Centrul de formare continuă, învăţământ la distanţă
şi învăţământ cu frecvenţă redusă

LOGICĂ COMPUTAŢIONALĂ

Laura Ciupală

2010

Introducere

În zilele noastre calculatorul a devenit nelipsit în majoritatea activităţilor cotidiene. Pentru un bun informatician este necesar nu numai să-l folosească, dar şi să-i înţeleagă modul de funcţionare. Cursul de *Logică computaţională* este o prezentare a fundamentelor aritmetice şi logice ale sistemelor de calcul, care, în general, rămân neschimbate, deşi tehnologiile folosite la fabricarea calculatoarelor sunt într-o continuă schimbare.

Obiectivele cursului

Cursul de *Logică computaţională* are ca obiectiv principal îmbogăţirea cunoştinţelor de fundamentele Informaticii ale studenţilor Programului de studii Informatică, forma de învăţământ ID. În acest sens, la sfârşitul acestui curs, studenţii vor fi capabili să:

- Efectueze operaţii aritmetice cu numere reprezentate în orice bază;
- Să efectueze conversii între diferite baze;
- Să identifice legăturile dintre calculatoare şi sistemele de numeraţie binare, octale şi hexazecimale;
- Să ştie cum sunt reprezentate în memoria calculatorului numerele întregi, reale şi caracterele;
- Să interpreteze conţinutul unei locaţii de memorie, cunoscând tipul de date memorate;
- Să determine forma normală disjunctivă şi forma normală conjunctivă a unei funcţii Booleene;
- Să minimize funcţii Boolene;
- Să construiască circuitul logic combinatorial optim pentru orice problemă practică care poate fi scrisă printr-o funcţie Booleană.

Cerinţe preliminare

Nu sunt necesare decât cunoştinţe din învăţământul preuniversitar, acesta fiind unul dintre cursurile din primul semestru.

Resurse

Parcurgerea unităţilor de învăţare aferente acestui curs nu necesită existenţa unor mijloace sau instrumente de lucru.

Structura cursului

Cursul de *Logică computaţională* este structurat în patru module, astfel: primul modul cuprinde trei unităţi de învăţare, al doilea modul trei unităţi de învăţare, al treilea modul cinci unităţi de învăţare, iar al patrulea modul cunprinde două unităţi de învăţare. La rândul său, fiecare unitate de învăţare cuprinde: obiective, aspecte teoretice privind tematica unităţii de învăţare respective, exemple, teste de evaluare precum şi probleme propuse spre discuţie şi rezolvare.

La sfârşitul ultimelor trei module sunt indicate teme de control. Rezolvarea acestor trei teme de control este obligatorie. Acestea vor fi transmise de către studenţi profesorului până la odată prestabilită.

Durata medie de studiu individual

Parcurgerea de către studenţi a unităţilor de învăţare ale cursului de *Logică computaţională* (atât aspectele teoretice cât şi rezolvarea testelor de autoevaluare şi rezolvarea problemelor propuse) se poate face în 2-3 ore pentru fiecare unitate de învăţare.

Evaluarea

La sfârşitul semestrului, fiecare student va primi o notă, care va cuprinde: un test, ce va conţine întrebări teoretice şi practice din materia prezentată în cadrul acestui material, test de va deţine o pondere de 75% în nota finală. Media notelor aferente testelor de evaluare, realizate pe parcursul semestrului, va deţine o pondere de 25%.

Spor la treaba !

Cuprins

Modulul 1. Sisteme de numeraţie

Cuprins

Introducere

Sistemele de numeraţie au apărut pentru a răspunde necesităţii oamenilor de a număra. Dacă iniţial oamenii peşterilor făceau doar diferenţa dintre *unu* şi *mulţi*, cu timpul limbile primitive au evoluat astfel încât se putea face distincţia între *unu*, *doi* şi *mulţi*, iar mai târziu între *unu*, *doi*, *trei* şi *mulţi*. Adică nu existau cuvinte care să exprime numere mai mari decât trei. Acestă lipsă mai persistă şi astăzi în unele limbi, de exemplu în limba vorbită de indienii Siriona din Bolivia.

Deşi nu aveau cuvinte care să exprime numere mai mari decât 2, anumite triburi (Bacairi şi Bororo din Brazilia) au creat sisteme numerice de genul *unu*, *doi*, *doi şi unu*, *doi şi doi*, *doi si doi şi unu* şi aşa mai departe. Acest sistem de numeraţie nu este altceva decât binecunoscutul sistem binar atât de răspândit în era calculatoarelor. Însă puţini oameni au numărat din 2 în 2. O altă modalitate de numărare era cu ajutorul unor crestături făcute într-un os. Crestăturile mici erau grupate câte 5, iar după 5 grupe exista o crestătură de alt tip. Acestă modalitate de grupare a crestăturilor în os corespunde sistemului de numeraţie în baza 5, care a fost preferat de multe culturi, fiind mai uşor de utilizat deoarece, pentru a număra, oamenii îşi foloseau degetele, care sunt grupate tot câte 5. Folosind apoi degetele ambelor mâini s-a ajuns la sistemul zecimal, iar folosindu-le şi pe cele ale picioarelor la cel în baza 20, utilizat de mayaşi. De asemenea, locuitorii Franţei se pare că au folosit sistemul în baza 20.

Un sistem care merită menţionat este cel babilonian care era sexagesimal, adică în baza 60, care, în mod bizar, nu avea 60 de simboluri diferit pentru fiecare „cifră” . Babilonienii nu foloseau decât două semne: un cui pentru 1 şi două cuie pentru 10. Celelalte „cifre” erau grupe de astfel de semne. Dar o mare problema a sistemului babilonian a fost faptul că un cui putea reprezenta 1, 60, 3600 sau o infinitate de alte numere.

În zilele noastre, este util să putem reprezenta numere şi să putem lucra cu numere nu numai în baza 10, dar şi în orice alt sistem de numeraţie şi să putem trece dintr-o bază în alta.

Competenţe

La sfârşitul acestui modul studenţii vor fi capabili să:

- Reprezinte orice număr într-o bază dată *b*;
- Efectueze schimbări de baze;
- Efectueze operaţii aritmetice cu numere reprezentate în orice bază;
- Să identifice legăturile dintre calculatoare şi sistemele de numeraţie binare, octale şi hexazecimale.

Unitatea de învăţare M1.U1. Conceptul de sistem de numeraţie

Cuprins

M1.U1.1. Introducere

Sistemele de numeraţie au apărut pentru a răspunde necesităţii oamenilor de a număra. În decursul timpului au fost folosite diferite sisteme de numeraţie, până ca sistemul zecimal să se impună.

M1.U1.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a conceptului de sistem de numeraţie.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- realizeze deosebirile dintre sistemele de numeraţie poziţionale şi cele nepoziţionale;
- Reprezinte orice număr într-o bază dată *b*.

Durata medie de parcurgere a primei unităţi de învăţare este de 3 ore.

M1.U1.3. Definirea sistemului de numeraţie

Un *sistem de numeraţie* este format din totalitatea regulilor de reprezentare a numerelor cu ajutorul unor simboluri distincte, numite *cifre*. Există două categorii de sisteme de numeraţie:

1. sisteme de numeraţie poziţionale
2. sisteme de numeraţie aditive sau nepoziţionale

Un sistem de numeraţie se numeşte *poziţional*, dacă valoarea unei cifre este dată de poziţia pe care aceasta o ocupă în cadrul numărului. Sistemele de numeraţie zecimal, binar, octal, hexazecimal sunt sisteme de numeraţie poziţionale.

Într-un sistem de numeraţie *aditiv*, cifrele au aceeaşi valoare, indiferent de poziţia pe care se află. De exemplu, sistemul de numeraţie roman este un sistem de numeraţie aditiv.

Numărul de cifre dintr-un sistem de numeraţie poziţional se numeşte *bază*.

Exemple de sisteme de numeraţie poziţionale

1. Sistemul binar are baza 2, iar cifrele sale sunt 0 si 1, numite şi *biţi* (bit = binary digit). Sistemul binar este utilizat pentru reprezentarea cele două stări posibile ale unui dispozitiv bistabil. De exemplu, un întrerupător poate fi deschis sau închis, un loc de pe un disc magnetic pot fi magnetizat sau nemagnetizat etc.
2. Sistemul octal are baza 8, iar cifre sale sunt 0, 1, 2, 3, 4, 5, 6 şi 7. A fost folosit şi este folosit în continuare în limbajul de asamblare.
3. Sistemul zecimal este sistemul de numeraţie pe care îl folosim în viaţa de zi cu zi, are la baza 10, iar cifre sale sunt 0, 1, 2, 3, 4, 5, 6, 7, 8 şi 9.
4. Sistemul hexazecimal are baza 16 şi cifre sale sunt 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E şi F. Sistemul de numeraţie hexazecimal este utilizat atunci când se lucrează direct cu adresele de memorie.

Exemplu de sistem de numeraţie aditiv

Sistemul roman este un sistem de numeraţie aditiv. Cifrele sale sunt I, V, X, L, C, D, M, ale căror valori sunt 1, 5, 10, 50, 100, 500, 1000.

Reguli de scriere:

1. mai multe cifre de aceeaşi valoare, scrise consecutiv, reprezintă suma acestor cifre. De exemplu, II = 2, XXX = 30.
2. două cifre diferite, cu cifra mai mare aflată în faţa cifrei mai mici, reprezintă suma acestor cifre. De exemplu, XI = 11, CX = 110.
3. două cifre diferite, cu cifra mai mare aflată după cifra mai mică, reprezintă diferenţa acestor cifre. De exemplu, IX = 9, XC = 90.

Sistemul roman are câteva dezavantaje majore: sunt posibile mai multe reprezentări ale aceluiaşi număr (VIII = IIX), reprezentări lungi ale unor numere relativ “mici” (CCCXXXIII = 333), operaţii aritmetice foarte dificil de efectuat, lipsa unei reprezentări pentru 0 etc.

M1.U1.4. Teorema sistemelor de numeraţie

Teorema sistemelor de numeraţie Fie $b\in\mathbf{N}^*$, $b > 1$. Pentru orice număr natural $N\in\mathbf{N}^*$, există numerele n, a_0, a_1,..., $a_n\in\mathbf{N}$ astfel încât $a_i < b$ pentru orice i, $0 \le i \le n$, $a_n > 0$ şi

$$N = a_nb^n + a_{n-1}b^{n-1} + ...+ a_1b + a_0.$$

Mai mult, aceste numere sunt unice.

Demonstraţie. Din Teorema împărţirii cu rest avem că:

$$N = N_0b + a_0 \text{ cu } 0 \le a_0 < b.$$

Evident, $N_0 < N$, pentru că în caz contrar am avea

$$N < Nb \le N_0b \le N_0b + a_0 = N, \text{ ceea ce este absurd.}$$

De asemenea, din Teorema împărţirii cu rest avem că:

$$N_0 = N_1b + a_1 \text{ cu } 0 \le a_1 < b\ ,\ N_1 < N_0.$$

În acelaşi mod, determinăm şirul finit:

$$N > N_0 > N_1 > N_2.> ... > 0.$$

Deci, există un număr natural n astfel încât

$$N_n = 0$$

sau, echivalent,

$$N_{n-1} = 0\cdot b + a_n, \text{ cu } 0 < a_n < b.$$

Deci, am obţinut:

$$N = N_0b + a_0$$
$$N_0 = N_1b + a_1$$
$$....$$
$$N_{n-2} = N_{n-1}\, b + a_{n-1}$$
$$N_{n-1} = a_n$$

Înmulţind aceste egalităţi cu puteri ale lui b (de la 0 la n) şi adunându-le, obţinem:

$$N = a_nb^n + a_{n-1}b^{n-1} + ...+ a_1b + a_0.$$

Vom demonstra prin reducere la absurd că numerele n, a_0, a_1,..., a_n sunt unice. Presupunem prin absurd că există numerele p, c_0, c_1,..., c_p astfel încât

$$N = c_pb^p + c_{p-1}b^{p-1} + ...+ c_1b + c_0.$$

Dacă $n < p$ atunci,

$$N = \sum_{i=0}^{n} a_ib^i < b^{n+1} \le b^p \le \sum_{i=0}^{p} c_ib^i = N, \text{ ceea ce este absurd.}$$

Deci, $n \ge p$.

În acelaşi mod, se poate arăta că $p \ge n$. Deci, $n = p$.

Mai trebuie demonstrat că $a_i = c_i$ for $0 \le i \le n$.

Dacă $n = 0$ atunci $a_0 = N = c_0$.

Dacă $n > 0$, atunci

$$N = a_0 + b(a_nb^{n-1} + a_{n-1}b^{n-2} + ...+ a_1),\ 0 \le a_0 < b$$

şi

$$N = c_0 + b(c_nb^{n-1} + c_{n-1}b^{n-2} + ...+ c_1)\ ,\ 0 \le c_0 < b.$$

Din Teorema împărţirii cu rest rezultă că

$a_0 = c_0$ pentru că sunt resturi în împărtirea a două numere întregi

şi

$a_nb^{n-1} + a_{n-1}b^{n-2} + ...+ a_1 = c_nb^{n-1} + c_{n-1}b^{n-2} + ...+ c_1$ pentru că sunt câturi în împărtirea a două numere întregi.

Succesiv, obţinem că $a_1 = c_1$, $a_2 = c_2$, ..., $a_n = c_n$.

Observaţie *a*) Teorema sistemelor de numeraţie arată că există o corespondenţă biunivocă între numerele naturale strict pozitive N şi şirurile finite de numere naturale $a_0, a_1,..., a_n$ unde $0 \le a_i < b$ pentru orice i, $0 \le i \le n$ şi $0 < a_n < b$. Deci, orice număr natural N poate fi scris în oricare dintre următoarele forme:

$$N = a_na_{n-1}...a_1a_0 = a_nb^n + a_{n-1}b^{n-1} + ...+ a_1b + a_0.$$

Suma $a_nb^n + a_{n-1}b^{n-1} + ...+ a_1b + a_0$ se numeşte *notaţie extinsă.*

a_n este *cea mai semnificativă cifră*, iar a_0 este *cea mai puţin semnificativă cifră* a numărului N.

b) Atunci când se lucrează cu diferite baze, se precizează şi baza: vom nota $a_na_{n-1}...a_1a_{0(b)}$ în loc de $a_na_{n-1}...a_1a_0$.

O consecinţă importantă a Teoremei sistemelor de numeraţie este algoritmul sistemelor de numeraţie, care este o metodă de conversie a unui număr întreg din baza 10 într-o bază oarecare *b*.

Algoritm sisteme numeratie;
begin
    read N, b;
    $i \leftarrow 0$;
    repeat
        $a_i \leftarrow N \% b$; //restul împăţirii
        $N \leftarrow [N / b]$; //câtul împărţirii
        $i \leftarrow i + 1$;
    until $N = 0$;
    write cifrele $a_{i-1}, a_{i-2}, ... , a_0$ ale numărului scris în baza b;
end.

Exemplu

Vom converti numărul zecimal 493 în baza 8.

Împărţind 493 la 8 obţinem câtul 61 şi restul 5.
Împărţind 61 la 8 obţinem câtul 7 şi restul 5.
Împărţind 7 la 8 obţinem câtul 0 şi restul 7.

Conversia în baza 8 a numărului zecimal 493 se obţine scriind, în ordine inversă, resturile obţinute la împărţirile succesive la 8. Deci, $493_{(10)} = 755_{(8)}$.

Exemplu
Vom converti numărul zecimal 168 în baza 16.

Împărţind 168 la 16 obţinem câtul 10 şi restul <u>8</u>.
Împărţind 10 la 16 obţinem câtul 0 şi restul 10, căruia îi corespunde cifra hexazecimală <u>A</u>.
Conversia în baza 16 a numărului 168 se obţine scriind, în ordine inversă, resturile obţinute la împărţirile succesive la 16. Deci, $168_{(10)} = A8_{(16)}$.

M1.U1.5. Rezumat

Există două categorii de sisteme de numeraţie: sisteme de numeraţie poziţionale şi sisteme de numeraţie aditive sau nepoziţionale. Sistemul roman este un sistem de numeraţie nepoziţional. Sistemul zecimal, pe care îl folosim de obicei şi sistemul binar folosit de calculatoare sunt sisteme poziţionale.

Orice număr natural strict pozitiv N poate fi scris în mod unic sub forma

$$\boldsymbol{N = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0}$$

unde $0 \le a_i < b$ pentru orice i, $0 \le i \le n$ şi $0 < a_n < b$,

sau echivalent $\boldsymbol{N = a_n a_{n-1} \ldots a_1 a_{0(b)}}$.

Algoritmul sistemelor de numeraţie este o metodă de conversie a unui număr întreg din baza 10 într-o bază oarecare $b>1$.

M1.U1.6. Test de evaluare a cunoştinţelor

1. Scrieţi numărul zecimal 123 în baza 2, în baza 5, în baza 8 şi în baza 16.
2. Scrieţi numărul zecimal 58674 în baza 5, în baza 8, în baza 9 şi în baza 16.

Unitatea de învăţare M1.U2. Aritmetica în baza *b*

Cuprins

M1.U2.1. Introducere

Sistemele de numeraţie au apărut din necesitatea oamenilor de a număra şi de a socoti. În zilele noastre, sistemul zecimal se foloseşte aproape pretutindeni, însă este uneori necesar să efectuăm operaţii în alte sisteme de numeraţie. În continuare vom studia aritmetica într-o bază oarecare *b*.

M1.U2.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a aritmeticii într-o bază oarecare *b*, $b > 1$.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să compare numere scrise în orice bază *b*
- Să efectueze adunări, scăderi şi înmulţiri de numere întregi într-o bază dată *b*.

Durata medie de parcurgere a acestei unităţi de învăţare este de 3 ore.

M1.U2.3. Compararea numerelor în baza *b*

Teoremă (Compararea numerelor naturale) Fie $b \in \mathbf{N}$, $b > 1$ şi fie a şi c două numere naturale scrise în baza b, $a = a_m a_{m-1} \ldots a_1 a_{0(b)}$, $c = c_n c_{n-1} \ldots c_1 c_{0(b)}$. Atunci

$a < c$ dacă şi numai dacă $m < n$ sau

$$m = n \text{ şi } a_p < c_p \text{ unde } p = \max \{i \mid a_i \neq c_i\}.$$

Demonstrație. Fie $a < c$.

Presupunem prin absurd că $m > n$.

Atunci $c = \sum_{i=0}^{n} c_i b^i \le \sum_{i=0}^{n} (b-1)b^i = b^{n+1}$ -1< $b^{n+1} \le b^m \le a$, ceea ce este absurd.

Deci, $m \le n$.

Dacă $m < n$ atunci stop, altfel înseamnă că $m = n$. În acest caz, evident că $a_p < c_p$, unde $p = \max\{i \mid a_i \ne c_i\}$.

Reciproc, dacă $m < n$ atunci $a = \sum_{i=0}^{m} a_i b^i \le \sum_{i=0}^{m} (b-1)b^i = b^{m+1}$ -1< $b^{m+1} \le b^n \le c$. Deci, $a < c$.

Dacă $m = n$ şi $a_p < c_p$ unde $p = \max\{i \mid a_i \ne c_i\}$, atunci

$$c - a = (c_p - a_p)\, b^p + (c_{p-1}b^{p-1} + ... + c_0) - (a_{p-1}b^{p-1} + ... + a_0) >$$
$$> b^p + (c_{p-1}b^{p-1} + ... + c_0) - b^p \ge 0.$$

Deci, $a < c$.

Observație Teorema anterioară ne oferă o metodă de comparare a numerelor naturale scrise în baza b. Dacă dorim să comparăm două numere întregi, nu neapărat pozitive, pot apărea 3 cazuri:

1. Amândouă sunt pozitive. În acest caz se aplică teorema anterioară.
2. Unul este negativ şi celălalt pozitiv. Evident, numărul negativ este mai mic decât cel pozitiv.
3. Amândouă sunt negative: $a, c < 0$. În acest caz $a > c$ dacă şi numai dacă $-a < -c$.

Exemple

$101011_{(2)} > 11111_{(2)}$

$AB56_{(16)} > A9FF_{(16)}$

$-1230_{(5)} > -2121_{(5)}$ pentru că $1230_{(5)} < 2121_{(5)}$.

M1.U2.4. Adunarea numerelor în baza *b*

Fie $b \in \mathbf{N}$, $b > 1$ şi fie a şi c două numere naturale scrise în baza b, $a = a_m a_{m-1} ... a_1 a_{0(b)}$, $c = c_n c_{n-1} ... c_1 c_{0(b)}$. Dorim să determinăm suma $s = a + c$.

Avem $a = a_0 + a_1 b + a_2 b^2 + ... + a_m b^m$ şi $c = c_0 + c_1 b + c_2 b^2 + ... + c_n b^n$.

Deoarece $a_0 < b$ şi $c_0 < b$, rezultă că $a_0 + c_0 < 2b$ sau, echivalent,

$$a_0 + c_0 = r_1 \cdot b + s_0,$$

unde $0 \le s_0 < b$, iar r_1 este 0 sau 1 şi se numeşte *transport*.

Mai exact,

dacă $a_0 + c_0 < b$ atunci $s_0 = a_0 + c_0$ şi $r_1 = 0$

dacă $a_0 + c_0 \ge b$ atunci $s_0 = a_0 + c_0 - b$ şi $r_1 = 1$.

Deci,

$$s = a + c = s_0 + (a_1 + c_1 + r_1)b + (a_2 + c_2)b^2 + ...$$

Evident că

$$a_1 + c_1 + r_1 < 2b.$$

Deci,

$$a_1 + c_1 + r_1 = r_2 b + s_1,$$

unde $0 \le s_1 < b$, iar transportul r_2 este 0 sau 1.

În acelaşi fel, determinăm toate cifrele sumei s:

$$s_i = (a_i + c_i + r_i)\%b,\ \forall\, i \ge 0,$$

unde $r_0 = 0$,

$$r_i = \begin{cases} 0, \text{dacă } a_{i-1} + c_{i-1} + r_{i-1} < b \\ 1, \text{altfel} \end{cases},\ \forall\, i > 0.$$

Observaţie 1) Dacă adunăm două numere naturale a şi c, fiecare având $(m+1)$ cifre, suma $s = a + c$ poate avea:

$(m+1)$ cifre dacă $a_m + c_m + r_m < b$ sau

$(m+2)$ cifre dacă $a_m + c_m + r_m \ge b$. În acest caz $s_{m+1} = 1$.

2) Dacă cele două numere naturale a şi c un un număr diferit de cifre atunci:

dacă $m > n$ considerăm $c_{n+1} = c_{n+2} = ... = c_m = 0$

dacă $m < n$ considerăm $a_{m+1} = a_{m+2} = ... = a_n = 0$.

Algoritm adunare;
begin
    read $b, m, a_m, a_{m-1}, ..., a_0, n, c_n, c_{n-1}, ..., c_0$;
    if $m > n$ **then**
        for $i = n+1$ **to** m **do** $c_i = 0$;
    else
        if $m < n$ **then begin**
                for $i = m+1$ **to** n **do** $a_i = 0$;
                $m \leftarrow n$;
        end;
    $r \leftarrow 0$;
    for $i = 0$ **to** m **do begin**
            $s_i \leftarrow (a_i + c_i + r)\%b$;
            $r \leftarrow (a_i + c_i + r)/b$;
    end;
    if $r = 1$ **then**
        write $1, s_m, s_{m-1}, ..., s_0$;
    else
        write $s_m, s_{m-1}, ..., s_0$;
end.

Exemple

' ' ' '	' '	' '
$101011_{(2)}$ +	$12345_{(8)}$ +	$A1B2C_{(16)}$ +
$\underline{111}_{(2)}$	$\underline{34274}_{(8)}$	$\underline{26F54}_{(16)}$
$110010_{(2)}$	$46641_{(8)}$	$C8A80_{(8)}$

Efectuaţi următoarele adunări:

$101011_{(2)} + 11111_{(2)}$

$AB56_{(16)} + A9FF_{(16)}$

$1230_{(5)} + 2121_{(5)}$.

M1.U2.5. Scăderea numerelor în baza *b*

Fie $b \in \mathbf{N}$, $b > 1$ şi fie a şi c două numere naturale scrise în baza b, $a = a_m a_{m-1} \ldots a_1 a_{0(b)}$, $c = c_n c_{n-1} \ldots c_1 c_{0(b)}$, cu $a \geq c$. Dorim să determinăm diferenţa $d = a - c$.

Se efectuează scăderi în baza 16 atunci când se lucrează direct cu conţinutul locaţiilor de memorie. Există aplicaţii practice în care trebuie să scădem o adresă de memorie dintr-o altă adresă de memorie mai mare.

Algoritm scadere;
begin
    read $b, m, a_m, a_{m-1}, \ldots, a_0, n, c_n, c_{n-1}, \ldots, c_0$;
    if $m > n$ **then**
        for $i = n + 1$ **to** m **do** $c_i = 0$;
    for $i = 0$ **to** m **do**
        if $a_i \geq c_i$ **then**
            $d_i \leftarrow a_i - c_i$;
        else
        begin
            se împrumută 1 de la a_{i+1}; // $a_{i+1} = a_{i+1} - 1$;
            $d_i \leftarrow b + a_i - c_i$;
        end;
    while $d_m = 0$ **do**
        $m \leftarrow m - 1$;
    write $d_m, d_{m-1}, \ldots, d_0$;
end.

Exemple

$101011_{(2)}$ -	$72345_{(8)}$ -	$A4EC_{(16)}$ -
$\underline{111_{(2)}}$	$\underline{34274_{(8)}}$	$\underline{34B4_{(16)}}$
$100100_{(2)}$	$36051_{(8)}$	$7038_{(16)}$

Efectuaţi următoarele scăderi:
$101011_{(2)} - 1111_{(2)}$
$AB56_{(16)} - A9FF_{(16)}$
$1230_{(5)} - 221_{(5)}$.

M1.U2.6. Înmulţirea numerelor în baza *b*

Înmulţirea a două numbere a şi c în baza b constă în următoarele operaţii:

1. Înmulţirea numărului a cu puteri ale bazei b
2. Înmulţirea numărului a cu cifre ale numărului c
3. Adunări de numere naturale.

<u>Înmulţirea numărului *a* cu o putere a bazei *b*</u>

Fie $a = a_m a_{m-1} \ldots a_1 a_{0(b)}$.
Atunci $a \cdot b^j = (a_0 + a_1 b + a_2 b^2 + \ldots + a_m b^m) \cdot b^j = a_0 b^j + a_1 b^{j+1} + a_2 b^{j+2} + \ldots + a_m b^{m+j} = a_m a_{m-1} \ldots a_0 \underbrace{00\ldots0}_{j \text{ ori}}$.

<u>Înmulţirea numărului *a* cu o cifră a numărului *c*</u>

Fie $a = a_m a_{m-1} \ldots a_1 a_{0(b)} = a_0 + a_1 b + a_2 b^2 + \ldots + a_m b^m$ şi c_j o cifră oarecare a numărului c.

Deoarece $a_i < b$ pentru orice $i = 0, \ldots, m$ şi $c_j < b$, rezultă că $a_i \cdot c_j < b^2$. Din Teorema împărţirii cu rest rezultă că există câtul unic $q(a_i, c_j)$ şi restul unic $r(a_i, c_j)$ astfel încât:

$$a_i c_j = b \cdot q(a_i, c_j) + r(a_i, c_j),$$

unde $0 \le r(a_i, c_j) < b$ şi $0 \le q(a_i, c_j) < b$.

Deci $a \cdot c_j = \sum_{i=0}^{m} a_i \cdot c_j \cdot b^i = \sum_{i=0}^{m} (b \cdot q(a_i, c_j) + r(a_i, c_j)) \cdot b^i = \sum_{i=0}^{m} q(a_i, c_j) b^{i+1} + \sum_{i=0}^{m} r(a_i, c_j) b^i$.

Exemple

```
   101011(2) ·          123(5)·          123(8)·
       101(2)            24(5)            24(8)
 ---------------        -------          -------
    101011              1102              514
  101011                301              246
----------------        -------          -------
  11010111(2)           4112(5)          3174(8)
```

Efectuaţi următoarele înmulţiri:

$101011_{(2)}$ - $1111_{(2)}$

$AB56_{(16)}$ - $A9FF_{(16)}$

$1230_{(5)}$ - $221_{(5)}$.

Observaţie Împărţirea a două numere în baza *b* constă în operaţii deja studiate: adunări, scăderi şi înmulţiri.

Exemplu

```
2144(5) | 12(5)
12      |132
 44
 41
  34
  24
  10
```

Deci $2144_{(5)} = 12_{(5)} \cdot 132_{(5)} + 10_{(5)}$.

```
1100101(2) | 101(2)
101        |10100
 10
  0
 101
 101
   0
   0
    1
    0
     1
```

Deci $1100101_{(2)} = 101_{(5)} \cdot 10100_{(2)} + 1_{(2)}$.

Efectuaţi următoarele împărţiri:

$101011_{(2)} : 11_{(2)}$

$1230_{(5)} : 21_{(5)}$

$A085_{(16)} : 1B_{(16)}$.

M1.U2.7. Rezumat

Sistemul de numeraţie pe care îl folosim în marea majoritate a timpului este cel zecimal, însă uneori este necesar să efectuăm calcule şi în alte baze. În acestă unitate de învăţare am arătat cum se efectuează adunările, scăderile, înmulţirile şi împărţirile într-o bază oarecare *b*, *b*>1. De asemenea, am văzut cum pot fi comparate două numere scrise în baza *b*, *b*>1. Ceea ce a putut fi observat cu uşurinţă este faptul că, indiferent de baza în care lucrăm, aplicăm acelaşi algoritm de adunare, de scădere, de înmulţire sau de împărţire.

M1.U2.8. Test de evaluare a cunoştinţelor

Efectuaţi următoarele operaţii:

$110111_{(2)} + 1011_{(2)}$

$110111_{(2)} - 1011_{(2)}$

$110111_{(2)} * 101_{(2)}$

$110111_{(2)} : 11_{(2)}$

$A1B5F_{(16)} + A9F22F_{(16)}$

$A1B5F_{(16)} – BCD1_{(16)}$

$A1B_{(16)} * A2F_{(16)}$

$A1B5F_{(16)} : 1A2_{(16)}$.

Unitatea de învăţare M1.U3. Schimbarea bazei

Cuprins

M1.U3.1. Introducere

Atunci când se lucrează cu mai multe sisteme de numeraţie, apare şi necesitatea de a efectua conversii între diferite baze.

Fie N un număr în baza b, care poate fi dat în oricare din următoarele trei forme echivalente:

$$N = \underbrace{a_n a_{n-1} \dots a_1 a_0}_{\text{partea intreagă}} . \underbrace{a_{-1} a_{-2} \dots a_{-m}}_{\text{partea fractională}}$$

$$N = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b + a_0 + a_{-1} b^{-1} + \dots + a_{-m} b^{-m}$$

$$N = \sum_{i=-m}^{n} a_i b^i$$

Spunem că a_n este *cea mai semnificativă cifră* a numărului N şi că a_{-m} este *cea mai puţin semnificativă cifră* a numărului N.

Observaţie Dacă $m = 0$ atunci N este un număr întreg.

Fie N un număr în baza b, $N = a_n b^n + a_{n-1} b^{n-1} + \dots + a_1 b + a_0 + a_{-1} b^{-1} + \dots + a_{-m} b^{-m}$.
Îl vom transforma pe N din baza b în baza h. Există mai multe metode de efectuare a acestei transformări:

1. Metoda substituţiei cu calcule în baza h
2. Metoda substituţiei cu calcule în baza b
3. Metoda substituţiei cu calcule într-o bază intermediară g.

M1.U3.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a capacităţii de a efectua conversii între diferite baze.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să treacă un număr natural dintr-o bază în alta;
- Să treacă un număr raţional dintr-o bază în alta.

Durata medie de parcurgere a acestei unităţi de învăţare este de 3 ore.

M1.U3.3. Metoda substituţiei cu calculele în noua bază

Această metodă constă în scrierea fiecărei cifre a_i a numărului N şi a bazei b în noua bază h. Toate calculele vor fi făcute în noua bază, h.

Vom efectua următoarele conversii:

$a_{i(b)} = c_{i(h)}, \forall i, -m \le i \le n$

$10_{(b)} = x_{(h)}$.

Deci, $N_{(b)} = (c_n x^n + c_{n-1} x^{n-1} + ... + c_1 x + c_0 + c_{-1} x^{-1} + ... + c_{-m} x^{-m})_{(h)}$.

Exemplu

Pentru trecerea numărului $234_{(5)}$ din baza 5 în baza 2, vom scrie fiecare cifră a numărului şi baza 5 în baza 2:

$2_{(5)} = 10_{(2)}$, $3_{(5)} = 11_{(2)}$, $4_{(5)} = 100_{(2)}$, $10_{(5)} = 101_{(2)}$.

Deci, $234_{(5)} = (2*5^2 + 3*5 + 4)_{(5)} = (10*101^2 + 11*101 + 100)_{(2)} = (10*11001+1111+100)_{(2)} = (110010+1111+100)_{(2)} = 1000101_{(2)}$.

Caz particular

Dacă $b = h^k$ fiecare cifră a numărului scris în baza b va transformată într-un număr în baza h cu k cifre. Apoi aceste numere în baza h sunt juxtapuse.

Exemplu

Pentru trecerea numărului $5436_{(8)}$ din baza 8 în baza 2, vom scrie fiecare cifră a numărului dat în baza 2 pe 3 poziţii, deoarece $8 = 2^3$:

$5_{(8)} = 101_{(2)}$, $4_{(8)} = 100_{(2)}$, $3_{(8)} = 011_{(2)}$, $6_{(8)} = 110_{(2)}$.

Deci, $5436_{(8)} = 101100011110_{(2)}$.

Exemplu

Pentru trecerea numărului A5F.61$_{(16)}$ din baza 16 în baza 2, vom scrie fiecare cifră a numărului hexazecimal în baza 2 pe 4 poziţii, iar apoi le vom concatena.

A$_{(16)}$ = 1010$_{(2)}$, 5$_{(16)}$ = 0101$_{(2)}$, F$_{(16)}$ = 1111$_{(2)}$, 6$_{(16)}$ = 0110$_{(2)}$, 1$_{(16)}$ = 0001$_{(2)}$.

Deci, A5F.61$_{(16)}$ = 101001011111.01100001$_{(2)}$.

Efectuaţi următoarele conversii:

101011$_{(2)}$ = ?$_{(10)}$

1230$_{(5)}$ = ? $_{(2)}$

A085$_{(16)}$ = ? $_{(4)}$

A085$_{(16)}$ = ? $_{(2)}$.

M1.U3.4. Metoda substituţiei cu calculele în vechea bază (metoda împărţirii/înmulţirii bazei)

Metoda substituţiei cu calculele în vechea bază se mai numeşte şi *metoda împărţirii/înmulţirii bazei*. Această metodă este folosită de calculatoare pentru afişarea rezultatelor, baza iniţială fiind 2, iar noua bază fiind 10, de obicei.

Fie $N_{(b)} = a_n a_{n-1}...a_1 a_0.a_{-1}...a_{-m(b)} = [N]_{(b)} + \{N\}_{(b)}$,

unde $[N]_{(b)}$ este partea întreagă a lui $N_{(b)}$, iar $\{N\}_{(b)}$ este partea sa fracţionară.

Pentru transformarea părţii întregi $[N]_{(b)}$ din baza b în baza h, vom folosi Teorema sistemelor de numeraţie. Partea întreagă a lui $N_{(b)}$, $[N]_{(b)}$, poate fi scrisă în forma:

$$[N]_{(b)} = c_{n'}h^{n'} + c_{n'-1}h^{n'-1} + ... + c_1 h + c_0,$$

unde nu se cunosc cifrele $c_{n'}$, $c_{n'-1}$, ...,c_1, c_0, dar pot fi determinate prin împărţiri succesive la h.

$[N]_{(b)} = (\underbrace{c_{n'}h^{n'-1} + c_{n'-1}h^{n'-2} + ... + c_1}_{[N_1]_{(b)}})h + c_0$. De aici putem determina c_0, care este restul într-o împărţire a două numere întregi,

$[N_1]_{(b)} = (\underbrace{c_{n'}h^{n'-2} + c_{n'-1}h^{n'-3} + ... + c_2}_{[N_2]_{(b)}})h + c_1$, deci putem determina c_1,

.....

$[N_{n'}]_{(b)} = c_{n'}$, deci putem determina $c_{n'}$.

În continuare vom trece partea fracţională $\{N\}_{(b)}$ a lui N din baza b în baza h. Putem scrie $\{N\}_{(b)}$ în forma

$$\{N\}_{(b)} = c_{-1}h^{-1} + c_{-2}h^{-2} + ... + c_{-m'}h^{-m'},$$

unde nu se cunosc cifrele c_{-1}, c_{-2},...,$c_{-m'}$, dar pot fi determinate prin înmulţiri succesive cu h.

$\{N\}_{(b)}h = c_{-1} + \underbrace{c_{-2}h^{-1} + c_{-3}h^{-2} + ... + c_{-m'}h^{-m'+1}}_{\{N_1\}_{(b)}}$, deci putem determina c_{-1},

$\{N_1\}_{(b)}h = c_{-2} + \underbrace{c_{-3}h^{-1} + ... + c_{-m'}h^{-m'+2}}_{\{N_2\}_{(b)}}$, deci putem determina c_{-2},

....

$\{N_{m'-1}\}_{(b)}h = c_{-m'}$, deci putem determina $c_{-m'}$.

Deci, am obţinut că $N_{(b)} = a_n a_{n-1} \ldots a_1 a_0 . a_{-1} \ldots a_{-m(b)} = c_{n'} c_{n'-1} \ldots c_1 c_0 . c_{-1} \ldots c_{-m'(h)}$.

Exemplu
Trecerea din baza 10 în baza 8 a numărului $234.128_{(10)}$, efectuând calculele în vechea bază, adică în baza 10.
Pentru transformarea părţii întregi, efectuăm împărţiri succesive la 8 şi reţinem resturile în ordinea inversă ordinii în care au fost obţinute:

$234 = 29 * 8 + 2$, rezultă că $c_0 = 2$.
$29 = 3 * 8 + 5$, deci $c_1 = 5$.
$3 = 0 * 8 + 3$, deci $c_2 = 3$.

Deci partea întreagă a numărului dat, $234_{(10)}$, este $352_{(8)}$.

Pentru transformarea părţii fracţionare, efectuăm înmulţiri succesive cu 8 şi reţinem părţile întregi ale produselor în ordinea în care au fost obţinute:

$0.128 * 8 = 1.024$, rezultă că $c_{-1} = 1$.
$0.024 * 8 = 0.192$, deci $c_{-2} = 0$.
$0.192 * 8 = 1.536$, deci $c_{-3} = 1$.

Deci, am obţinut că, $0.128_{(10)} = 0.101_{(8)}$ şi $234.128_{(10)} = 352.101_{(8)}$.

Exemplu
Trecerea din baza 5 în baza 2 a numărului $234.21_{(5)}$, efectuând calculele în vechea bază, adică în baza 5.
Pentru transformarea părţii întregi, efectuăm împărţiri succesive la 2, efectuând calculele în baza 5, şi reţinem resturile în ordinea inversă ordinii în care au fost obţinute:

$234_{(5)} = 114_{(5)} * 2_{(5)} + 1_{(5)}$, rezultă că $c_0 = 1$.
$114_{(5)} = 32_{(5)} * 2_{(5)} + 0_{(5)}$, deci $c_1 = 0$.
$32_{(5)} = 13_{(5)} * 2_{(5)} + 1_{(5)}$, deci $c_2 = 1$.
$13_{(5)} = 4_{(5)} * 2_{(5)} + 0_{(5)}$, rezultă că $c_3 = 0$.
$4_{(5)} = 2_{(5)} * 2_{(5)} + 0_{(5)}$, deci $c_4 = 0$.
$2_{(5)} = 1_{(5)} * 2_{(5)} + 0_{(5)}$, deci $c_5 = 0$.
$1_{(5)} = 0_{(5)} * 2_{(5)} + 1_{(5)}$, deci $c_6 = 1$.

Deci partea întreagă a numărului dat, $234_{(5)}$, este $1000101_{(2)}$.

Pentru transformarea părţii fracţionare, efectuăm înmulţiri succesive cu 2, efectuând calculele în baza 5, şi reţinem părţile întregi ale produselor în ordinea în care au fost obţinute:

$0.21_{(5)} * 2_{(5)} = 0.42_{(5)}$, rezultă că $c_{-1} = 0$.
$0.42_{(5)} * 2_{(5)} = 1.34_{(5)}$, rezultă că $c_{-2} = 1$.
$0.34_{(5)} * 2_{(5)} = 1.23_{(5)}$, rezultă că $c_{-3} = 1$.

Deci, am obţinut că, $0.21_{(5)} = 0.011_{(2)}$ şi $234.21_{(5)} = 1000101.011_{(2)}$.

Caz particular

Dacă $h = b^k$ atunci, pornind de la virgulă către stânga şi către dreapta (dacă numărul are şi parte fracţionară), se formează grupe de câte k cifre. Dacă este necesar, se adaugă zerouri nesemnificative la partea fracţionară astfel încât grupele din dreapta virgulei să aibă toate exact k cifre. Fiecare grup se transformă apoi într-o cifră în baza h. În final, juxtapunem ciferele obţinute în baza h.

Exemple

1) Pentru transformarea numărului binar $1101110111001_{(2)}$ într-un număr în baza 8, vom grupa biţii numărului binar câte 3 de la dreapta la stânga deoarece numărul este întreg, iar $8 = 2^3$:

$$\underbrace{1}\underbrace{101}\underbrace{110}\underbrace{111}\underbrace{001} = \underbrace{001}\underbrace{101}\underbrace{110}\underbrace{111}\underbrace{001}.$$

Apoi transformăm fiecare grup de 3 biţi în cifra corespunzătoare lor din baza 8:

$001_{(2)} = 1_{(8)}$, $101_{(2)} = 5_{(8)}$, $110_{(2)} = 6_{(8)}$, $111_{(2)} = 7_{(8)}$.

Deci, $1101110111001_{(2)} = 15671_{(8)}$.

2) Pentru transformarea numărului binar $10011111.11001_{(2)}$ într-un număr în baza 16, vom grupa biţii numărului binar câte 4 pornind de la punct către stânga şi către dreapta deoarece $16 = 2^4$:

$$\underbrace{1001}\underbrace{1111}.\underbrace{1100}\underbrace{1} = \underbrace{1001}\underbrace{1111}.\underbrace{1100}\underbrace{1000}$$

Apoi transformăm fiecare grup de 4 biţi într-o cifră din baza 16:

$1001_{(2)} = 9_{(16)}$, $1111_{(2)} = F_{(16)}$, $1101_{(2)} = D_{(16)}$, $1000_{(2)} = 8_{(16)}$.

deci, $10011111.11001_{(2)} = 9F.D8_{(16)}$.

M1.U3.5. Metoda substituţiei cu calculele într-o bază intermediară

În general, această metodă este folosită atunci când, folosind creionul şi hârtia, trebuie să transformăm un număr din baza b în baza h, cu $b, h \neq 10$. Deoarece suntem familiarizaţi cu aritmetica în baza 10, preferăm să facem toate calculele în baza 10. Deci, vom folosi baza 10 ca bază intermediară.

Metoda substituţiei cu calcule într-o bază intermediară constă în:

1. Transformarea numărului din baza b în baza g, efectuând calculele în baza g, urmată de
2. Transformarea numărului obţinut din baza g în baza h, efectuând calculele în baza g.

Exemplu

Pentru a transforma numărul $111442_{(5)}$ în baza 8, vom folosi baza 10 ca bază intermediară. Mai întâi transformăm numărul $111442_{(5)}$ în baza 10 astfel:

$$111442_{(5)} = (1*5^5 + 1*5^4 + 1*5^3 + 4*5^2 + 4*5^1 + 2*5^0)_{(10)} = 3997_{(10)}.$$

Apoi, transformăm numărul $3997_{(10)}$ în baza 8:

$3997 = 499*8 + 5$
$499 = 62*8 + 3$
$62 = 7*8 + 6$
$7 = 0*8 + 7$

Deci, $3997_{(10)} = 7635_{(8)}$ şi $111442_{(5)} = 7635_{(8)}$.

Exemplu

Pentru a transforma numărul $208.7_{(9)}$ în baza 7, vom folosi baza 10 ca bază intermediară. Mai întâi transformăm numărul $208.7_{(9)}$ în baza 10 astfel:

$$208.7_{(9)} = (2*9^2 + 0*9^1 + 8*9^0 + 7*9^{-1})_{(10)} = 170.077_{(10)}.$$

Apoi, transformăm numărul $170.077_{(10)}$ în baza 7:

Pentru transformarea părţii întregi, efectuăm împărţiri succesive la 7, efectuând calculele în baza 10, şi reţinem resturile în ordinea inversă ordinii în care au fost obţinute:

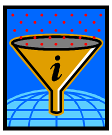
$170_{(10)} = 24_{(10)}* 7_{(10)} + 2_{(10)}$, rezultă că $c_0 = 2$.
$24_{(10)} = 3_{(10)}*7_{(10)} + 3_{(10)}$, deci $c_1 = 3$.
$3_{(10)} = 0_{(10)}* 7_{(10)} + 3_{(10)}$, deci $c_2 = 3$.

Deci partea întreagă a numărului, $170_{(10)}$, este $332_{(7)}$.

Pentru transformarea părţii fracţionare, efectuăm înmulţiri succesive cu 7, efectuând calculele în baza 10, şi reţinem părţile întregi ale produselor în ordinea în care au fost obţinute:

$0.077_{(10)}*7_{(10)} = 0.539_{(10)}$, rezultă că $c_{-1} = 0$.
$0.539_{(10)}*7_{(10)} = 3.773_{(10)}$, rezultă că $c_{-2} = 3$.
$0.773_{(10)}*7_{(10)} = 5.511_{(10)}$, rezultă că $c_{-3} = 5$.

Deci, am obţinut că, $0.077_{(10)} = 0.035_{(7)}$ şi $170.077_{(10)} = 332.035_{(7)}$. În consecinţă, $208.7_{(9)} = 332.035_{(7)}$.

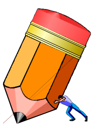


M1.U3.6. Rezumat

Atunci când lucrăm în mai multe sisteme de numeraţie, apare şi necesitatea de a efectua conversii între diferite baze. În acestă unitate de învăţare, am prezentat trei metode generale de trecere a unui număr oarecare $N = a_n a_{n-1}... a_1 a_0.a_{-1}... a_{-m}$ din baza b în baza h:

1. Metoda substituţiei cu calcule în baza h
2. Metoda substituţiei cu calcule în baza b
3. Metoda substituţiei cu calcule într-o bază intermediară g.

De asemenea, am arătat că cele mai rapide conversii se pot face între baze care sunt puteri ale aceluiaşi număr.



M1.U3.7. Test de evaluare a cunoştinţelor

Efectuaţi următoarele conversii:

$1485.4_{(10)} = ?_{(2)}$

$110111_{(2)} = ?_{(10)}$

$110111_{(2)} = ?_{(8)}$

$110111_{(2)} = ?_{(16)}$

$A1B5F_{(16)} = ?_{(10)}$

$A1B_{(16)} = ?_{(5)}$

$A1B_{(16)} = ?_{(4)}$

$10101.1_{(2)} = ?_{(8)}$

$10101.1_{(2)} = ?_{(16)}$

$1230_{(5)} = ?_{(2)}$

$A085_{(16)} = ?_{(8)}$.

Modulul 2. Reprezentarea datelor

Cuprins

Introducere

Folosirea calculatorului implică şi utilizarea unor modalităţi de reprezentare a informaţiilor în memoria lui ca şiruri de biţi. Orice informatician trebuie să cunoască cum sunt reprezentate informaţiile numerice şi alfa-numerice în memorie. De asemenea, este uneori necesar să poată interpreta conţinutul locaţiilor de memorie. Asupra fiecărui tip de date sunt permise anumite operaţii. Vom prezenta modul în care se efectuează adunările, scăderile şi înmulţirile numerelor întregi reprezentate în cod complementar.

Competenţe

La sfârşitul acestui modul studenţii vor fi capabili să:

- Determine reprezentarea în memoria calculatorului a numerelor întregi, raţionale şi a caracterelor;
- Determine ce informaţie este memorată într-o locaţie de memorie, cunoscând tipul informaţiei (număr întreg, raţional sau caracter);
- Să efectueze operaţii aritmetice cu numere întregi reprezentate în cod complementar.

Unitatea de învăţare M2.U1. Reprezentarea numerelor întregi

Cuprins

M2.U1.1. Introducere

Se ştie că în memoria calculatoarelor informaţiile sunt reprezentate ca succesiuni de 0 şi 1. Deseori informaţiile sunt numere întregi, a căror reprezentare va fi studiată în această unitate de învăţare. Cele mai cunoscute 3 modalităţi de reprezentare a numerelor întregi sunt:

1. Codul direct
2. Codul invers
3. Codul complementar

Codul direct este cel mai „apropiat" de modul în care suntem obişnuiţi să scriem numerele, însă cel mai eficient este codul complementar, care este folosit pentru reprezentarea numerelor întregi în memoria calculatoarelor.

M2.U1.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a celor mai cunoscute 3 modalităţi de reprezentare a numerelor întregi.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Reprezinte numere întregi în cod direct;
- Reprezinte numere întregi în cod invers;
- Reprezinte numere întregi în cod complementar.

Durata medie de parcurgere a primei unităţi de învăţare este de 2 ore.

M1.U1.3. Codul direct

În reprezentarea în *cod direct* pe *n* biţi a unui număr întreg, cel mai semnificativ bit este rezervat pentru semn, iar ceilalţi *n*-1 biţi conţin reprezentarea în baza 2 a modulului numărului. Pentru numerele pozitive bitul de semn este 0, iar pentru cele negative bitul de semn este 1.

Exemple

1. Reprezentarea numărului 12 în cod direct pe 8 biţi este

$$\underbrace{0}_{\text{Semn}}\ \underbrace{0001100}_{\text{Mărime}}.$$

2. Reprezentarea numărului -12 în cod direct pe 8 biţi este

$$\underbrace{1}_{\text{Semn}}\ \underbrace{0001100}_{\text{Mărime}}.$$

Se observă cu uşurinţă că singura deosebire între codul direct al lui 12 şi cel al lui -12 este bitul de semn.

Observaţii 1) Reprezentarea în cod direct a unui număr negativ se obţine din reprezentarea modulului numărului prin schimbarea bitului de semn din 0 în 1.
2) Un dezavantaj al codului direct este faptul că +0 and -0 au două reprezentări diferite, deşi sunt unul şi acelaşi număr: 0. De exemplu, codul direct pe 8 biţi al lui +0 este 00000000, iar al lui -0 este 10000000.

În consecinţă, pe *n* biţi pot fi reprezentate în cod direct numai 2^n-1 numere întregi diferite (de la $-2^{n-1}+1$ la $2^{n-1}-1$) chiar dacă există 2^n şiruri diferite de *n* biţi. De exemplu, pe 8 biţi se pot reprezenta folosind codul direct 255 de numere întregi: de la – 127 la 127, iar pe 16 biţi se pot reprezenta 65535 de numere întregi: de la – 32767 la 32767.

Determinaţi codurile directe pe 8 biţi ale următoarelor numere: -127, -56, -3, 3, 5, 111, 127.

Până acum am văzut cum determinăm codul direct al unui număr întreg dat. Acum vom studia problem inversă: Presupunem că cunoaştem un cod şi vrem să aflăm numărul al cărui cod direct se dă.

1. Determinăm numărul binar obţinut din codul direct prin eliminarea bitului de semn, îl convertim în baza 10 şi obţinem modulul numărului căutat. Fie acesta *m*.
2. Dacă bitul de semn este 0 atunci numărul căutat este chiar *m*, iar dacă bitul de semn este 1 atunci numărul este -*m*.

Exemple

- Determinăm numărul întreg al cărui cod direct este 00010101.
  Deoarece bitul de semn este 0, înseamnă că este un număr pozitiv, care este echivalentul zecimal al numărului binar 0010101. Acesta este 21.
- Determinăm numărul întreg al cărui cod direct este 10010101.
  Deoarece bitul de semn este 1, înseamnă că este un număr negativ, al cărui modul este echivalentul zecimal al numărului binar 0010101, adică 21. Deci, numărul căutat este -21.

M2.U1.4. Codul invers

Ca şi în reprezentarea în cod direct, în reprezentarea în *cod invers* pe *n* biţi a unui număr întreg, cel mai semnificativ bit este rezervat pentru semn. Pentru numerele pozitive bitul de semn este 0, iar pentru cele negative bitul de semn este 1. Pentru numerele pozitive, ceilalţi *n*-1 biţi conţin reprezentarea lor în binar. Codul invers al unui număr negativ se obţine din codul invers al modulului său prin înlocuirea biţilor 0 cu 1 şi a biţilor 1 cu 0. Această operaţie se numeşte *complementarea biţilor*.

Exemple

1. Reprezentarea numărului 12 în cod invers pe 8 biţi este 00001100.
2. Reprezentarea numărului -12 în cod invers pe 8 biţi este 11110011.

Observaţii 1) Pentru numere întregi pozitive, reprezentarea în cod invers coincide cu cea în cod direct.
2) Un dezavantaj al reprezentării în cod invers este, ca şi în cazul codului direct, faptul că +0 and -0 au două reprezentări diferite, deşi sunt unul şi acelaşi număr: 0. De exemplu, codul invers pe 8 biţi al lui +0 este 00000000, iar al lui -0 este 11111111.

În consecinţă, pe *n* biţi pot fi reprezentate în cod invers, ca şi în cod direct, numai 2^n-1 numere întregi diferite (de la $-2^{n-1}+1$ la $2^{n-1}-1$) chiar dacă există 2^n şiruri diferite de *n* biţi. De exemplu, pe 8 biţi se pot reprezenta folosind codul invers 255 de numere întregi: de la – 127 la 127, iar pe 16 biţi se pot reprezenta 65535 de numere întregi: de la – 32767 la 32767.

Reprezentarea în cod invers nu este folosită în mod obişnuit decât ca pas intermediar în determinarea codului complementar.

Determinaţi codurile inverse pe 8 biţi ale următoarelor numere: -127, -56, -3, 3, 5, 111, 127.

Până acum ne-am pus problema determinării codului invers al unui număr întreg dat. Acum vom studia problem inversă: Presupunem că ştim un cod şi vrem să aflăm numărul al cărui cod invers se cunoaşte.

1. Dacă este numărul reprezentat este pozitiv (adică dacă bitul de semn este 0), atunci convertim codul invers în baza 10 şi obţinem numărul.
2. Dacă este numărul reprezentat este negativ (adică dacă bitul de semn este 1), atunci aplicăm aceeaşi metodă pe care am aplicat-o pentru determinarea codului invers: înlocuim toţi biţii 0 cu 1 şi toţi biţii 1 cu 0. Am obţinut acum codul unui număr pozitiv, care este modulul numărului negativ pe care-l căutăm. Acesta se obţine prin convertirea codului în baza 10.

Exemple

- Determinăm numărul întreg al cărui cod invers este 00010101.
  Deoarece bitul de semn este 0, înseamnă că este un număr pozitiv, care este echivalentul zecimal al numărului binar 0010101. Acesta este 21.
- Determinăm numărul întreg al cărui cod invers este 10010101.
  Deoarece bitul de semn este 1, înseamnă că este un număr negativ, al cărui modul este echivalentul zecimal al numărului binar 01101010 obţinut prin inversarea biţilor. Acesta este 104. Deci, numărul căutat este -104.

M2.U1.5. Codul complementar

Ca şi în codurile direct şi invers, în reprezentarea în *cod complementar* pe *n* biţi a unui număr întreg, cel mai semnificativ bit este rezervat pentru semn. Pentru numerele pozitive bitul de semn este 0, iar pentru cele negative bitul de semn este 1. Pentru numerele pozitive, ceilalţi *n*-1 biţi conţin reprezentarea lor în binar. Codul complementar al unui număr negativ se obţine din său codul invers la care se adună 1. Ultimii *n* biţi ai acestei sume reprezintă codul complementar al numărului negativ dat.

Majoritatea calculatoarele lucrează cu numere reprezentate în cod complementar.

Exemple

1. Reprezentarea numărului 12 în cod complemntar pe 8 biţi este 00001100.
2. Reprezentarea numărului -12 în cod complementar pe 8 biţi este

```
11110011+   // codul invers al lui -12
       1
--------
11110100    // codul complementar al lui -12.
```

Observaţii 1) Pentru numere întregi pozitive, reprezentările în cod direct, invers şi complementar coincid.
2) În cod complementar +0 şi -0 au aceeaşi reprezentare. De exemplu, codul complementar pe 8 biţi al lui +0 este 00000000, iar al lui -0 este format din ultimii 8 biţi ai sumei

11111111+

1

100000000

Adică, codul complementar al lui -0 este 00000000 ca şi cel al lui +0.

În consecinţă, pe n biţi pot fi reprezentate în cod complementar 2^n numere întregi diferite: de la -2^{n-1} la 2^{n-1}-1. De exemplu, pe 8 biţi se pot reprezenta folosind codul invers 256 de numere întregi: de la –128 la 127, iar pe 16 biţi se pot reprezenta 65536 de numere întregi: de la –32768 la 32767. Adică, datorită unicei reprezentări pentru +0 şi -0, se câştigă încă un număr negativ, -2^{n-1}, care poate fi reprezentat în cod complementar pe n biţi, dar care nu putea fi reprezentat nici în cod direct, nici în cod invers pe n biţi.

O altă metodă de determinare a codului complementar al unui număr negativ este următoarea:

1. Se determină codul complementar al modulului numărului.
2. Pornind de la dreapta spre stânga:
   - Toţi biţii 0 rămân neschimbaţi până la întâlnirea primului 1
   - Primul bit 1 rămâne neschimbat
   - Toţi ceilalţi biţi vor fi schimbaţi: biţii 1 vor deveni 0, iar biţii 0 vor deveni 1.

Până acum ne-am pus problema determinării codului complementar al unui număr întreg dat. Acum vom studia problem inversă: Presupunem că se cunosc cei n biţi prin care este memorat un anumit număr întreg şi vrem să aflăm numărul.

1. Dacă este numărul reprezentat este pozitiv (adică dacă bitul de semn este 0), atunci convertim codul complementar în baza 10 şi obţinem numărul.
2. Dacă este numărul reprezentat este negativ (adică dacă bitul de semn este 1), atunci aplicăm aceeaşi metodă pe care am aplicat-o pentru determinarea codului complementar: mai întâi înlocuim toţi biţii 0 cu 1 şi toţi biţii 1 cu 0, apoi adunăm 1 şi păstrăm ultimii n biţi ai sumei. Aceştia trebuie să conţină reprezentarea în cod complementar a unui număr pozitiv, care se se obţine prin convertirea codului în baza 10 şi care reprezintă modulul numărului negativ pe care-l căutăm.

Exemple

- Determinăm numărul întreg al cărui cod complementar este 00010101.

  Deoarece bitul de semn este 0, înseamnă că este un număr pozitiv, care este echivalentul zecimal al numărului binar 00010101. Acesta este 21.

- Determinăm numărul întreg al cărui cod complementar este 10010101.

  Deoarece bitul de semn este 1, înseamnă că este un număr negativ. Inversăm biţii şi obţinem 01101010. Apoi adunăm 1 şi obţinem 01101011. Acesta este codul complementar al unui număr pozitiv, care este modulul numărului pe care îl căutăm. Deoarece $01101011_{(2)} = 107_{(10)}$, rezultă că numărul căutat este -107.

Determinaţi codurile complementare pe 8 biţi ale următoarelor numere: -127, -56, -3, 3, 5, 111, 127.

M2.U1.6. Rezumat

Cele mai cunoscute modalităţi de reprezentare a numerelor întregi sunt: codul direct, codul invers şi codul complementar.

Majoritatea calculatoarele lucrează cu numere reprezentate în cod complementar.

În toate cele 3 coduri cel mai semnificativ bit este folosit pentru semn: 0 pentru numere pozitive şi 1 pentru cele negative.

Pentru numere întregi pozitive, reprezentările în cod direct, invers şi complementar coincid şi sunt egale cu reprezentarea numerelor în binar.

Codul complementar al unui număr negativ se obţine din său codul invers la care se adună 1. Ultimii *n* biţi ai acestei sume reprezintă codul complementar al numărului negativ dat.

M2.U1.7. Test de evaluare a cunoştinţelor

1. Determinaţi codurile directe pe 8 biţi ale următoarelor numere:
   11, 12, +0, -0, -111.
2. Determinaţi codurile inverse pe 8 biţi ale următoarelor numere:
   11, 12, +0, -0, -111
3. Determinaţi codurile complementare pe 8 biţi ale următoarelor numere:
   11, 12, +0, -0, -111.

Unitatea de învăţare M2.U2. Operaţii aritmetice cu numere întregi în cod complementar

Cuprins

M2.U2.1. Introducere

Majoritatea calculatoarele lucrează cu numere întregi reprezentate în cod complementar. Vom studia adunarea şi înmulţirea întregilor în cod complementar. Scăderea poate fi realizată prin adunarea descăzutului cu opusul scăzătorului.

M2.U2.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a modului în care calculatorul efectuează operaţii aritmetice cu numere întregi reprezentate în cod complementar.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să efectueze adunări, scăderi şi înmulţiri de numere întregi în cod complementar.

Durata medie de parcurgere a acestei unităţi de învăţare este de 3 ore.

M2.U2.3. Adunarea numerelor întregi în cod complementar

Majoritatea calculatoarelor utilizează numerele întregi reprezentate în cod complementar pentru că acest cod are multiple avantaje. Unul dintre ele a fost deja menţionat în unitatea de învăţare anterioară: unica reprezentare a lui 0. Alt avantaj este faptul că cele mai rapide circuite pot fi folosite dacă numerele sunt reprezentate în cod complementar. Circuitul pentru scădere poate fi eliminat, deoarece scăderea poate fi efectuată cu ajutorul circuitului destinat adunării, cu o mică modificare.

Circuitul destinat adunării, numit şi *sumator*, adună două numere în cod complementar. Transportul către bitul de semn şi transportul de la bitul de semn sunt comparate. Dacă aceste ultime două transporturi sunt egale atunci suma (fără transportul de la bitul de semn) este corectă. În caz contrar, rezultatul este incorect şi programatorul va primi un mesaj de eroare (depăşire).

Exemplu

Vom arăta cum se adună 2 numere pozitive reprezentate în cod complementar pe 8 biţi.

$[22]_c$ = 00010110+
$[34]_c$ = 00100010
            00111000

Transportul către bitul de semn şi transportul de la bitul de semn sunt ambele 0, deci rezultatul corect. Deoarece bitul de semn al sumei este 0, înseamnă că s-a obţinut un număr pozitiv. Acesta este 56, deoarece $00111000_{(2)} = 56_{(10)}$.

Exemplu

Vom arăta cum se adună 2 numere negative reprezentate în cod complementar pe 8 biţi.

$[-22]_c$ = 11101010+
$[-34]_c$ = 11011110
            111001000

Transportul către bitul de semn şi transportul de la bitul de semn sunt ambele 1, deci rezultatul, care constă în ultimii 8 biţi ai sumei, este corect. Deoarece bitul de semn al sumei este 1, înseamnă că s-a obţinut un număr negativ. Valoarea modulului acestui număr negativ este 56, deoarece $00111000_{(2)} = 56_{(10)}$. Această ultimă transformare a fost făcută doar din motive didactice. Calculatoarele nu o mai efectuează deoarece numerele negative sunt memorate în cod complementar.

Exemplu

Vom arăta cum se adună un număr pozitiv cu unul negativ în cod complementar pe 8 biţi, suma fiind pozitivă.

$[-22]_c$ = 11101010+
$[34]_c$ = 00100010
            100001100

Transportul către bitul de semn şi transportul de la bitul de semn sunt ambele 1, deci rezultatul, care constă în ultimii 8 biţi ai sumei, este corect. Deoarece bitul de semn al sumei este 0, înseamnă că s-a obţinut un număr pozitiv. Valoarea acestui număr este 12, deoarece $00001100_{(2)} = 12_{(10)}$.

Exemplu

Vom arăta cum se adună un număr pozitiv cu unul negativ în cod complementar pe 8 biţi, suma fiind un număr negativ.

$[22]_c$ = 00010110+
$[-34]_c$ = $\underline{11011110}$
11110100

Transportul către bitul de semn şi transportul de la bitul de semn sunt ambele 0, deci rezultatul, care constă în ultimii 8 biţi ai sumei, este corect. Deoarece bitul de semn al sumei este 1, înseamnă că s-a obţinut un număr negativ. Valoarea acestui număr negativ este -12, deoarece $00001100_{(2)} = 12_{(10)}$. Această ultimă transformare a fost făcută doar din motive didactice. Calculatoarele nu o mai efectuează deoarece numerele negative sunt memorate în cod complementar.

Exemplu

Vom arăta cum apare o eroare de depăşire. Vom încerca să adunăm 73 cu 65 în cod complementar pe 8 biţi.

$[73]_c$ = 01001001+
$[65]_c$ = $\underline{01000001}$
10001010

Transportul către bitul de semn este 1, iar transportul de la bitul de semn este 0, ceea ce înseamnă că a avut loc o depăşire şi, ca urmare, rezultatul este incorect. Într-adevăr, bitul de semn al sumei este 1, ceea ce înseamnă că s-a obţinut un număr negativ, -108, care, evident nu poate fi rezultatul unei adunări corecte a două numere pozitive. Eroarea a apărut din cauză că suma, care era 138, nu este un număr care poate fi reprezentat pe 8 biţi în cod complementar.

Pentru scădere majoritatea calculatoarelor nu au un circuit special, deoarece scăderea poate fi efectuată cu ajutorul circuitului destinat adunării, cu o mică modificare.

Exemplu

Scăderea 22 – 33 poate fi transformată în adunarea 22 + (-33).

$[22]_c$ = 00010110+
$[-33]_c$ = $\underline{11011111}$
11110101

Transportul către bitul de semn şi transportul de la bitul de semn sunt ambele 0, deci rezultatul, care constă în ultimii 8 biţi, este corect. Deoarece bitul de semn al sumei este 1, înseamnă că s-a obţinut un număr negativ. Valoarea acestui număr negativ este -11, deoarece $00001011_{(2)} = 11_{(10)}$. Această ultimă transformare a fost făcută doar din motive didactice. Calculatoarele nu o mai efectuează deoarece numerele negative sunt memorate în cod complementar.

1. Efectuaţi următoarele adunări în cod complementar pe 8 biţi:

56+12

-127+51,

-3+ 3,

5 +(-111).

M2.U2.4. Înmulţirea numerelor întregi în cod complementar

Unele calculatoare efectuează înmulţirile prin adunări repretate. Dar, majoritatea calculatoarelor au circuite speciale pentru înmulţire. În general, se implementează diferite variante ale algoritmului clasic de înmulţire.

Algoritmul de înmulţire

Acest algoritm este asemănător cu algoritmul pe care îl folosim pentru înmulţirea a 2 numere întregi utilizând creionul şi hârtia: pentru fiecare cifră a înmulţitorului, se scrie o sumă parţială cu o poziţie la stânga, sub precedenta sumă parţială. Înmulţirea cu 1 înseamnă copierea deînmulţitului cu o poziţie la stânga, sub precedenta sumă parţială. Înmulţirea cu 0 înseamnă scrierea un şir de zerouri cu o poziţie la stânga, sub precedenta sumă parţială. În final, rezultatul se obţine adunând toate sumele parţiale.

Algoritmul foloseşte doi regiştri dubli. Un *registru* este un dispozitiv de stocare temporară a informaţiei, având, de obicei 16 biţi şi fiind accesat de către unitatea aritmetico-logică. Unul dintre registrii dubli este un registru de deplasare. Un *registru de deplasare* poate deplasa fiecare bit memorat cu o poziţie spre stânga, completând cel mai puţin semnificativ bit cu 0 şi eliminând cel mai semnificativ bit.

Algoritmul de înmulţire a două numere întregi pozitive reprezentate în cod complementar pe *n* biţi este următorul:

Algoritm inmultire;
begin
se pune deînmulţitul în registrul de deplasare SR;
se iniţializează cu 0 registrul dublu DR care va conţine rezultatul;
if cel mai puţin semnificativ bit al înmulţitorului este 1 **then**
DR = DR + SR;
for fiecare din cei *n*-1 biţi rămaşi ai înmulţitorului, mergând de la dreapta spre stânga **do**
begin
se deplasează deînmulţitul din SR cu o poziţie spre stânga;
if bitul curent al înmulţitorului este 1 **then**

DR = DR + SR;

end;

if cei mai semnificativi n+1 biţi ai registrului DR sunt 0 **then**

cei mai puţin semnificativi n biţi ai registrului DR conţin produsul corect;

else produsul nu poate fi reprezentat pe n biţi;

end.

Exemplu

Vom aplica algoritmul pentru a înmulţi 13 cu 6 în cod complementar pe 8 biţi.

$[13]_c$ = 00001101
$[6]_c$ = 00000110
Iniţial,
SR: 00000000 00001101
DR: 00000000 00000000

Cel mai puţin semnificativ bit al înmulţitorului este 0, deci la primul pas nu facem nimc.

Pentru bitul subliniat al înmulţitorului 000001$\underline{1}$0:
Se deplasează deînmulţitul din registrul SR cu o poziţie spre stânga:
SR: 00000000 00011010
Se adună conţinutul celor 2 regiştri, rezultatul păstrându-se în registrul DR:
DR: 00000000 00011010

Pentru bitul subliniat al înmulţitorului 00000$\underline{1}$10:
Se deplasează deînmulţitul din registrul SR cu o poziţie spre stânga:
SR: 00000000 00110100
Se adună conţinutul celor 2 regiştri, rezultatul păstrându-se în registrul DR:
DR: 00000000 01001110

Deoarece ceilalţi biţi ai înmulţitorului sunt 0, conţinutul registrul DR nu se va mai modifica. Deci, nu mai are rost să efectuăm ultimii 5 paşi ai algoritmului, care constau doar în deplasări ale deînmulţitului din registrul SR.

Deoarece cei mai semnificativi 9 biţi ai registrului DR sunt 0, înseamnă că cei mai puţin semnificativi 8 biţi ai lui DR conţin rezultatul corect, care este 78.

Observaţie Dacă unul sau ambii operanzi sunt negativi, atunci el sau ei pot fi înlocuiţi cu modulul lor înainte de începerea algoritmului, iar la sfârşitul algortimului rezultatul se înlocuieşte cu opusul său dacă este necesar.

Exemplu

Putem determina rezultatul înmulţirii lui -11 cu 9 în cod complementar pe 8 biţi aplicând algoritmul de înmulţire pentru 11 şi 9 şi determinând apoi opusul produsului furnizat de algoritm.

$[11]_c$ = 00001011
$[9]_c$ = 00001001
Iniţial,
SR: 00000000 00001011
DR: 00000000 00000000

Cel mai puţin semnificativ bit al înmulţitorului este 1, deci vom aduna SR la DR.

DR: 00000000 00001011

Pentru bitul subliniat al înmulţitorului 00001001:
Se deplasează deînmulţitul din registrul SR cu o poziţie spre stânga:
SR: 00000000 00010110

Pentru bitul subliniat al înmulţitorului 00001001:
Se deplasează deînmulţitul din registrul SR cu o poziţie spre stânga:
SR: 00000000 00101100

Pentru bitul subliniat al înmulţitorului 00001001:
Se deplasează deînmulţitul din registrul SR cu o poziţie spre stânga:
SR: 00000000 01011000
Se adună conţinutul celor 2 regiştri, rezultatul păstrându-se în registrul DR:

DR: 00000000 01100011

Deoarece ceilalţi biţi ai înmulţitorului sunt 0, conţinutul registrul DR nu se va mai modifica. Deci, nu mai are rost să efectuăm ultimii 4 paşi ai algoritmului, care constau doar în deplasări ale deînmulţitului din registrul SR.

Deoarece cei mai semnificativi 9 biţi ai registrului DR sunt 0, înseamnă că cei mai puţin semnificativi 8 biţi ai lui DR, 01100011, conţin rezultatul corect al înmulţirii lui 11 cu 9. Pentru a obţine produsul dintre -11 şi 9 vom determina opusul numărul furnizat de algoritm, care este 10011101, adică reprezentarea în cod complementar pe 8 biţi a lui -99.

Efectuaţi următoarele înmulţiri în cod complementar pe 8 biţi:

5*12

(-17)*4,

(-3)*(-19),

5 *(-11).

Algoritmul lui Booth

Se ştie că

$$2^m + 2^{m-1} + ... + 2^{n+1} + 2^n = 2^{m+1} – 2^n,$$

pentru orice $n, m \in \mathbf{N}, m > n$.

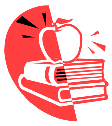
În consecinţă, pentru un şir de biţi 1 consecutivi ai înmulţitorului, în loc de un şir de adunări, se pot efectua doar o singură adunare şi o singură scădere, obţinându-se acelaşi rezultat.

Algoritmul lui Booth foloseşte 3 regiştri: *M*, *A* şi *Q*. Iniţial, *M* conţine deînmulţitul, toţi biţii registrului *A* sunt 0, iar *Q* conţine înmulţitorul.

Folosind metoda lui Booth, se poate implementa un circuit pentru înmulţire, care utilizează cei mai puţin semnificativi 2 biţi ai registrului *Q*. Dacă aceşti 2 biţi sunt:

0	0	atunci se efectuează o deplasare a lui *AQ*
0	1	atunci se adună *M* la *A* şi se efectuează o deplasare a lui *AQ*
1	0	atunci se scade *M* din *A* şi se efectuează o deplasare a lui *AQ*
1	1	atunci se efectuează o deplasare a lui *AQ*

În acest algoritm, o deplasare a lui *AQ* înseamnă deplasarea fiecărui bit cu o poziţie la dreapta, piederea celui mai puţin semnificativ bit şi păstrarea celui mai semnificativ bit.



Exemplu

Aplicăm algoritmul lui Booth pentru a înmulţi 15 cu 3 în cod complementar pe 8 biţi.

$[15]_c$ =00001111

$[3]_c$ = 00000011

M	*A*	*Q*	Q_{prec}	Operaţii
00001111	00000000 11110001	00000011	0	Scădere şi deplasare
	11111000	10000001	1	deplasare
	11111100 00001011	01000000	1	adunare şi deplasare
	00000101	10100000	0	deplasare
	00000010	11010000	0	deplasare
	00000001	01101000	0	deplasare
	00000000	10110100	0	deplasare
	00000000	01011010	0	deplasare
	00000000	00101101	0	

Toţi biţii din registrul *A* sunt zero şi cel mai semnificativ bit al lui *Q* este tot 0, deci *Q* conţine codul complementar al unui număr pozitiv, 45, care este produsul corect.

Algoritmul lui Booth este mai eficient decât algoritmul de înmulţire dacă unul dintre operanzi, care va fi înmulţitorul, conţine unul sau câteva şiruri de biţi 1 consecutivi.

Observaţie Algoritmul lui Booth se poate folosi şi dacă unul sau ambii operanzi sunt numere întregi negative.

Exemplu

Aplicăm algoritmul lui Booth pentru a înmulţi 41 cu -2 în cod complementar pe 8 biţi.

$[41]_c$ =00101001
$[-2]_c$ = 11111110

M	*A*	*Q*	Q_{prec}	Operaţii
00101001	00000000	11111110	0	deplasare
	00000000 11010111	01111111	0	scădere şi deplasare
	11101011	10111111	1	deplasare
	11110101	11011111	1	deplasare
	11111010	11101111	1	deplasare
	11111101	01110111	1	deplasare
	11111110	10111011	1	deplasare
	11111111	01011101	1	deplasare
	11111111	10101110	1	

Interpretarea rezultatului: toţi biţii din registrul *A* sunt 1 şi cel mai semnificativ bit al lui *Q* este tot 1, deci *Q* conţine codul complementar al unui număr negativ, reprezentabil pe 8 biţi. Modul acestui număr este echivalentul zecimal al numărului binar 01010010, care este 82.

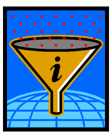
Efectuaţi următoarele înmulţiri în cod complementar pe 8 biţi, folosind algoritmul lui Booth:

5*15

(-17)*6,

(-3)*(31),

25*(-2).



M2.U2.5. Rezumat

Majoritatea calculatoarelor utilizează numerele întregi reprezentate în cod complementar datorită multiplelor avantaje pe care le are acest cod: 0 are o reprezentare unică, iar cele mai rapide circuite pot fi folosite dacă numerele sunt reprezentate în cod complementar.

Circuitul destinat adunării, numit şi *sumator*, adună două numere în cod complementar. Transportul către bitul de semn şi transportul de la bitul de semn sunt comparate. Dacă aceste ultime două transporturi sunt egale atunci suma (fără transportul de la bitul de semn) este corectă. În caz contrar, rezultatul este incorect şi programatorul va primi un mesaj de eroare (depăşire).

Circuitul pentru scădere poate fi eliminat, deoarece scăderea poate fi efectuată cu ajutorul circuitului destinat adunării, cu o mică modificare.

Înmulţirea numerelor întregi reprezentate în cod complementar se poate face folosind un algoritm de înmulţire asemănător celui pe care îl folosim când înmulţim numerele scriind cu creionul pe hârtie sau folosind algoritmul lui Booth. Acesta din urmă este eficient a fi aplicat atunci când unul dintre operanzi, care va fi înmulţitorul, conţine unul sau câteva şiruri de biţi 1 consecutivi.

M2.U2.6. Test de evaluare a cunoştinţelor

1. Adunaţi numerele:
   19 şi 62,
   45 şi -65,
   -111 şi -56
   în cod complementar pe 8 biţi.
2. Scădeţi numerele:
   94 şi 52,
   48 şi -5,
   -104 şi -12
   în cod complementar pe 8 biţi.
3. Înmulţiţi numerele:
   9 şi 11,
   5 şi 15,
   -12 şi 6
   în cod complementar pe 8 biţi.
   La sfârşitul fiecărei operaţii spuneţi dacă rezultatul este corect. Justificaţi-vă răspunsul.

Unitatea de învăţare M2.U3. Reprezentarea numerelor raţionale şi a caracterelor

Cuprins

M2.U3.1. Introducere

Calculatoarele lucrează cu numere reale aproximate prin numere raţionale. Acestea pot fi reprezentate în virgulă fixă sau în virgulă mobilă. În reprezentarea în virgulă fixă este nevoie un număr mare de biţi pentru a putea memora un interval rezonabil de numere. De aceea, este preferată reprezentarea în virgulă mobilă. Majoritatea calculatoarelor folosesc pentru reprezentarea numerelor reale în virgulă mobilă standardul IEEE 754.

În finalul acestei unităţi de învătare, vom prezenta câteva coduri folosite pentru reprezentarea caracterelor: ASCII, EBCDIC, Unicode.

M2.U3.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a capacităţii de a determina reprezentarea numerelor reale (raţionale) în virgulă mobilă în standardul IEEE 754. De asemenea, studenţii se vor familiariza cu diferite coduri pentru reprezentarea caracterelor: ASCII, EBCDIC, Unicode.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să determine reprezentarea în virgulă mobilă în standardul IEEE 754 în precizie simplă a unui număr raţional dat.
- Să determine reprezentarea în virgulă mobilă în standardul IEEE 754 în precizie dublă a unui număr raţional dat.
- Să determine numărul raţional, ştiind reprezentarea sa în virgulă mobilă în standardul IEEE 754 în precizie simplă sau dublă.

Durata medie de parcurgere a celei de-a treia unităţi de învăţare este de 3 ore.

M2.U3.3. Reprezentarea numerelor reale

Calculatoarele lucrează cu numere reale aproximate prin numere raţionale. Există două abordări în ce priveşte reprezentarea numerelor raţionale:

- Reprezentarea în virgulă fixă
- Reprezentarea în virgulă mobilă

În *reprezentarea în virgulă fixă*, după cum sugerează şi numele reprezentării, există o poziţie fixă a virgulei, un număr fixat de biţi la stânga virgulei şi un număr fixat de biţi la dreapta virgulei. Dezavantajul major al reprezentării în virgulă fixă constă în faptul că în acestă reprezentare este nevoie un număr mare de biţi pentru a putea memora un interval rezonabil de numere.

Să considerăm numărul: $0.10111_{(2)}*2^4{}_{(10)}$. *Mantisa* sa este 0.10111, iar *exponentul* este 4. Exponentul indică ordinul de mărime al numărului printr-o putere a bazei, iar mantisa exprimă mărimea numărului în cadrul ordinului respectiv.

Folosind reprezentarea în virgulă mobilă, se pot memora pe acelaşi număr de biţi, mai multe numere decât în reprezentarea în virgulă fixă deoarece pentru numerele "mari" se alocă mai mulţi biţi pentru exponent şi mai puţini pentru mantisă, iar la cele "mici" se alocă mai puţini biţi pentru exponent şi mai mulţi pentru mantisă.

O problemă care ar putea apărea când se lucrează cu numere reprezentate în virgulă mobilă constă în faptul că există multiple reprezentări ale aceluiaşi număr, ceea ce face ca operaţiile aritmetice să devină mai dificile.

De exemplu, să considerăm din nou numărul: $0.10111_{(2)}*2^4{}_{(10)}$. Acesta poate fi scris şi într-una din următoarele forme:

$0.10111_{(2)}*2^4{}_{(10)} = 1.0111_{(2)}*2^3{}_{(10)} = 101.11_{(2)}*2^1{}_{(10)} = 10111_{(2)}*2^{-1}{}_{(10)}$ etc.

Pentru a evita multiplele reprezentări ale aceluiaşi număr, se poate conveni, de exemplu, să se aşeze virgula în faţa celei mai semnificative cifre nenule, iar exponentul să se modifice corespunzător cu deplasarea virgulei. Pentru reprezentarea lui 0, se va face o excepţie de la această regulă. Dacă folosim această reprezentare atunci, dintre toate formele echivalente ale numărului de mai sus, se va alege următoarea: $0.10111_{(2)}*2^4{}_{(10)}$.

Calculatoarele lucrează cu numere reale (raţionale) reprezentate, în general, în virgulă mobilă în *formă normalizată*.

În *forma normalizată*, virgula se aşează după cea mai semnificativă cifră nenulă şi exponentul se modifică corespunzător cu deplasarea virgulei. Folosind această modalitate de reprezentare, se câştigă un bit suplimentar pentru mantisă. Pentru reprezentarea lui 0, se va face o excepţie de la această regulă. În general, există câteva valori ale exponentului care sunt rezervate pentru cazuri speciale: +0, -0, +∞, -∞, NaN (Not a Number). Dacă folosim forma normalizată atunci, dintre toate formele echivalente ale numărului de mai sus, cea utilizată va fi $1.0111_{(2)}*2^3{}_{(10)}$.

Exemple

$1.101_{(2)} * 2^{7}{}_{(10)} = 1.101_{(2)} * 10^{111}{}_{(2)}$ este un număr binar normalizat.

Majoritatea calculatoarelor folosesc pentru reprezentarea numerelor reale în virgulă mobilă standardul IEEE 754.

Standardul IEEE 754 în precizie simplă pentru reprezentarea numerelor reale în virgulă mobilă are următoarea structură:

- Cel mai semnificativ bit este bitul de semn, care dacă este 0 indică un număr pozitiv, iar dacă este 1 indică un număr negativ.
- Următorii 8 biţi conţin exponentul, care este memorat adunat cu 127. Pentru aceşti 8 biţi, valorile 00000000 şi 11111111 sunt rezervate pentru cazurile speciale.
- Ultimii 23 de biţi sunt destinaţi memorării părţii fracţionare, care, împreună cu bitul 1 din stânga virgulei, formează un număr cu 24 de biţi: $1.\underbrace{ff....fff}_{23}$, unde $\underbrace{ff....fff}_{23}$ reprezintă partea fracţionară pe 23 de biţi.

Există 5 tipuri de numere care pot fi reprezentate:

1. *Numerele normalizate nenule*, care sunt în formatul descris mai sus.
2. *Zero*, care este reprezentat prin exponentul 00000000 şi cu toţi biţii 0 în partea fracţionară. Bitul de semn poate fi 0 sau 1. Deci, sunt două reprezentări pentru zero: +0 şi -0.
3. *Infinitul* este reprezentat prin exponentul 11111111 şi cu toţi biţii 0 în partea fracţionară. Bitul de semn poate fi 0 sau 1. Deci, atât $+\propto$ cât şi $-\propto$ pot fi reprezentaţi.
4. *NaN* (*Not a Number*). Dacă încercăm să împărţim 0 la 0 sau $\propto$ la $\propto$, atunci rezultatul nu este definit. Acest rezultat nedefinit va fi NaN, care este repezentat prin exponentul 11111111, iar partea fracţionară conţinând orice şir de biţi exceptând şirul de 23 de zerouri. Bitul de semn poate fi 0 sau 1.
5. *Numere nenormalizate*. În reprezentarea normalizată, exisă un interval destul de mare între 0 şi cel mai mic număr strict pozitiv care poate fi reprezentat. Numere din acest interval pot fi reprezentate, dacă nu sunt în forma normalizată. Pentru reprezentarea numerelor nenormalizate, exponentul este întotdeauna -127, care, adunat cu 127, este reprezentat prin 00000000. Bitul de semn poate fi 0 (pentru numere pozitive) sau 1 (pentru numere negative). Cei 23 de biţi ai părţii fracţionare conţin chiar numărul de reprezentat scris în baza 2. Deci, lipseşte acel bit 1 din stânga virgulei aşa cum aveau numerele nenule normalizate.

Exemple

Semn	Exponent	Parte fracţionară	Valoare
0	10000101	10100...0	$+1.101_{(2)}*2^{6}$
1	00000001	01100...0	$-1.011_{(2)}*2^{-126}$ (-126 este cel mai mic

			exponent pentru numere normalizate)
0	11111110	000...0	$+1.0_{(2)}*2^{127}$ (127 este cel mai mare exponent pentru numere normalizate)
0	00000000	000...0	+0
1	00000000	000...0	-0
0	11111111	000...0	$+\infty$
1	11111111	000...0	$-\infty$
0	11111111	10100...	NaN
0	00000000	10...0	$+0.1_{(2)}*2^{-127}$ (număr nenormalizat)
1	00000000	010...0	$-0.01_{(2)}*2^{-127}$ (număr nenormalizat)

Scrieţi reprezentarea în standardul IEEE 754 în precizie simplă a următoarelor numere: $+1.11101_{(2)}*2^{13}$, $-1.01_{(2)}*2^{-3}$, $+0.001_{(2)}*2^{-127}$, $-0.101_{(2)}*2^{-127}$.

Standardul IEEE 754 în precizie dublă pentru reprezentarea numerelor reale în virgulă mobilă are următoarea structură:

- Cel mai semnificativ bit este bitul de semn, care dacă este 0 indică un număr pozitiv, iar dacă este 1 indică un număr negativ.
- Următorii 11 biţi conţin exponentul, care este memorat adunat cu 1023. Pentru aceşti 11 biţi, valorile 00000000000 şi 11111111111 sunt rezervate pentru cazurile speciale.
- Ultimii 52 de biţi sunt destinaţi memorării părţii fracţionare, care, împreună cu bitul 1 din stânga virgulei, formează un număr cu 53 de biţi: $1.\underbrace{ff....fff}_{52}$, unde $\underbrace{ff....fff}_{52}$ reprezintă partea fracţionară pe 52 de biţi.

În standardul IEEE 754 în precizie dublă sunt aceleaşi 5 tipuri de numere care pot fi reprezentate ca şi în standardul IEEE 754 în precizie simplă.

În standardul IEEE 754, în afară de formatul în precizie simplă şi formatul în precizie dublă, există şi formatul în precizie simplă extinsă şi în precizie dublă extinsă. Aceste formate în precizie extinsă nu sunt vizibile utilizatorului, dar sunt folosite pentru a reţine un număr mai mare de biţi pe parcursul calculelor, astfel încât să se reducă efectul erorilor cauzate de rotunjirile care se efectuează în urma calculelor. În formatele în precizie extinsă se folosesc biţi suplimentari atât pentru exponent cât şi pentru partea fracţionară. Numărul acestor biţi suplimentari poate varia de la o implementare la alta. De exemplu, un format pentru reprezentarea numerelor reale în precizie simplă extinsă poate avea 3 biţi în plus la exponent şi 8 biţi suplimentari la partea fracţionară faţă de formatul în precizie simplă. În general, formatul pentru reprezentarea numerelor reale în precizie dublă extinsă are 80 de biţi, din care 15 biţi pentru exponent şi 64 de biţi pentru partea fracţionară.

Observaţie Orice implementare a standardului IEEE 754 trebuie să conţină cel puţin reprezentarea în precizie simplă. Celelalte formate sunt opţionale.

Pentru ca rezultatul operaţiilor cu numere reale să fie cât mai exact, se folosesc biţi suplimentari pentru partea fracţionară pe parcursul calculelor şi diferite metode de rotunjire a rezultatelor. În standardul IEEE 754 există 4 moduri de rotunjire:

1. Rotunjirea spre 0
2. Rotunjirea spre +∝
3. Rotunjirea spre -∝
4. Rotunjirea spre cel mai apropiat număr care poate fi reprezentat în formatul dat.

Rotunjirea implicită este cea spre cel mai apropiat număr care poate fi reprezentat în formatul dat.

M2.U3.4. Reprezentarea caracterelor

ASCII

ASCII este un acronim pentru American Standard Code for Information Interchange. ASCII a fost creat de către American National Standard Institute (ANSI).

ASCII este un cod binar folosit pentru codificarea caracterelor pentru ca acestea să poată fi transmise procesorului de către periferice (de exemplu, de la tastatură).

ASCII a fost conceput iniţial ca un cod pe 7 biţi, numit şi ASCII-7, cu ajutorul căruia se puteau reprezenta 128 de caractere. Deoarece acum majoritatea calculatoarelor folosesc un byte (8 biţi) pentru a reprezenta un caracter, codul ASCII a fost extins la 8 biţi, prin adăugarea unui bit suplimentar la stânga. Acest bit suplimentar este adesea 0. Unele procesoare îl folosesc ca bit de control, iar altele îl folosesc pentru a dubla numărul de caractere care pot fi reprezentate.

ASCII are următoarele caracteristici:

1. Atât codurile cifrelor cât şi ale literelor sunt consecutive. Adică, codul caracterului '1' este egal cu codul caracterului '0' +1. De asemenea, codul lui 'B' este codul lui 'A' + 1 ş.a.m.d.
2. La fiecare 32 de caractere, începe o nou şir. De exemplu, şirul cu codurile literelor mici este imediat după şirul cu codurile literelor mari: 'A' + 32 = 'a', 'B' + 32 = 'b' etc.

EBCDIC

EBCDIC este un acronim pentru Extended Binary Coded Decimal Interchange Code şi a fost creat de IBM.

EBCDIC este codul folosit pentru a trimite informaţii între dispozitivele periferice şi procesor.

Fiecare cod EBCDIC conţine două părţi, fiecare constând într-un *nibble* (4 biţi). Cel mai semnificativ nibble defineşte clasa în care se află caracterul, iar celălalt nibble defineşte un anumit caracter din cadrul clasei.

Cu ajutorul codului EBCDIC, care este un cod pe 8 biţi, se pot reprezenta 256 de caractere. De exemplu:

- Cei 4 biţi care indică clasa în care se află codurile literelor de la 'A' la 'I' sunt 1100, iar ceilalţi 4 biţi care definesc caracterele propriu-zise în cadrul clasei variază de la 0001 la 1001.

- Nibble-ul care indică clasa în care se află codurile literelor de la 'J' la 'R' este 1101, iar cei 4 biţi care definesc caracterele propriu-zise în cadrul clasei variază de la 0001 la 1001.
- Cei 4 biţi care indică clasa în care se află codurile literelor de la 'S' la 'Z' sunt 1110, iar ceilalţi 4 biţi care definesc caracterele propriu-zise în cadrul clasei variază de la 0001 la 1001.
- Cei 4 biţi care indică clasa în care se află codurile cifrelor de la '0' la '9' sunt 1111, iar ceilalţi 4 biţi conţin echivalentul binar (pe 4 poziţii) al cifrei de reprezentat.

Unicode

Deoarece calculatoarele lucrează cu numere, caracterele sunt memorate prin intermediul unor numere. Unicode asociază un număr unic fiecărui caracter, independent de platformă, program sau limbaj. Standardul Unicode a fost adoptat de Apple, HP, IBM, Microsoft, Oracle, SAP, Sun, Sybase, Unisys etc. Multe sisteme de operare, toate browserele şi, de asemenea, XML, Java, WML etc. utilizează Unicode.

Unicode este produs de Unicode Consortium. Acest cod asigură baza pentru procesarea, stocarea datelor de tip caracter.

Unicode asociază fiecărui caracter un cod, care este un număr hexazecimal.

Codurile din Basic Multilingual Plane (BMP) sunt formate din 4 cifre hexazecimale (de exemplu, U+0030 este codul lui ‘0’, U+0031 este codul lui ‘1’,..., U+0039 este codul lui ‘9’, U+0041 este codul lui ‘A’, U+0042 este codul lui ‘B’,..., U+005A este codul lui ‘Z’, U+0061 este codul lui ‘a’, U+0062 este codul lui ‘b’,..., U+007A este codul lui ‘z’). Ca şi în codul ASCII, şi în Unicode atât codurile cifrelor cât şi ale literelor sunt consecutive.

Codurile care nu fac parte din BMP sunt formate din 5 sau 6 cifre hexazecimale.

M2.U3.5. Rezumat

Există două modalităţi de reprezentare a numerelor reale: în virgulă fixă sau în virgulă mobilă. În prima reprezentare, după cum sugerează şi numele, există o poziţie fixă a virgulei, un număr fixat de biţi la stânga virgulei şi un număr fixat de biţi la dreapta virgulei. În reprezentarea în virgulă mobilă, deoarece poziţia virgulei nu este prestabilită, se pot memora, pe acelaşi număr de biţi, mai multe numere decât puteau fi memorate folosind reperzentarea în virgulă fixă. Majoritatea calculatoarelor lucrează cu numere raţionale reprezentate în virgulă mobilă în standardul IEEE 754 şi cu numere reale aproximate prin numere raţionale.

În finalul acestei unităţi de învătare, am prezentat cele mai cunoscute coduri folosite pentru reprezentarea caracterelor ASCII şi Unicode şi am făcut o succintă descrire a codului EBCDIC.

M2.U3.6. Test de evaluare a cunoştinţelor

Scrieţi reprezentarea în standardul IEEE 754 în precizie simplă a următoarelor numere: $+1.10101_{(2)}*2^{13}$, $-1.001_{(2)}*2^{-13}$, $+0.101_{(2)}*2^{-127}$, $-0.111_{(2)}*2^{-127}$.

Temă de control

1. Determinaţi codurile directe pe 8 biţi ale următoarelor numere:
   14, 52, +0, -0, -121.
2. Determinaţi codurile inverse pe 8 biţi ale următoarelor numere:
   14, 52, +0, -0, -121.
3. Determinaţi codurile complementare pe 8 biţi ale următoarelor numere:
   14, 52, +0, -0, -121.
4. Adunaţi numerele:
   29 şi 62,
   55 şi -65,
   -101 şi -56
   în cod complementar pe 8 biţi.
5. Scădeţi numerele:
   84 şi 52,
   88 şi -5,
   -94 şi -12
   în cod complementar pe 8 biţi.
6. Înmulţiţi numerele:
   6 şi 11,
   7 şi 15,
   -12 şi 7
   în cod complementar pe 8 biţi.
   La sfârşitul fiecărei operaţii spuneţi dacă rezultatul este corect. Justificaţi-vă răspunsul.
7. Scrieţi reprezentarea în standardul IEEE 754 în precizie simplă a următoarelor numere: $+1.10101_{(2)}*2^{33}$, $-1.0101_{(2)}*2^{-3}$, $+0.1001_{(2)}*2^{-127}$, $-0.0101_{(2)}*2^{-127}$.

Modulul 3. Bazele logicii

Cuprins

Introducere

În acest modul vom studia funcţiile boolene şi reprezentarea lor cu ajutorul circuitelor logice. Funcţiile booleene pot avea ca valori pot fi 0 sau 1, iar variabilele lor pot lua, de asemenea, valorile 0 sau 1. Vom determina formele normale ale funcţiilor booleene, care sunt folosite ca pas intermediar în simplificarea lor. Vom prezenta două metode de simplificare: una grafică – cu ajutorul diagramelor Karnaugh, cealaltă analitică – metoda Quine-McCluskey a implicaţilor primi.

Atunci când vom dori să determinăm un circuit logic care să rezolve o problemă care apare în practică şi care poate fi exprimată printr-o funcţie booleană, vom identifica mai întâi funcţia, apoi o vom simplifica şi abia după ce am obţinut forma sa minimală, vom determina circuitul logic corespunzător. Toate aceste etape sunt necesare deoarece, pentru fiecare problemă vom determina circuitul de complexitate minimă care produce soluţia problemei.

De asemenea, vom studia şi funcţiile booleene incomplet definite deoarece, în general, problemele care apar în practică sunt modelate cu ajutorul lor. Vom arăta cum cele două metode de simplificare a funţiilor booleene (complet definite) amintite mai sus pot fi adaptate pentru a simplifica funcţii booleene incomplet definite.

Competenţe

La sfârşitul acestui modul studenţii vor fi capabili să:

- Determine forma normală disjunctivă şi forma normală conjunctivă a unei funcţii booleene;

- Determine forma minimală a unei funcţii booleene complet sau incomplet definite;
- Determine circuitul logic corespunzător unei funcţii booleene complet sau incomplet definite.

Unitatea de învăţare M3.U1. Algebra booleană

Cuprins

M3.U1.1. Introducere

Algebra booleană îşi leagă numele de matematicianul englez George Boole, care a trăit în secolul XIX şi a publicat în 1854 *The Laws of Thought* (de fapt, titlul complet era *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities*), lucrare care a fost iniţial ignorată sau criticată. O contribuţie importantă a lui George Boole a fost introducerea tabelelor de adevăr.

M3.U1.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a conceptelor de expresie booleană şi funcţie booleană şi a relaţiilor dintre ele.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Determine tabelul de adevăr al unei expresii booleene;
- Determine tabelul de adevăr al unei funcţii booleene;
- Demonstreze egalităţi de expresii/funcţii booleene folosind deducţia sau tabelele de adevăr.

Durata medie de parcurgere a primei unităţi de învăţare este de 2 ore.

M3.U1.3. Expresii booleene

Fie mulţimea $B = \{0, 1\}$. Se definesc următoarele operaţii: *conjuncţia* sau *AND*, *disjuncţia* sau *OR* şi *negaţia* sau *NOT*:

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

x	$\bar{x}$
0	1
1	0

Mulţimea *B* pe care s-au definit aceşti 3 operatori formează o *algebra booleană*, notată cu $(B, +, \cdot, \overline{\ })$.

O variabilă *x* se numeşte *variabilă booleană* dacă ia numai valori din *B*.

Observaţii 1) Valorile din algebra booleană *B* sunt 0 şi 1 (şi nu FALS şi ADEVĂRAT) deoarece calculatoarele lucrează cu biţi.

2) Pentru orice variabilă booleană *x*, avem:

$x + x = x$

$x^2 = x \cdot x = x.$

3) Pentru orice variabile booleene *x* şi *y*, avem:

$x + y = 0$ dacă şi numai dacă $x = y = 0$

$x \cdot y = 1$ dacă şi numai dacă $x = y = 1$.

Vom nota expresiile boolene, care pot fi simple sau compuse, cu *a*, *b*, *c* ...

Reguli de formare a expresiilor booleene corecte:

1. O variabilă booleană este o expresie booleană, numită *expresie simplă*.
2. Dacă *a* este o expresie booleană, atunci $\bar{a}$ şi (a) sunt expresii booleene.
3. Dacă *a* şi *b* sunt expresii booleene, atunci $a+b$ şi $a \cdot b$ sunt expresii booleene.

Expresiile booleene se evaluează de la stânga la dreapta în conformitate cu priorităţile operatorilor. Operatorul cu cea mai mare prioritate este $\overline{\ }$, urmat de · şi +.

Teoremă (Proprietăţile expresiilor booleene) Dacă *a*, *b* şi *c* sunt expresii booleene, atunci au loc:

1. Legile comutativităţii:

   $a+b = b+a$

   $a \cdot b = b \cdot a$
2. Legile asociativităţii:

   $(a+b) + c = a + (b+c)$

   $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
3. Legile distributivităţii:

   $a \cdot (b+c) = a \cdot b + a \cdot c$

   $a+(b \cdot c) = (a+b) \cdot (a+c)$
4. Idempotenţele:

   $a+a = a$

   $a \cdot a = a$
5. Elementele neutre:

   $a+0 = a$

   $a \cdot 1 = a$
6. Elementele inverse/simetrizabile:

$a+\bar{a} = 1$

$a\cdot \bar{a} = 0$

7. Elementele dominante:

   $a+1 = 1$

   $a\cdot 0 = 0$

8. Legile absorbţiei:

   $a + a\cdot b = a$

   $a\cdot(a+b) = a$

9. Legea dublei negaţii:

   $\bar{\bar{a}}=a$

10. Legile lui de Morgan:

    $\overline{a+b} = \bar{a}\cdot\bar{b}$

    $\overline{a\cdot b} = \bar{a}+\bar{b}$

Observaţie Pentru o lege dată, putem obţine legea sa duală prin înlocuirea tuturor operatorilor + cu ·, · cu + şi a tututor apariţiilor lui 0 cu 1 şi a tuturor apariţiilor lui 1 cu 0.

Aceste 10 proprietăţi pot fi demonstrate folosind tabele de adevăr sau folosind deducţia.

Într-un *tabel de valori de adevăr* se scriu toate combinaţiile posibile de valori ale variabilelelor din expresia booleană şi fiecărei astfel combinaţii i se asociază câte valoarea expresiei, care poate fi 0 sau 1.

Exemplu

Putem demonstra legea absorbţiei $a + a\cdot b = a$ folosind deducţia în modul următor:

$a + a\cdot b =$ //elementul neutru

$= a\cdot 1+ a\cdot b =$ //legea distributivităţii

$= a\cdot(1+ b) =$ //elementul dominant

$= a\cdot 1 =$ // elementul neutru

$= a$

De asemenea, putem demonstra legea absorbţiei $a + a\cdot b = a$ folosind tabelul de adevăr:

a	b	$a\cdot b$	$a+a\cdot b$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Se observă că, în tabelul de adevăr, coloana corespunzătoare expresiei $a + a \cdot b$ este egală cu coloana corespunzătoare lui a. Deci, $a + a \cdot b = a$.

Exemplu

Putem demonstra legea dublei negaţii $\bar{\bar{a}} = a$ folosind deducţia în modul următor:

$\bar{\bar{a}} =$	// elementul neutru
$= \bar{\bar{a}} \cdot 1 =$	// elementul invers
$= \bar{\bar{a}} \cdot (a + \bar{a}) =$	// legea distributivităţii
$= \bar{\bar{a}} \cdot a + \bar{\bar{a}} \cdot \bar{a} =$	// elementul invers
$= \bar{\bar{a}} \cdot a + 0 =$	// elementul invers
$= \bar{\bar{a}} \cdot a + \bar{a} \cdot a =$	// legea comutativităţii
$= a \cdot \bar{\bar{a}} + a \cdot \bar{a} =$	// legea distributivităţii
$= a \cdot (\bar{\bar{a}} + \bar{a}) =$	// elementul invers
$= a \cdot 1 =$	// elementul neutru
$= a$	

De asemenea, putem demonstra legea dublei negaţii $\bar{\bar{a}} = a$ folosind tabelul de adevăr:

a	$\bar{a}$	$\bar{\bar{a}}$
0	1	0
1	0	1

Se observă că, în tabelul de adevăr, coloana corespunzătoare expresiei $\bar{\bar{a}}$ este egală cu coloana corespunzătoare lui a. Deci, $\bar{\bar{a}} = a$.

Folosind tabele de adevăr sau deducţia, demonstraţi şi celelalte proprietăţi ale expresiilor booleene enunţate în teorema anterioară.

M3.U1.4. Funcţii booleene

Fie $B^n = \{(b_1, ..., b_n) \mid b_i \in B, 1 \le i \le n\}, n \in \mathbf{N}^*$.

O funcţie $f : B^n \rightarrow B$ se numeşte *funcţie booleană de n variabile*. Există 2^{2^n} funcţii booleene distincte de n variabile.

Exemplu

Există 4 funcţii booleene distincte de o variabilă.

x	f_1	f_2	f_3	f_4
0	0	0	1	1
1	0	1	0	1

$f_1(x) = 0$, $f_2(x) = x$, $f_3(x) = \bar{x}$, $f_4(x) = 1$.

Exemplu

Există 16 funcţii booleene distincte de două variabile.

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Observaţie Expresiile booleene pot fi privite ca funcţii booleene care asociază unui n-uplu de elemente zero şi unu, unde n este numărul de variabile din expresie, un număr care poate fi 0 sau 1.

Deci, expresia

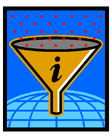
$$\bar{x}\cdot y+x\cdot y\cdot z+x\cdot\bar{z}$$

unde $x = 1$, $y = 0$, $z = 0$ face ca tripletului (1, 0, 0) să-i corespundă valoarea 1.

În consecinţă, proprietăţile expresiilor booleene enunţate în teorema anterioară sunt valabile şi pentru funcţii booleene. Fie f şi g două funcţii booleene de n variabile. Definim:

$$\bar{f}(x_1,\ldots,x_n) = \overline{f(x_1,\ldots,x_n)}$$
$$(f+g)(x_1,\ldots,x_n) = f(x_1,\ldots,x_n)+g(x_1,\ldots,x_n)$$
$$(f\cdot g)(x_1,\ldots,x_n) = f(x_1,\ldots,x_n)\cdot g(x_1,\ldots,x_n)$$

Spunem că funcţiile booleene f şi g sunt egale şi scriem $f = g$, dacă ultima coloană din tabelul de adevăr al lui f coincide cu ultima coloană din tabelul de adevăr al lui g.



M3.U1.5. Rezumat

Funcţiile booleene pot avea ca valori pot fi 0 sau 1, iar variabilele lor pot lua, de asemenea, valorile 0 sau 1. Deoarece expresiile booleene pot fi privite ca funcţii booleene, toate proprietăţile expresiilor booleene pot fi extinse şi la funcţii booleene. De asemenea, pentru a demonstra egalităţi de funcţii putem folosi metodele de demonstrare a egalităţilor de expresii booleene: deducţia şi tabelele de adevăr.

M3.U1.6. Test de evaluare a cunoştinţelor

1. Folosind tabelele de adevăr sau deducţia, arătaţi că:
    - $\bar{a}bc+a\bar{b}c+ab\bar{c}+abc = bc+ac+ab$
    - $\bar{a}\bar{b}cd+\bar{a}bcd+ a\bar{b}cd+abcd = cd$
2. Determinaţi tabelele de adevăr ale următoarelor funcţii:

- $f_1(x_1, x_2) = x_1+\bar{x}_2$
- $f_1(x_1, x_2) = x_1\ x_2+\bar{x}_1x_2$
- $f_1(x_1, x_2, x_3) = x_1\ x_2+\bar{x}_1x_2\bar{x}_3+x_2$

Unitatea de învăţare M3.U2. Forme normale ale funcţiilor booleene

Cuprins

M3.U2.1. Introducere

În această unitate de învăţare vom introduce formele normale ale funcţiilor booleene: forma normală disjunctivă şi forma normală conjunctivă. Vom arăta cum se determină aceste forme normale ale unei funcţii booleene date fie prin tabelul său de adevăr, fie în formă algebrică.

Aceste forme normale sunt folosite ca pas intermediar în simplificarea funcţiilor booleene, iar simplificarea lor este o necesitate deoarece se urmăreşte, pentru orice funcţie, crearea unui circuit minimal care să o implementeze.

M3.U2.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a modului de determinare a formelor normale ale funcţiilor booleene.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Determine forma normală disjunctivă a unei funcţii booleene;
- Determine forma normală conjunctivă a unei funcţii booleene.

Durata medie de parcurgere a acestei unităţi de învăţare este de 2 ore.

M3.U2.3. Forma normală disjunctivă

Fie f o funcţie booleană de n variabile $x_1,...,x_n$. Un *constituent al unităţii* sau un *minterm* este un termen de forma $y_1\cdot...\cdot y_n$, unde $y_i = x_i$ sau $y_i = \bar{x}_i$, pentru $1 \leq i \leq n$.

O reprezentare a funcţiei f ca sumă de mintermi se numeşte *forma normală disjunctivă* sau *forma canonică disjunctivă* a lui f.

Exemplu

Funcţia booleană $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2x_3$ este în forma normală disjunctivă.

Dacă funcţia f se dă în formă algebrică, putem să-i determinăm forma normală disjunctivă aplicând următorul algoritm:

Pasul 1. Inserăm în fiecare termen, pentru fiecare variabilă care lipseşte, suma dintre ea şi negaţia ei.

Pasul 2. Aplicăm legea distributivităţii şi idempotenţa.

Exemplu

Vom determina forma normală disjunctivă a funcţiei booleene $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_3+x_1\bar{x}_2+x_1x_3$:

$$f(x_1, x_2, x_3) = \bar{x}_1(x_2+\bar{x}_2)\bar{x}_3+x_1\bar{x}_2(x_3+\bar{x}_3)+x_1(x_2+\bar{x}_2)x_3=$$
$$=\bar{x}_1x_2\bar{x}_3+\bar{x}_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2x_3+x_1\bar{x}_2\bar{x}_3+x_1x_2x_3+x_1\bar{x}_2x_3=$$
$$=\bar{x}_1x_2\bar{x}_3+\bar{x}_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2x_3+x_1\bar{x}_2\bar{x}_3+x_1x_2x_3.$$

Determinaţi forma normală disjunctivă a funcţiilor booleene:

1. $f(x_1, x_2) = \bar{x}_1+x_1\bar{x}_2+x_2$.
2. $f(x_1, x_2, x_3) = \bar{x}_1+x_1\bar{x}_2\bar{x}_3+x_2x_3$.

Dacă se dă tabelul de adevăr al funcţiei booleene f, fiecare valoare 1 din coloana cu valorile lui f corespunde unui minterm din forma normală disjunctivă. Acest minterm conţine toate variabilele a căror valoare este 1 în tabelul de adevăr şi negaţiile variabilelor care au valoarea 0 în tabelul de adevăr.

Exemplu

Vom determina forma normală disjunctivă a funcţiei booleene de 3 variabile date prin tabelul său de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

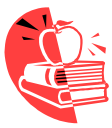
Obţinem $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3$.

Determinaţi forma normală disjunctivă a funcţiei booleene date prin următorul tabel de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Orice minterm poate fi identificat prin intermediul unui *indice*, care este echivalentul în baza 10 al numărului binar $e_1e_2...e_n$, unde

$$e_i = \begin{cases} 0, \text{daca } \bar{x}_i \text{ apare in minterm} \\ 1, \text{daca } x_i \text{ apare in minterm} \end{cases}$$



Exemplu

Considerăm funcţia booleană $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3$ în formă normală disjunctivă.

Primul termen din sumă este $m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$, al doilea este $m_1 = \bar{x}_1\bar{x}_2x_3$, iar ultimul termen este $m_4 = x_1\bar{x}_2\bar{x}_3$.

Deci, $f(x_1, x_2, x_3) = m_0 + m_1 + m_4$, sau, echivalent

$f(x_1, x_2, x_3) = \Sigma m(0, 1, 4)$.

În consecinţă, pentru a defini o funcţie booleană, în locul tabelului de adevăr, se pot indica doar mintermii pentru care funcţia are valoarea 1.

Exemplu
Funcţia booleană de 3 variabile dată prin tabelul său de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

poate fi scrisă şi în forma echivalentă:
$f(x_1, x_2, x_3) = \Sigma m(0, 4, 6, 7)$.

M3.U2.4. Forma normală conjunctivă

Fie f o funcţie booleană de n variabile $x_1,...,x_n$. Un *constituent al nulului* sau un *maxterm* este un termen de forma $y_1+...+y_n$, unde $y_i = x_i$ sau $y_i = \bar{x}_i$, pentru $1 \le i \le n$.

O reprezentare a funcţiei f ca produs de maxtermi se numeşte *forma normală conjunctivă* sau *forma canonică conjunctivă* a lui f.

Exemplu
Funcţia booleană $f(x_1, x_2, x_3) = (\bar{x}_1+\bar{x}_2+\bar{x}_3)(x_1+\bar{x}_2+\bar{x}_3)(x_1+\bar{x}_2+x_3)$ este în forma normală conjunctivă.

Dacă se dă tabelul de adevăr al funcţiei booleene f, fiecare valoare 0 din coloana cu valorile lui f corespunde unui maxterm din forma normală conjunctivă. Acest maxterm conţine toate variabilele a căror valoare este 0 în tabelul de adevăr şi negaţiile variabilelor care au valoarea 1 în tabelul de adevăr.

Exemplu
Vom determina forma normală conjunctivă a funcţiei booleene de 3 variabile date prin tabelul său de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Obţinem $f(x_1, x_2, x_3) = (x_1+\bar{x}_2+x_3)(x_1+\bar{x}_2+\bar{x}_3)(\bar{x}_1+x_2+\bar{x}_3)(\bar{x}_1+\bar{x}_2+x_3)(\bar{x}_1+\bar{x}_2+\bar{x}_3)$.

Determinaţi forma normală conjunctivă a funcţiei booleene date prin următorul tabel de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Orice maxterm poate fi identificat prin intermediul unui *indice*, care este echivalentul în baza 10 al numărului binar $e_1e_2{\ldots}e_n$, unde

$$e_i = \begin{cases} 0, \text{daca } x_i \text{ apare in maxterm} \\ 1, \text{daca } \bar{x}_i \text{ apare in maxterm} \end{cases}$$

Exemplu

Considerăm funcţia booleană

$f(x_1, x_2, x_3) = (x_1+\bar{x}_2+x_3)(x_1+\bar{x}_2+\bar{x}_3)(\bar{x}_1+\bar{x}_2+\bar{x}_3)$

în formă normală conjunctivă.

Primul factor din produs este $M_2 = x_1+\bar{x}_2+x_3$, al doilea este $M_3 = x_1+\bar{x}_2+\bar{x}_3$, iar ultimul este $M_7 = \bar{x}_1+\bar{x}_2+\bar{x}_3$.

Deci, $f(x_1, x_2, x_3) = M_2\, M_3\, M_7$, sau, echivalent

$f(x_1, x_2, x_3) = \Pi M(2, 3, 7)$.

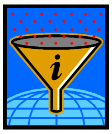
În consecinţă, pentru a defini o funcţie booleană, în locul tabelului de adevăr, se pot specifica doar maxtermii pentru care funcţia are valoarea 0.

Exemplu

Funcţia booleană de 3 variabile dată prin tabelul său de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

poate fi dată şi în forma echivalentă $f(x_1, x_2, x_3) = \Pi M(2, 3, 5, 6, 7)$



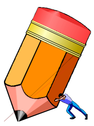
M3.U2.5. Rezumat

În această unitate de învăţare am introdus formele normale ale funcţiilor booleene. Forma normală disjunctivă este o reprezentare a unei funcţii booleene ca sumă de mintermi, un minterm fiind un produs în care apar toate variabilele fie negate fie nenegate.

Forma normală conjunctivă este o reprezentare a unei funcţii booleene ca produs de maxtermi, un maxterm fiind o sumă în care apar toate variabilele fie negate fie nenegate.

Am arătat cum se determină formele normale ale unei funcţii booleene date fie prin tabelul său de adevăr, fie în formă algebrică.

Determinarea formelor normale ale funcţiilor booleene este foarte importantă deoarece formele normale sunt folosite ca pas intermediar în simplificarea funcţiilor booleene, iar simplificarea este o necesitate deoarece se urmăreşte, pentru orice funcţie, crearea unui circuit minimal care să o implementeze.



M3.U2.6. Test de evaluare a cunoştinţelor

1. Determinaţi forma normală disjunctivă pentru următoarele funcţii booleene:
   - $f(x_1, x_2) = \bar{x}_1+x_2$.
   - $f(x_1, x_2, x_3) = \bar{x}_1+x_1\bar{x}_3+x_2x_3$.
2. Determinaţi forma normală disjunctivă şi forma normală conjunctivă a funcţiei booleene date prin următorul tabel de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Unitatea de învăţare M3.U3. Simplificarea funcţiilor booleene

Cuprins

M3.U3.1. Introducere

În această unitate de învăţare vom prezenta două metode de simplificare: una grafică – cu ajutorul diagramelor Karnaugh, cealaltă analitică – metoda Quine-McCluskey a implicaţilor primi.

Atunci când vom dori să determinăm un circuit logic care să rezolve o problemă care apare în practică şi care poate fi exprimată printr-o funcţie booleană, vom identifica mai întâi funcţia, apoi o vom simplifica şi abia după ce am obţinut forma sa minimală, vom determina circuitul logic corespunzător. Dintre toate aceste etape, cea mai consistentă este etapa de simplificare a funcţiei deoarece, pentru fiecare problemă de urmăreşte determinarea circuitului de complexitate minimă care produce soluţia problemei.

M3.U3.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a capacităţii de a simplifica funcţiile booleene folosind atât metode grafice – diagrame Karnaugh, cât şi metode analitice – metoda Quine-McCluskey.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să simplifice funcţii booleene folosind diagrame Karnaugh;
- Să simplifice funcţii booleene folosind metoda Quine-McCluskey.

Durata medie de parcurgere a celei de-a treia unităţi de învăţare este de 3 ore.

M3.U3.3. Simplificarea funcţiilor booleene

Complexitatea unei funcţii boolene este dată de numărul de operatori pe care îi conţine.

Exemplu

Funcţia $f(x_1, x_2) = \bar{x}_1+x_2+x_1\bar{x}_2$ conţine 5 operatori ($\bar{\ }$, +, +, ·, $\bar{\ }$).
Dar, acestă funcţie poate fi scrisă şi în forma:

$f(x_1, x_2) = \bar{x}_1+x_2+x_1\bar{x}_2 =$ //legea lui de Morgan

$= \overline{x_1\bar{x}_2}+x_1\bar{x}_2=$ // elementul invers

$= 1$

În forma simplificată, funcţia nu conţine niciun operator.

A *simplifica* o funcţie înseamnă a reduce numărul de operatori care apar în scrierea funcţiei în formă algebrică. Simplificarea funcţiilor booleene este necesară deoarece funcţiile booleene se folosesc la conceperea circuitelor electronice şi se doreşte crearea circuitelor cât mai simple pentru rezolvarea oricărei probleme.

Există mai multe metode de simplificare a funcţiilor booleene:

1. Aplicarea proprietăţilor funcţiilor booleene (ca în exemplul anterior)
2. Metoda grafică utilizând diagrame Karnaugh
3. Metoda analitică a lui Quine-McCluskey.

M3.U3.4. Diagrame Karnaugh

O binecunoscută metodă grafică de simplificare a funcţiilor booleene a fost introdusă de fizicianul american Maurice Karnaugh în 1953.

Simplificare funcţiilor booleene date în formă normală disjunctivă

Simplificarea se bazează pe determinarea de perechi, cvatruple, ..., grupuri de 2^n mintermi, în care oricare doi mintermi diferă prin exact o variabilă. Pentru o identificare mai uşoară a acestor perechi, cvatruple, ..., grupuri de 2^n mintermi, vom folosi diagrame Karnaugh.

O diagramă Karnaugh conţine câte o celulă pentru fiecare posibil minterm din forma normală disjunctivă a funcţiei. Deoarece pentru o funcţie booleană de *n* variabile există 2^n posibili mintermi în forma sa normală disjunctivă, rezultă că diagrama Karnaugh va conţine 2^n celule. Aceste 2^n celule pot fi aşezate în mai multe moduri, însă trebuie să respecte condiţia ca oricare două celule vecine să corespundă unor mintermi care diferă prin exact o variabilă. Cele mai cunoscute moduri de aranjare a celulelor sunt cunoscute sub numele de diagrame Karnaugh de forma I şi diagrame Karnaugh de forma II, care se mai numesc şi diagrame Veitch-Karnaugh.

Diagrame Karnaugh de forma I

Pentru funcţii de 2 variabile

	x_2	$\bar{x}_2$
x_1		
$\bar{x}_1$		

Pentru funcţii de 3 variabile

	x_2	x_2	$\bar{x}_2$	$\bar{x}_2$
x_1				
$\bar{x}_1$				
	$\bar{x}_3$	x_3	x_3	$\bar{x}_3$

Pentru funcţii de 4 variabile

	x_2	x_2	$\bar{x}_2$	$\bar{x}_2$	
x_1					$\bar{x}_3$
x_1					x_3
$\bar{x}_1$					x_3
$\bar{x}_1$					$\bar{x}_3$
	$\bar{x}_4$	x_4	x_4	$\bar{x}_4$	

Diagrame Karnaugh de forma II

Pentru funcţii de 2 variabile

$x_1 \backslash x_2$	0	1
0		
1		

Pentru funcţii de 3 variabile

$x_1 \backslash x_2\, x_3$	00	01	11	10
0				
1				

Pentru funcţii de 4 variabile

$x_1\, x_2 \backslash x_3\, x_4$	00	01	11	10
00				
01				
11				
10				

Pentru funcţii de 5 variabile, se construiesc două diagrame Karnaugh pentru funcţii de 4 variabile, prima diagramă corespunzând lui $x_5 = 0$ şi cealaltă pentru $x_5 = 1$.

Pentru funcţii de 6 variabile sunt necesare patru diagrame Karnaugh pentru funcţii de 4 variabile, care, însă, sunt foarte dificil de utilizat. De aceea, pentru minimizarea funcţiilor booleene de 6 sau mai multe variabile se va folosi metoda Quine-McCluskey, care va fi studiată în unitatea de învăţare M3.U4.

Observaţie În diagramele Karnaugh descrise mai sus celulele adiacente corespund unor mintermi care diferă prin exact o variabilă. Fiecare celulă are 4 celule adiacente: sus, jos, stânga şi dreapta. Putem determina celulele adiacente cu celulele de pe prima sau ultima linie (sau coloană) a diagramei, închizând diagrama astfel ultima linie (sau coloană) se lipeşte de prima linie (sau coloană).

Observaţie Diagramele Karnaugh descrise mai sus nu sunt singurele diagrame care pot fi folosite pentru simplificarea funcţiilor booleene. Se poate face orice altă rearanjare a celulelor, care să respecte condiţia ca oricare celulele adiacente să corespundă unor mintermi care diferă prin exact o variabilă.

Pentru a simplifica o funcţie booleană dată, folosim diagrama Karnaugh în modul următor:

1. Scriem câte un 1 în fiecare celulă care corespunde unui minterm din forma normală disjunctivă a funcţiei de simplificat.
2. În orice simplificare, vom încerca să determinăm cel mai mare grup compact, care conţine o putere a lui 2 de celule adiacente care conţin 1, pentru a obţine un produs minimal. Un grup de 2 celule adiacente care conţin 1 înseamnă dispariţia unei variabile din produs, un grup de 4 celule adiacente care conţin 1 înseamnă dispariţia a 2 variabile din produs, un grup de 8 celule adiacente care conţin 1 înseamnă dispariţia a 3 variabile din produs ş.a.m.d. Putem include o celulă, care conţine 1, în mai multe grupuri, dacă astfel obţinem grupuri mai mari, datorită idempotenţei ($a + a = a$).
3. Începem prin a forma grupuri care conţin celule cu 1 care pot fi grupate într-un singur mod.
4. Verificăm cele 4 colţuri ale diagramei. Acestea ar putea fi celule adicente care conţin 1, chiar dacă par a fi izolate. De asemenea, celulele de pe aceeaşi coloană (sau linie) de pe prima linie (sau coloană) şi de pe ultima linie (sau coloană) ar putea fi celule adicente care conţin 1, chiar dacă par a fi izolate.
5. Dacă există mai multe posibilităţi de grupare a celulelor, prin care s-ar obţine grupuri cu acelaşi număr de celule, se aleg celulele care nu fac parte din niciun alt grup.

Exemplu

Utilizăm diagrama Karnaugh pentru a minimiza funcţia booleană:

$f(x_1, x_2, x_3, x_4) = \Sigma m(0, 3, 6, 9, 11, 13, 14, 15).$

$x_1\,x_2 \backslash x_3\,x_4$	00	01	11	10
00	1		1	
01				1
11		1	1	1
10		1	1	

Deci, forma minimală a acestei funcţii este:

$$f(x_1, x_2, x_3, x_4) = x_1x_4 + x_2x_3\bar{x}_4 + \bar{x}_2x_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4.$$

Observaţie Dacă funcţia nu este dată în formă normală disjunctivă, mai întâi îi determinăm forma normală disjunctivă (printr-una din metodele prezentate în unitatea de învăţăre M3.U2) şi apoi o simplificăm folosind diagrame Karnaugh şi obţinem forma minimală (suma minimală de produse).

Conceptul dual conceptului de sumă minimală de produse este produsul minimal de sume. Acesta poate fi determinat folosind tot diagramele Karnaugh, dar pornind de la forma normală conjunctivă a funcţiei de minimizat.

Simplificarea funcţiilor booleene date în formă normală conjunctivă

Simplificarea funcţiilor booleene date în formă normală conjunctivă, utilizând diagrame Karnaugh, se face asemănător cu simplificarea funcţiilor booleene date în formă normală disjunctivă. Deosebirea între cele două metode de simplificare constă în faptul că, în cazul funcţiilor booleene date în formă normală conjunctivă, vom scrie câte un 0 în fiecare celulă a diagramei care corespunde unui maxterm al funcţiei de simplificat. Apoi procesul de simplificare este acelaşi ca şi în cazul simplificării funcţiilor booleene date în formă normală disjunctivă. Vom căuta să identificăm cele mai mari blocuri care conţin puteri ale lui 2 de celule adiacente care conţin 0, de acestă dată, cu scopul de a obţine sume minimale.

Exemplu

Utilizăm diagrama Karnaugh pentru a minimiza funcţia booleană:

$f(x_1, x_2, x_3, x_4) = \Pi M(1, 5, 7, 9, 10, 13, 14, 15)$.

$x_1\ x_2 \backslash x_3\ x_4$	00	01	11	10
00		0		
01		0	0	
11		0	0	0
10		0		0

Deci, forma minimală (produsul minimal de sume) a (al) acestei funcţii este:

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_1+\bar{x}_3+x_4)(\bar{x}_2+\bar{x}_4)(x_3+\bar{x}_4).$$

M3.U3.5. Metoda Quine-McCluskey

Metoda Quine–McCluskey, numită şi *metoda implicanţilor primi*, este o metodă analitică de simplificare a funcţiilor booleene, care a aparţine logicianului american Quine şi inginerului american McCluskey. Din punct de vedere funcţional, acestă metodă este identică cu metoda grafică cu diagrame Karnaugh, însă are câteva avantaje importante:

1. Poate fi folosită pentru funcţii cu oricâte variabile, spre deosebire de metoda cu diagrame Karnaugh care este inadecvată pentru funcţii cu mai mult de 5 variabile.
2. Forma sa tabelară, face ca metoda Quine–McCluskey să poată fi programată mai eficient.
3. Metoda Quine–McCluskey poate fi folosită pentru a determina dacă o funcţie booleană dată este sau nu în formă minimală.

Metoda Quine–McCluskey constă în următorii 2 paşi:

Pasul 1. Determinarea implicanţilor primi

Pasul 2. Determinarea implicanţilor primi esenţiali, folosind tablelul implicanţilor primi.

Pasul 1.

1. Se determină forma normală disjunctivă a funcţiei booleene pe care dorim să o simplificăm.
2. Fiecărui minterm m_i din forma normală disjunctivă i se asociază şirul binar a_i, definit astfel:

$$a_{ij} = \begin{cases} 1, \text{daca } m_i \text{ conţine } x_j \\ 0, \text{daca } m_i \text{ conţine } \bar{x}_j \end{cases}$$

3. Se sortează şirurile binare a_i în ordine crescătoare după numărul de elemente 1 pe care le conţin.
4. Se aplică operaţia de compunere tuturor perechilor de şiruri binare care diferă prin exact un element. Dacă 2 şiruri diferă doar printr-un bit, atunci acel bit va fi înlocuit printr-o liniuţă care va indica faptul că variabila corespunzătoare acelui bit nu contează.
5. Se determină un nou tabel care va conţine pe lângă şirurile obţinute prin operaţia de compunere de la pasul anterior şi toate şirurile care nu au intrat în nicio compunere.
6. Dacă şirurilor din acest nou tabel li se mai poate aplica operaţia de compunere atunci se revine la Pasul 5, altfel algoritmul se termină (cu un tabel ireductibil).

Fiecărui şir a_i din tabelul ireductibil determinat la sfârşitul Pasului 1 îi corespunde un produs care:

- conţine $\bar{x}_j$, dacă $a_{ij} = 0$
- conţine x_j, dacă $a_{ij} = 1$
- nu conţine nici $\bar{x}_j$, nici x_j, dacă a_{ij} = -.

Fiecare astfel de produs corespunzător unui şir din tabelul ireductibil determinat la sfârşitul Pasului 1 se numeşte *implicant prim.*

Pasul 2.

1. Se determină tabelul implicanţilor primi. Acesta conţine câte o linie pentru fiecare implicant prim determinat la sfârşitul Pasului 1 şi cate o coloană pentru fiecare minterm al funcţiei de minimizat.
2. Se aleg din acest tabel, acei implicanţi primi care acoperă toţi mintermii, adică întreaga funcţie. Aceştia se vor numi *implicanţi primi esenţiali*.

Exemplu

Vom aplica metoda Quine-McCluskey pentru a minimiza funcţia

$$f(x_1, x_2, x_3, x_4) = \Sigma m(1, 5, 7, 9, 11, 13).$$

Număr de elemente 1	Minterm	Şir binar
1	m_1	0001
2 2	m_5 m_9	0101 1001
3 3 3	m_7 m_{11} m_{13}	0111 1011 1101

Aplicând operaţia de compunere tuturor perechilor de şiruri binare care diferă prin exact un element obţinem tabelul următor, care conţine şirurile obţinute în urma compunerilor.

Implicant	Operaţie de compunere
0-01 -001	$m_1\bullet m_5$ $m_1\bullet m_9$
01-1 -101 10-1 1-01	$m_5\bullet m_7$ $m_5\bullet m_{13}$ $m_9\bullet m_{11}$ $m_9\bullet m_{13}$

Acest tabel nu este ireductibil, deci mai aplicăm o dată operaţia de compunere şi obţinem următorul tabel:

Implicant	Operaţie de compunere
--01	$m_1\bullet m_5 \bullet m_9\bullet m_{13}$
01-1 10-1	$m_5\bullet m_7$ $m_9\bullet m_{11}$

Acest tabel conţine un şir obţinut în urma aplicării operaţiei de compunere şi cele două şiruri din tabelul anterior care nu au intrat în nicio operaţie de compunere.
Din şirurile din acest tabel, care este ireductibil, determinăm implicanţii primi: $\bar{x}_3x_4$, $\bar{x}_1x_2x_4$, $x_1\bar{x}_2x_4$.
În continuare, vom determina tabelul implicanţilor primi, în care vom avea câte o linie pentru fiecare implicant prim şi câte o coloană pentru fiecare minterm:

Implicant\minterm	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$	$\bar{x}_1x_2\bar{x}_3x_4$	$\bar{x}_1x_2x_3x_4$	$x_1\bar{x}_2\bar{x}_3x_4$	$x_1\bar{x}_2x_3x_4$	$x_1x_2\bar{x}_3x_4$
$\bar{x}_3x_4$	*	*		*		*
$\bar{x}_1x_2x_4$		*	*			
$x_1\bar{x}_2x_4$				*	*	

În consecinţă, toţi cei trei implicanţi primi sunt implicanţi primi esenţiali. Deci, forma minimală a acestei funcţii este:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_3x_4+\bar{x}_1x_2x_4+ x_1\bar{x}_2x_4.$$

M3.U3.6. Rezumat

În această unitate de învăţare am arătat cum se pot simplifica funcţiile booleene pornind de la formele lor normale disjunctive prin două metode. Prima este o metodă grafică şi foloseşte diagrame Karnaugh, cealaltă metodă este analitică – metoda implicanţilor primi a lui Quine şi McCluskey. Principiul de simplificare pe care îl folosesc ambele metode este acelaşi: identificarea perechilor, cvatruplelor, ..., grupurilor de 2^n mintermi, în care oricare doi mintermi diferă prin exact o variabilă. Diferenţa dintre cele două metode constă în felul în care se face această identificare. Metoda cu diagrame Karnaugh presupune determinarea celor mai mari grupuri compacte, care conţin o putere a lui 2 de celule adiacente care corespund unor mintermi, pentru a obţine produse minimale. Un grup de 2 celule înseamnă dispariţia unei variabile din produs, un grup de 4 celule înseamnă dispariţia a 2 variabile din produs, un grup de 8 celule înseamnă dispariţia a 3 variabile din produs ş.a.m.d. Putem include o celulă care corespunde unui minterm în mai multe grupuri datorită idempotenţei, dacă astfel obţinem grupuri mai mari.

Metoda Quine–McCluskey este o metodă în 2 paşi. Mai întâi, cu ajutorul unor şiruri binare asociate mintermilor, se determină implicanţii primi. Apoi, dintre aceştia se selectează implicanţii primi esenţiali, folosind tablelul implicanţilor primi.

Simplificarea funcţiilor booleene este probabil cel mai important pas dintre toţi paşii care trebuie parcurşi de la identificarea funcţiei corespunzătoare unui probleme care apare în practică pană la construirea circuitului care furnizează soluţia respectivei probleme deoarece se urmăreşte crearea unui circuit minimal care rezolvă problema.

M3.U3.7. Test de evaluare a cunoştinţelor

1. Folosind diagrame Karnaugh, simplificaţi următoarele funcţii booleene:
    - $f(x_1, x_2, x_3) = \Sigma m(0, 2, 4, 6)$.
    - $f(x_1, x_2, x_3, x_4) = \Sigma m(2, 4, 6, 10, 14)$.
    - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 1, 2, 4, 6, 9, 12, 13)$.

2. Folosind diagrame Karnaugh, simplificaţi următoarele funcţii booleene:
    - $f(x_1, x_2, x_3) = \Pi M(0, 2, 4, 5)$.
    - $f(x_1, x_2, x_3, x_4) = \Pi M (2, 4, 6, 11, 12, 14)$.
    - $f(x_1, x_2, x_3, x_4) = \Pi M (0, 1, 2, 4, 8, 12, 13)$.

3. Folosind metoda Quine-McCluskey, simplificaţi următoarele funcţii booleene:
   - $f(x_1, x_2, x_3) = \Sigma m(1, 3, 5, 7)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(1, 2, 4, 6, 11, 12, 15)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 4, 5, 8, 9, 12, 14)$.

Unitatea de învăţare M3.U4. Funcţii incomplet definite

Cuprins

M3.U4.1. Introducere

Până acum ne-am ocupat de funcţii booleene complet definite, adică funcţii care au o valoare pentru orice posibilă combinaţie de valori ale variabilelor. Dar, atunci când determinăm funcţia booleană corespunzătoare unei anumite probleme care apare în practică este posibil să existe combinaţii de valori ale variabilelor pentru care valoarea funcţiei să nu conteze, deoarece acele combinaţii nu vor apărea niciodată. O asemenea funcţie se va numi funcţie boooleană incomplet definită.

În această unitate de învăţare vom studia simplificarea funcţiilor booleene incomplet definite. Vom arăta cum cele două metode de simplificare a funţiilor booleene complet definite – cu diagrame Karnaugh şi metoda Quine-McCluskey – pot fi adaptate pentru a simplifica funcţii booleene incomplet definite.

M3.U4.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a noţiunii de funcţie booleană incomplet definită şi a capacităţii de a simplifica funcţiile booleene incomplet definite folosind atât metode grafice – diagrame Karnaugh, cât şi metode analitice – metoda Quine-McCluskey.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Să simplifice funcţii booleene incomplet definite folosind diagrame Karnaugh;
- Să simplifice funcţii booleene incomplet definite folosind metoda Quine-McCluskey.

Durata medie de parcurgere a celei de-a patra unităţi de învăţare este de 3 ore.

M3.U4.3. Funcţii incomplet definite

În rezolvarea problemelor din lumea reală se ajunge adesea la folosirea funcţiilor incomplet definite.

Să presupunem că dorim să creăm un "detector de multiplu de 3". Acesta va trebui să returneze 1 pentru fiecare cifră zecimală care este un multiplu de 3 şi 0 pentru fiecare cifră zecimală care nu este multiplu de 3. Vom defini o funcţie booleană f de 4 variabile x_1, x_2, x_3, x_4, unde $x_1x_2x_3x_4$ este scrierea în baza 2 a cifrei zecimale pe care vrem să o testăm pentru a vedea dacă este sau nu multiplu de 3. Tabelul de adevăr al acestei funcţii este următorul:

x_1	x_2	x_3	x_4	$f(x_1, x_2, x_3, x_4)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

În acest tabel de adevăr, ultimele 6 combinaţii de valori ale variabilelor x_1, x_2, x_3 şi x_4 nu vor apărea deoarece ele nu reprezintă scrierea în baza 2 a niciunei cifre zecimale. În consecinţă, nu contează ce valoare are funcţia f în aceste cazuri. Spunem că valoarea funcţiei este *nespecificată* în aceste cazuri şi o vom nota cu x în tabelul de adevăr.

O funcţie cu cel puţin o valoare nespecificată se numeşte *funcţie incomplet definită.*

Funcţia booleană pentru detectorul de multiplu de 3 se poate scrie şi sub forma:

$$f(x_1, x_2, x_3, x_4) = \Sigma m(0, 3, 6, 9) + d(10, 11, 12, 13, 14, 15),$$

unde $d(10, 11, 12, 13, 14, 15)$ indică cele 6 combinaţii de valori ale variabilelor pentru care valoarea funcţiei nu este specificată.

M3.U4.4. Minimizarea funcţiilor incomplet definite

Pentru a minimiza o funcţie booleană incomplet definită putem folosi oricare din metodele de minimizare a funcţiilor booleene (complet definite) prezentate în unitatea anterioară de învăţare.

Putem folosi diagramele Karnaugh pentru a minimiza funcţii incomplet definite în acelaşi mod ca pentru minimizarea funcţiilor booleene cu o singură modificare: în loc de a căuta cele mai mari grupuri compacte de celule care 1, *vom identifica cele mai mari grupuri compacte de celule care conţin* 1 *şi, eventual,* x. Deci, în procesul de simplificare se pot folosi şi oricâţi dintre termenii ai căror coeficienţi nu sunt specificaţi: de la niciunul la toţi.

Exemplu

Diagrama Karnaugh pentru minimizarea funcţiei incomplet definite corespunzătoare detectorului de multiplu de 3 este următoarea:

$x_1\, x_2 \backslash x_3\, x_4$	00	01	11	10
00	1		1	
01				1
11	x	x	x	x
10		1	x	x

Deci, forma minimală a acestei funcţii este:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_2x_3x_4+x_2x_3\bar{x}_4+x_1x_4.$$

De asemenea, se poate utiliza metoda analitică Quine-McCluskey pentru a minimiza funcţii booleene incomplet definite. În acest caz, vom folosi termenii pentru care valoarea funcţiei nu este definită în acelaşi mod ca şi mintermii, cu o singură excepţie: la Pasul 2, aceştia vor fi adăugaţi în tabelul implicaţilor primi.

Exemplu

Vom aplica metoda Quine-McCluskey pentru a minimiza funcţia incomplet definită corespunzătoare detectorului de multiplu de 3.

Număr de elemente 1	Minterm	Şir binar
0	m_0	0000
2	m_3	0011
2	m_6	0110
2	m_9	1001
2	m_{10}	1010
2	m_{12}	1100
3	m_{11}	1011
3	m_{13}	1101
3	m_{14}	1110
4	m_{15}	1111

Aplicând operaţia de compunere tuturor perechilor de şiruri binare care diferă prin exact un element obţinem tabelul următor, care conţine şirurile obţinute în urma compunerilor şi şirul m_0 care nu a putut fi compus cu niciun alt şir.

Implicant	Operaţie de compunere
0000	m_0
-011	$m_3 \bullet m_{11}$
-110	$m_6 \bullet m_{14}$
10-1	$m_9 \bullet m_{11}$
1-01	$m_9 \bullet m_{13}$
101-	$m_{10} \bullet m_{11}$
1-10	$m_{10} \bullet m_{14}$
110-	$m_{12} \bullet m_{13}$
11-0	$m_{12} \bullet m_{14}$
1-11	$m_{11} \bullet m_{15}$
11-1	$m_{13} \bullet m_{15}$
111-	$m_{14} \bullet m_{15}$

Acest tabel nu este ireductibil, deci mai aplicăm o dată operaţia de compunere şi obţinem următorul tabel:

Implicant	Operaţie de compunere
0000	m_0
-011	$m_3 \bullet m_{11}$
-110	$m_6 \bullet m_{14}$
1--1	$m_9 \bullet m_{11} \bullet m_{13} \bullet m_{15}$
1-1-	$m_{10} \bullet m_{11} \bullet m_{14} \bullet m_{15}$
11--	$m_{12} \bullet m_{13} \bullet m_{14} \bullet m_{15}$

Din şirurile din acest tabel ireductibil determinăm implicanţii primi: $\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$, $\bar{x}_2x_3x_4$, $x_2x_3\bar{x}_4$, x_1x_4, x_1x_3, x_1x_2.
În continuare, vom determina tabelul implicanţilor primi, în care vom avea câte o linie pentru fiecare implicant prim şi câte o coloană pentru fiecare minterm (termenii pentru care valoarea funcţiei nu este specificată nu apar):

Implicant\minterm	$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	$\bar{x}_1\bar{x}_2x_3x_4$	$\bar{x}_1x_2x_3\bar{x}_4$	$x_1\bar{x}_2\bar{x}_3x_4$
$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$	*			
$\bar{x}_2x_3x_4$		*		
$x_2x_3\bar{x}_4$			*	
x_1x_4				*
x_1x_3				
x_1x_2				

În consecinţă, primii patru implicanţi primi sunt implicanţi primi esenţiali. Deci, forma minimală a acestei funcţii este:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_2x_3x_4+x_2x_3\bar{x}_4+x_1x_4.$$

M3.U4.5. Rezumat

În acestă unitate de învăţare am studiat funcţiile booleene incomplet definite. Acestea apar în modelarea problemelor practice. Se poate întâmpla ca anumite combinaţii de valori ale varibilelor funcţiei să nu apară niciodată din cauza unor condiţii externe. În aceste cazuri valoarea funcţiei nu este specificată. O funcţie care are cel puţin o valoare nespecificată este o funcţie incomplet definită.

Am arătat cum cele două metode de simplificare a funcţiilor booleene complet definite – cu diagrame Karnaugh şi metoda Quine-McCluskey – pot fi adaptate pentru a simplifica funcţiile booleene incomplet definite.

M3.U4.6. Test de evaluare a cunoştinţelor

1. Folosind diagrame Karnaugh, simplificaţi următoarele funcţii booleene incomplet definite:
   - $f(x_1, x_2, x_3) = \Sigma m(0, 4, 6) + d(1, 2)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(2, 4, 6, 10, 14) + d(1, 12, 15)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 1, 2, 4, 6, 9) + d(11, 12, 13)$.
2. Folosind metoda Quine-McCluskey, simplificaţi următoarele funcţii booleene incomplet definite:
   - $f(x_1, x_2, x_3) = \Sigma m(1, 5, 7) + d(2, 3)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(1, 2, 4, 6, 11, 12) + d(13, 14, 15)$.
   - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 4, 5, 8, 9, 10, 14) + d(11, 12, 13)$.

Unitatea de învăţare M3.U5. Circuite logice

Cuprins

M3.U5.1. Introducere

În acest modul vom studia implementarea funcţiilor boolene cu ajutorul circuitelor logice. Vom da o clasificare a circuitelor în funcţie de complexitatea lor, măsurată în numărul de porţi logice pe care le conţin. Circuitele mai complexe impun folosirea multiplexoarelor, care sunt componente care leagă un număr oarecare de intrări de una sau mai multe ieşiri.

M3.U5.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a modului de creare a circuitelor logice pentru rezolvarea problemelor care pot fi scrise sub formă de funcţii booleene.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- Determine circuitul logic corespunzător unei funcţii booleene date;
- Creeze şi să utilizeze multiplexoare.

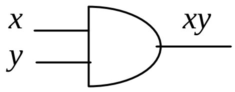
Durata medie de parcurgere a primei unităţi de învăţare este de 2 ore.

M3.U5.3. Circuite

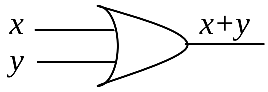
0 *poartă logică* este un dispozitiv care implementează o funcţie booleană simplă: AND, OR, NOT etc.

Simbolurile pentru porţile logice corespunzătoare funcţiilor booleene AND, OR şi NOT sunt următoarele:

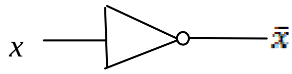
AND



OR

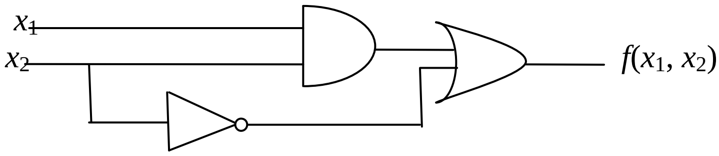


NOT



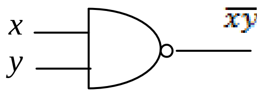
Exemplu
Determinăm circuitul logic corespunzător funcţiei booleene:

$$f(x_1, x_2) = x_1x_2+\bar{x}_2.$$

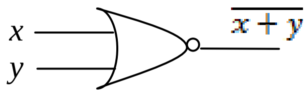


Alte simboluri pentru porţi logice:

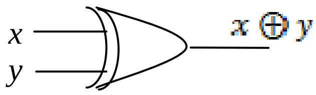
NAND



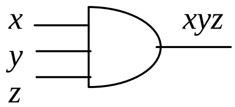
NOR



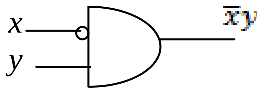
XOR



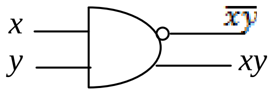
Acestea sunt formele de bază ale porţilor logice, însă ele pot fi modificate. De exemplu, pot exista mai multe intrări:



sau o intrare poate fi înlocuită cu negaţia ei:



sau poate exista şi o ieşire negată:



Numărul de porţi şi numărul de intrări în porţi sunt două măsuri ale complexităţii circuitelor. Pentru o orice funcţie, vom încerca să creăm un circuit de complexitate cât mai mică. Pentru aceasta, înainte de a crea circuitul, vom simplifica funcţia folosind una din metodele deja învăţate: cu diagrame Karnaugh sau aplicând metoda Quine-McCluskey.

În exemplul anterior, avem un circuit cu 3 porţi şi 5 intrări.

În funcţie de numărul de porţi pe care îl conţin, circuitele sunt:

1. SSI (Small Scale Integration) când se lucrează cu câteva porţi logice
2. MSI (Medium Scale Integration) când se lucrează cu 10 până la aproximativ 50 porţi logice
3. LSI (Large Scale Integration) când se lucrează cu 50 până la 1000 porţi logice
4. VLSI (Very Large Scale Integration) când se lucrează cu mai mult de 1000 porţi logice.

M3.U5.4. Multiplexoare

Un *multiplexor* (*MUX*) este o componentă a circuitelor logice care leagă un număr oarecare de intrări de una sau mai multe ieşiri.

Un multiplexor este folosit pe post de cutie neagră atunci când se creează circuite mai complexe (MSI, LSI sau VLSI).

Vom arăta utilitatea mutiplexoarelor, creând circuitul pentru adunarea a două numere binare reprezentate pe un nibble (4 biţi).

Pentru determinarea celui mai puţin semnificativ bit al sumei ar putea fi folosit un multiplexor cu două intrări:

1. cel mai puţin semnificativ bit al primului termen
2. cel mai puţin semnificativ bit al celui de-al doilea termen

şi două ieşiri:

1. cel mai puţin semnificativ bit al sumei
2. transportul către următorul bit.

Pentru ceilalţi biţi, trebuie folosit un *sumator*, care este un multiplexor cu 3 intrări:

1. un bit al primului termen
2. un bit al celui de-al doilea termen
3. transportul de la bitul anterior

şi două ieşiri:

1. un bit al sumei
2. transportul către următorul bit.

Deoarece se urmăreşte minimizarea numărului de componente diferite pe care le conţine un circuit, vom folosi şi pentru determinarea celui mai puţin semnificativ bit al sumei tot un sumator, considerând transportul iniţial 0.

Fie $a = a_1a_2a_3a_4$ şi $b = b_1b_2b_3b_4$ cei doi termeni, iar $s = s_1s_2s_3s_4 = a + b$ suma pe care vrem să o determinăm.

Tabelul de adevăr pentru o cifră oarecare sumei s_i şi pentru transportul c_{i+1} către următoarea cifră este:

a_i	b_i	c_i	$s_i(a_i, b_i, c_i)$	$c_{i+1}(a_i, b_i, c_i)$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Deci, forma normală disjunctivă a funcţiei $s_i(a_i, b_i, c_i)$ pentru determinarea cifrei s_i a sumei s este

$$s_i(a_i, b_i, c_i) = \bar{a}_i\bar{b}_ic_i + \bar{a}_ib_i\bar{c}_i + a_i\bar{b}_i\bar{c}_i + a_ib_ic_i,$$

iar forma normală disjunctivă a funcţiei $c_{i+1}(a_i, b_i, c_i)$ pentru determinarea transportului c_{i+1} este

$$c_{i+1}(a_i, b_i, c_i) = \bar{a}_ib_ic_i + a_i\bar{b}_ic_i + a_ib_i\bar{c}_i + a_ib_ic_i.$$

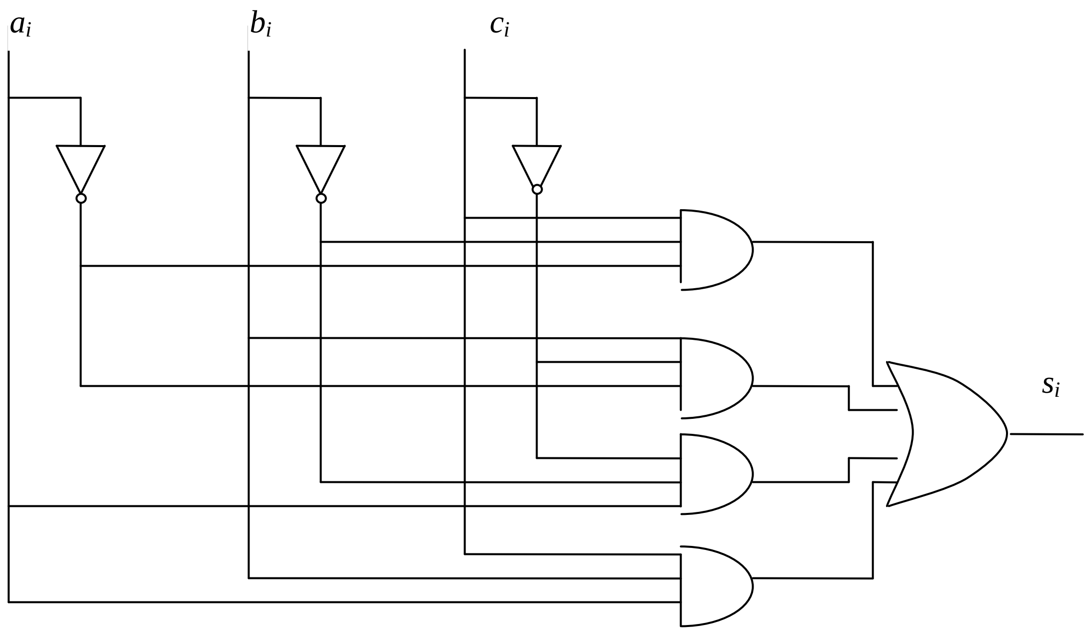
Încercăm să simplificăm funcţia $s_i(a_i, b_i, c_i)$ folosind următoarea diagramă Karnaugh:

$a_i \backslash b_ic_i$	00	01	11	10
0		1		1
1	1		1	

Se observă că toate celulele care conţin 1 sunt izolate, deci funcţia

$$s_i(a_i, b_i, c_i) = \bar{a}_i\bar{b}_ic_i + \bar{a}_ib_i\bar{c}_i + a_i\bar{b}_i\bar{c}_i + a_ib_ic_i$$

este deja în formă minimală şi nu mai poate fi simplificată. Circuitul logic corespunzător acestei funcţii este:



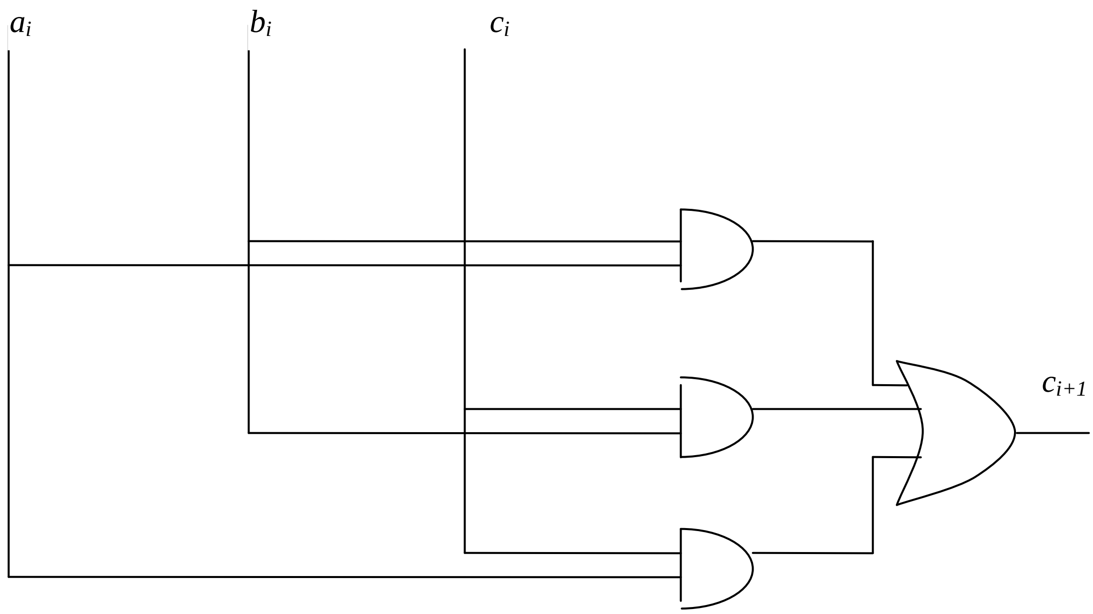
Vom simplifica funcţia $c_{i+1}(a_i, b_i, c_i)$ folosind următoarea diagramă Karnaugh:

$a_i \backslash b_i c_i$	00	01	11	10
0			1	
1		1	1	1

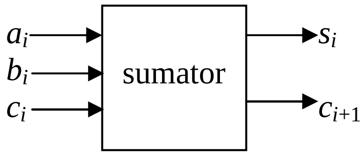
Deci, forma minimală a acestei funcţii este:

$c_{i+1}(a_i, b_i, c_i) = b_i c_i + a_i c_i + a_i b_i$.

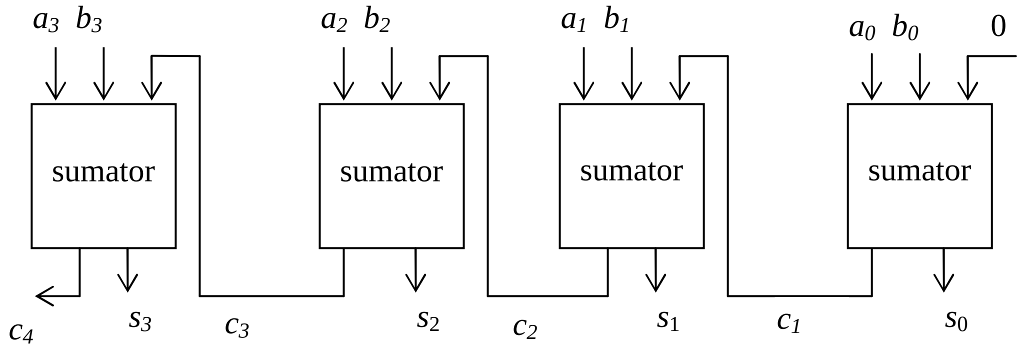
Circuitul logic corespunzător ei este următorul:



Vom "închide" cele 2 circuite de mai sus într-o cutie neagră, care este sumatorul, adică multiplexorul care adună 3 biţi şi produce un bit sumă şi un transport.



Folosind 4 sumatoare, putem determina următorul circuit care adună 2 numere binare pe 4 biţi:



M3.U5.5. Rezumat

În acest modul am studiat implementarea funcţiilor boolene cu ajutorul circuitelor logice. Crearea circuitului logic este ultima etapă dintre cele care se parcurg de la identificarea problemei până la rezolvarea ei, etape pe care le-am studiat în acest modul: scrierea problemei sub forma unei funcţii booleene, determinarea formei normale disjunctive (sau conjunctive), simplificarea funcţiei şi, în final, crearea circuitului corespunzător corespunzător formei minimale a funcţiei.

De asemenea, am clasificat circuitele în funcţie de complexitatea lor, măsurată în numărul de porţi logice pe care le conţin. Am arătat avantajele utilizării multiplexoarelor, care sunt componente care leagă un număr oarecare de intrări de una sau mai multe ieşiri, în circuitele mai complexe.

M3.U5.6. Test de evaluare a cunoştinţelor

Determinaţi circuitele logice corespunzătoare următoarelor funcţii:

- $f_1(x_1, x_2) = x_1x_2+\bar{x}_2$
- $f_1(x_1, x_2, x_3) = x_1x_2+\bar{x}_1x_2\, x_3$
- $f_1(x_1, x_2, x_3) = x_1x_2+\bar{x}_1x_2\bar{x}_3+x_2$

Temă de control

1. Determinaţi forma normală disjunctivă pentru următoarele funcţii booleene:
   - $f(x_1, x_2) = \bar{x}_2+x_1$.

- $f(x_1, x_2, x_3) = \bar{x}_2+x_1\bar{x}_3+x_1x_3$.

2. Determinaţi forma normală disjunctivă şi forma normală conjunctivă a funcţiei booleene date prin următorul tabel de adevăr:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

3. Folosind diagrame Karnaugh, simplificaţi următoarele funcţii booleene:
    - $f(x_1, x_2, x_3) = \Sigma m(0, 2, 4, 6)$.
    - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 1, 2, 4, 6, 9, 12, 13)$.

Apoi determinaţi circuitele logice corepsunzătoare formelor lor minimale.

4. Folosind diagrame Karnaugh, simplificaţi următoarele funcţii booleene:
    - $f(x_1, x_2, x_3) = \Pi M(0, 2, 4, 5)$.
    - $f(x_1, x_2, x_3, x_4) = \Pi M (2, 4, 6, 11, 12, 14)$.

Apoi determinaţi circuitele logice corepsunzătoare formelor lor minimale.

5. Folosind metoda Quine-McCluskey, simplificaţi următoarele funcţii booleene:
    - $f(x_1, x_2, x_3) = \Sigma m(1, 3, 5, 7)$.
    - $f(x_1, x_2, x_3, x_4) = \Sigma m(0, 4, 5, 8, 9, 12, 14)$.

Apoi determinaţi circuitele logice corepsunzătoare formelor lor minimale.

Modulul 4. Expresii Reed-Müller

Cuprins

Introducere

În modulul anterior am arătat cum putem determina circuitele logice care rezolvă probleme care pot fi exprimate prin funcţii booleene. În consecinţă, ne bazam pe algebra booleană. În locul algebrei booleene, putem folosi algebra modulo-2 şi obţinem o altă modalitate de descriere a problemelor. În acestă nouă scriere, care este expresia Reed-Müller, intervin doar doi operatori: adunarea modulo-2 şi înmulţirea modulo-2. Ca urmare, circuitele logice vor fi formate doar din porţile logice corespunzătoare acestor doi operatori.

Competenţe

La sfârşitul acestui modul studenţii vor fi capabili să:

- determine expresia Reed-Müller corespunzătoare unei funcţii booleene;
- determine funcţia booleană corespunzătoare unei expresii Reed-Müller;
- determine expresia Reed-Müller generalizată de orice polaritate corespunzătoare unei expresii Reed-Müller;
- determine expresia Reed-Müller generalizată de orice polaritate corespunzătoare unei funcţii booleene.

Unitatea de învăţare M4.U1. Expresii Reed-Müller

Cuprins

M4.U1.1. Introducere

În modulul anterior am arătat cum se pot crea circuite logice pentru funcţii booleene. O alternativă la scrierea funcţiilor folosind algebra booleană ar fi scrierea care utilizează operaţii din algebra modulo-2.

Operaţii fundamentale din algebra modulo-2:

		b	
	$\oplus$	0	1
a	0	0	1
	1	1	0

adunarea modulo-2

		b	
	$\otimes$	0	1
a	0	0	0
	1	0	1

înmulţirea modulo-2

şi operaţiile echivalente cu ele din algebra booleană:

		b	
	$\oplus$	0	1
a	0	0	1
	1	1	0

XOR

		b	
	$\cdot$	0	1
a	0	0	0
	1	0	1

AND

Din tabelele de adevăr ale acestor operaţii fundamentale, rezultă că singura diferenţă dintre descrierea care foloseşte algebra modulo-2 şi cea care foloseşte algebra booleană este diferenţa dintre operaţiile booleene XOR şi OR. Această diferenţă constă în faptul că 1+1=1 în algebra booleană, pe când $1\oplus1=0$ în algebra modulo-2. Această diferenţă aparent nesemnificativă va duce la diferenţe semnificative între cele două algebre şi la modificări majore ale metodelor de creare şi implementare a circuitelor bazate pe cele două algebre.

M4.U1.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a conceptului de expresie Reed-Müller.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- determine expresia Reed-Müller corespunzătoare unei funcţii booleene;
- determine funcţia booleană corespunzătoare unei expresii Reed-Müller.

Durata medie de parcurgere a primei unităţi de învăţare este de 2 ore.

M4.U1.3. Algebra GF(2)

Algebra modulo-2 va fi notată cu GF(2).

Dacă a, b şi c sunt variabile bivalente şi dacă $\oplus$ şi $\otimes$ reprezintă adunarea şi înmulţirea modulo-2 atunci au loc următoarele proprietăţi:

1. închiderea:
   $a\oplus b$ şi $a\otimes b$ sunt tot bivalente

2. comutativitatea:
   $a\oplus b = b\oplus a$
   $a\otimes b = b\otimes a$

3. asociativitatea:
   $a\oplus(b\oplus c) = (a\oplus b)\oplus c$
   $a\otimes(b\otimes c) = (a\otimes b)\otimes c$

4. distributivitatea:
   $a\otimes(b\oplus c) = a\otimes b\oplus a\otimes c$

5. elemente neutre:
   $a\oplus 0 = a$
   $a\otimes 1= a$

   Până aici sunt similarităţi cu algebra numerelor reale.

6. elemente inverse:
   $a\oplus a = 0$, deci $a = -a$, adică orice element este propriul său invers în rapot cu adunarea modulo-2. În consecinţă, în GF(2) adunarea şi scăderea coincid.
   $a\otimes a=a$

Este interesant să vedem care este legătura dintre algebra GF(2) şi algebra booleană.

Putem exprima operatorii din algebra GF(2) folosind operatorii din algebra booleană astfel:

$a \otimes b = ab$

$a \oplus b = a\bar{b} + \bar{a}b$

Reciproc, pentru a exprima operatorii din algebra booleană folosind operatorii din algebra GF(2), mai întâi observăm că

$a \oplus 1 = a \cdot \bar{1} + \bar{a} \cdot 1 = \bar{a}$.

Deoarece, conform legii lui de Morgan avem că: $a + b = \overline{\bar{a} \cdot \bar{b}}$, rezultă că

$a + b = (a \oplus 1) \otimes (b \oplus 1) \oplus 1$.

Deci,

$ab = a \otimes b$

$a + b = (a \oplus 1) \otimes (b \oplus 1) \oplus 1$

$\bar{a} = a \oplus 1$.

În consecinţă, toţi operatorii algebrei booleene pot fi exprimaţi folosind operatorii din algebra modulo-2.

În continuare nu vom mai face distincţie între AND logic şi înmulţirea modulo-2 şi vom folosi pentru ambii operatori acelaşi simbol, ·, sau doar juxtapunerea.

M4.U1.4. Expresii Reed-Müller

Funcţia booleană standard de o variabilă, x_1, poate fi scrisă sub forma:

$f(x_1) = d_0 \bar{x}_1 + d_1 x_1$,

unde coeficienţii d_i, pentru $i = 0, 1$ pot fi 0 sau 1 şi sunt valorile din tabelul de adevăr.

Deoarece x_1 şi $\bar{x}_1$ nu pot fi simultan egali cu 1, putem înlocui OR (+) cu XOR(⊕) fără să modificăm valoarea expresiei.

$f(x_1) = d_0 \bar{x}_1 \oplus d_1 x_1 = d_0(x_1 \oplus 1) \oplus d_1 x_1 = d_0 \oplus (d_0 \oplus d_1)x_1$.

Deci, funcţia poate fi scrisă în forma echivalentă:

$f(x_1) = c_0 \oplus c_1 x_1$ peste $GF(2)$, unde coeficienţii c_i , pentru $i = 0, 1$ pot fi 0 sau 1.

Coeficienţii c_i, pentru $i = 0, 1$ ai acestei noi expresii <u>nu</u> sunt egali cu valorile din tabelul de adevăr, dar pot fi determinaţi din acestea folosind matricea:

$$T_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Putem scrie

$c = T_1 d$ peste $GF(2)$,

unde $c = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$, $d = \begin{pmatrix} d_0 \\ d_1 \end{pmatrix}$ şi toate operaţiile se fac în algebra modulo-2.

Acestă transformare pe care am făcut-o asupra funcţiilor booleene de o variabilă poate fi aplicată şi funcţiilor de 2 sau mai multe variabile.

Forma normală disjunctivă a unei funcţii booleene de două variabile x_1 şi x_2 este:

$$f(x_2, x_1) = d_0\,\bar{x}_2\bar{x}_1 + d_1\,\bar{x}_2x_1 + d_2\,x_2\bar{x}_1 + d_3\,x_2x_1,$$

unde coeficienţii d_i, pentru $i = 0,...3$, pot fi 0 sau 1 şi sunt valorile din tabelul de adevăr.

Deoarece oricare doi mintermi nu pot fi simultan egali cu 1, putem înlocui OR (+) cu XOR(⊕) fără să modificăm valoarea expresiei:

$$f(x_2, x_1) = d_0\,\bar{x}_2\bar{x}_1 \oplus d_1\,\bar{x}_2x_1 \oplus d_2\,x_2\bar{x}_1 \oplus d_3\,x_2x_1 =$$

$= d_0\,(x_2\oplus 1)\,(x_1\oplus 1) \oplus d_1(x_2\oplus 1)x_1 \oplus d_2x_2(x_1\oplus 1) \oplus d_3\,x_2x_1 =$

$= d_0\,x_2x_1 \oplus d_0\,x_2 \oplus d_0\,x_1 \oplus d_0 \oplus d_1x_2x_1 \oplus d_1\,x_1 \oplus d_2\,x_2x_1 \oplus d_2\,x_2 \oplus d_3\,x_2x_1 =$

$= d_0 \oplus (d_0 \oplus d_1)\,x_1 \oplus (d_0 \oplus d_2)\,x_2 \oplus (d_0 \oplus d_1 \oplus d_2 \oplus d_3)\,x_2x_1.$

Deci, funcţia poate fi scrisă în forma echivalentă:

$f(x_2, x_1) = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus c_3x_2x_1$ peste $GF(2)$, unde coeficienţii c_i, pentru $i = 0,...,3$ pot fi 0 sau 1 şi pot fi determinaţi astfel:

$$c = T_2\,d \text{ peste } GF(2),$$

unde $c = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$, $d = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$ şi $T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$.

Se poate observa că $T_2 = \begin{pmatrix} T_1 & 0 \\ T_1 & T_1 \end{pmatrix}$.

În general, pentru n variabile matricea de transformare T_n este:

$$T_n = \begin{pmatrix} T_{n-1} & 0 \\ T_{n-1} & T_{n-1} \end{pmatrix} \quad \text{pentru } n > 0$$

$$T_0 = (1)$$

O funcţie de n variabile poate fi scrisă:

$f(x_n, x_{n-1},..., x_1) = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus c_3x_2x_1 \oplus ... \oplus c_{2^n-1}x_nx_{n-1}...\,x_1$ peste $GF(2)$, unde coeficienţii c_i, pentru $i = 0,...,2^n - 1$ pot fi 0 sau 1 şi pot fi determinaţi astfel:

$$c = T_n\,d \text{ peste } GF(2),$$

unde $c = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2^n-1} \end{pmatrix}$ şi $d = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{2^n-1} \end{pmatrix}$.

Scrierea funcţiei f ca sumă de produse modulo-2 se numeşte *expresie Reed-Müller*.

O expresie Reed-Müller poate fi scrisă şi în următoarea formă

$$f(x_n, x_{n-1},..., x_1) = \sum_{i=0}^{2^n-1} c_i x_n^{e_{i,n}} x_{n-1}^{e_{i,n-1}} ... x_1^{e_{i,1}},$$

unde suma se face peste $GF(2)$, iar exponenţii $e_{i,j}$ pot fi 0 sau 1, unde $x_j^0 = 1$ şi $x_j^1 = x_j$.

Într-o expresie Reed-Müller a unei funcţii de n variabile pot fi cel mult 2^n termeni, care se numesc π*-termi* sau *pitermi* şi se notează π_i. Indicele i este echivalentul zecimal al numărului binar $e_{i,n}\, e_{i,n-1} \ldots e_{i,1}$.

Deci, expresia Reed-Müller poate fi scrisă şi sub forma

$$f(x_n, x_{n-1}, \ldots, x_1) = \sum_{i=0}^{2^n-1} c_i \pi_i \text{ , peste } GF(2).$$

Observaţii 1) $\pi_{2^{i-1}} = x_i$ pentru $i = 1, \ldots, n$.

2) $\pi_0 = 1$

Exemplu

Vom determina expresia Reed-Müller corespunzătoare funcţii booleene:

$$f(x_3, x_2, x_1) = \Sigma m(1, 2, 5, 7).$$

Avem $d = (0, 1, 1, 0, 0, 1, 0, 1)$ şi ştim că matricea de transformare T_3 pentru funcţii de 3 variabile este:

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Coeficienţii c ai expresiei Reed-Müller se obţin din următoarea înmulţire peste algebra GF(2):

$$c = T_3 \cdot d = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Deci, expresia Reed-Müller corespunzătoare funcţii booleene:

$$f(x_3, x_2, x_1) = \Sigma m(1, 2, 5, 7)$$

este

$$f(x_3, x_2, x_1) == x_1 \oplus x_2 \oplus x_3 x_2.$$

Determinaţi expresiile Reed-Müller ale următoarelor funcţii booleene:

- $f_1(x_2, x_1) = \Sigma m(0, 2)$
- $f_1(x_3, x_2, x_1) = \Sigma m(0, 2, 4)$
- $f_1(x_3, x_2, x_1) = \Sigma m(1, 2, 4, 7)$.

M4.U1.5. Domeniul operaţional şi domeniul funcţiei

Coeficienţii expresiei Reed-Müller nu sunt egali cu valorile din tabelul de adevăr. De aceea sunt necesare două forme de reprezentare.

1. Tabelul de adevăr şi diagrama Karnaugh arată comportarea operaţională a funcţiei. Ca urmare, valorile din tabelul de adevăr (diagrama Karnaugh) vor forma *domeniul operaţional*. *Tabelul domeniului operaţional* are aceeaşi structură ca diagrama Karnaugh.
2. Coeficienţii expresiei Reed-Müller formează *domeniul funcţiei* şi vor fi scrişi în *tabelul domeniului funcţiei*. Acesta are o structură asemănătoare cu cea a diagramei Karnaugh, cu deosebirea că etichetarea celulelor se va face în alt mod. Şi acest tabel este împărţit în două părţi egale, în moduri diferite pentru fiecare variabilă (ca şi în cazul diagramelor Kanaugh). O parte conţine celulele corespunzătoare pitermilor care conţin acea variabilă, iar celaltă parte conţine celulele corespunzătoare pitermilor care nu conţin acea variabilă. În acestea din urmă, variabila va fi considerată 1 în calculul produsului.

Putem aşeza mintermii şi pitermii în cele două tabele astfel încât să ocupe aceleaşi poziţii relative.

Pentru funcţii de 2 variabile:

	$\overline{x}_1$	x_1
$\overline{x}_2$	d_0	d_1
x_2	d_1	d_2

Domeniul operaţional

	1	x_1
1	c_0	c_1
x_2	c_2	c_3

Domeniul funcţiei

Pentru funcţii de 3 variabile:

	$\overline{x}_1$	x_1	x_1	$\overline{x}_1$
$\overline{x}_3$	d_0	d_1	d_3	d_2
x_3	d_4	d_5	d_7	d_6
	$\overline{x}_2$	$\overline{x}_2$	x_2	x_2

Domeniul operaţional

	1	x_1	x_1	1
1	c_0	c_1	c_3	c_2
x_3	c_4	c_5	c_7	c_6
	1	1	x_2	x_2

Domeniul funcţiei

Pentru funcţii de 4 variabile:

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$	
$\bar{x}_4$	d_0	d_1	d_3	d_2	$\bar{x}_3$
$\bar{x}_4$	d_4	d_5	d_7	d_6	x_3
x_4	d_{12}	d_{13}	d_{15}	d_{14}	x_3
x_4	d_8	d_9	d_{10}	d_{11}	$\bar{x}_3$
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2	

Domeniul operaţional

	1	x_1	x_1	1	
1	c_0	c_1	c_3	c_2	1
1	c_4	c_5	c_7	c_6	x_3
x_4	c_{12}	c_{13}	c_{15}	c_{14}	x_3
x_4	c_8	c_9	c_{10}	c_{11}	1
	1	1	x_2	x_2	

Domeniul funcţiei

Exemplu

Se dă domeniul operaţional al unei funcţii de 3 variabile:

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	0	1	1	0
x_3	0	0	1	0
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

Vom determina domeniul funcţiei folosind matricea de transformare

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Extragem din domeniul operaţional vectorul d cu valorile din tabelul de adevăr $d = (0, 1, 0, 1, 0, 0, 0, 1)$.

Coeficienţii c care formează domeniul funcţiei se obţin din următoarea înmulţire peste algebra GF(2):

$$c = T_3 \cdot d = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Deci, domeniul funcţiei va fi

	1	x_1	x_1	1
1	0	1	0	0
x_3	0	1	1	0
	1	1	x_2	x_2

1. Determinaţi domeniul funcţiei ştiind că domeniul operaţional este

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	0	0	1	1
x_3	0	1	0	1
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

2. Determinaţi domeniul funcţiei ştiind că domeniul operaţional este

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$	
$\bar{x}_4$	0	1	0	1	$\bar{x}_3$
$\bar{x}_4$	1	0	1	0	x_3
x_4	0	0	1	0	x_3
x_4	1	0	0	0	$\bar{x}_3$
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2	

Cunoscând coeficienţii din domeniul operaţional, putem determina domeniul funcţiei folosind matricea de transformare T_n, unde n este numărul de variabile ale funcţiei.

Vom arăta cum se determină domeniul operaţional sau, echivalent, tabelul de adevăr, al unei funcţii date prin expresia Reed-Müller.

Expresia Reed-Müller pentru o variabilă are forma:

$$f(x_1) = c_0 \oplus c_1x_1$$

Putem determina valorile din tabelul de adevăr, d, în modul următor:

$$d_0 = f(0) = c_0$$

$$d_1 = f(1) = c_0 \oplus c_1$$

Ecuaţiile de mai sus pot fi scrise şi în următoarea formă:

$d = T_1\, c$ peste $GF(2)$,

unde $c = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$, $d = \begin{pmatrix} d_0 \\ d_1 \end{pmatrix}$, $T_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ şi toate operaţiile se fac în algebra modulo-2.

Expresia Reed-Müller pentru 2 variabile este următoarea:

$$f(x_2, x_1) = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus c_3x_2x_1$$

Putem determina valorile din tabelul de adevăr, d, în modul următor:

$$d_0 = f(0, 0) = c_0$$

$d_1 = f(0, 1) = c_0 \oplus c_1$
$d_2 = f(1, 0) = c_0 \oplus c_2$
$d_3 = f(1, 1) = c_0 \oplus c_1 \oplus c_2 \oplus c_3$

Ecuaţiile de mai sus pot fi scrise şi în următoarea formă:

$d = T_2\, c$ peste $GF(2)$,

unde $c = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$, $d = \begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix}$, $T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$ şi toate operaţiile se fac în algebra modulo-2.

În general, pentru n variabile, expresia Reed-Müller este:

$f(x_n, x_{n-1},\ldots, x_1) = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus c_3x_2x_1 \oplus \ldots \oplus c_{2^n-1}x_nx_{n-1}\ldots x_1$ peste $GF(2)$.

Putem determina valorile din tabelul de adevăr d_i , pentru $i = 0,\ldots,2^n-1$, folosind matricea de transformare T_n:

$d = T_n\, c$ peste $GF(2)$,

unde $c = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2^n-1} \end{pmatrix}$, $d = \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{2^n-1} \end{pmatrix}$ şi T_n este:

$T_n = \begin{pmatrix} T_{n-1} & 0 \\ T_{n-1} & T_{n-1} \end{pmatrix}$ pentru $n > 0$

$T_0 = (1)$.

Observaţie Transformarea de la domeniul operaţional la domeniul funcţiei este identică cu transformarea inversă, de la domeniul funcţiei la domeniul operaţional.

Exemplu

Se dă domeniul funcţiei al unei funcţii de 3 variabile:

	1	x_1	x_1	1
1	0	1	1	1
x_3	0	0	1	0
	1	1	x_2	x_2

Vom determina domeniul funcţiei folosind matricea de transformare

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Extragem din domeniul funcţiei vectorul c, care conţine coeficienţii expresiei Reed-Müller $c = (0, 1, 1, 1, 0, 0, 0, 1)$.

Vectorul d cu valorile din tabelul de adevăr se obţine din următoarea înmulţire peste algebra GF(2):

$$d = T_3 \cdot c = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Deci, domeniul operaţional va fi

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	0	1	1	1
x_3	0	1	0	1
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

1. Determinaţi domeniul operaţional ştiind că domeniul funcţiei este

	1	x_1	x_1	1
1	0	1	1	0
x_3	0	0	1	0
	1	1	x_2	x_2

2. Determinaţi domeniul operaţional ştiind că domeniul funcţiei este

	1	x_1	x_1	1	
1	0	1	1	0	1
1	0	0	0	1	x_3
x_4	0	1	0	0	x_3
x_4	0	0	1	0	1
	1	1	x_2	x_2	

O altă metodă, mai simplă, de trecere de la domeniul operaţional la domeniul funcţiei sau invers, de la domeniul funcţiei la domeniul operaţional, se obţine pornind de la structura triunghiulară a matricii transformării T_n. Putem construi un triunghi al lui Pascal inversat în modul următor:

1. pe prima linie scriem cei 2^n coeficienţi pe care îi cunoaştem (d_i sau c_i, pentru i = 0,...,2^n-1).
2. Formăm următoarea linie, care va conţine sumele modulo-2 ale coeficienţilor de deasupra.
3. Repetăm acest procedeu până ajungem la o linie formată dintr-un singur element.

Se poate observa că primul element de pe linia i, pentru i = 0,...,2^n-1 a fost calculat exact în acelaşi fel în care se calculează d_i cunoscând d sau c_i cunoscând d, dacă se foloseşte matricea transformării T_n.

Exemplu

Vom utiliza metoda care foloseşte un triunghi al lui Pascal inversat pentru a determina domeniul funcţiei cunoscând domeniul operaţional:

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	1	0	1	0
x_3	0	0	1	1
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

Extragem din domeniul operaţional valorile din tabelul de adevăr:

d = (1, 0, 0, 1, 0, 0, 1, 1).

Formăm următorul triunghi inversat al lui Pascal:

1 0 0 1 0 0 1 1
1 0 1 1 0 1 0
1 1 0 1 1 1
0 1 1 0 0
1 0 1 0
1 1 1
0 0
0

Coeficienţii c care vor forma domeniul funcţiei se extrag de pe prima poziţie a fiecărei linii a triunghiului. Deci, c = (1, 1, 1, 0, 1, 1, 0, 0), iar domeniul funcţiei va fi:

	1	x_1	x_1	1
1	1	1	0	1
x_3	1	1	0	0
	1	1	x_2	x_2

1. Folosind triunghiul inversat al lui Pascal, determinaţi domeniul funcţiei ştiind că domeniul operaţional este

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	0	0	1	1
x_3	0	1	0	1
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

2. Folosind triunghiul inversat al lui Pascal, determinaţi domeniul funcţiei ştiind că domeniul operaţional este

	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$	
$\bar{x}_4$	0	1	0	1	$\bar{x}_3$
$\bar{x}_4$	1	0	1	0	x_3
x_4	0	0	1	0	x_3
x_4	1	0	0	0	$\bar{x}_3$
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2	

Deoarece, transformarea de la domeniul operaţional la domeniul funcţiei este identică cu transformarea inversă de la domeniul funcţiei la domeniul operaţional, putem aplica metoda triunghiului inversat al lui Pascal şi pentru a determina domeniul operaţional din domeniul funcţiei.

Exemplu

Vom utiliza metoda care foloseşte un triunghi al lui Pascal inversat pentru a determina domeniul operaţional cunoscând domeniul funcţiei:

	1	x_1	x_1	1
1	1	1	0	1
x_3	1	0	1	1
	1	1	x_2	x_2

Extragem din domeniul funcţiei coeficienţii expresiei Reed-Müller:

$c = (1, 1, 1, 0, 1, 0, 1, 1)$.

Formăm următorul triunghi inversat al lui Pascal:

1 1 1 0 1 0 1 1
0 0 1 1 1 1 0
0 1 0 0 0 1
1 1 0 0 1
0 1 0 1
1 1 1
0 0
0

Valorile din tabelul de adevăr *d* care vor forma domeniul operaţional se extrag de pe prima poziţie a fiecărei linii a triunghiului. Deci, *d* = (1, 0, 1, 0, 1, 0, 1, 1), iar domeniul operaţional va fi:

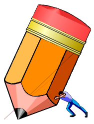
	$\bar{x}_1$	x_1	x_1	$\bar{x}_1$
$\bar{x}_3$	1	0	1	0
x_3	1	0	1	1
	$\bar{x}_2$	$\bar{x}_2$	x_2	x_2

M4.U1.6. Rezumat

În această unitate de învăţare am arătat că algebra booleană nu este singura algebră pe care o putem folosi pentru scrierea funcţiilor. O descriere alternativă a fost obţinută înlocuind algebra booleană cu algebra modulo-2. Aparent diferenţa dintre cele două descrieri este minoră. Această diferenţă constă în faptul că 1+1=1 în algebra booleană, pe când $1\oplus1=0$ în algebra modulo-2. Această diferenţă aparent nesemnificativă duce la diferenţe semnificative între cele două algebre şi între descrierile funcţiilor bazate pe cele două algebre.

Expresiile Reed-Müller sunt descrierile funcţiilor bazate pe algebra modulo-2, în care intervin doar doi operatori: adunarea modulo-2 şi înmulţirea modulo-2. Ca urmare, circuitele logice vor fi formate doar din porţile logice corespunzătoare acestor doi operatori.

Deoarece coeficienţii expresiei Reed-Müller nu sunt egali cu valorile din tabelul de adevăr, sunt necesare două forme de reprezentare: domeniul operaţional, care conţine valorile din tabelul de adevăr şi domeniul funcţiei care conţine coeficienţii expresiei Reed-Müller. Am arătat că transformarea pe care trebuie să o aplicăm pentru a trece de la un domeniu la celălalt este identică în ambele sensuri.



M4.U1.7.Test de evaluare a cunoştinţelor

1. Determinaţi expresiile Reed-Müller corespunzătoare funcţiilor booleene:
   - $f_1(x_2, x_1) = \Sigma m(0, 2)$
   - $f_1(x_3, x_2, x_1) = \Sigma m(0, 2, 4)$
   - $f_1(x_3, x_2, x_1) = \Sigma m(1, 2, 5, 7)$.
2. Determinaţi funcţiile boolene corespunzătoare expresiilor Reed-Müller:
   - $f(x_2, x_1) = 1 \oplus x_1 \oplus x_2x_1$
   - $f(x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_3x_1 \oplus x_3x_2.$

Unitatea de învăţare M4.U2. Expresii Reed-Müller generalizate

Cuprins

M4.U2.1. Introducere

În unitatea anterioară de învăţare am studiat expresiile Reed-Müller, care sunt forme canonice constând în sume de produse modulo-2. Dar, expresiile Reed-Müller nu sunt singura modalitate de scriere a funcţiilor booleene ca sume de produse modulo-2. Expresiile Reed-Müller generalizate constituie o altă scriere a funcţiilor booleene ca sume de produse modulo-2.

M4.U1.2. Obiectivele unităţii de învăţare

Această unitate de învăţare îşi propune ca obiectiv principal însuşirea de către studenţi a conceptului de expresie Reed-Müller generalizată.

La sfârşitul acestei unităţi de învăţare studenţii vor fi capabili să:

- determine expresia Reed-Müller generalizată de orice polaritate corespunzătoare unei expresii Reed-Müller;
- determine expresia Reed-Müller generalizată de orice polaritate corespunzătoare unei funcţii booleene.

Durata medie de parcurgere a acestei unităţi de învăţare este de 2 ore.

M4.U1.3. Expresii Reed-Müller generalizate

Expresia Reed-Müller, care este o sumă de produse modulo-2, este o formă canonicaă a funcţiei. Această formă canonică nu este unică. Putem înlocui orice variabilă x_i cu $\bar{x}_i$ şi vom obţine o altă formă canonică.

Pentru o funcţie de n variabile, există 2^n astfel de substituţii, care vor duce la obţinerea a 2^n forme canonice. Acestea se numesc *expresii Reed-Müller generalizate* şi fiecare dintre ele poate fi identificată printr-un număr, numit *polaritate*. Acest număr este echivalentul zecimal al numărului binar format din n biţi, format prin scrierea unui 0 sau a unui 1 pentru fiecare variabilă care apare sau, respectiv, este înlocuită cu negaţia ei.

Orice expresie Reed-Müller poate fi considerată o expresie Reed-Müller generalizată de polaritate 0, deoarece ea nu conţine negaţia niciunei variabile.

O metodă de determinare a expresiilor Reed-Müller generalizate constă în aplicarea câte unei transformări asupra domeniului operaţional al funcţiei pentru fiecare polaritate in parte. Dar, este mai eficient să determinăm o expresie Reed-Müller generalizată de o anumită polaritate aplicând o transformare asupra expresiei Reed-Müller, sau, echivalent, asupra domeniului funcţiei.

Pentru o variabilă, expresia Reed-Müller generalizată are următoarea formă:

$$f(\dot{x}_1) = c_0 \oplus c_1\dot{x}_1$$

unde $\dot{x}_1$ este fie x_1 fie $\overline{x}_1$.

Pentru a determina expresia Reed-Müller generalizată de polaritate 0 vom considera $\dot{x}_1 = x_1$:

$$f(\dot{x}_1) = c_0 \oplus c_1\, x_1$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_1) = a_0 \oplus a_1x_1,$$

unde $a_0 = c_0$ and $a_1 = c_1$. Deci, aşa cum era de aşteptat, expresia Reed-Müller generalizată de polaritate 0 se obţine din expresia Reed-Müller, matricea de transformare fiind matricea unitate, pe care o notăm cu Z_0. Deci,

$$a = Z_0\, c \text{ peste } GF(2),$$

unde $c = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}$, $a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$, $Z_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ şi toate operaţiile se fac în algebra modulo-2.

Pentru a determina expresia Reed-Müller generalizată de polaritate 1 vom considera $\dot{x}_1 = \overline{x}_1$:

$$f(\dot{x}_1) = c_0 \oplus c_1\, \overline{x}_1 = c_0 \oplus c_1\, (x_1 \oplus 1) = c_0 \oplus c_1 \oplus c_1x_1.$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_1) = a_0 \oplus a_1x_1,$$

unde $a_0 = c_0 \oplus c_1$ şi $a_1 = c_1$. Deci, coeficienţii a ai expresiei Reed-Müller generalizate de polaritate 1 se obţin din coeficienţii expresiei Reed-Müller, folosind matricea de transformare Z_1:

$$a = Z_1\, c \text{ peste } GF(2),$$

unde $Z_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.

Pentru două variabile, expresia Reed-Müller generalizată are următoarea formă:

$$f(\dot{x}_2, \dot{x}_1) = c_0 \oplus c_1\dot{x}_1 \oplus c_2\dot{x}_2 \oplus c_3\dot{x}_2\dot{x}_1$$

unde $\dot{x}_i$ este fie x_i fie $\overline{x}_i$, i=1, 2.

Pentru a determina expresia Reed-Müller generalizată de polaritate 0 vom considera $\dot{x}_2 = x_2$ şi $\dot{x}_1 = x_1$. Atunci

$$f(\dot{x}_2, \dot{x}_1) = c_0 \oplus c_1x_1 \oplus c_2x_2 \oplus c_3x_2x_1$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_2, x_1) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_2x_1,$$

unde

$$a_0 = c_0,$$
$$a_1 = c_1,$$
$$a_2 = c_2,$$
$$a_3 = c_3.$$

Deci, aşa cum era de aşteptat, expresia Reed-Müller generalizată de polaritate 0 se obţine din expresia Reed-Müller, considerând matricea de transformare ca fiind matricea unitate:

$$a = Z_{00}\, c \text{ peste } GF(2),$$

unde $c = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix}$, $a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$, $\mathrm{Z}_{00} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ şi toate operaţiile se fac în algebra modulo-2.

Pentru a determina expresia Reed-Müller generalizată de polaritate 1 vom considera $\dot{x}_2 = x_2$ şi $\dot{x}_1 = \bar{x}_1$:

$$f(\dot{x}_2, \dot{x}_1) = c_0 \oplus c_1\bar{x}_1 \oplus c_2x_2 \oplus c_3x_2\bar{x}_1 =$$
$$= c_0 \oplus c_1(x_1 \oplus 1) \oplus c_2x_2 \oplus c_3x_2(x_1 \oplus 1) =$$
$$= c_0 \oplus c_1 \oplus c_1x_1 \oplus (c_2 \oplus c_3)\, x_2 \oplus c_3x_2x_1$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_2, x_1) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_2x_1,$$

unde

$$a_0 = c_0 \oplus c_1,$$
$$a_1 = c_1,$$
$$a_2 = c_2 \oplus c_3,$$
$$a_3 = c_3.$$

Deci, coeficienţii a ai expresiei Reed-Müller generalizate de polaritate 1 se obţin din coeficienţii expresiei Reed-Müller, folosind matricea de transformare Z_{01}:

$$a = Z_{01}\, c \text{ peste } GF(2),$$

unde $\mathrm{Z}_{01} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Pentru a determina expresia Reed-Müller generalizată de polaritate 2 vom considera $\dot{x}_2 = \bar{x}_2$ şi $\dot{x}_1 = x_1$:

$$f(\dot{x}_2, \dot{x}_1) = c_0 \oplus c_1x_1 \oplus c_2\bar{x}_2 \oplus c_3\bar{x}_2x_1 =$$
$$= c_0 \oplus c_1x_1 \oplus c_2(x_2 \oplus 1) \oplus c_3(x_2 \oplus 1)x_1 =$$
$$= c_0 \oplus c_2 \oplus (c_1 \oplus c_3)x_1 \oplus c_2x_2 \oplus c_3x_2x_1$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_2, x_1) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_2x_1,$$

unde

$$a_0 = c_0 \oplus c_2,$$
$$a_1 = c_1 \oplus c_3,$$
$$a_2 = c_2,$$
$$a_3 = c_3.$$

Echivalent:

$$a = Z_{10}\, c \text{ peste } GF(2),$$

unde $Z_{10} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Pentru a determina expresia Reed-Müller generalizată de polaritate 3 vom considera $\dot{x}_2 = \bar{x}_2$ şi $\dot{x}_1 = \bar{x}_1$:

$$f(\dot{x}_2, \dot{x}_1) = c_0 \oplus c_1\bar{x}_1 \oplus c_2\bar{x}_2 \oplus c_3\bar{x}_2\bar{x}_1 =$$
$$= c_0 \oplus c_1(x_1 \oplus 1) \oplus c_2(x_2 \oplus 1) \oplus c_3(x_2 \oplus 1)(x_1 \oplus 1) =$$
$$= c_0 \oplus c_1 \oplus c_2 \oplus c_3 \oplus (c_1 \oplus c_3)x_1 \oplus (c_2 \oplus c_3)x_2 \oplus c_3x_2x_1$$

Deci, funcţia poate fi scrisă în următoarea formă echivalentă:

$$f(x_2, x_1) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus a_3x_2x_1,$$

unde

$$a_0 = c_0 \oplus c_1 \oplus c_2 \oplus c_3,$$
$$a_1 = c_1 \oplus c_3,$$
$$a_2 = c_2 \oplus c_3,$$
$$a_3 = c_3.$$

Echivalent:

$$a = Z_{11}\, c \text{ peste } GF(2),$$

unde $Z_{11} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Ştim că produsul Krönecker al matricilor $A = \begin{pmatrix} a_{11} & \dots & a_{1q} \\ \dots & & \\ a_{p1} & \dots & a_{pq} \end{pmatrix}$ de dimensiune $p \times q$ şi $B = \begin{pmatrix} b_{11} & \dots & b_{1s} \\ \dots & & \\ b_{r1} & \dots & b_{rs} \end{pmatrix}$ de dimensiune $r \times s$ este

$$A*B=\begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1q}B \\ \dots & & & \\ a_{p1}B & a_{p2}B & \dots & a_{pq}B \end{pmatrix}, \text{ de dimensiune } pr\times qs.$$

Deci, matricile de transformare pentru determinarea expresiilor Reed-Müller generalizate pentru două variabile pot fi scrise şi în forma:

$Z_{00}= Z_0*Z_0$
$Z_{01}= Z_0*Z_1$
$Z_{10}= Z_1*Z_0$
$Z_{11}= Z_1*Z_1$.

Acest rezultat poate fi generalizat şi pentru n variabile şi o polaritate oarecare i. Matricea de transformare pentru determinarea expresiei Reed-Müller generalizate de polaritate i pentru n variabile este:

$$Z_{e_n e_{n-1}\dots e_1} = Z_{e_n} * Z_{e_{n-1}} * \dots * Z_{e_1}$$

unde $e_n\, e_{n-1\dots}\, e_1$ este scrierea în baza 2 pe n biţi a polarităţii i.

Exemplu

Vom determina expresia Reed-Müller generalizată de polaritate 5 cunoscând domeniul funcţiei:

	1	x_1	x_1	1
1	1	1	1	0
x_3	0	0	1	1
	1	1	x_2	x_2

Extragem din domeniul funcţiei coeficienţii expresiei Reed-Müller:

c = (1, 1, 0, 1, 0, 0, 1, 1).

Scriem polaritatea 5 în baza 2 pe 3 biţi, 101, şi determinăm matricea transformării

$$Z_{101}=Z_1*Z_0*Z_1=\begin{pmatrix}1&1\\0&1\end{pmatrix}*\begin{pmatrix}1&0\\0&1\end{pmatrix}*\begin{pmatrix}1&1\\0&1\end{pmatrix}=\begin{pmatrix}1&1\\0&1\end{pmatrix}*\begin{pmatrix}1&1&0&0\\0&1&0&0\\0&0&1&1\\0&0&0&1\end{pmatrix}=$$

$$=\begin{pmatrix}1&1&0&0&1&1&0&0\\0&1&0&0&0&1&0&0\\0&0&1&1&0&0&1&1\\0&0&0&1&0&0&0&1\\0&0&0&0&1&1&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&1\\0&0&0&0&0&0&0&1\end{pmatrix}.$$

Coeficienţii expresiei Reed-Müller generalizate de polaritate 5 vor fi:

$$a = Z_{101}c = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Deci, expresia Reed-Müller generalizată de polaritate 5 va fi:
$f(x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_3 x_2 x_1.$

M4.U2.4. Rezumat
Expresiile Reed-Müller sunt forme canonice bazate pe algebra modulo-2, adică sume de produse modulo-2. Într-o expersie Reed-Müller, putem înlocui orice variabilă x_i cu $\bar{x}_i$ şi vom obţine o altă formă canonică. Pentru o funcţie de n variabile, există 2^n astfel de substituţii, care vor duce la obţinerea a 2^n noi forme canonice. Acestea se numesc expresii Reed-Müller generalizate şi fiecare dintre ele poate fi identificată printr-un număr, numit polaritate. Coeficienţii acestor expresii Reed-Müller generalizate se determină aplicând o transformare asupra expresiei Reed-Müller de polaritate 0.

M4.U2.5.Test de evaluare a cunoştinţelor

1. Determinaţi expresia Reed-Müller generalizată de polaritate 2 pentru $f(x_2, x_1) = 1 \oplus x_1 \oplus x_2 x_1$
2. Determinaţi expresia Reed-Müller generalizată de polaritate 6 pentru $f(x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_3 x_1 \oplus x_3 x_2.$

Temă de control

1. Determinaţi expresiile Reed-Müller corespunzătoare funcţiilor booleene:
   - $f_1(x_2, x_1) = \Sigma m(0, 3)$

- $f_1(x_3, x_2, x_1) = \Sigma m(1, 2, 4, 7)$.

2. Determinaţi funcţiile boolene corespunzătoare expresiilor Reed-Müller:
   - $f(x_2, x_1) = 1 \oplus x_2 \oplus x_2x_1$
   - $f(x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_2x_1 \oplus x_3x_2$.

3. Determinaţi expresia Reed-Müller generalizată de polaritate 1 pentru $f(x_2, x_1) = 1 \oplus x_1 \oplus x_2x_1$

4. Determinaţi expresia Reed-Müller generalizată de polaritate 4 pentru $f(x_3, x_2, x_1) = x_1 \oplus x_2 \oplus x_3x_1 \oplus x_3x_2$.