezentarea algebrică

III.2.1 FORMA DISJUNCTIVĂ NORMALĂ A UNEI FUNCȚII

Definiție: Fie B={0,1} și o funcție booleană de n variabile x_0, x_1, ... x_{n-1}. Atunci f: $B^n \rightarrow B$ se poate scrie sub forma: $f(x_{n-1}, ..., x_0) = \Sigma\ y_{n-1} * ... * y_0$, unde $y_i = \begin{cases} x_i \\ \overline{x_i} \end{cases}, \forall i = \overline{1,n}$. Fiecare termen al sumei ($y_{n-1} * ... * y_0$) se numește **minterm** realizat cu variabilele x_{n-1}, … , x_0. Astfel, mintermii sunt niște produse (monoame) ce depind de toate variabilele de care depinde funcția (adică sunt complete).

Teoremă: Orice funcție booleană se poate scrie ca o sumă de mintermi, astfel obținându-se **forma disjunctivă normală (FDN)** a funcției.

Exemplu:

Fie f: B^3 ->B $f(x,y,z) = x*y+\overline{z}+y*z$. Se cere să se aducă f la FDN. Pentru ca f să fie în FDN, trebuie ca fiecare monom să conțină toate variabilele (lucru care acum nu se întâmplă). Pentru a putea introduce variabilele care lipsesc din monoame se va folosi principiul terțului exclus ($x+\overline{x}=1$) și proprietățile elementului neutru (x * 1 = x). În plus, se va mai folosi și principiul idempotenței (x + x = x) pentru a elimina duplicatele ce pot apare în urma desfacerii parantezelor.

$$f(x,y,z) = x*y+\overline{z}+y*z = x*y*1+\overline{z}*1*1+y*z*1$$
$$f(x,y,z) = x*y*(z+\overline{z})+\overline{z}*(x+\overline{x})*(y+\overline{y})+y*z*(x+\overline{x})$$
$$f(x,y,z) = x*y*z+x*y*\overline{z}+x*y*\overline{z}+x*\overline{y}*\overline{z}+\overline{x}*y*\overline{z}+$$
$$+\overline{x}*\overline{y}*\overline{z}+x*y*z+\overline{x}*y*z$$
$$f(x,y,z) = x*y*z+x*y*\overline{z}+x*\overline{y}*\overline{z}+\overline{x}*y*\overline{z}+\overline{x}*\overline{y}*\overline{z}+$$
$$+\overline{x}*y*z$$

O funcție ce conține toți mintermii posibili se spune că este în **forma disjunctivă normală completă (FDNC)**, iar valoarea acestei funcții este 1.

III.2.2 FORMA CONJUNCTIVĂ NORMALĂ A UNEI FUNCȚII

Suma logică a tuturor variabilelor în forma lor directă sau negată se numește **maxterm**.

O funcție este reprezentată în **forma conjunctivă normală (FCN)** dacă este reprezentată ca produs logic al maxtermilor săi.

Teoremă: Orice funcție booleană se poate scrie ca un produs de maxtermi, astfel obținându-se **forma conjunctivă normală (FCN)** a funcției.

Exemplu:

Fie f: B^3 ->B $f(x,y)=x$. Se cere să se aducă f la FCN.

$$f(x,y)=x=x+x=x+x*1=x+x*(y+\bar{y})=x+x*(y+\bar{y})+0$$
$$f(x,y)=x+x*y+x*\bar{y}+y*\bar{y}=x+x*\bar{y}+x*y+y*\bar{y}$$
$$f(x,y)=x*x+x*\bar{y}+x*y+y*\bar{y}=x*(x+\bar{y})+y*(x+\bar{y})$$
$$f(x,y)=(x+y)*(x+\bar{y})$$

Cele două forme (FCN și FDN) sunt interschimbabile; pentru a trece dintr-o formă în alta se folosesc formulele lui De Morgan.

Exemple:

1. Fie f: B^3 ->B $f(x,y)=x*y+\bar{y}$. Să se aducă f la FDN și apoi la FCN.

$$f(x,y)=x*y+\bar{y}=x*y+1*\bar{y}=x*y+(x+\bar{x})*\bar{y}$$
$$f(x,y)=x*y+x*\bar{y}+\bar{x}*\bar{y}\rightarrow FDN$$
$$f(x,y)=x*y+x*\bar{y}+\bar{x}*\bar{y}=x*y+x*\bar{y}+x*\bar{y}+\bar{x}*\bar{y}$$
$$f(x,y)=x*(y+\bar{y})+(x+\bar{x})*\bar{y}=x*1+1*\bar{y}=x+\bar{y}\rightarrow FCN$$

2. Fie f: B^3 ->B $f(x,y,z) = x * y + \bar{y}$. Să se aducă f direct la FCN.

$$f(x,y) = \overline{\overline{x * y + \bar{y}}} = \overline{\overline{x * y} * y} = \overline{(\bar{x} + \bar{y}) * y}$$
$$f(x,y) = \overline{(\bar{x} + \bar{y}) * (y + x) * (y + \bar{x})} = \overline{(\bar{x} + \bar{y}) * (x + y) * (\bar{x} + y)}$$
$$f(x,y) = \overline{(\bar{x} + \bar{y}) * (x + y) * (\bar{x} + y)} \rightarrow FCN$$

3. Fie : B^3 ->B $f(x,y) = (x + y) * (x + \bar{y})$. Să se aducă f la FDN.

$$f(x,y) = \overline{\overline{(x + y) * (x + \bar{y})}} = \overline{\bar{x} * \bar{y} + \bar{x} * y} \rightarrow FDN$$

III.3 Reprezentarea prin tabele de adevăr. Funcții echivalente

Funcțiile booleene se pot reprezenta prin tabele de adevăr. Pentru aceasta, se scriu într-un tabel toate combinațiile valorilor variabilelor de intrare, împreună cu valoarea pe care trebuie să o aibă funcția la ieșire.

Exemplu:

Fie funcția $f(x,y) = x + \bar{y}$. Tabelul de adevăr al acestei funcții se construiește prin identificarea tuturor combinațiilor de valori pe care le pot lua variabilele de intrare, precum și a valorilor funcției pentru aceste combinații. Astfel, se construiește tabelul de mai jos:

x	**y**	**f**
0	0	**1**
0	1	**0**
1	0	**1**
1	1	**1**

Observație: Pentru o mai ușoară construcție a tabelului de valori a unei funcții se poate folosi următoarea regulă: dacă funcția este dată în formă disjunctivă (sumă de produse), atunci se identifică combinațiile de valori pentru care fiecare monom (produs) ia valoarea 1 și se trece în tabelul de adevăr 1 pe acele poziții, la sfârșit completându-le pe restul cu 0-uri. Dacă funcția este dată în formă conjunctivă (produs de sume), atunci se identifică unde fiecare monom (produs) ia valoarea 0 și se trece în tabelul de adevăr 0 pe acele poziții, la sfârșit completându-le pe restul cu unități.

Pe baza tabelului de adevăr al funcțiilor, se poate stabili dacă două funcții sunt sau nu **echivalente**. Două funcții se numesc **echivalente** dacă iau aceleași valori la ieșire pentru fiecare combinație de valori ale variabilelor de intrare.

Exemplu:

Fie funcțiile f și g de mai jos. Verificați dacă sunt echivalente sau nu.

$$f(x,y,z) = x*y*z + x*y*\bar{z} + x*\bar{y}*\bar{z} + \bar{x}*y*\bar{z} + \bar{x}*\bar{y}*\bar{z} + \bar{x}*y*z$$

$$g(x,y,z) = x*y + \bar{z} + y*z$$

Construim tabelul de adevăr al celor 2 funcții pentru a verifica dacă iau aceleași valori pentru aceleași combinații ale variabilelor de intrare.

x	**y**	**z**	**f**	**g**
0	0	0	**1**	**1**
0	0	1	**0**	**0**
0	1	0	**1**	**1**
0	1	1	**1**	**1**
1	0	0	**1**	**1**
1	0	1	**0**	**0**
1	1	0	**1**	**1**
1	1	1	**1**	**1**

⇨ Cum f și g iau aceleași valori pentru toate combinațiile de valori ale variabilelor de intrare → f și g sunt echivalente.

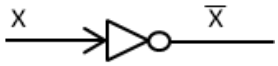
III.4 Reprezentarea grafică sub formă de circuite

Implementarea diferitelor funcții booleene se realizează folosind o serie de dispozitive (elemente de comutație) standard. Aceste dispozitive se numesc porți și sunt de mai multe feluri, fiecare generând o altă funcție.

III.4.1 CELE MAI IMPORTANTE ELEMENTE STANDARD DE COMUTAȚIE

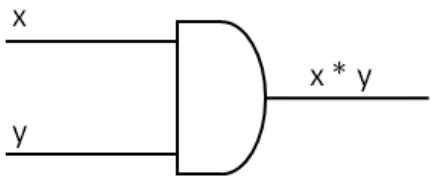
1. **Circuitul NOT (Negare)** implementează funcția f(x) = $\bar{x}$, are tabelul de adevăr și reprezentarea de mai jos:

x	f
0	**1**
1	**0**



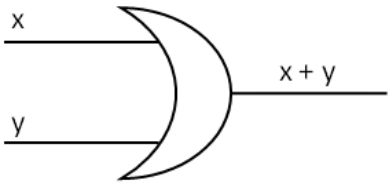
2. **Circuitul AND (ȘI)** implementează funcția f(x, y) = x * y, are tabelul de adevăr și reprezentarea de mai jos:

x	y	f
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**



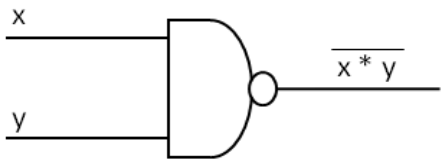
3. **Circuitul OR (SAU)** implementează funcția f(x, y) = x + y, are tabelul de adevăr și reprezentarea de mai jos:

x	y	f
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**



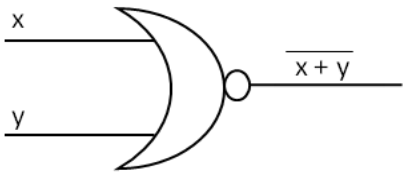
4. **Circuitul NAND (ȘI-NU)** implementează funcția f(x, y) = $\overline{x * y}$, are tabelul de adevăr și reprezentarea de mai jos:

x	**y**	**f**
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**



5. **Circuitul NOR (SAU-NU)** implementează funcția f(x, y) = $\overline{x + y}$, are tabelul de adevăr și reprezentarea de mai jos:

x	**y**	**f**
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**



Setul de operatori necesar și suficient pentru a reprezenta orice funcție booleană se numește **set complet de operatori**. Drept urmare, orice funcție booleană se poate reprezenta folosind diferite combinații ale elementelor de comutație corespunzătoare acestor operatori.

Astfel, setul (NOT, ȘI, SAU) este un set complet de operatori și orice funcție poate fi implementată folosind acești operatori.

În plus, folosind formulele lui De Morgan, operatorul SAU poate fi înlocuit de o combinație de operatori NOT și ȘI ($x + y = \overline{\overline{x} * \overline{y}}$). Similar, operatorul ȘI poate fi înlocuit de o combinație de operatori NOT și SAU ($x * y = \overline{\overline{x} + \overline{y}}$). Drept urmare, seturile (NOT, ȘI) și (NOT, SAU) sunt și ele seturi complete de operatori, iar orice funcție poate fi reprezentată folosind exclusiv elemente de comutație de tip NOT și ȘI sau de tip NOT și SAU.

Mai mult, orice funcție booleană se poate reprezenta folosind numai elemente de tip NAND sau NOR, de aceea existând elemente de comutație și pentru aceste funcții. Pentru a reprezenta o funcție doar cu elemente de tip NAND sau NOR se folosesc din nou formulele lui De Morgan.

III.5 Exemple rezolvate

1. Aduceți funcția $f(x,y,z) = x*\bar{y}*z + \bar{y}*\bar{z} + x*y + z$ în FDN și construiți tabelul de adevăr al celor două funcții.

Fie g forma FDN a funcției f. Pentru a o construi, trebuie ca fiecare monom să conțină toate variabilele. Drept urmare, se va folosi principiul terțului exclus și proprietățile elementului neutru pentru a introduce variabilele lipsă. În plus, se va folosi principiul idempotenței pentru a elimina monoamele duplicate.

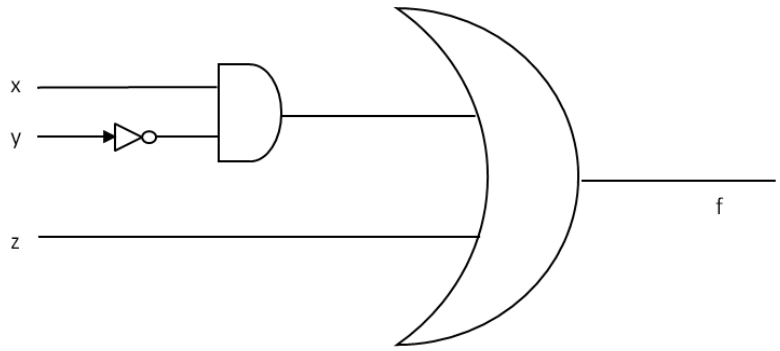
$$g(x,y,z) = x*\bar{y}*z + \bar{y}*\bar{z} + x*y + z = x*\bar{y}*z + \bar{y}*\bar{z}*1 + x*y*1 + z*1$$
$$g(x,y,z) = x*\bar{y}*z + \bar{y}*\bar{z}*(x+\bar{x}) + x*y*(z+\bar{z}) + z*(x+\bar{x})*(y+\bar{y})$$
$$g(x,y,z) = x*\bar{y}*z + x*\bar{y}*\bar{z} + \bar{x}*\bar{y}*\bar{z} + x*y*z + x*y*\bar{z} + x*y*z$$
$$+ x*\bar{y}*z + \bar{x}*y*z + \bar{x}*\bar{y}*z$$
$$g(x,y,z) = x*\bar{y}*z + x*\bar{y}*\bar{z} + \bar{x}*\bar{y}*\bar{z} + x*y*z + x*y*\bar{z} + +x*\bar{y}*z$$
$$+ \bar{x}*\bar{y}*z$$

Tabelul de adevăr al celor două funcții este:

x	y	z	f	g
0	0	0	**1**	**1**
0	0	1	**1**	**1**
0	1	0	**0**	**0**
0	1	1	**1**	**1**
1	0	0	**1**	**1**
1	0	1	**1**	**1**
1	1	0	**1**	**1**
1	1	1	**1**	**1**

Așa cum se poate vedea și din tabelul de adevăr, cele două funcții sunt echivalente.

2. Fie funcția $f(x,y,z) = x * \overline{y} + z$. Realizați circuitul corespunzător acestei funcții.

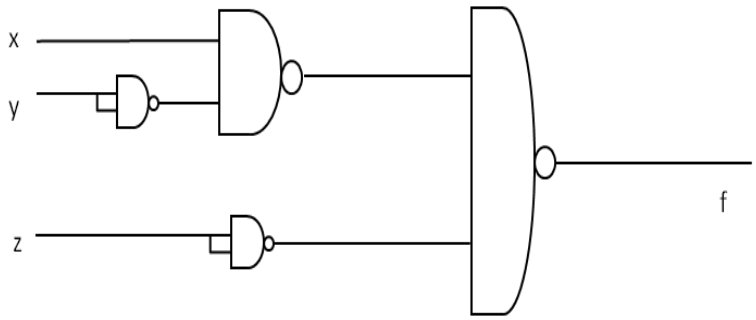


3. Implementați circuitul anterior folosind numai porți de tip NAND.

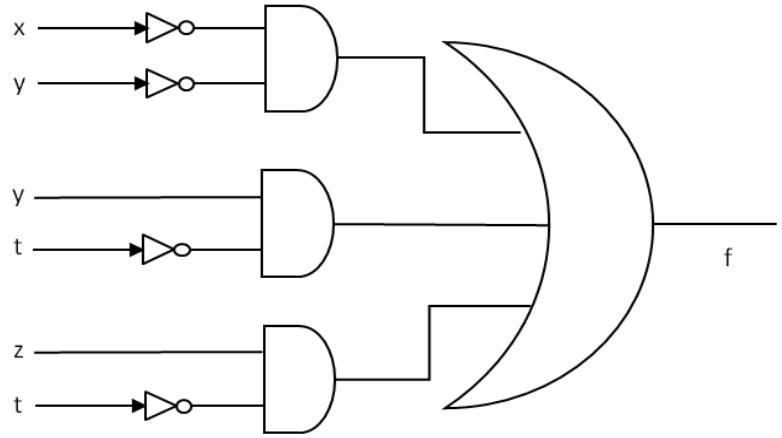
Pentru a putea implementa funcția f folosind doar elemente de tip NAND va trebui să se ajungă la o formă în care să existe doar produse negate. Drept urmare, va trebui să se nege de două ori expresia lui f și apoi să se aplice formulele lui De Morgan:

$$f(x,y,z) = x * \overline{y} + z = \overline{\overline{x * \overline{y} + z}} = \overline{\overline{x * \overline{y}} * \overline{z}}.$$

Astfel, circuitul devine:



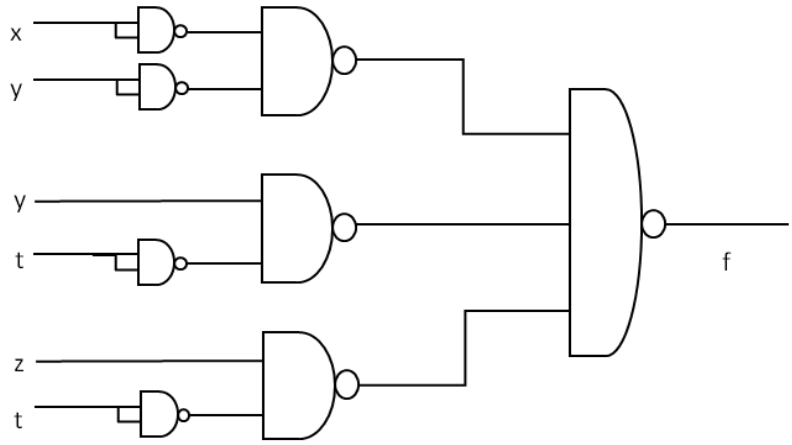
4. Realizați circuitul echivalent al funcției $f(x,y,z)=\bar{x}*\bar{y}+y*\bar{t}+z*\bar{t}$.



Aplicând formulele lui De Morgan, se obține funcția :

$$f(x,y,z)=\bar{x}*\bar{z}+\bar{x}*y+y*\bar{z}=\overline{\overline{\bar{x}*\bar{z}+\bar{x}*y+y*\bar{z}}}=\overline{\overline{\bar{x}*\bar{z}}*\overline{\bar{x}*y}*\overline{y*\bar{z}}}$$

Astfel, circuitul devine:



III.6 Probleme propuse

1. Verificați dacă $f(x,y)=x*y+x*\bar{y}+\bar{x}*\bar{y}$ și $g(x,y)=\overline{\bar{x}*\bar{y}+\bar{x}*y}$ sunt echivalente.

2. Verificați dacă $f(x,y)=(x+y)*(x+\bar{y})$ și $g(x,y)=x$ sunt echivalente.

3. Aduceți la forma disjunctivă normală funcția $f(x,y,z)=y+\bar{z}$.

4. Aduceți la forma conjunctivă normală funcția $f(x,y,z)=x*y*z+x*y*\bar{z}+x*\bar{y}*\bar{z}+\bar{x}*y*\bar{z}+\bar{x}*\bar{y}*\bar{z}+\bar{x}*y*z$

5. Aduceți la forma disjunctivă normală funcția $f(x,y,z)=(x+z)*(y+\bar{z})*(\bar{x}+y)*(x+\bar{y}+\bar{z})$.

6. Construiți tabelul de adevăr al funcției $f(x,y,z,t)=\bar{x}*\bar{y}+\bar{y}*\bar{z}*\bar{t}$.

7. Construiți tabelul de adevăr al funcției $f(x,y,z,t)=(\bar{x}+\bar{y})*(\bar{y}+\bar{z}+\bar{t})$.

8. Construiți tabelul de adevăr al funcției $f(x,y,z,t)=(x+\bar{y})*(\bar{x}+z+\bar{t})*(x+y+z)*(\bar{y}+t)*(x+\bar{z})$.

9. Realizați circuitul funcției de mai jos folosind numai porți de tip NAND: $f(x,y,z)=x*\bar{y}+\bar{y}*z+\bar{x}*y$

10. Realizați circuitul funcției de mai jos folosind numai porți de tip NAND:

$$f(x,y,z,t)=\bar{x}*\bar{y}*\bar{t}+\bar{x}*z*\bar{t}+\bar{x}*y*\bar{z}*t+x*y*z*t+x*\bar{y}*\bar{z}$$

11. Realizați circuitul funcției de mai jos folosind numai porți de tip NAND:

$$f(x,y,z,t)=\bar{x}*\bar{y}*\bar{t}+\bar{x}*y*\bar{z}*t+x*y*z*\bar{t}+x*\bar{y}*\bar{z}*\bar{t}+x*\bar{y}*\bar{z}*t$$

IV Reprezentarea grafică folosind diagrame Karnaugh

IV.1 Diagramele Karnaugh

Diagramele Karnaugh sunt reprezentări derivate din diagramele Venn pentru mulțimi și sunt folosite pentru minimizarea funcțiilor booleene.

Observație: Într-o diagramă Karnaugh, două căsuțe adiacente diferă printr-un singur bit!

Pentru a construi diagrama Karnaugh a unei funcții booleene, aceasta trebuie să fie adusă în prealabil în FDN. Apoi, pentru fiecare monom prezent în FDN a funcției de minimizat se trece valoarea 1 în căsuța corespunzătoare din diagramă.

Exemple:

1. Realizați tabelul de adevăr și diagrama Karnaugh pentru funcția:

$$f(x,y,z) = x*y*z + x*y*\bar{z} + x*\bar{y}*\bar{z} + \bar{x}*y*\bar{z} + \bar{x}*\bar{y}*\bar{z} + \bar{x}*y*z$$

x	y	z	f
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

Diagrama Karnaugh este:

yz / x	00	01	11	10
0	1	0	1	1
1	1	0	1	1

2. Realizați diagrama Karnaugh pentru funcția dată prin următorul tabel de adevăr:

x	y	z	g
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Diagrama Karnaugh este:

yz / x	00	01	11	10
0	1	1	1	1
1	0	0	0	0

3. Realizați diagrama Karnaugh pentru funcția dată prin următorul tabel de adevăr:

x	**y**	**z**	**t**	**h**
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**

Diagrama Karnaugh este:

zt / xy	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

IV.2 Minimizarea funcțiilor booleene folosind Diagramele Karnaugh

Grupând căsuțe adiacente din diagrama Karnaugh putem obține monoame care depind de mai puține variabile decât cele inițiale. Astfel se pot grupa **2, 4, 8, 16 căsuțe adiacente** care să formeze un **dreptunghi sau un pătrat** (căsuțele unite trebuie să fie ori pe aceeași linie, ori pe aceeași coloană, ori să creeze pătrate).

Algoritm de minimizare:

Pasul 1: Pentru fiecare produs din funcția de minimizat se trece valoarea 1 în căsuța corespunzătoare din diagramă;

Pasul 2: Se formează suprafețe de minimizare pătrate sau dreptunghiulare alcătuite din 1, 2, 4, 8 sau 16 pătrățele adiacente **ce conțin unități**;

Observație: Căsuțele de pe prima linie sunt adiacente cu cele de pe ultima linie, iar cele de pe prima coloană sunt adiacente cu cele de pe ultima coloană.

Proprietăți ale suprafețelor de minimizare:

a. trebuie să aibă aria cât mai mare;

b. trebuie să fie în număr cât mai redus;

c. o căsuță ce conține 1 trebuie să fie conținută în cel puțin o suprafață.

Pasul 3: Se formează produse pe baza suprafețelor de minimizare astfel:

a. dacă în interiorul unei suprafețe de minimizare o variabilă își schimbă valoarea, atunci aceasta nu mai apare în produs;

b. dacă o variabilă își păstrează valoarea 0, atunci ea apare în produs în forma negată;

c. dacă o variabilă își păstrează valoarea 1, atunci ea apare în produs în forma directă;

Pasul 4: Expresia minimizată este dată de suma produselor obținute la pasul 3.

Observații:

Observația 1: Căsuțele ce conțin 0-uri trebuie să **nu** fie cuprinse în nicio suprafață de minimizare.

Observația 2: Pentru o mai rapidă completare a diagramelor și pentru a ușura vizualizarea, căsuțele ce conțin 0-uri pot să rămână necompletate (căsuțele ce conțin 0 pot rămâne goale în diagrama Karnaugh).

Exemple:

1. Minimizați funcția f definită mai sus (în exemplul 1) folosind diagrama Karnaugh deja realizată.

x \ yz	00	01	11	10
0	1	0	1	1
1	1	0	1	1

$f(x, y, z) = y + \bar{z}$

2. Minimizați funcția h definită mai sus (în exemplul 3) folosind diagrama Karnaugh deja realizată.

xy \ zt	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	1	1	0
10	0	1	1	0

$g(x, y, z, t) = \bar{x} + t$

3. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**

Diagrama Karnaugh este:

zt / xy	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$$f(x,y,z,t) = y * t + \bar{y} * \bar{t}$$

4. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

Diagrama Karnaugh este:

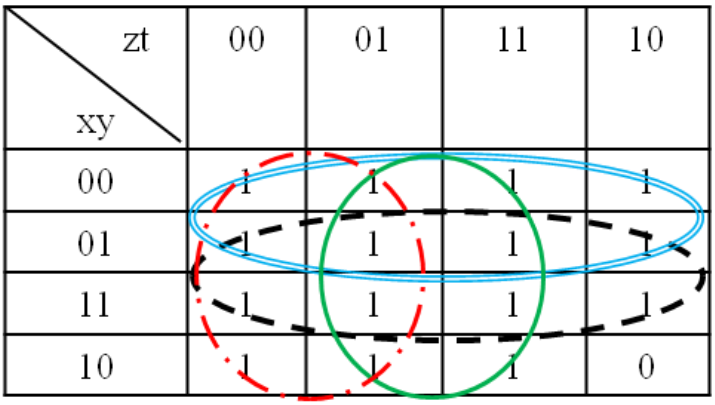
zt / xy	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

f(x, y, z, t) = 1 → FDNC

5. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**	$\overline{f}$
0	0	0	0	**1**	**0**
0	0	0	1	**1**	**0**
0	0	1	0	**1**	**0**
0	0	1	1	**1**	**0**
0	1	0	0	**1**	**0**
0	1	0	1	**1**	**0**
0	1	1	0	**1**	**0**
0	1	1	1	**1**	**0**
1	0	0	0	**1**	**0**
1	0	0	1	**1**	**0**
1	0	1	0	**0**	**1**
1	0	1	1	**1**	**0**
1	1	0	0	**1**	**0**
1	1	0	1	**1**	**0**
1	1	1	0	**1**	**0**
1	1	1	1	**1**	**0**

Diagrama Karnaugh este:



$$f(x,y,z,t) = \overline{x} + y + \overline{z} + t$$

În particular, această funcție se poate rezolva mai ușor folosind principiul dublei negații: $f(x,y,z,t) = \overline{\overline{f(x,y,z,t)}}$. De aceea, în tabelul de adevăr au fost trecute și valorile lui $\overline{f(x,y,z,t)}$, aceasta fiind de fapt funcția care va fi minimizată, iar rezultatul obținut va fi negat pentru a obține valoarea lui $f(x,y,z,t)$.

xy \ zt	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	1

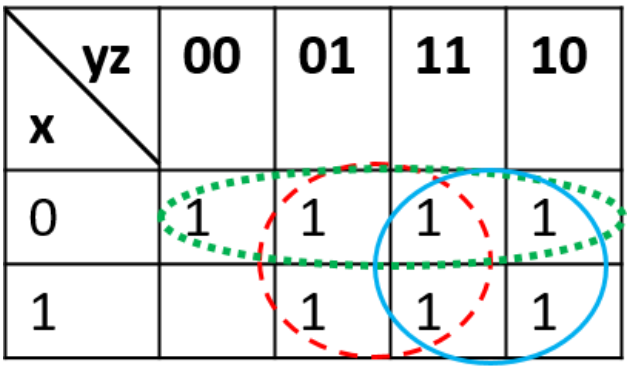
$$\overline{f(x,y,z,t)} = x * \bar{y} * z * \bar{t} \rightarrow f(x,y,z,t) = \overline{\overline{f(x,y,z,t)}}$$

$$f(x,y,z,t) = \overline{x * \bar{y} * z * \bar{t}} = \bar{x} + y + \bar{z} + t$$

6. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

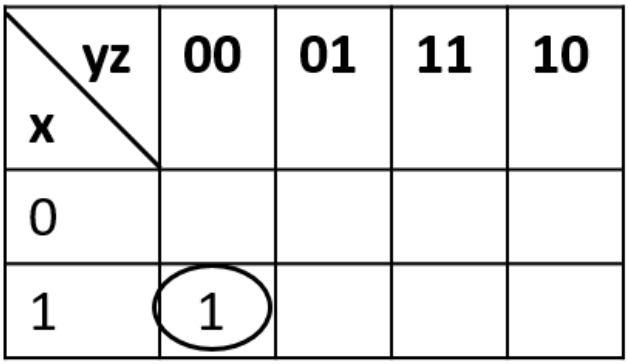
x	**y**	**z**	**f**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Diagrama Karnaugh este:



$$f(x,y,z) = \bar{x} + y + z$$

Întrucât lipsește un singur produs din funcție, este mai ușor să se nege funcția, să se minimizeze funcția rezultată și apoi să se nege rezultatul minimizării pentru a ajunge la minimizarea funcției inițiale, folosind proprietatea că $\bar{\bar{x}} = x$.



$$\overline{f(x,y,z)} = x * \bar{y} * \bar{z} \rightarrow$$

$$f(x,y,z) = \overline{\overline{f(x,y,z)}} = \overline{x * \bar{y} * \bar{z}} = \bar{x} + y + z$$

7. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	y	z	t	f
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

Diagrama Karnaugh este:

xy \ zt	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	0	1	1	1
10	1	0	0	0

$$f(x,y,z,t) = x * \bar{y} * \bar{z} * \bar{t} + y * t + y * z + \bar{x} * z * \bar{t}$$

IV.3 Minimizarea funcțiilor incomplet specificate

Funcțiile incomplet specificate sunt acele funcții la care nu este important rezultatul pentru anumite combinații ale datelor de intrare (în general datorită faptului că acele combinații nu se vor întâlni niciodată). Din această cauză, rezultatul funcției pentru acele combinații poate fi ales liber, cât mai convenabil pentru minimizare. Rezultatele care nu sunt importante (pentru combinațiile de intrare care nu sunt folosite) se trec în tabelul de adevăr cu *.

Observații:

Observația 1: La minimizarea funcțiilor incomplet specificate, **nu** se realizează suprafețe alcătuite doar din *!

Observația 2: Numărul maxim de * din tabelul de adevăr al unei funcții este 2^{N-1}-1, unde N este numărul de variabile de care depinde funcția. Cu alte cuvinte, numărul maxim de combinații de intrare care nu sunt folosite (valori care pot să nu fie importante) la o funcție este de jumătate din valori – 1. Dacă ar fi mai multe combinații de intrare nefolosite, ar însemna că funcția se poate reprezenta pe mai puține variabile de intrare și nu s-ar justifica folosirea unei astfel de variabile suplimentare.

Exemple:

1. Minimizați funcția dată prin diagrama Karnaugh de mai jos:

x \ yz	00	01	11	10
0	1	*	1	1
1	0	*	0	1

$f(x,y,z,t) = \bar{x} + y * \bar{z}$

2. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**
0	0	0	0	*
0	0	0	1	**1**
0	0	1	0	*
0	0	1	1	**1**
0	1	0	0	*
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	*
1	0	1	0	*
1	0	1	1	**0**
1	1	0	0	**1**
1	1	0	1	*
1	1	1	0	*
1	1	1	1	**0**

Diagrama Karnaugh este:

xy \ zt	00	01	11	10
00	*	1	1	*
01	*	1		1
11	1	*		*
10		*		*

$$f(x,y,z,t) = \bar{x} * \bar{y} + y * \bar{z} + z * \bar{t}$$

3. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	*
0	0	1	1	**1**
0	1	0	0	*
0	1	0	1	*
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	*
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	*

Diagrama Karnaugh este:

xy \ zt	00	01	11	10
00		1	1	*
01	*	*		1
11		1	*	
10		1	*	1

$$f(x,y,z,t) = \bar{z} * t + \bar{y} * z + \bar{x} * z * \bar{t}$$

IV.4 Funcții cu mai mult de 4 variabile – trecerea variabilelor în coloana rezultat

Așa cum s-a specificat anterior, diagramele Karnaugh sunt potrivite pentru minimizarea funcțiilor cu un număr mic de variabile (până acum s-a arătat cum se minimizează funcții ce depind de maxim 4 variabile). Există totuși situații când funcțiile depind de mai multe variabile și se dorește să se folosească aceleași diagrame Karnaugh pentru minimizarea lor. În astfel de cazuri, variabilele suplimentare se introduc în coloana rezultat, considerând că funcția depinde de mai puține variabile și putând aplica o versiune modificată ce permite minimizarea, folosind principiul superpoziției.

Pentru introducerea variabilelor în coloana rezultat trebuie să se țină cont de valorile inițiale ale funcției, precum și de numărul de variabile ce se doresc a fi introduse în coloana rezultat. Astfel, dacă se dorește eliminarea unei singure variabile de care depinde funcția, trebuie ca în tabelul de adevăr să avem 2^{n-1} rânduri în loc de 2^n câte aveam inițial. Cu alte cuvinte, fiecare linie din noul tabel trebuie să conțină informația existentă pe 2 linii în tabelul inițial. Dacă valorile din tabelul inițial erau identice, atunci în noul tabel se trece acea valoare. Dacă în schimb valorile erau diferite, atunci se folosește variabila ce se elimină pentru a codifica cele două valori.

Exemple:

1. Pentru funcția dată prin tabelul de adevăr de mai jos, introduceți variabila z în coloana rezultat:

x	y	z	f
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Prin eliminarea variabilei z, noul tabel va avea doar 4 linii (x = 0, y = 0; x = 0, y = 1; x = 1, y = 0 și x = 1, y = 1). Drept urmare, valorile de pe primele două linii vor trebui să fie combinate și apoi trecută o singură valoare pe prima linie din noul tabel; valorile de pe următoarele două linii vor trebui combinate și apoi trecută o singură valoare pe cea de-a doua linie din noul tabel etc.

Cum pe primele două linii este aceeași valoare (1) a funcției f, înseamnă că pe prima linie din noul tabel se va trece valoarea respectivă (1). În schimb, pe liniile 3 și 4 din tabelul inițial al lui f sunt valori diferite. Când z = 0, f = 1, iar când z = 1, f = 0. Cum în noul tabel trebuie trecută o singură valoare, va trebui să se facă o legătură între valoarea lui f și valoarea lui z, această variabilă fiind singura care influențează ieșirea lui f când x = 0 și y = 1. Astfel, se observă că valoarea lui f este exact negatul valorii lui z și drept urmare în noul tabel se va trece $\bar{z}$ ca valoare a lui f. În continuare, din nou sunt două linii ce au aceeași valoare (0) a lui f, astfel că valoarea respectivă se trece în noul tabel. În schimb, pe ultimele două linii f ia valori diferite și se observă că atunci când z = 0, f = 0, iar când z = 1, f = 1. Drept urmare, f = z și în noul tabel se va trece valoarea z pe linia corespunzătoare combinației x = 1, y = 1. Astfel, tabelul devine:

x	**y**	**f**
0	0	**1**
0	1	$\bar{z}$
1	0	**0**
1	1	**z**

2. Pentru funcția din exemplul anterior, introduceți variabila x în coloana rezultat:

În acest caz, trebuie combinate liniile 1 și 5 din tabelul inițial, apoi 2 și 6, 3 și 7 și, în final, 4 și 8. Așa cum se poate observa, pentru primele 3 linii din noul tabel avem aceleași combinații de valori, și anume când x = 0, f = 1, iar când x = 1, f = 0. Drept urmare, f = $\bar{x}$ și în noul tabel, pe primele 3 linii, se va trece această valoare. În cazul ultimei linii din tabel se observă că atunci când x = 0, f = 0, iar când x = 1, f = 1. Cu alte cuvinte, f = x. Astfel, tabelul devine:

y	**z**	**f**
0	0	$\bar{x}$
0	1	$\bar{x}$
1	0	$\bar{x}$
1	1	**x**

3. Pentru funcția dată prin tabelul de adevăr de mai jos, introduceți variabila t în coloana rezultat:

x	y	z	t	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

În noul tabel vor fi 8 intrări, valoarea de pe linia i a acestuia fiind obținută prin combinarea valorilor de pe liniile $2^i - 1$ și 2^i din vechiul tabel. Astfel noul tabel devine:

x	**y**	**z**	**f**
0	0	0	$\bar{t}$
0	0	1	**t**
0	1	0	$\bar{t}$
0	1	1	$\bar{t}$
1	0	0	**0**
1	0	1	**t**
1	1	0	**1**
1	1	1	**1**

Odată învățată modalitatea prin care se pot trece variabile în coloana rezultat, se poate trece la prezentarea algoritmului de minimizare al acestor funcții.

IV.5 Minimizarea funcțiilor cu mai mult de 4 variabile (cu variabile în coloana rezultat)

Algoritm de minimizare:

Pasul 1: Se construiește diagrama Karnaugh pornind de la valorile lui f.

Pasul 2: Se realizează suprafețe de minimizare care să conțină toate variabilele. Fiecare suprafață formată în acest pas trebuie să conțină un același literal (variabilă). Două suprafețe de minimizare formate în această etapă pot să aibă în comun doar căsuțe 1, * sau același literal. Pe baza suprafețelor de minimizare se vor forma produse (monoame) astfel: la produsul (monomul) obținut doar cu unități (conform algoritmului standard de minimizare) se adaugă variabila ce apare în interiorul suprafeței.

Observație: În acest pas **nu** se formează suprafețe de minimizare care nu conțin literali (variabile) – nu se formează suprafețe care să conțină numai căsuțe 1 sau *.

Pasul 3: Toate căsuțele ce conțin variabile sunt înlocuite cu valoarea 0 și se minimizează diagrama rezultată în mod obișnuit.

Pasul 4: Forma minimizată este dată de suma produselor obținute în pașii 2 si 3.

Observații:

Observația 1: Dacă în pasul 2 apar variabile ce sunt una negata celeilalte (de exemplu A și $\overline{A}$), cele două variabile se consideră a fi **independente** și drept urmare **nu** se pot grupa în aceeași suprafață de minimizare.

Observația 2: Dacă în pasul 2 apar variabile ce sunt una negata celeilalte (de exemplu A și $\overline{A}$), cele două variabile se consideră a fi **independente** datorită principiului superpoziției pe care se bazează minimizarea acestui gen de diagrame Karnaugh. Cu alte cuvinte, în pasul 3 ambele variabile sunt înlocuite cu 0, chiar dacă $\overline{0}=1$ (nu se consideră că dacă A = 0, atunci $\overline{A}$=1, ci ambele devin 0 întrucât sunt evaluate „pe rând").

Exemple:

1. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	y	z	t	f
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**A**
0	1	0	0	**1**
0	1	0	1	**A**
0	1	1	0	**0**
0	1	1	1	**A**
1	0	0	0	**1**
1	0	0	1	**A**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**A**
1	1	1	1	**1**

Pasul 1:

zt / xy	00	01	11	10
00		1	A	1
01	1	A	A	
11		1	1	A
10	1	A	1	

Pasul 2: minimizarea suprafețelor ce conțin literali:

$$f_1\ (x,y,z,t,A) = t * A + x * y * z * A$$

Pasul 3: refacerea diagramei Karnaugh prin eliminarea tuturor variabilelor și apoi minimizarea acesteia:

zt / xy	00	01	11	10
00		1		1
01	1			
11		1	1	
10	1		1	

$$f_2\ (x,y,z,t,A) = \bar{x} * \bar{y} * \bar{z} * t + \bar{x} * \bar{y} * z * \bar{t} + \bar{x} * y * \bar{z} * \bar{t} +$$
$$+ x * y * t + x * z * t + x * \bar{y} * \bar{z} * \bar{t}$$

Pasul 4: însumarea rezultatelor din pașii 2 și 3 pentru obținerea formei finale a funcției.

$$f\ (x,y,z,t,A) = f_1\ (x,y,z,t,A) + f_2\ (x,y,z,t,A) = t * A +$$
$$+ x * y * z * A + \bar{x} * \bar{y} * \bar{z} * t + \bar{x} * \bar{y} * z * \bar{t} + \bar{x} * y * \bar{z} * \bar{t} +$$
$$+ x * y * t + x * z * t + x * \bar{y} * \bar{z} * \bar{t}$$

2. Realizați diagrama Karnaugh și apoi minimizați funcția dată prin tabelul de adevăr de mai jos:

x	y	z	t	f
0	0	0	0	**1**
0	0	0	1	**A**
0	0	1	0	**1**
0	0	1	1	$\overline{A}$
0	1	0	0	$\overline{A}$
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**A**
1	0	0	0	**A**
1	0	0	1	**1**
1	0	1	0	**0**
1	0	1	1	$\overline{A}$
1	1	0	0	**0**
1	1	0	1	$\overline{A}$
1	1	1	0	**1**
1	1	1	1	$\overline{A}$

Pașii 1+2:

zt / xy	00	01	11	10
00	1	A	$\overline{A}$	1
01	$\overline{A}$	1	A	
11		$\overline{A}$	$\overline{A}$	1
10	A	1	$\overline{A}$	

$$f_1(x,y,z,t,A,\overline{A}) = \overline{y} * \overline{z} * A + \overline{x} * y * t * A + \overline{x} * \overline{y} * z * \overline{A} +$$
$$+ \overline{x} * y * \overline{z} * \overline{A} + x * t * \overline{A}$$

Pasul 3:

zt / xy	00	01	11	10
00	1			1
01		1		
11				1
10		1		

$$f_2(x,y,z,t,A,\overline{A}) = \overline{x} * \overline{y} * \overline{t} + \overline{x} * y * \overline{z} * t + x * y * z * \overline{t} + x * \overline{y} * \overline{z} * t$$

Pasul 4:

$$f(x,y,z,t,A,\overline{A}) = f_1(x,y,z,t,A,\overline{A}) + f_2(x,y,z,t,A,\overline{A})$$

$$f(x,y,z,t,A,\overline{A}) = \overline{y} * \overline{z} * A + \overline{x} * y * t * A + \overline{x} * \overline{y} * z * \overline{A} + x * t * \overline{A} +$$
$$+ \overline{x} * y * \overline{z} * \overline{A} + \overline{x} * \overline{y} * \overline{t} + \overline{x} * y * \overline{z} * t + x * y * z * \overline{t} + x * \overline{y} * \overline{z} * t$$

3. Minimizați funcția dată prin diagrama Karnaugh de mai jos:

zt / xy	00	01	11	10
00	1		A	1
01		1		1
11	A		1	A
10	1	1	A	A

Pasul 2:

xy \ zt	00	01	11	10
00	1		A	1
01		1		1
11	A		1	A
10	1	1	A	A

$$f_1(x,y,z,t,A) = x * \bar{t} * A + \bar{y} * z * A$$

Pasul 3:

xy \ zt	00	01	11	10
00	1			1
01		1		1
11			1	
10	1	1		

$$f_2(x,y,z,t,A) = \bar{x} * \bar{y} * \bar{t} + x * \bar{y} * \bar{z} + \bar{x} * z * \bar{t} + \bar{x} * y * \bar{z} * t + \\ + x * y * z * t$$

Pasul 4:

$$f(x,y,z,t,A) = x * \bar{t} * A + \bar{y} * z * A + \bar{x} * \bar{y} * \bar{t} + x * \bar{y} * \bar{z} + \\ + \bar{x} * z * \bar{t} + \bar{x} * y * \bar{z} * t + x * y * z * t$$

4. Minimizați funcția dată prin diagrama Karnaugh de mai jos:

xy \ zt	00	01	11	10
00	A	1		$\overline{A}$
01	1	A	$\overline{A}$	
11			$\overline{A}$	1
10	A	1	1	

Pasul 2:

xy \ zt	00	01	11	10
00	A	1		$\overline{A}$
01	1	A	$\overline{A}$	
11			$\overline{A}$	1
10	A	1	1	

$$f_1(x,y,z,t,A,\overline{A}) = \overline{x} * \overline{z} * A + \overline{y} * \overline{z} * A + y * z * t * \overline{A} +$$
$$+ \overline{x} * \overline{y} * z * \overline{t} * \overline{A}$$

Pasul 3:

xy \ zt	00	01	11	10
00		1		
01	1			
11				1
10		1	1	

$$f_2(x,y,z,t,A,\overline{A}) = \overline{y} * \overline{z} * t + x * \overline{y} * t + \overline{x} * y * \overline{z} * \overline{t} + x * y * z * \overline{t}$$

Pasul 4:

$$f(x,y,z,t,A,\overline{A}) = \overline{x} * \overline{z} * A + \overline{y} * \overline{z} * A + y * z * t * \overline{A} +$$
$$+ \overline{x} * \overline{y} * z * \overline{t} * \overline{A} + \overline{y} * \overline{z} * t + x * \overline{y} * t + \overline{x} * y * \overline{z} * \overline{t} + x * y * z * \overline{t}$$

IV.6 Minimizarea funcțiilor cu mai mult de 4 variabile incomplet specificate

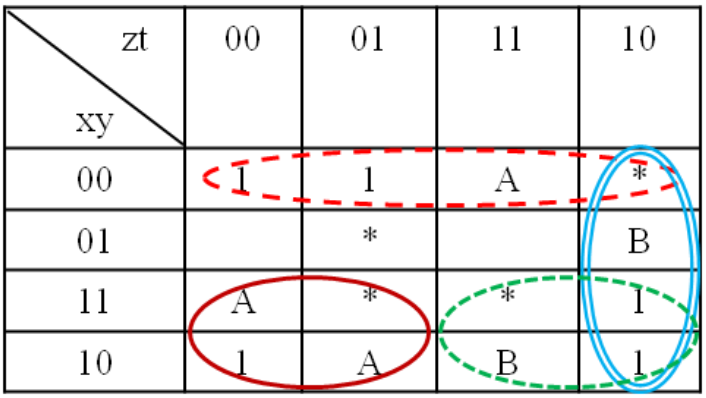
Observație: La minimizarea funcțiilor incomplet specificate cu variabile în coloana rezultat, evaluarea steluțelor dintr-un pas este independentă de evaluarea steluțelor din celălalt pas.

Exemple:

1. Realizați circuitul următoarei funcții folosind porți de tip NAND:

x	y	z	t	f
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	*****
0	0	1	1	**A**
0	1	0	0	**0**
0	1	0	1	*****
0	1	1	0	**B**
0	1	1	1	**0**
1	0	0	0	**1**
1	0	0	1	**A**
1	0	1	0	**1**
1	0	1	1	**B**
1	1	0	0	**A**
1	1	0	1	*****
1	1	1	0	**1**
1	1	1	1	*****

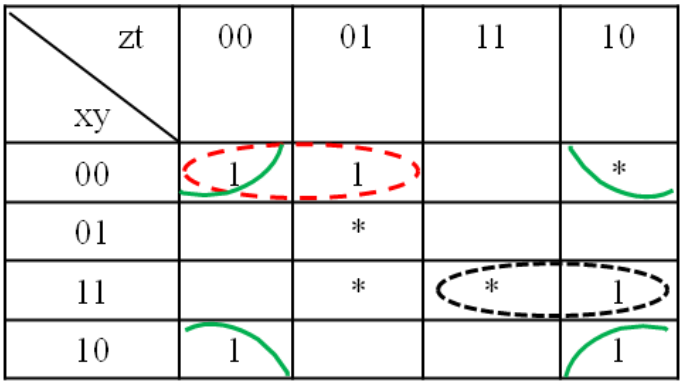
Pașii 1+2:



xy \ zt	00	01	11	10
00	1	1	A	*
01		*		B
11	A	*	*	1
10	1	A	B	1

$$f_1(x,y,z,t,A,B) = \bar{x} * \bar{y} * A + x * \bar{z} * A + z * \bar{t} * B + x * z * B$$

Pasul 3:



xy \ zt	00	01	11	10
00	1	1		*
01		*		
11		*	*	1
10	1			1

$$f_2(x,y,z,t,A,B) = \bar{y} * \bar{t} + \bar{x} * \bar{y} * \bar{z} + x * y * z$$

Pasul 4:

$$f(x,y,z,t,A,B) = \bar{x} * \bar{y} * A + x * \bar{z} * A + z * \bar{t} * B + x * z * B +$$
$$+ \bar{y} * \bar{t} + \bar{x} * \bar{y} * \bar{z} + x * y * z$$

Pentru a implementa circuitul cu porți de tip NAND, trebuie să se nege de două ori funcția și apoi să se desfacă una dintre negări folosind formulele lui De Morgan, astfel transformând suma în produs negat.

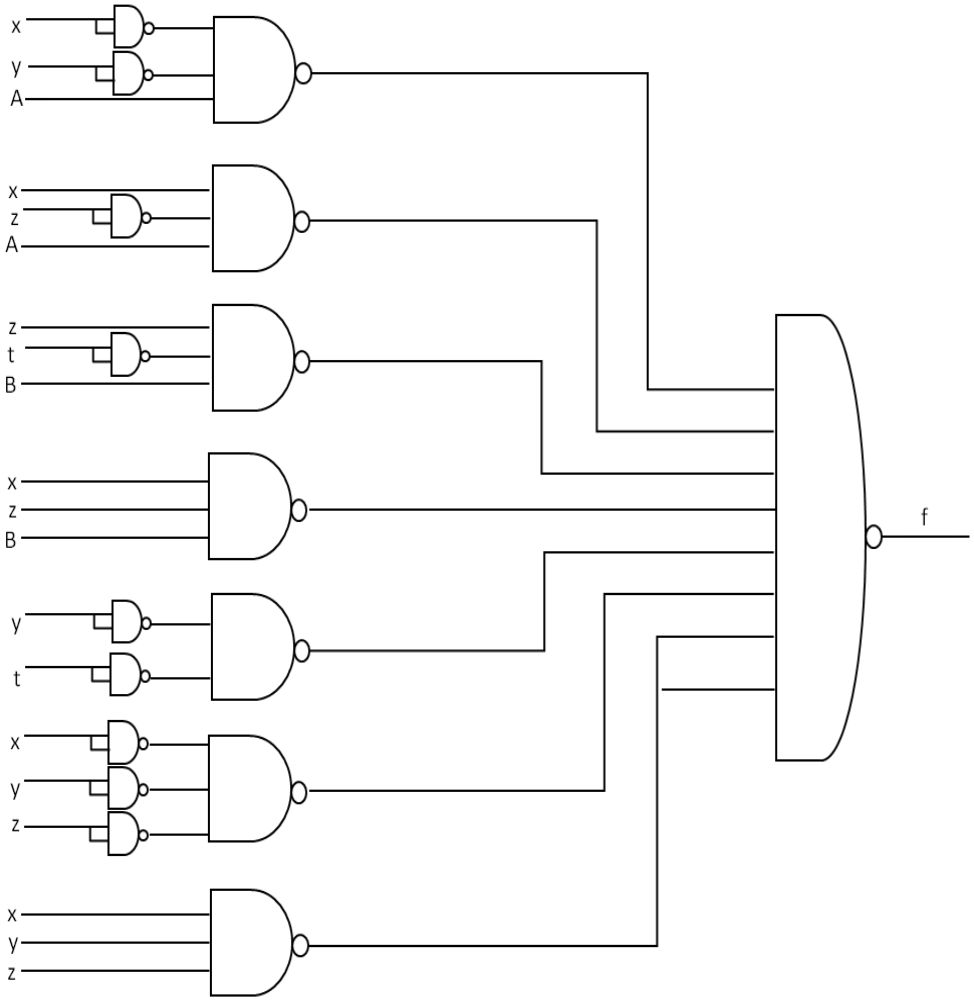
$$f(x,y,z,t,A,B) = \overline{\overline{f(x,y,z,t,A,B)}} = \overline{\overline{\bar{x} * \bar{y} * A + x * \bar{z} * A + }}$$

$$\overline{\overline{+ z * \bar{t} * B + x * z * B + \bar{y} * \bar{t} + \bar{x} * \bar{y} * \bar{z} + x * y * z}}$$

$$f(x,y,z,t,A,B) = \overline{\overline{\bar{x} * \bar{y} * A}} * \overline{\overline{x * \bar{z} * A}} * \overline{\overline{z * \bar{t} * B}} * \overline{\overline{x * z * B}} *$$

$$* \overline{\overline{\bar{y} * \bar{t}}} * \overline{\overline{\bar{x} * \bar{y} * \bar{z}}} * \overline{x * y * z}$$

Astfel, în acest moment fiecare termen din funcție poate fi reprezentat folosind un circuit de tip NAND și circuitul se implementează ca mai jos:



2. Realizați circuitul următoarei funcții folosind porți de tip NAND:

x	y	z	t	f
0	0	0	0	**0**
0	0	0	1	*
0	0	1	0	**A**
0	0	1	1	**1**
0	1	0	0	**A**
0	1	0	1	**1**
0	1	1	0	**B**
0	1	1	1	*
1	0	0	0	*
1	0	0	1	**A**
1	0	1	0	**1**
1	0	1	1	**B**
1	1	0	0	**0**
1	1	0	1	*
1	1	1	0	**B**
1	1	1	1	**A**

Pașii 1+2:

zt / xy	00	01	11	10
00		*	1	A
01	A	1	*	B
11		*	A	B
10	*	A	B	1

$$f_1(x,y,z,t,A,B) = \bar{z} * t * A + y * t * A + \bar{x} * y * \bar{z} * A + \bar{x} * \bar{y} * z * A$$
$$+ y * z * \bar{t} * B + x * \bar{y} * z * B$$

Pasul 3:

zt / xy	00	01	11	10
00		*	1	
01		1	*	
11		*		
10	*			1

$$f_2(x,y,z,t,A,B) = \bar{x}*t + x*\bar{y}*\bar{t}$$

Pasul 4:

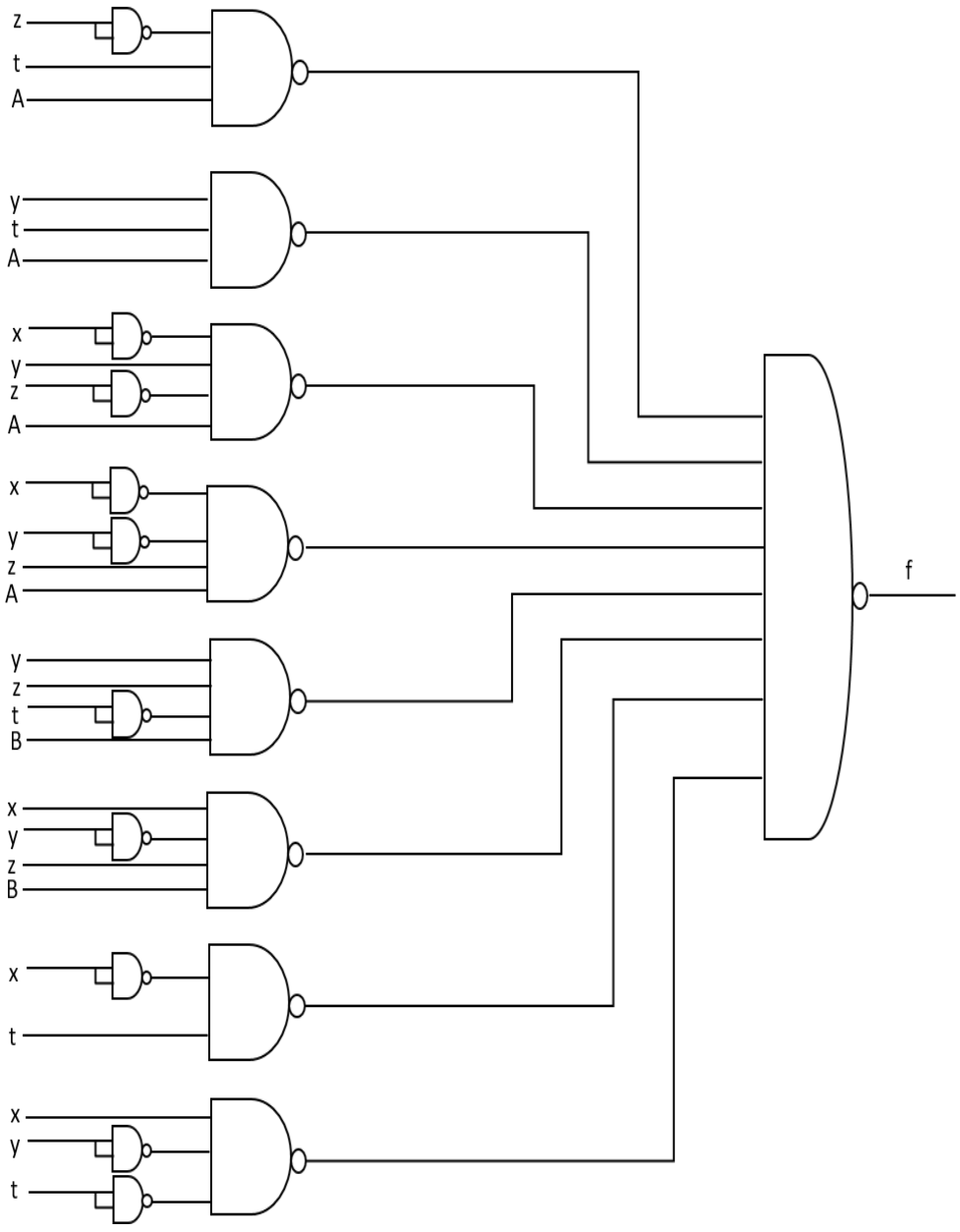
$$f(x,y,z,t,A,B) = \bar{z}*t*A + y*t*A + \bar{x}*y*\bar{z}*A + \bar{x}*\bar{y}*z*A$$
$$+ y*z*\bar{t}*B + x*\bar{y}*z*B + \bar{x}*t + x*\bar{y}*\bar{t}$$

Pentru a implementa circuitul cu porți de tip NAND, trebuie să se nege de două ori funcția și apoi să se desfacă una dintre negări folosind formulele lui De Morgan, astfel transformând suma în produs negat.

$$f(x,y,z,t,A,B) = \overline{\overline{\bar{z}*t*A + y*t*A + \bar{x}*y*\bar{z}*A +}}$$
$$\overline{\overline{+\bar{x}*\bar{y}*z*A + y*z*\bar{t}*B + x*\bar{y}*z*B + \bar{x}*t + x*\bar{y}*\bar{t}}}$$

$$f(x,y,z,t,A,B) = \overline{\overline{\bar{z}*t*A} * \overline{y*t*A} * \overline{\bar{x}*y*\bar{z}*A} *}$$
$$* \overline{\bar{x}*\bar{y}*z*A} * \overline{y*z*\bar{t}*B} * \overline{x*\bar{y}*z*B} * \overline{\bar{x}*t} * \overline{x*\bar{y}*\bar{t}}$$

Astfel, în acest moment fiecare termen din funcție poate fi reprezentat folosind un circuit de tip NAND și circuitul se implementează ca mai jos:

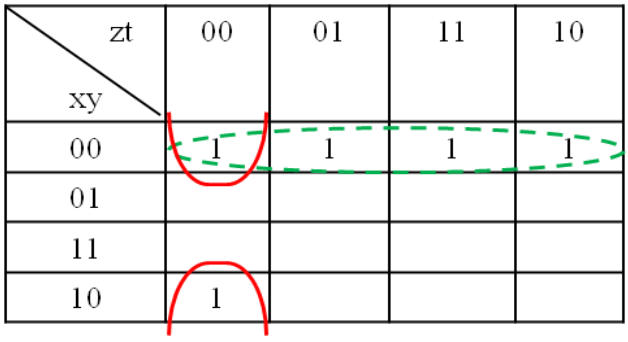

z
t
A
y
t
A
x
y
z
A
x
y
z
A
y
z
t
B
x
y
z
B
x
t
x
y
t
f

IV.7 Exemple rezolvate

1. Minimizați funcția dată prin tabelul de adevăr de mai jos:

x	y	z	t	f
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**0**
0	1	1	0	**0**
0	1	1	1	**0**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**0**

Diagrama Karnaugh este:

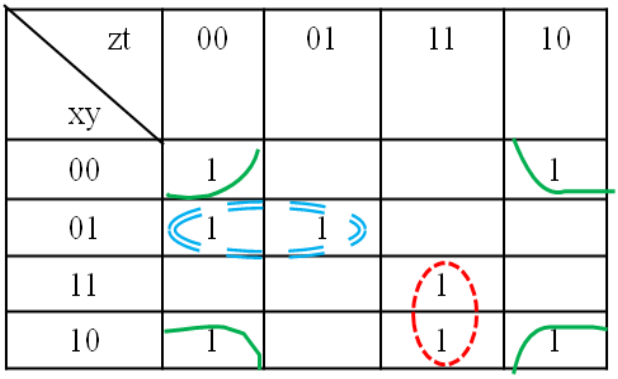


$$f(x,y,z,t) = \bar{x} * \bar{y} + \bar{y} * \bar{z} * \bar{t}$$

2. Minimizați funcția dată prin tabelul de adevăr de mai jos:

x	**y**	**z**	**t**	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**1**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**0**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**1**

Diagrama Karnaugh este:



$$f(x,y,z,t) = \bar{y} * \bar{t} + \bar{x} * y * \bar{z} + x * z * t$$

3. Minimizați funcția dată prin tabelul de adevăr de mai jos:

x	y	z	t	f
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

Diagrama Karnaugh este:

zt / xy	00	01	11	10
00			1	1
01		1	1	1
11		1	1	1
10			1	1

$f(x, y, z, t) = z + y * t$

4. Minimizați funcția dată prin tabelul de adevăr de mai jos și realizați circuitul ei echivalent:

x	y	z	f
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

Diagrama Karnaugh este:

x \ yz	00	01	11	10
0	1	0	1	1
1	0	0	0	1

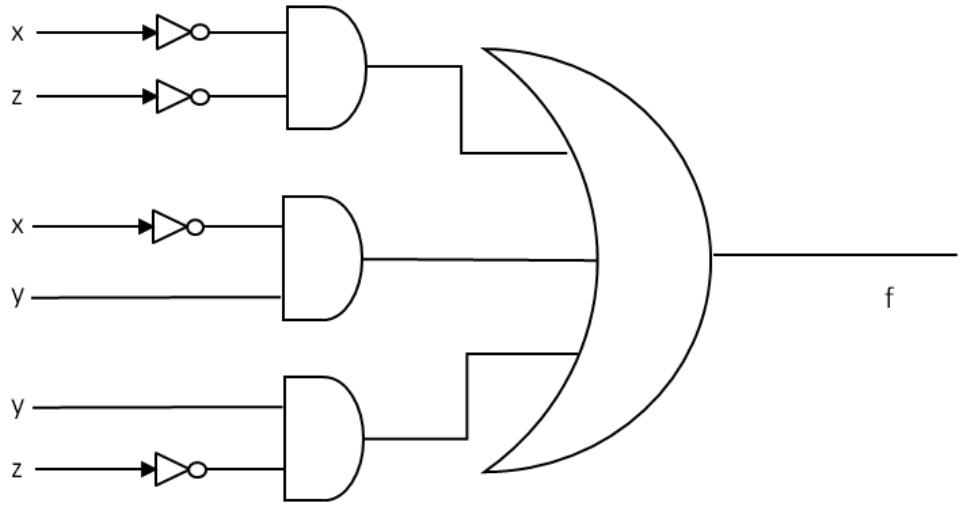
$$f(x,y,z) = \bar{x} * \bar{z} + \bar{x} * y + y * \bar{z}$$

Dacă se dorește implementarea doar cu circuite de tip NAND, trebuie să se aplice formulele lui De Morgan, obținându-se funcția:

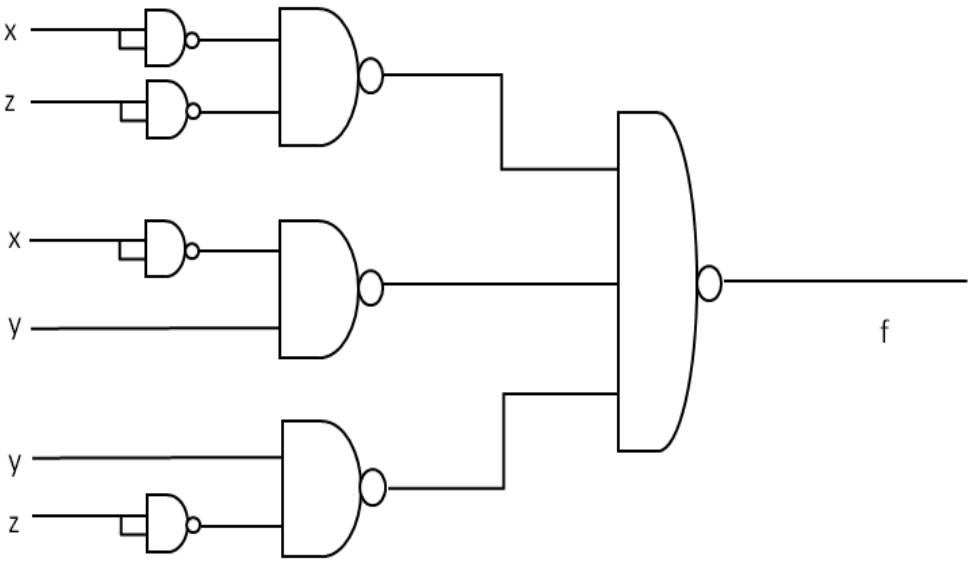
$$f(x,y,z) = \bar{x} * \bar{z} + \bar{x} * y + y * \bar{z} = \overline{\overline{\bar{x} * \bar{z} + \bar{x} * y + y * \bar{z}}}$$

$$f(x,y,z) = \overline{\overline{\bar{x} * \bar{z}} * \overline{\bar{x} * y} * \overline{y * \bar{z}}}$$

Circuitul echivalent al circuitului folosind orice tip de porți este:



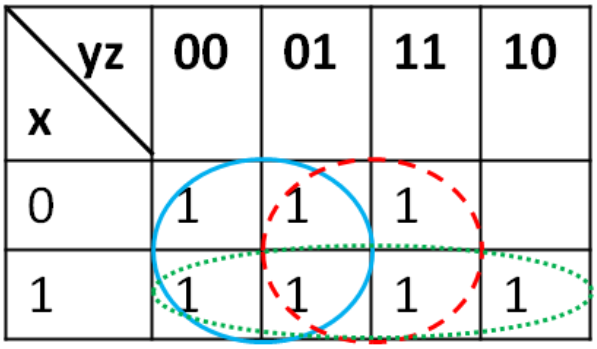
Pentru a implementa funcția f folosind doar circuite de tip NAND, circuitul devine:



5. Minimizați funcția dată prin tabelul de adevăr de mai jos folosind două metode diferite.

x	**y**	**z**	**f**
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Diagrama Karnaugh este:



$$f(x,y,z) = x + \bar{y} + z$$

Întrucât lipsește un singur produs din funcție, este mai ușor să se nege funcția, să se minimize funcția rezultată și apoi să se nege din nou pentru a ajunge la minimizarea funcției inițiale, folosind proprietatea $\bar{\bar{x}} = x$.

x \ yz	00	01	11	10
0				1
1				

$$\overline{f(x,y,z)} = \bar{x} * y * \bar{z} \rightarrow f(x,y,z) = \overline{\overline{f(x,y,z)}} = \overline{\bar{x} * y * \bar{z}} = x + \bar{y} + z$$

IV.8 Probleme propuse

1. Minimizați funcția dată prin tabelul de adevăr de mai jos și realizați circuitul ei echivalent:

x	y	z	t	f
0	0	0	0	*
0	0	0	1	**1**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	*
0	1	1	0	**1**
0	1	1	1	*
1	0	0	0	*
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	*
1	1	1	0	**1**
1	1	1	1	*

2. Minimizați funcția reprezentată de diagrama Karnaugh de mai jos și realizați circuitul ei echivalent folosind porți de tip NAND:

zt / xy	00	01	11	10
00	1	1		
01	1		*	*
11	1		*	*
10	1	1		

3. Minimizați funcția dată prin tabelul de adevăr de mai jos și realizați circuitul ei echivalent:

x	**y**	**z**	**t**	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**A**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**A**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**A**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**A**

4. Minimizați funcția reprezentată de diagrama Karnaugh de mai jos și realizați circuitul ei echivalent folosind porți de tip NAND:

yz / x	00	01	11	10
0	A	A	1	
1	1	1	1	$\overline{A}$

5. Minimizați funcția reprezentată de diagrama Karnaugh de mai jos și realizați circuitul ei echivalent folosind porți de tip NAND:

zt / xy	00	01	11	10
00	A			A
01	1		1	
11	$\overline{A}$	1	1	$\overline{A}$
10		1	A	1

6. Minimizați funcția reprezentată de diagrama Karnaugh de mai jos și realizați circuitul ei echivalent folosind porți de tip NAND:

zt / xy	00	01	11	10
00	1		1	1
01	1	A	B	*
11		1	A	1
10	A	B	*	

7. Pentru funcția dată prin tabelul de adevăr următor, introduceți variabila y în coloana rezultat, după care minimizați funcția rezultată folosind diagrame Karnaugh și realizați circuitul ei echivalent folosind porți de tip NAND:

x	**y**	**z**	**t**	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**0**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**0**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	**1**

8 Pentru funcția dată prin tabelul de adevăr de mai sus, introduceți variabila z în coloana rezultat, după care minimizați funcția rezultată folosind diagrame Karnaugh și realizați circuitul ei echivalent folosind porți de tip NAND. Seamănă rezultatul obținut și implementarea realizată cu cele de la exercițiul anterior?

9 Pentru funcția de la exercițiul 7, introduceți variabilele y și z în coloana rezultat, după care minimizați funcția rezultată folosind diagrame Karnaugh și realizați circuitul ei echivalent folosind porți de tip NAND. Seamănă rezultatul obținut și implementarea realizată cu cele de la exercițiul anterior?

V Reprezentarea zecimală a funcțiilor. Algoritmul Quine – McCluskey

V.1 Reprezentarea zecimală a funcțiilor

Reprezentarea zecimală a unei funcții este dată de suma termenilor care apar în cadrul funcției respective, scrișI în baza 10. Pentru a construi reprezentarea zecimală a unei funcții există două posibilități:

- Fie se construiește tabelul de adevăr al funcției și apoi se identifică valoarea zecimală a combinațiilor de valori a variabilelor de intrare pentru care funcția ia valoarea 1, aceștia fiind termenii din suma reprezentării zecimale,
- Fie, se scrie funcția în FDN și apoi se identifică valorile zecimale ale termenilor ce fac parte din funcție prin înlocuirea valorilor directe cu 1 și a celor negate cu 0 și apoi transformarea numărului din binar în zecimal.

Observații:

Observația 1: numărul termenilor ce vor apărea în reprezentarea zecimală este dependent de funcție și este dat de numărul monoamelor (produselor) din FDN a funcției sau numărul de unități din tabelul de adevăr al acesteia.

Observația 2: valoarea minimă a termenilor ce pot apare în sumă este 0 și se obține atunci când funcția ia valoarea 1 pentru cazul în care toate variabilele de intrare sunt negate, iar valoarea maximă este 2^N-1, unde N este numărul de variabile de care depinde funcția, și se obține atunci când funcția ia valoarea 1 pentru cazul în care toate variabilele de intrare sunt în formă directă.

Exemplu:

Fie funcția $f(x,y,z) = \bar{y} + \bar{x} * \bar{z}$. Să se construiască reprezentarea zecimală a acestei funcții.

Metoda 1: Construim tabelul de adevăr:

x	**y**	**z**	**f**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

În continuare, pentru fiecare valoare 1 a funcției, se transformă în zecimal numărul binar obținut din valorile variabilelor de intrare. De exemplu, prima valoare pentru care f = 1 este x = 0, y = 0, z = 0. Drept urmare, se va transforma $000_{(2)}$ în $0_{(10)}$, acesta fiind primul termen din reprezentarea zecimală a funcției. Următoarea valoare pentru care f = 1 este x = 0, y = 0, z = 1. Drept urmare, cel de-al doilea termen din reprezentarea zecimală a funcției este $010_{(2)} = 1_{(10)}$. În final, reprezentarea zecimală a funcției f este: $f(x,y,z) = \sum\{0,1,2,4,5\}$.

Metoda 2: Aducem funcția la FDN:

$$f(x,y,z) = \bar{y} + \bar{x} * \bar{z} = 1 * \bar{y} * 1 + \bar{x} * 1 * \bar{z} = (x + \bar{x}) * \bar{y} * (z + \bar{z}) +$$
$$+ \bar{x} * (y + \bar{y}) * \bar{z} = x * \bar{y} * z + x * \bar{y} * \bar{z} + \bar{x} * \bar{y} * z + \bar{x} * \bar{y} * \bar{z} +$$
$$+ \bar{x} * y * \bar{z} + \bar{x} * \bar{y} * \bar{z}$$
$$f(x,y,z) = x * \bar{y} * z + x * \bar{y} * \bar{z} + \bar{x} * \bar{y} * z + \bar{x} * \bar{y} * \bar{z} + \bar{x} * y * \bar{z}$$

În continuare, înlocuim variabilele în formă directă cu 1 și pe cele în formă negată cu 0 și apoi transformăm numerele binare obținute în numere zecimale:

101	100	001	000	010
5	4	1	0	2

Astfel, funcția se scrie ca sumă a termenilor zecimali obținuți: $f(x,y,z) = \sum\{0,1,2,4,5\}$. Așa cum se poate observa, s-a ajuns la aceeași valoare, semn că cele două metode sunt echivalente.

V.2 Algoritmul de minimizare Quine–McCluskey

Algoritmul de minimizare Quine–McCluskey este folosit în general pentru minimizarea funcțiilor booleene ce depind de mai mult de 5 variabile. Pentru a minimiza o funcție folosind algoritmul Quine–McCluskey, aceasta trebuie să fie mai întâi adusă în forma zecimală. Avantajul major al acestui algoritm este faptul că poate fi automatizat prin implementarea într-un limbaj de programare.

V.2.1 DEFINIȚII

Implicant: orice monom care apare în reprezentarea unei funcții booleene în formă disjunctivă (nu neapărat FND!).

Observație: în diagramele Karnaugh, orice element este un implicant.

Implicant prim: un implicant care nu este conținut într-un alt implicant format dintr-un număr mai mic de variabile.

Forma minimală a unei funcții booleene reprezintă o sumă de implicanți primi. Totuși, este posibil ca mintermii să fie conținuți mai mult decât o singură dată în setul de implicanți primi și atunci înseamnă că nu s-a obținut o reprezentare minimală.

Implicant prim esențial: un implicant prim care acoperă un minterm neacoperit de alți implicanți primi.

V.2.2 PAȘII ALGORITMULUI

Pasul 1: Plecând de la valoarea maximă a termenilor din reprezentarea zecimală, se determină numărul de variabile ale funcției de minimizat astfel:

$$nr\_var = \lceil \log(\max(t \mid t \in \text{termeni din reprezentarea lui f})) \rceil$$

Unde $\lceil x \rceil$ reprezintă partea întreagă superioară a lui x (cel mai mic număr întreg mai mare sau egal cu x).

Pasul 2: Se aduce funcția la o formă echivalentă cu FND, pornind de la termenii din reprezentarea zecimală. Practic se face operația inversă transformării din FND în reprezentarea zecimală, dar nu se duce până la capăt. Astfel, pentru fiecare termen din reprezentarea zecimală se identifică reprezentarea binară a acestuia pe numărul de biți identificat la pasul anterior. Fiecare bit din reprezentarea binară obținută corespunde unei variabile diferite. Aceste reprezentări sunt echivalente mintermilor din FDN. În această formă, dacă valorile de 0 se înlocuiesc cu variabile negate, iar cele de 1 cu variabile în formă directă, atunci se ajunge chiar la FDN. Totuși, pentru acest algoritm, acest pas nu mai este necesar.

Pasul 3: Se efectuează o serie de comparații exhaustive ale tuturor perechilor de produse pentru a descoperi toate posibilitățile de reducere.

Pasul 3.1: Se construiește un tabel cu 3 coloane: termenul de minimizat, reprezentarea binară a lui și numărul de unități (biți „1") din reprezentarea binară. Termenii trebuie să fie grupați astfel încât termenii cu același număr de unități să fie adiacenți.

Pasul 3.2: Se compară fiecare termen din grupul cu index i cu toți cei din grupul i + 1. Dacă termenii diferă printr-un singur bit, pe poziția respectivă se trage o linie (-) și se consideră că cei 2 termeni au fost combinați. De asemenea, se bifează toți termenii care au participat la

combinări. Dacă în schimb termenii diferă prin mai mulți biți, atunci înseamnă că aceștia nu se pot combina și drept urmare se trece la următoarea comparație fără a face nimic suplimentar.

Pasul 3.3: Termenii obținuți la pasul 3.2 se vor combina în aceeași manieră până când se epuizează toate combinațiile posibile, ținând cont și de poziția liniilor din pașii anteriori (termenii nu pot fi combinați dacă nu au – pe aceeași poziție). Termenii rămași nebifați constituie setul de implicanți primi ai funcției. Unii din implicanţii primi sunt implicanți primi esențiali, și prin urmare nu trebuie să lipsească din expresia minimizată a funcției.

Pasul 3.4: Pentru identificarea implicanţilor primi esențiali se construiește *tabelul acoperirilor*. În acesta avem pe o dimensiune termenii de acoperit, iar pe cealaltă implicanţii primi obținuți. Implicanţii primi esențiali sunt cei care acoperă în mod unic un anumit termen (unici pe linia/coloana termenilor de acoperit).

Pasul 3.5: Se identifică termenii acoperiți de implicanţii primi esențiali, iar dacă rămân termeni ce nu au fost acoperiți, atunci se alege numărul minim de implicanți primi ce au număr minim de variabile (numărul maxim de linii -) și care acoperă toți termenii rămași.

Pasul 3.6: Se construiește funcția plecând de la monoamele rezultate în pasul anterior astfel: dacă pe poziția unei variabile avem linie (-), înseamnă că acea variabilă nu va mai apărea în monom (produs), dacă avem 1, atunci ea va apare în formă directă, iar dacă avem 0, atunci va apare în formă negată. Funcția va fi dată de suma monoamelor construite pe baza implicanţilor primi.

Exemple:

1. Fie funcția definită astfel: $f = \sum\{1,4,7,12,16,19,31\}$. Determinați forma minimizată a acesteia.

Pasul 1: Determinarea numărului de variabile de care depinde funcția:

$$nr\_\text{var} = \lceil \log(\max(t \mid t \in \text{termeni din reprezentare})) \rceil = \lceil \log(31) \rceil = 5$$

Pasul 2: Aducerea funcției la o formă echivalentă cu FDN: pentru fiecare termen din sumă trebuie găsită reprezentarea binară pe 5 biți, unde 5 este valoarea identificată în pasul anterior.

1 – 00001

4 – 00100

7 – 00111

12 – 01100

16 – 10000

19 – 10011

31 – 11111

Pașii 3.1 – 3.3: Se construiește tabelul cu 3 coloane:

Termenul de minimizat	**Reprezentarea binară**					**Nr de unități**
1	0	0	0	0	1	1
4	0	0	1	0	0	1
7	0	0	1	1	1	3
12	0	1	1	0	0	2
16	1	0	0	0	0	1
19	1	0	0	1	1	3
31	1	1	1	1	1	5

Apoi, tabelul se reorganizează astfel încât reprezentările cu același număr de unități să fie adiacente.

Termenul de minimizat	**Reprezentarea binară**					**Nr de unități**
1	0	0	0	0	1	1
4	0	0	1	0	0	**1 ***
16	1	0	0	0	0	1
12	0	1	1	0	0	**2 ***
7	0	0	1	1	1	3
19	1	0	0	1	1	3
31	1	1	1	1	1	5

Așa cum se poate observa, se vor combina termenii 4 cu 12 și va rezulta reprezentarea 0-100, aceasta fiind singura combinație posibilă. De asemenea, termenii care se combină se bifează pentru a ne ajuta în următorii pași.

Pasul 3.4: Se construiește tabelul acoperirilor.

1	**4**	**7**	**12**	**16**	**19**	**31**	
*							**1**
		*					**7**
				*			**16**
					*		**19**
						*	**31**
	*		*				**(4,12)**

Cum fiecare termen de acoperit este acoperit în mod unic de către un implicant prim, înseamnă că toți implicanţii primi sunt implicanţi primi esențiali și vor apărea în forma minimizată a funcției. Astfel, vom sări direct la pasul 3.6.

Pasul 3.6: Se construiește funcția pe baza implicanţilor primi esențiali descoperiți.

Implicanţii primi esențiali sunt 1 – 00001, 7 – 00111, 16 – 10000, 19 – 10011, 31 – 11111 și (4,12) – 0-100. Drept urmare:

$$f(x_5, x_4, x_3, x_2, x_1) = \overline{x_5} * \overline{x_4} * \overline{x_3} * \overline{x_2} * x_1 + \overline{x_5} * \overline{x_4} * x_3 * x_2 * x_1 +$$
$$+ x_5 * \overline{x_4} * \overline{x_3} * \overline{x_2} * \overline{x_1} + x_5 * \overline{x_4} * \overline{x_3} * x_2 * x_1 + x_5 * x_4 * x_3 * x_2 * x_1 +$$
$$+ \overline{x_5} * x_3 * \overline{x_2} * \overline{x_1}$$

2. Fie funcția definită astfel: $f = \sum\{1,3,4,6,8,9,12,14,19,23,27,29,30\}$. Determinați forma minimizată a acesteia.

Pasul 1: Determinarea numărului de variabile de care depinde funcția:

$$nr\_\text{var} = \lceil \log(\max(t \mid t \in \text{termeni din reprezentare})) \rceil = \lceil \log(30) \rceil = 5$$

Pasul 2: Aducerea funcției la o formă echivalentă cu FDN: pentru fiecare termen din sumă trebuie găsită reprezentarea binară pe 5 biți, unde 5 este valoarea identificată în pasul anterior.

1 – 00001

3 – 00011

4 – 00100

6 – 00110

8 – 01000

9 – 01001

12 – 01100

14 – 01110

19 – 10011

23 – 10111

27 – 11011

29 – 11101

30 – 11110

Pașii 3.1 – 3.3: Se construiește tabelul cu 3 coloane:

Termenul de minimizat	Reprezentarea binară					Nr de unități	
1	0	0	0	0	1	**1**	*
4	0	0	1	0	0	**1**	*
8	0	1	0	0	0	**1**	*
3	0	0	0	1	1	**2**	*
6	0	0	1	1	0	**2**	*
9	0	1	0	0	1	**2**	*
12	0	1	1	0	0	**2**	*
14	0	1	1	1	0	**3**	*
9	1	0	0	1	1	**3**	*
23	1	0	1	1	1	**4**	*
27	1	1	0	1	1	**4**	*
29	1	1	1	0	1	4	
30	1	1	1	1	0	**4**	*

În urma combinării termenilor, se obține tabelul de mai jos, iar în primul tabel bifăm termenii care se combină:

Termenul de minimizat	Reprezentarea binară					Nr de unități
(1, 3)	0	0	0	-	1	1
(4, 6)	0	0	1	-	0	**1** *
(8, 12)	0	1	-	0	0	1
(1, 9)	0	-	0	0	1	1
(4, 12)	0	-	1	0	0	**1** *
(8, 9)	0	1	0	0	-	1
(3, 19)	-	0	0	1	1	2
(6, 14)	0	-	1	1	0	**2** *
(12, 14)	0	1	1	-	0	**2** *
(14, 30)	-	1	1	1	0	3
(19, 23)	1	0	-	1	1	3
(19, 27)	1	-	0	1	1	3

Combinând termenii în continuare (și bifându-i pe cei care se combină) obținem termenii:

Termenul de minimizat	Reprezentarea binară					Nr de unități
(4, 6, 12, 14)	0	-	1	-	0	1
(4, 12, 6, 14)	0	-	1	-	0	1

Pasul 3.4: Se construiește tabelul acoperirilor și se identifică implicanţii primi esențiali.

1	3	4	6	8	9	12	14	19	23	27	29	30	
											*		**29**
*	*												**(1,3)**
*					*								**(1,9)**
				*	*								**(8,9)**
				*		*							**(8,12)**
	*							*					**(3,19)**
							*					*	**(14,30)**
								*	*				**(19,23)**
								*		*			**(19,27)**
		*	*			*	*						**(4,6,12,14)**

În cazul de față, coloana 4 este acoperită în mod unic de implicantul prim (4, 6, 12, 14), acesta acoperind în același timp și coloanele 6, 12 și 14. În continuare, coloana 23 este acoperită în mod unic de implicantul prim (19, 23), care acoperă și coloana 19. Mai departe, toți termenii de acoperit mai mari (27, 29 și 30) au o singură * pe coloana lor, deci sunt acoperiți de implicanți primi esențiali: (19, 27), 29 și (14, 30).

Pasul 3.5: Se identifică termenii acoperiți de implicanţii primi esențiali și se remarcă faptul că au rămas 4 termeni neacoperiți: 1,3,8 și 9. Pentru acoperirea acestor termeni se caută numărul minim de implicanți primi care să îi acopere. Astfel, se remarcă faptul că o combinație de doi implicanți primi ((1, 3) și (8, 9)) acoperă toți cei 4 termeni, ne-existând o altă combinație mai convenabilă de implicanți primi care să îi acopere pe aceștia. Drept urmare, implicanţilor primi esențiali descoperiți în pasul anterior li se adaugă acești doi implicanți primi pentru a definitiva valoarea lui f.

Pasul 3.6: Se construiește funcția pe baza implicanţilor primi ce vor fi utilizați.

Implicanţii primi ce vor fi utilizați la minimizarea funcției sunt: (4, 6, 12, 14) – 0-1-0, (19, 23) – 10-11, (19, 27) – 1-011, 29 – 11101, (14, 30) – -1110, (1, 3) – 000-1 și (8, 9) – 0100-. Drept urmare, valoarea finală a funcției va fi:

$$f(x_5, x_4, x_3, x_2, x_1) = x_5 * x_4 * x_3 * \overline{x_2} * x_1 + \overline{x_5} * \overline{x_4} * \overline{x_3} * x_1 +$$
$$+ \overline{x_5} * x_4 * \overline{x_3} * \overline{x_2} + x_4 * x_3 * x_2 * \overline{x_1} + x_5 * \overline{x_4} * x_2 * x_1 +$$
$$+ x_5 * \overline{x_3} * x_2 * x_1 + \overline{x_5} * x_3 * \overline{x_1}$$

V.3 Probleme propuse

1. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{0,1,2,4,6,8,10,12,13,14,15\}$

2. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{2,4,6,8,9,10,12,13,15\}$

3. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{0,1,3,4,6,9,12,15,18,19,21,24,26,28,30\}$

4. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{1,3,5,6,6,12,16,19,21,22,24,37,28\}$

5. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{2,6,8,9,10,12,16,18,20,21,22,24,26,29,30,31\}$

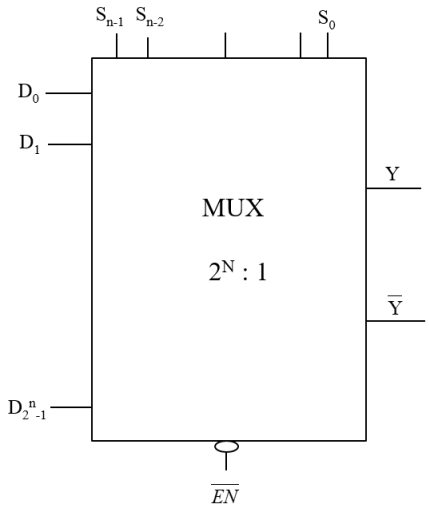
6. Minimizați funcția dată prin următoarea reprezentare zecimală: $f = \sum\{1,3,5,6,8,12,16,19,21,22,24,27,28\}$

VI Alte circuite combinaționale. Multiplexoare și Demultiplexoare

La circuitele combinaționale, ieșirea la un moment dat depinde doar de intrările de la momentul respectiv de timp. Până în prezent, din această clasă de circuite au fost prezentate porțile, dar mai există și alte categorii de circuite ce fac parte din această clasă. Astfel de exemple sunt multiplexoarele și demultiplexoarele ce vor fi prezentate în acest capitol.

VI.1 Multiplexoare (MUX)

Multiplexoarele (MUX) sunt circuite combinaționale cu n intrări de selecție (S_i), 2^n intrări de date (D_i), o intrare de enable ($\overline{EN}$) și o ieșire (Y) și eventual ieșirea negată ($\overline{Y}$).



Rolul multiplexorului este de a comuta la ieșire una dintre valorile primite pe intrările de date. Decizia legată de ce valoare se transmite la ieșire se ia în funcție de intrările de selecție. Astfel, ieșirea $Y = D_k$, unde k este numărul în baza 10 obținut din intrările respective ($k = (S_{n-1}S_{n-2}...S_0)_{10}$).

Observații:

Observația 1: Pentru a se implementa o funcție cu multiplexoare, acea funcție nu se mai minimizează în prealabil!

Observația 2: Dacă ordinea în care sunt aplicate intrările de selecție coincide cu ordinea variabilelor din tabelul de adevăr al funcției, atunci completarea intrărilor MUX-ului este trivială, tot ce trebuie făcut fiind să se copieze în ordine valorile funcției pe intrările de date ale MUX-ului.

Observația 3: Dacă ordinea în care sunt aplicate intrările de selecție se schimbă, atunci automat se vor schimba și valorile de la intrările de date, astfel încât să se respecte valorile pe care funcția trebuie să le aibă la ieșire. Acest lucru se poate face în două feluri:

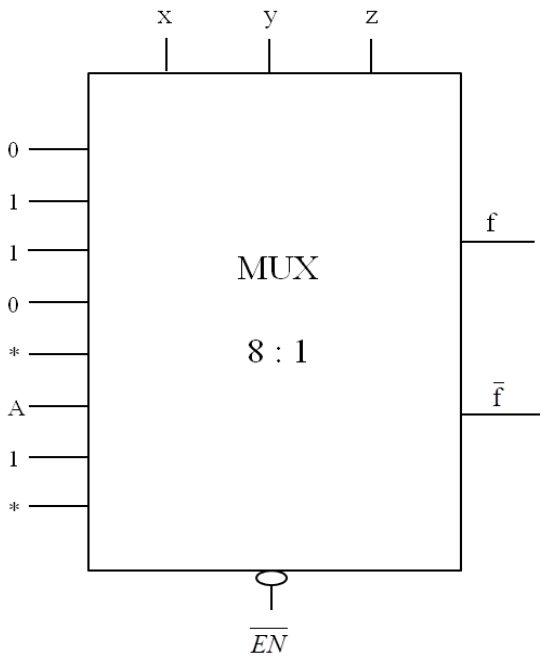
- Fie se rescrie tabelul de adevăr având variabilele de care depinde funcția puse în ordinea în care sunt impuse variabilele de selecție în MUX, după care se copiază valorile funcției în ordine pe intrările de date ale MUX-ului,
- Fie, mai rapid, se identifică în tabelul de adevăr al funcției valoarea pe care aceasta trebuie să o ia pentru fiecare combinație de valori ale variabilelor de selecție din MUX și această valoare se va trece pe intrarea corespunzătoare a acestuia.

Exemple:

1. Implementați funcția dată prin tabelul de adevăr de mai jos utilizând MUX 8:1, având ca variabile de selecție combinația x, y, z:

x	y	z	f
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	*****
1	0	1	**A**
1	1	0	**1**
1	1	1	*****

$Y = D_k$, unde $k = (xyz)_{10}$. Drept urmare, implementarea funcției utilizând MUX 8:1 este:



2. Implementați funcția de mai sus utilizând MUX 8:1, având ca variabile de selecție combinația x, z, y:

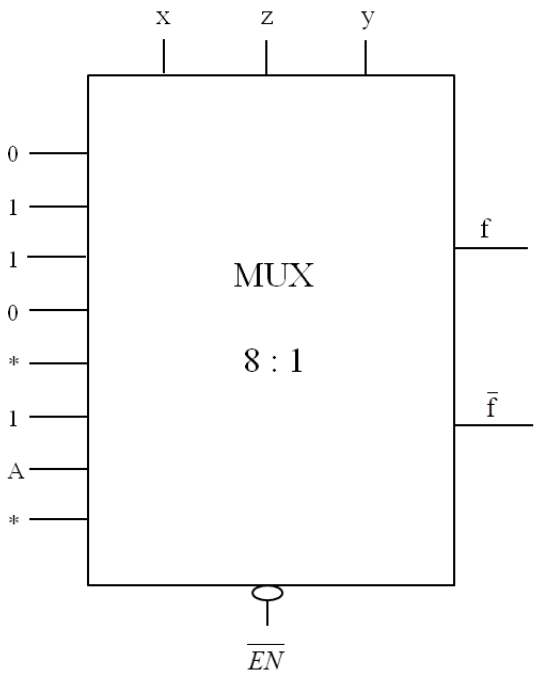
Metoda 1: Rescrierea tabelului de adevăr și apoi copierea în ordine a valorilor pe intrările MUX-ului:

x	y	z	f
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	*****
1	0	1	**A**
1	1	0	**1**
1	1	1	*****

→

x	z	y	f
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	*****
1	0	1	**1**
1	1	0	**A**
1	1	1	*****

Astfel, implementarea funcției utilizând MUX 8:1 și variabilele de selecție x, z, y va fi:

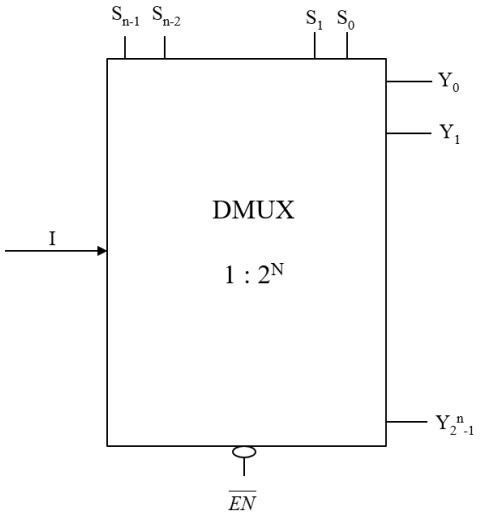


Metoda 2: Identificarea valorii funcției pe baza tabelului de adevăr și copierea acestei valori pe intrarea corespunzătoare din MUX. De exemplu, pentru combinația 101 a variabilelor de selecție ale MUX-ului, linia corespunzătoare din tabelul de adevăr al funcției este x = 1, y = 1, z = 0, iar valoarea lui f pentru această combinație este 1. Drept urmare, pe intrarea de date cu numărul $k = (xzy)_{10} = (101)_{10} = 5$ se va pune valoarea 1.

Atenție la faptul că indexarea se face de la 0, deci această intrare este de fapt a 6-a intrare în MUX. Utilizând această metodă se obține același rezultat ca în cazul metodei anterioare, dar se câștigă timp deoarece nu mai este nevoie să se rescrie tabelul, ci doar să se identifice corect combinațiile corespunzătoare fiecărei intrări de date din MUX.

VI.2 Demultiplexoare (DMUX)

Demultiplexoarele sunt circuite complementare multiplexoarelor care permit transmiterea datelor de pe o intrare de date comună pe una din ieșirile selectate. Un demultiplexor are n intrări de selecție (S_i), o intrare de date (I), o intrare de enable ($\overline{EN}$) și 2^n ieșiri (Y_i).



În general, DMUX primesc valoarea 1 pe intrare, această valoare fiind comutată pe una dintre ieșiri, în funcție de valorile intrărilor de selecție: $Y_k = I$, unde $k = (S_{n-1}\ldots S_0)_{10}$.

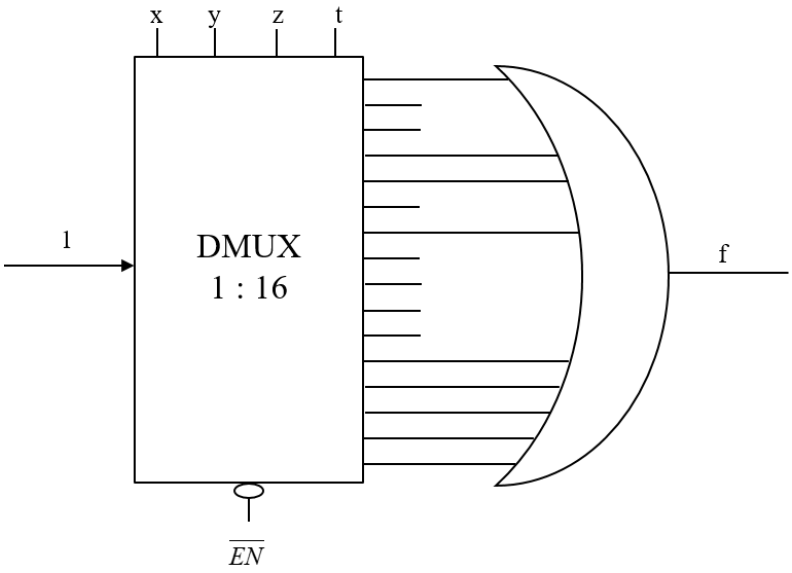
Pentru a implementa o funcție folosind un DMUX, mai întâi se scrie acea funcție în FDN. Apoi se folosește DMUX, având valoarea 1 pe intrare, iar ieșirile corespunzătoare termenilor ce apar în suma dată de reprezentarea FDN a funcției sunt legate la un circuit de tip SAU. Ieșirea circuitului SAU reprezintă chiar funcția de implementat.

Observație: Dacă se dorește implementarea funcției folosind DMUX de dimensiuni mai mici, și implicit un număr mai mic de variabile de selecție, atunci se realizează tabelul de adevăr al funcției în care se trec în coloana rezultat variabilele ce nu se doresc a fi variabile de selecție și apoi implementarea se face ca mai sus. Pentru realizarea valorilor funcției ce depind de variabile, între DMUX și circuitul de tip SAU se intercalează porți de tip ȘI în care intră variabila respectivă și ieșirea corespunzătoare ei.

Exemple:

1. Implementați funcția de mai jos utilizând DMUX 16:1.

x	y	z	t	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

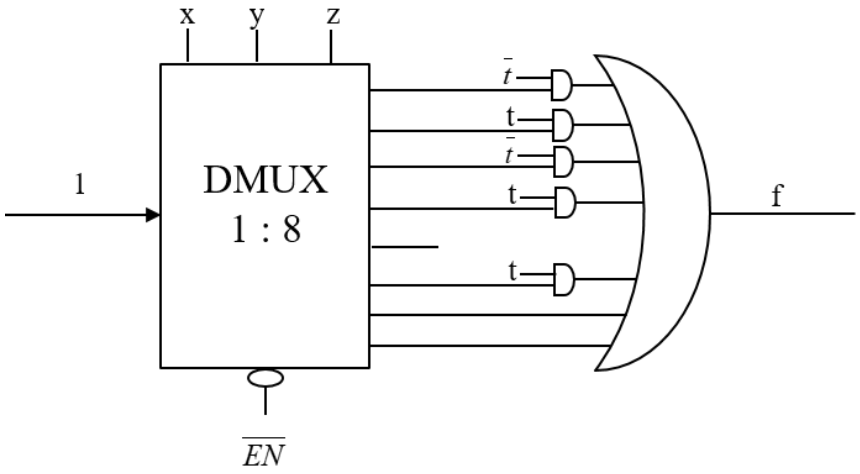


2. Implementați funcția de mai sus utilizând DMUX 8:1.

Pentru a putea implementa funcția folosind DMUX 8:1 trebuie ca, în prealabil, să construim tabelul său de adevăr în care să păstrăm doar 3 variabile de selecție. Cel mai simplu este să eliminăm ultima variabilă întrucât, în felul acesta, liniile din tabelul inițial se vor combina pe rând, două câte două, pentru a genera valorile din noul tabel.

x	**z**	**y**	**f**
0	0	0	$\bar{t}$
0	0	1	**t**
0	1	0	$\bar{t}$
0	1	1	**t**
1	0	0	**0**
1	0	1	**t**
1	1	0	**1**
1	1	1	**1**

Având noul tabel de adevăr, implementarea cu DMUX 8:1 este:



VI.3 Legarea MUX în cascadă

Pentru a putea folosi și în cazul multiplexoarelor elemente de dimensiuni mai mici (așa cum facem în cazul demultiplexoarelor) se pot aplica două metode diferite: legarea în cascadă a MUX și multiplexarea intrărilor. În cele ce urmează se va detalia prima variantă, urmând ca în următorul subcapitol să se discute și cea de-a doua alternativă.

La legarea în cascadă a multiplexoarelor practic se descompune un MUX mai mare în mai multe MUX-uri mai mici. În final, numărul total de intrări în MUX-urile legate în cascadă coincide cu numărul de intrări din MUX-ul mare și cu numărul de linii din tabelul de adevăr. Cum dimensiunea MUX-ului impune numărul de intrări în MUX, precum și numărul de variabile de selecție, înseamnă că trebuie să se identifice de la bun început de câte MUX-uri este nevoie pentru implementare, după care să se decidă cum se leagă acele MUX-uri pentru a obține funcția respectivă.

Având o funcție ce depinde de n variabile de selecție, aceasta poate fi implementată printr-un MUX 2^n:1 sau prin mai multe MUX-uri mai mici. În funcție de ce dispozitive sunt disponibile, se poate folosi, de exemplu, o combinație de două MUX 2^{n-1}:1, în care să se folosească n-1 din variabilele de selecție, urmând ca ultima variabilă de selecție să fie folosită pentru a decide care dintre ieșirile celor două MUX-uri va fi folosită pentru a stabili valoarea finală a funcției. Cu alte cuvinte, ieșirile celor două MUX 2^{n-1}:1 vor fi introduse într-un MUX 2:1 ce va avea ca variabilă de selecție acea variabilă care nu a fost folosită în pasul anterior. Cele două MUX 2^{n-1}:1 vor avea în total $2^{n-1} + 2^{n-1} = 2 * 2^{n-1} = 2^n$ intrări și vor putea acoperi toate valorile pe care trebuie să le aibă la ieșire funcția de implementat.

Dacă se dorește să se lucreze cu MUX-uri de dimensiuni mai mici, se pot folosi 4 MUX de dimensiune MUX 2^{n-2}:1. Acestea vor folosi n-2 intrări de selecție și vor avea $4 * 2^{n-2} = 2^n$ intrări, acoperind toate valorile de ieșire ale funcției. În final, ieșirile celor 4 MUX 2^{n-2}:1 vor fi introduse într-un nou MUX 4:1 având ca variabile de selecție variabilele rămase nefolosite în MUX-urile anterioare. Acest ultim MUX decide care dintre valorile furnizate de MUX-urile anterioare va fi folosită pentru a stabili valoarea finală a funcției.

Astfel, se creează două niveluri ale circuitului: primul nivel este cel pe care sunt MUX-urile inițiale, în care avem ca intrări ieșirile pe care trebuie să le furnizeze funcția, iar cel de-al doilea este nivelul pe care se stabilește care dintre MUX-uri va fi folosit pentru a decide valoarea finală a funcției. O astfel de structură se poate întinde pe mai multe niveluri, în funcție de dimensiunea

MUX-urilor folosite pe fiecare nivel, astfel obținându-se o structură ce poartă denumirea de **legare în cascadă**, întrucât ieșirile nivelului anterior sunt folosite ca intrări în nivelul curent.

Observații:

Observația 1: Dacă avem mai multe MUX-uri pe un nivel, atunci toate aceste MUX-uri se spune că sunt comandate simultan, ceea ce înseamnă că li se aplică tuturor exact aceleași variabile de selecție, în aceeași ordine și la același moment de timp.

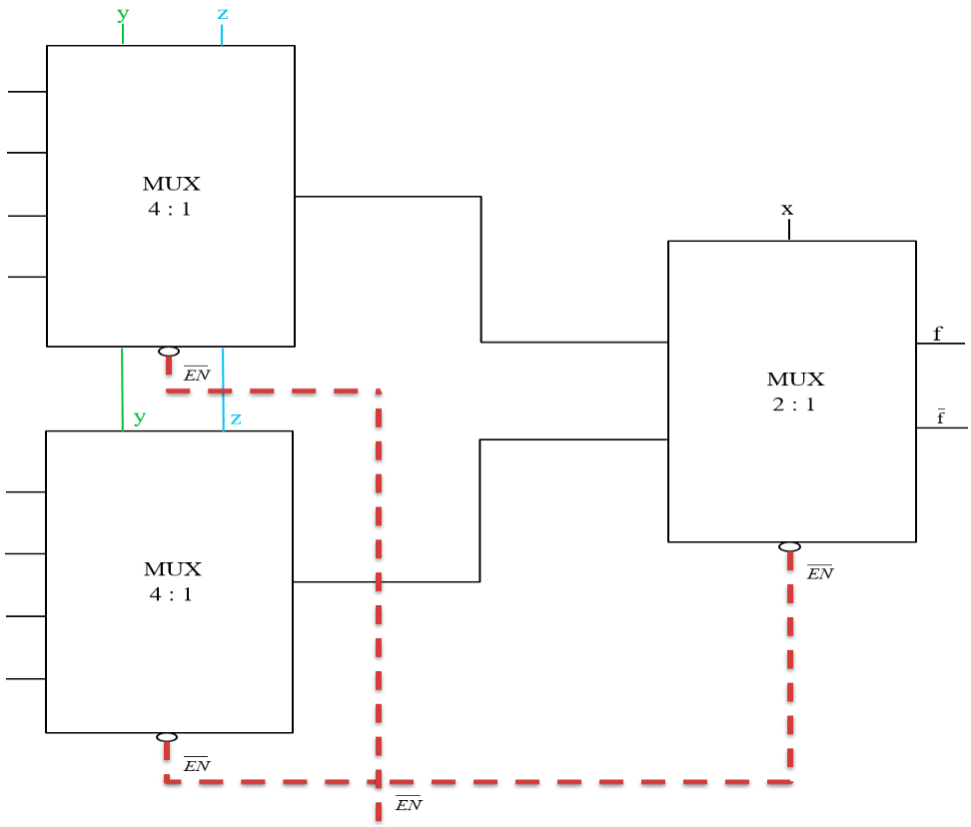
Observația 2: Variabilele de selecție folosite pe diferitele niveluri ale unui circuit cu MUX-uri legate în cascadă trebuie să formeze mulțimi disjuncte și să acopere toate variabilele de care depinde funcția de implementat (fiecare dintre variabile trebuie să fie folosită o singură dată pe unul dintre nivele, dar numai pe unul singur). Alegerea variabilelor de selecție ce sunt folosite pe fiecare nivel în parte, precum și ordinea în care sunt acestea folosite rămâne la latitudinea implementatorului.

Exemple:

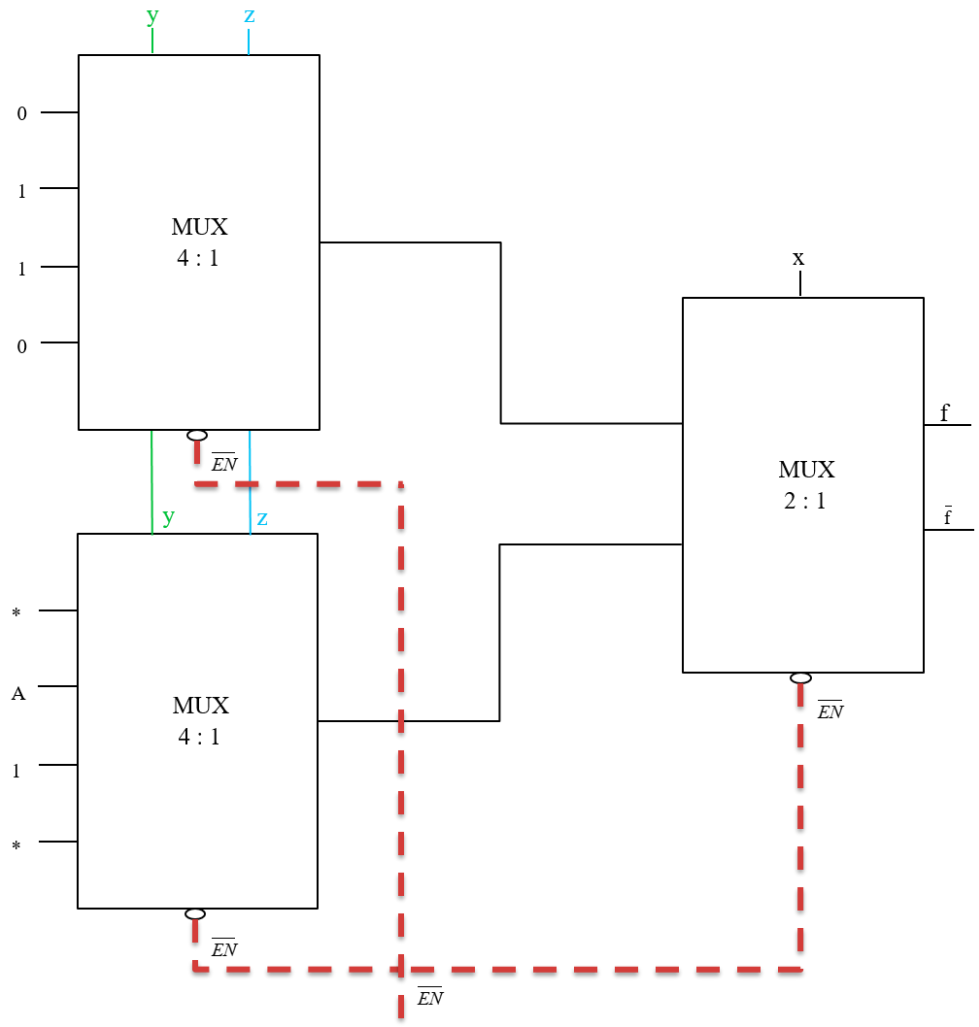
1. Implementați funcția de mai jos folosind două MUX-uri 4:1 și un MUX 2:1.

x	y	z	f
0	0	0	**0**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	*****
1	0	1	**A**
1	1	0	**1**
1	1	1	*****

Pentru această implementare, vom folosi ca variabile de selecție pe primul nivel variabilele mai puțin semnificative din tabel (y și z), iar pe cel de-al doilea vom folosi variabila cea mai semnificativă (x).



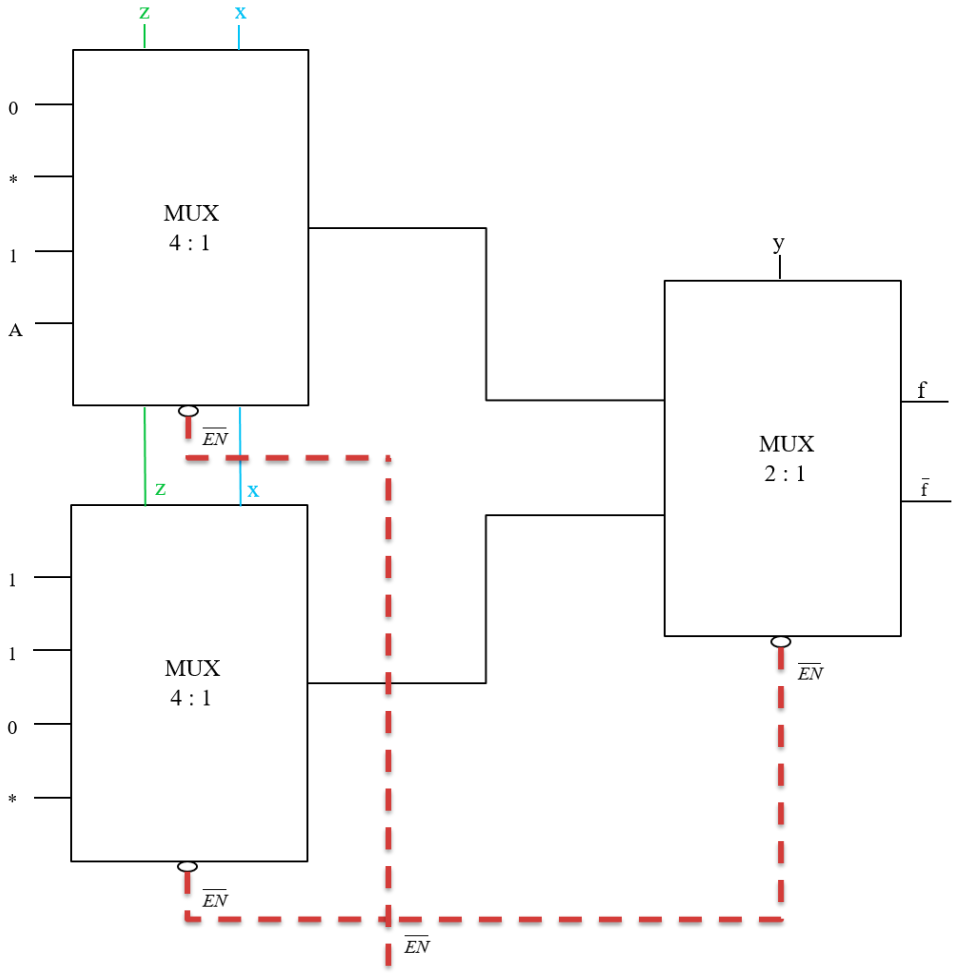
Cu această alegere, primul MUX 4:1 corespunde valorilor funcției pentru care x = 0, iar cel de-al doilea corespunde valorilor funcției pentru care x = 1. Mai mult, datorită faptului că pe primul nivel variabilele de selecție sunt folosite în aceeași ordine ca în tabelul de adevăr, valorile de la ieșirea funcției se vor copia în aceeași ordine pe intrările MUX-urilor 4:1.



2. Implementați funcția de mai sus folosind două MUX-uri 4:1 și un MUX 2:1, având ca variabile de selecție z și x pe primul nivel (în această ordine), respectiv y pe al doilea.

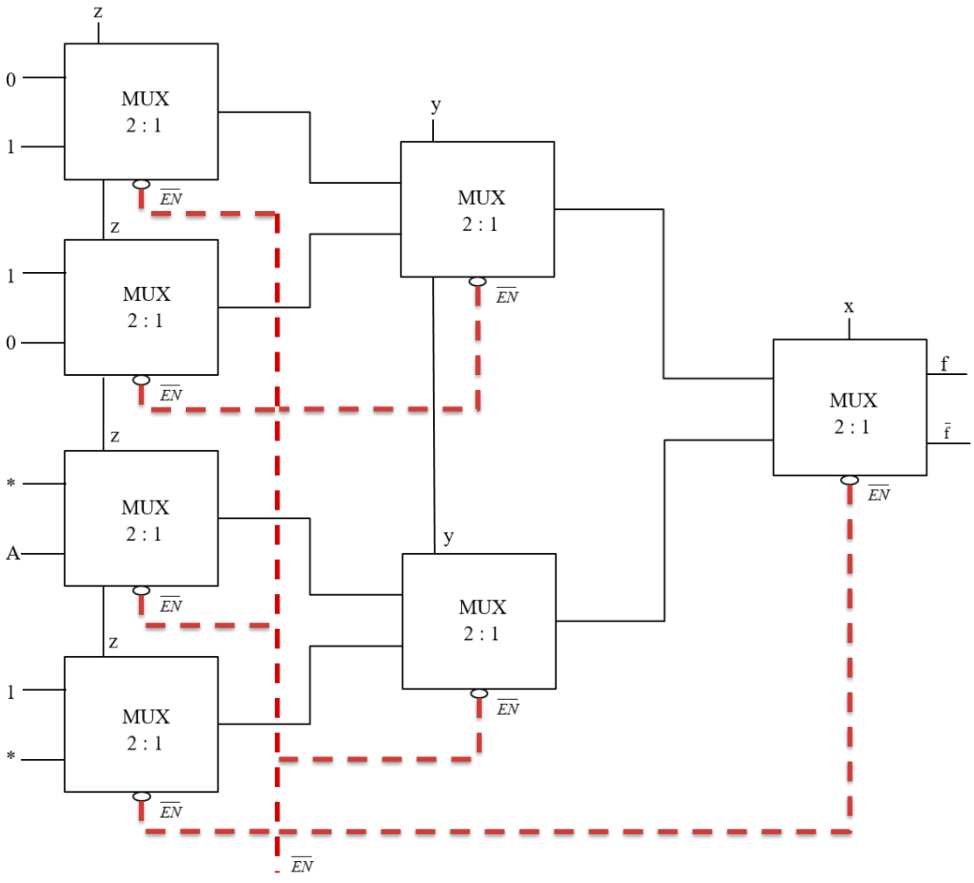
Pentru a rezolva acest exercițiu, se pot folosi cele două tehnici prezentate la MUX: fie se rescrie tabelul folosind variabilele din tabelul de adevăr în ordinea în care sunt impuse ca variabile de selecție (y, z, x – variabilele de pe nivelurile superioare sunt mai semnificative) și apoi se trec în ordine valorile pe intrările MUX-urilor; fie pentru fiecare combinație de valori ale variabilelor de

selecție se identifică direct în tabelul de adevăr valoarea pe care trebuie să o ia funcția și se trece acea valoare pe intrarea corespunzătoare a MUX-ului. Mai jos este prezentat rezultatul obținut direct folosind cea de-a doua metodă, lăsând cititorului posibilitatea de verificare a corectitudinii folosind prima metodă.



3. Implementați aceeași funcție utilizând numai MUX-uri 2:1 (în total vor fi 7 MUX 2:1).

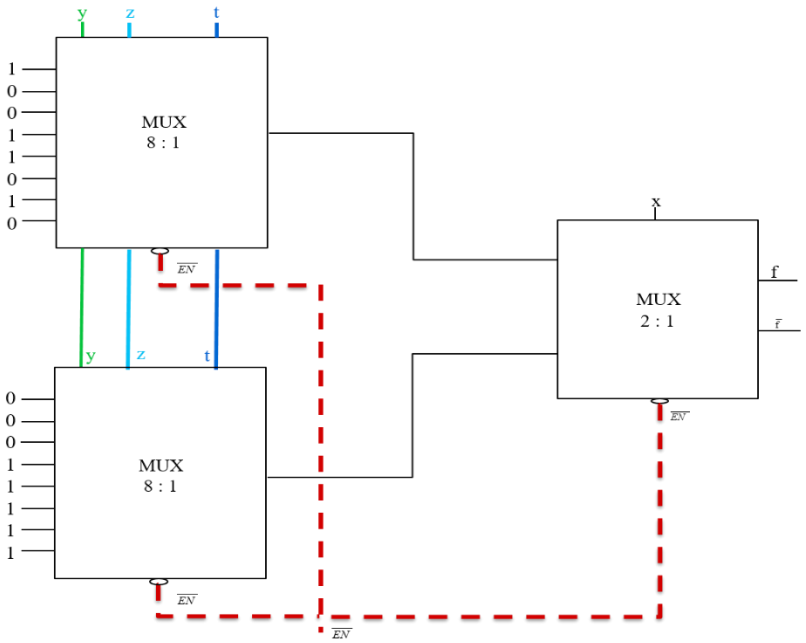
Dacă se folosesc variabilele de selecție pentru MUX-uri în ordinea importanței lor (pe primul nivel, z – cea mai puțin importantă, apoi y și în fine x pe ultimul nivel), atunci valorile de intrare ale MUX-urilor sunt exact ieșirile funcției copiate în aceeași ordine. Astfel, implementarea este:



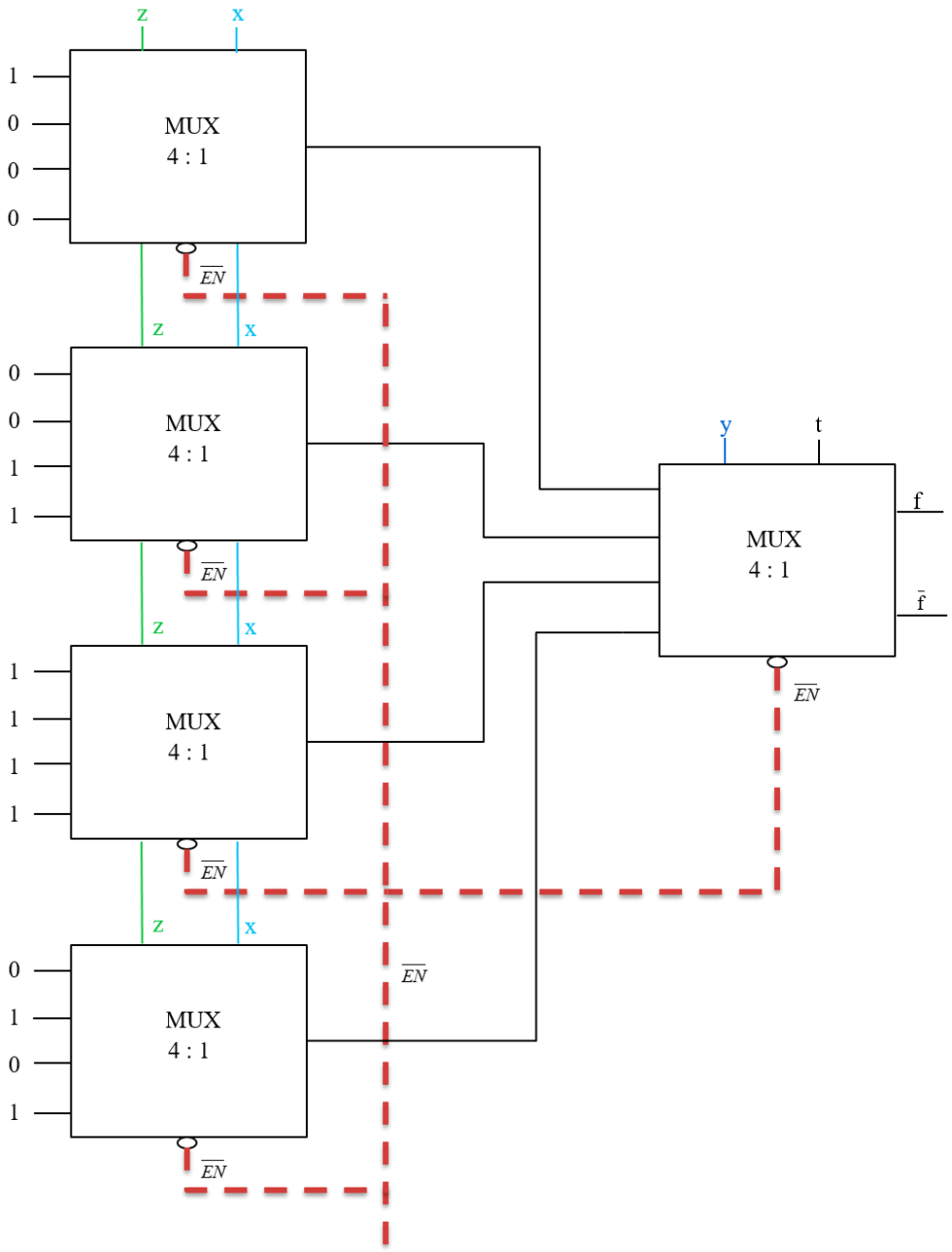
Dacă, în schimb, ordinea variabilelor de selecție pentru MUX-uri este diferită, structura circuitului rămâne aceeași și tot ce se va schimba sunt valorile intrărilor în MUX-uri (și ordinea variabilelor de selecție, desigur).

4. Implementați funcția de mai jos folosind MUX-uri 8:1.

x	y	z	t	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**



5. Implementați funcția anterioară folosind MUX-uri 4:1 și având variabilele z și x pe primul nivel, respectiv y și t pe al doilea.



VI.4 Multiplexarea intrărilor

Așa cum s-a menționat anterior, pentru a putea implementa o funcție cu MUX-uri de dimensiuni mai mici se pot aplica două metode diferite: legarea în cascadă a MUX-urilor și multiplexarea intrărilor. În primul caz, se folosesc mai multe MUX-uri de dimensiuni mai mici, în timp ce în al doilea caz se folosește un singur MUX de dimensiuni mai mici, împreună cu porți logice legate înaintea acestuia pentru a permite totuși implementarea funcției.

Astfel, pentru această metodă se folosește o tehnică similară celei prezentate în subcapitolul IV.4 , unde s-a arătat cum se poate face trecerea variabilelor în coloana rezultat, deoarece și aici, folosind un MUX de dimensiuni mai mici, vor trebui să se combine mai multe valori ale funcției pe o singură intrare a MUX-ului. Așa cum s-a putut observa și acolo, atunci când funcțiile sunt complet specificate și nu se renunță la mai mult de o variabilă, lucrurile se pot face ușor pornind de la tabelul de adevăr al funcției. Totuși, când cele două condiții de mai sus nu sunt îndeplinite simultan, trebuie să se apeleze la o altă soluție, aceasta fiind bazată tot pe folosirea diagramelor Karnaugh.

Drept urmare, pentru a implementa o funcție cu un MUX ce acceptă un număr de intrări mai mic decât numărul total de valori pe care trebuie să le aibă funcția la ieșire se procedează în felul următor:

- Se construiește diagrama Karnaugh pornind de la valorile funcției;
- Se identifică variabilele ce sunt folosite ca variabile de selecție în MUX;
- Se identifică zonele acoperite de fiecare combinație de valori ale acestor variabile (ce sunt variabile de selecție în MUX);
- Respectivele zone sunt extrase și minimizate separat, ignorând variabilele folosite ca variabile de selecție în MUX (se extrage zona corespunzătoare într-o nouă diagramă Karnaugh care nu va mai depinde de acele variabile);
- Rezultatele minimizărilor sunt trecute pe intrările corespunzătoare fiecărei combinații de valori a variabilelor de selecție din MUX.

Observații:

Observația 1: La extragerea zonei corespunzătoare unei anumite combinații a variabilelor de selecție din MUX trebuie să se țină cont de valorile celorlalte variabile rămase în diagrama Karnaugh pentru că este posibil ca ordinea căsuțelor să nu mai fie cea standard (întâi căsuțele corespunzătoare valorii 0 a variabilei și apoi cele corespunzătoare valorii 1 a acesteia).

Observația 2: Toate regulile menționate în capitolul IV rămân valide la minimizarea zonelor extrase – acestea sunt, de fapt, noi diagrame Karnaugh de dimensiuni mai mici. Atenție la zonele ce conțin * (sunt incomplet specificate), dar mai ales la cele ce conțin variabile și care trebuie minimizate în 2 pași.

Exemple:

1. Implementați funcția de mai jos folosind un singur MUX 8:1 (punând o variabilă în coloana rezultat).

x	y	z	t	**f**
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**1**
0	1	0	0	**1**
0	1	0	1	**0**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**1**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

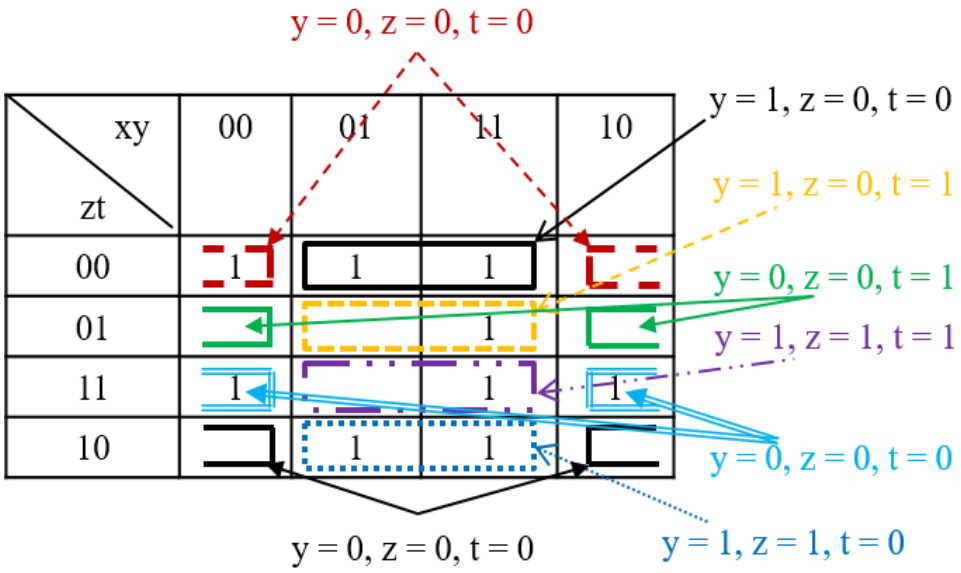
Funcția dată mai sus este aceeași funcție folosită în exemplul 4 din subcapitolul anterior (VI.3). Astfel, se pot observa cele două metode diferite de a implementa aceeași funcție folosind MUX de dimensiune mai mici.

Pasul 1: Realizarea diagramei Karnaugh:

xy / zt	00	01	11	10
00	1	1	1	
01			1	
11	1		1	1
10		1	1	

Pasul 2: Identificarea variabilelor ce sunt folosite ca variabile de selecție în MUX → cum nu au fost impuse variabilele ce trebuie folosite ca variabile de selecție în MUX, se vor alege variabilele y, z și t.

Pasul 3: Identificarea zonelor acoperite de fiecare combinație de valori ale acestor variabile. Pentru rezolvarea acestui pas, se vor considera toate combinațiile de valori (0 și 1) ale variabilelor y, z și t care au fost alese ca variabile de selecție pentru MUX. Astfel, prima combinație (y = 0, z = 0, t = 0) corespunde căsuțelor aflate pe prima linie, prima și ultima coloană. Cea de-a doua combinație (y = 0, z = 0, t = 1) corespunde căsuțelor aflate pe cea de-a doua linie, prima și ultima coloană etc.



Pasul 4: În continuare, fiecare astfel de zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Astfel pentru prima combinație, se va obține o diagramă Karnaugh având 2 pătrățele, ce depinde doar de variabila x, iar în urma minimizării se obține valoarea $\bar{x}$:

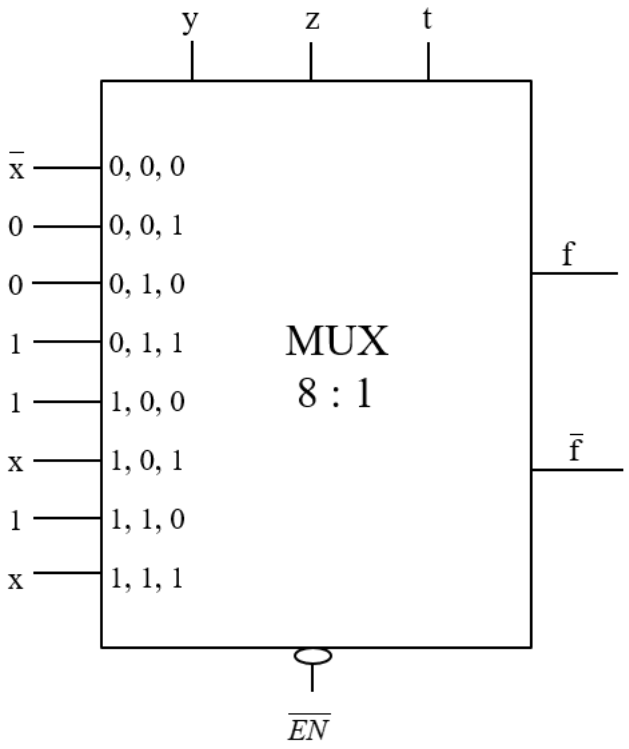
X	**0**	**1**
	1	0

Similar, pentru a doua zonă, se va obține o diagramă Karnaugh goală care, minimizată, va genera valoarea 0:

X	**0**	**1**
	0	0

La fel se va proceda în cazul tuturor celor 8 zone corespunzătoare celor 8 intrări în MUX.

Pasul 5: implementarea cu MUX 8:1 folosind variabilele de selecție y, z și t în această ordine este:



2. Implementați aceeași funcție folosind un singur MUX 4:1 (punând două variabile în coloana rezultat):

Pasul 1: Diagrama Karnaugh este identică cu cea de la problema anterioară, așa că nu se va repeta.

Pasul 2: Identificarea variabilelor de selecție din MUX → vor fi alese variabilele z și t ca variabile de selecție.

Pasul 3: Identificarea zonelor acoperite de fiecare combinație de valori ale acestor variabile.

xy / zt	00	01	11	10
00	1	1	1	
01			1	
11	1		1	1
10		1	1	

Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Astfel diagramele Karnaugh corespunzătoare celor 4 combinații (dependente doar de variabilele x și y) sunt:

xy	00	01	11	10
	1	1	1	

$$f_1(x, y) = \bar{x} + y$$

xy	00	01	11	10
			1	

$$f_2(x, y) = x * y$$

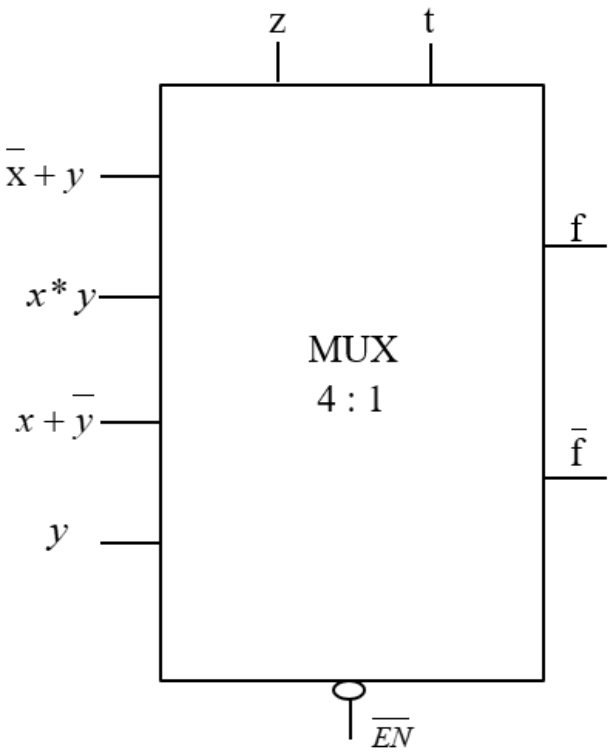
xy	00	01	11	10
	1		1	1

$f_3(x,y) = \bar{y} + x$

xy	00	01	11	10
		1	1	

$f_4(x,y) = y$

Pasul 5: Implementarea cu MUX 4:1 folosind variabilele de selecție z și t în această ordine este:



3. Implementați aceeași funcție folosind un singur MUX 4:1, având ca variabile de selecție x și z:

Pasul 1: Diagrama Karnaugh este identică cu cea de la problema 1, așa că nu se va repeta.

Pasul 2: Identificarea variabilelor de selecție din MUX: x și z.

Pasul 3: Identificarea zonelor acoperite de fiecare combinație de valori ale acestor variabile.

zt \ xy	00	01	11	10
00	1	1	1	
01			1	
11	1		1	1
10		1	1	

Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. În acest caz trebuie ținut cont de valorile variabilelor de care depind noile diagrame Karnaugh (y și t): pentru jumătatea din dreapta a diagramei (combinațiile 3 și 4), prima căsuță din dreapta corespunde valorii lui y = 1 și nu valorii y = 0, **deci diagramele Karnaugh trebuie rearanjate**. Același lucru se întâmplă în cazul variabilei t, pentru jumătatea de jos a diagramei Karnaugh inițiale (combinațiile 2 și 4), unde linia de sus corespunde valorii t = 1 și nu t = 0. Astfel, ținând cont de aceste rearanjări, diagramele Karnaugh corespunzătoare celor 4 combinații (dependente doar de variabilele y și t) sunt:

y / t	0	1
0	1	1
1		

$f_1(y,t) = \bar{t}$

y / t	0	1
0		1
1	1	

$f_2(y,t) = \bar{y} * t + y * \bar{t}$

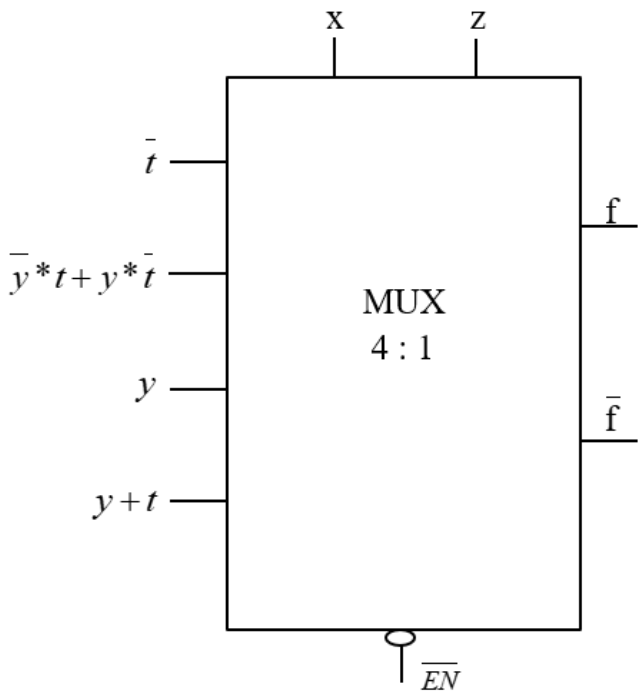
y / t	0	1
0		1
1		1

$f_3(y,t) = y$

y / t	0	1
0		1
1	1	1

$f_4(y,t) = y + t$

Pasul 5: Implementarea cu MUX 4:1 folosind variabilele de selecție x și z în această ordine este:



4. Implementați aceeași funcție folosind un singur MUX 4:1, având ca variabile de selecție y și z:

Pasul 1: Diagrama Karnaugh este identică cu cea de la problema 1, așa că nu se va repeta.

Pasul 2: Identificarea variabilelor de selecție din MUX: y și z.

Pasul 3: Identificarea zonelor acoperite de fiecare combinație de valori ale acestor variabile.

zt \ xy	00	01	11	10
00	1	1	1	
01			1	
11	1		1	1
10		1	1	

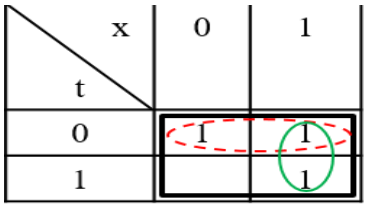
Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Ținând cont de rearanjările care apar, diagramele Karnaugh corespunzătoare celor 4 combinații (dependente doar de variabilele x și t) sunt:

t \ x	0	1
0	1	
1		

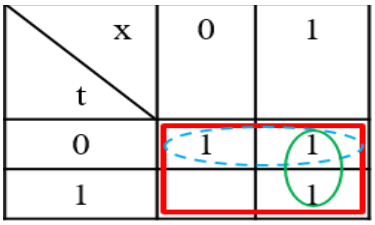
$$f_1(x,t) = \bar{x} * \bar{t}$$

t \ x	0	1
0		
1	1	1

$$f_2(x,t) = t$$

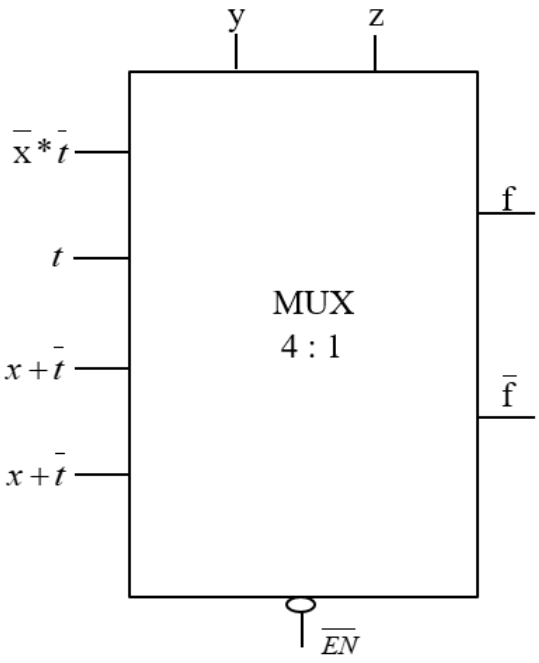


$$f_3(x,t) = x + \bar{t}$$



$$f_4(x,t) = x + \bar{t}$$

Pasul 5: Implementarea cu MUX 4:1 folosind variabilele de selecție y și z în această ordine este:

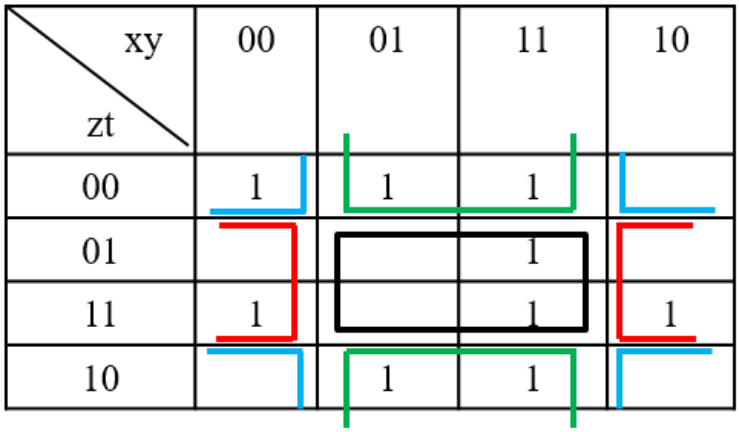


5. Implementați aceeași funcție folosind un singur MUX 4:1, având ca variabile de selecție y și t:

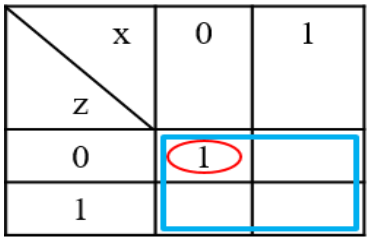
Pasul 1: Diagrama Karnaugh este identică cu cea de la problema 1, așa că nu se va repeta.

Pasul 2: Identificarea variabilelor de selecție din MUX: y și t.

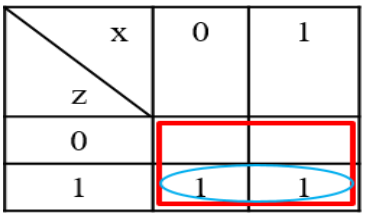
Pasul 3: Identificarea zonelor acoperite de fiecare combinație de valori ale acestor variabile.



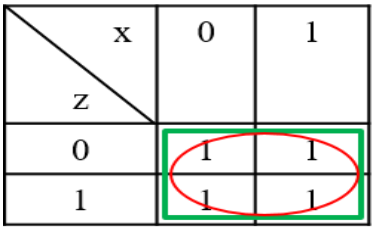
Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Diagramele Karnaugh corespunzătoare celor 4 combinații (dependente doar de variabilele x și z) sunt:



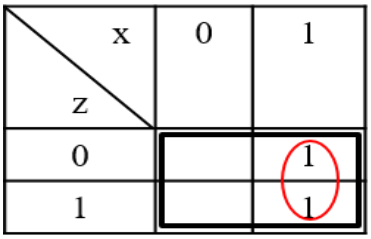
$$f_1(x,z) = \overline{x} * \overline{z}$$



$f_2(x,z) = z$

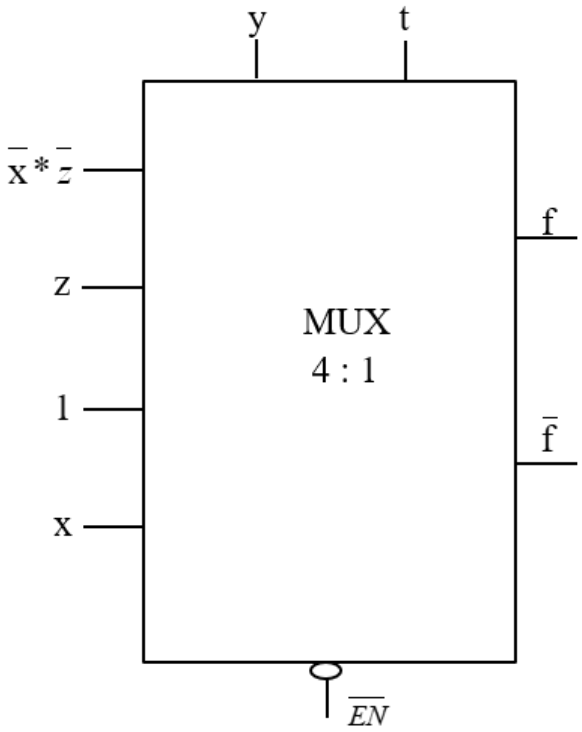


$f_3(x,z) = 1$



$f_4(x,z) = x$

Pasul 5: implementarea cu MUX 4:1 folosind variabilele de selecție y și t în această ordine:

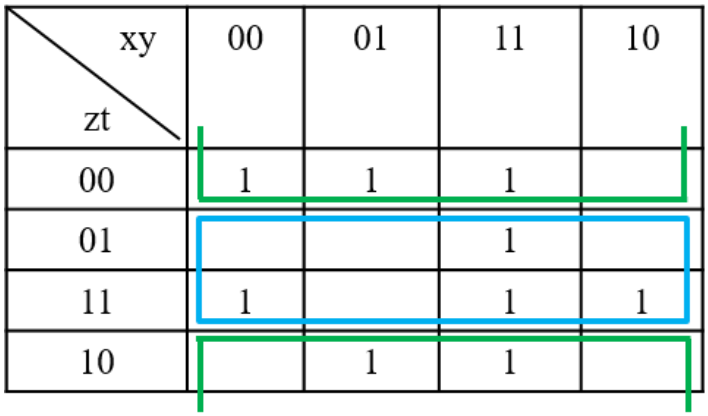


6. Implementați aceeași funcție folosind un singur MUX 2:1.

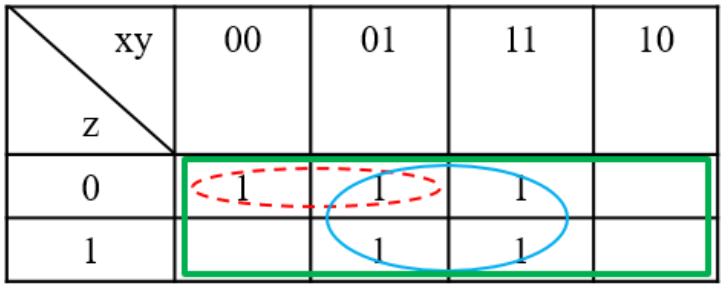
Pasul 1: Diagrama Karnaugh este identică cu cea de la problema 1, așa că nu se va repeta.

Pasul 2: Identificarea variabilelor de selecție din MUX: se va folosi variabila t drept variabilă de selecție.

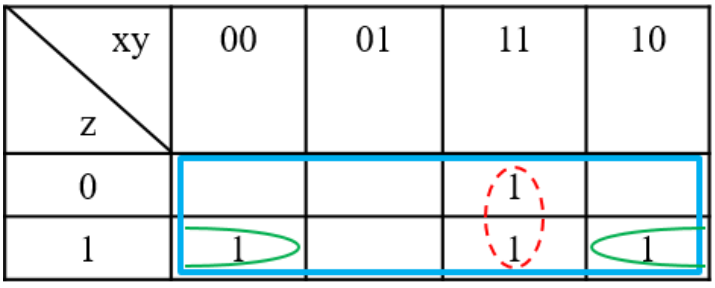
Pasul 3: Identificarea zonelor acoperite de fiecare valoare a acestei variabile.



Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Diagramele Karnaugh corespunzătoare celor 2 combinații (dependente doar de variabilele x, y și z) sunt:

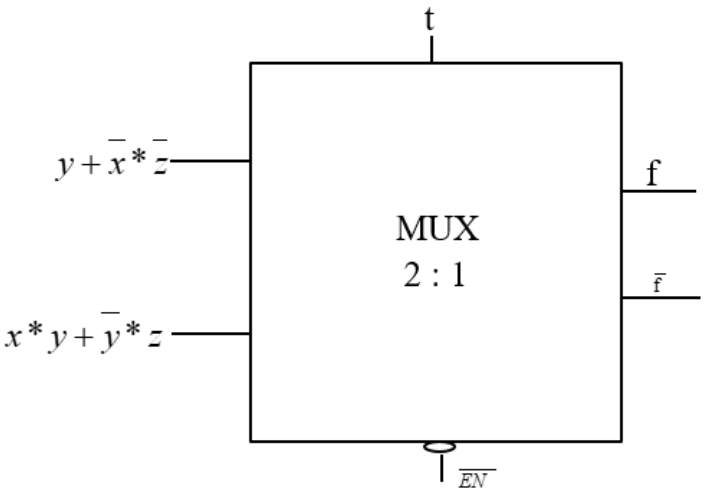


$$f_1(x, y, z) = y + \bar{x} * \bar{z}$$



$$f_2(x, y, z) = x * y + \bar{x} * z$$

Pasul 5: Implementarea cu MUX 2:1 folosind variabila de selecție t este:



7. Implementați funcția de mai jos folosind un MUX 2:1.

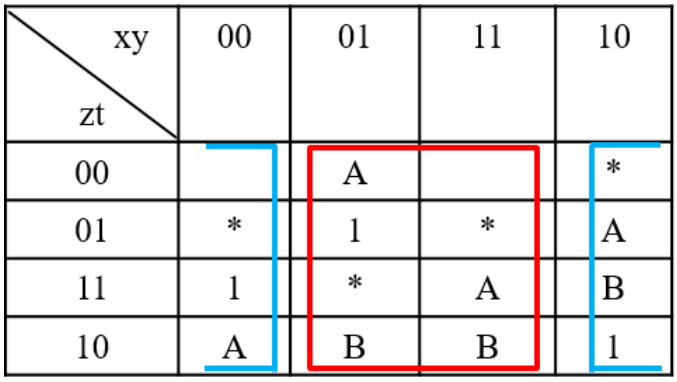
x	**y**	**z**	**t**	**f**
0	0	0	0	**0**
0	0	0	1	*
0	0	1	0	**A**
0	0	1	1	**1**
0	1	0	0	**A**
0	1	0	1	**1**
0	1	1	0	**B**
0	1	1	1	*
1	0	0	0	*
1	0	0	1	**A**
1	0	1	0	**1**
1	0	1	1	**B**
1	1	0	0	**0**
1	1	0	1	*
1	1	1	0	**B**
1	1	1	1	**A**

Pasul 1: Realizarea diagramei Karnaugh:

xy / zt	00	01	11	10
00		A		*
01	*	1	*	A
11	1	*	A	B
10	A	B	B	1

Pasul 2: Identificarea variabilelor de selecție din MUX: se va folosi variabila y drept variabilă de selecție.

Pasul 3: Identificarea zonelor acoperite de fiecare valoare a acestei variabile.



Pasul 4: În continuare, fiecare zonă trebuie extrasă într-o nouă diagramă Karnaugh și minimizată separat. Diagramele Karnaugh corespunzătoare celor 2 combinații (dependente doar de variabilele x, z și t) sunt prezentate în continuare. Așa cum se poate observa, cele două diagrame sunt de tipul funcțiilor incomplet specificate, cu variabile în coloana rezultat, deci minimizarea se face în doi pași. Pentru a nu reface diagrama de două ori, s-au realizat minimizările

dintr-un pas cu linie întreruptă (simplă pentru suprafețele ce corespund variabilei A și dublă pentru suprafețele corespunzătoare lui B), iar cele din celălalt pas cu linie continuă îngroșată.

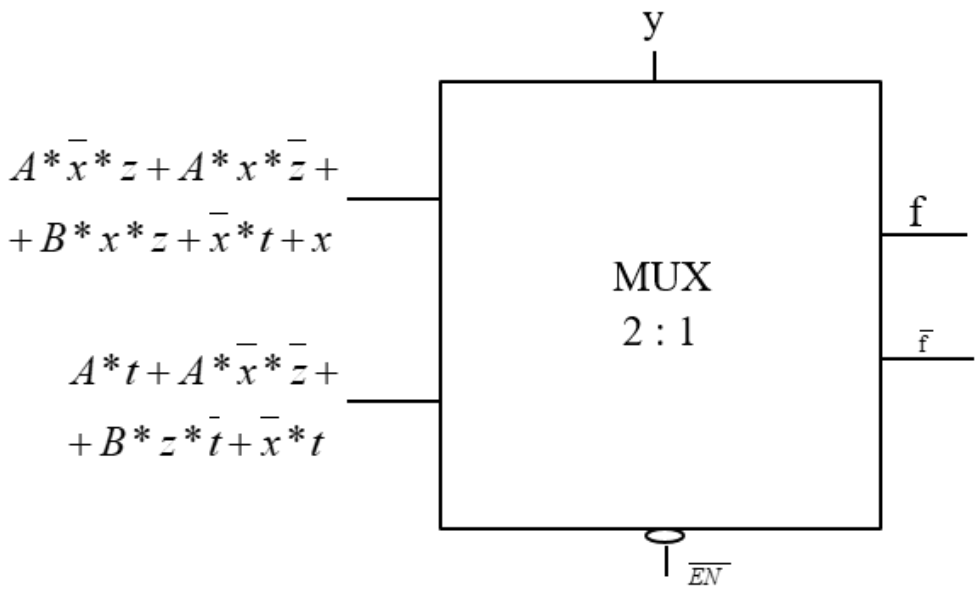
x / zt	0	1
00		*
01	*	A
11	1	B
10	A	1

$$f_1(x,z,t) = A*\bar{x}*z + A*x*\bar{z} + B*x*z + \bar{x}*t + x*\bar{t}$$

x / zt	0	1
00	A	
01	1	*
11	*	A
10	B	B

$$f_2(x,z,t) = A*t + A*\bar{x}*\bar{z} + B*z*\bar{t} + \bar{x}*t$$

Pasul 5: Implementarea cu MUX 4:1 folosind variabila de selecție t este:



VI.5 Probleme propuse

1. Implementați funcția de mai jos folosind un MUX 8:1.

x	y	z	f
0	0	0	**1**
0	0	1	*
0	1	0	**1**
0	1	1	**t**
1	0	0	**0**
1	0	1	**1**
1	1	0	*
1	1	1	**1**

2. Implementați funcția de la exercițiul anterior folosind un DMUX 8:1.

3. Implementați funcția de la exercițiul 1 folosind o structură cu multiplexoare cuplate în cascadă, având pe primul nivel MUX-uri 4:1 comandate de variabilele y, x (în această ordine), iar pe al doilea un MUX 2:1.

4. Implementați funcția de la exercițiul 1 folosind MUX-uri 2:1 cuplate în cascadă.

5. Implementați funcția de mai jos folosind un MUX 16:1.

x	**y**	**z**	**t**	**f**
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	*****
0	0	1	1	**1**
0	1	0	0	*****
0	1	0	1	*****
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	*****
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**0**
1	1	1	1	*****

6. Implementați funcția de la exercițiul anterior folosind un DMUX 16:1.

7. Implementați funcția de la exercițiul 5 folosind MUX-uri 4:1 cuplate în cascadă și având variabilele y, t pe primul nivel și x, z pe al doilea.

8. Implementați funcția de la exercițiul 5 folosind MUX-uri 4:1 cuplate în cascadă și având variabilele t, x pe primul nivel și y, z pe al doilea.

9. Implementați funcția de la exercițiul 1 folosind un singur MUX 4:1, având variabilele de selecție x și z.

10. Implementați funcția de la exercițiul 1 folosind un singur MUX 2:1, având y ca variabilă de selecție.

11. Implementați funcția de la exercițiul 5 folosind un singur MUX 8:1, având variabilele de selecție y, x și z.

12. Implementați funcția de la exercițiul 5 folosind un singur MUX 4:1, având variabilele de selecție x și t.

13. Implementați funcția de la exercițiul 5 folosind un singur MUX 4:1, având variabilele de selecție z și y.

14. Implementați funcția de la exercițiul 5 folosind un singur MUX 2:1, având pe z drept variabilă de selecție.

15. Implementați funcția de la exercițiul 5 folosind un MUX 2:1 comandat de variabila y.

16. Implementați funcția de la exercițiul 5 folosind un MUX 2:1 comandat de variabila x.

Bibliografie

Culegerea de probleme a rezultat din studierea referințelor bibliografice menționate mai jos, din care au fost extrase și aplicate metode de rezolvare pentru problemele prezentate.

- Notițele de la curs
- *L. Văcariu, O. Creţ*, Analiza şi sinteza dispozitivelor numerice - Îndrumător de laborator, ediţia a 3-a., Ed. U.T. Press, Cluj-Napoca, 2009
- *L. Văcariu, O. Creţ*, Probleme de proiectare logică - a sistemelor numerice - / Logic design problems - for digital systems -, ediţia a 2-a, rev., Ed. U.T. Press, Cluj-Napoca, 2013
- *B. Holdsworth*, Digital Logic Design, , Ed. Butterworths, 1987
- *Th. F. Bogart Jr.*, Introduction to Digital Circuits, Ed. Glencoe, 1992
- *J. F.Wakerly*, Digital Design Principles And Practices Solutions, 1999
- *M. Morris Mano and M. Ciletti*, Digital Design, 5th Edition, 2013
- *Gh. Toacse și D. Nicula*, Electronica Digitala, vol 1, Ed. Tehnica, 2005
- *D. Nicula și Gh. Toacse*, Electronica Digitala, vol 2, Ed. Tehnica, 2005
- *Gh. Stefan*, Circuite si Sisteme Digitale, Ed. Tehnica, 2000
- *Spanulescu și S. Spanulescu*, Circuite Integrate Digitale si Sisteme cu Microprocesoare, Ed. Victor, 1996
- *R. Popa și M. Iliev*, Analiza si sinteza sistemelor numerice. Aplicatii, Editura Fundatiei Universitare "Dunarea de Jos" Galati, 2003, ISBN 973-627-080-7
- *B. Wilkinson*, Electronica Digitala. Bazele proiectarii, Editura Teora, Bucuresti, 2002 (Original English title: *B. Wilkinson*, The Essence of Digital Design, Prentice Hall, 1997)
- *Gh. Andronescu*, Sisteme digitale, Editura MATRIX ROM, Bucuresti,2001
- *D. Potorac*, Bazele proiectarii circuitelor numerice, Editura MATRIX ROM, Bucuresti, 2002
- *D. Friedman*, Fundamentals of Logic Design and Switching Theory, Computer Science Press, Rockville, 1986