ARHITECTURA SISTEMELOR DE CALCUL ŞI SISTEME DE OPERARE

LUCRĂRILE DE LABORATOR Nr. 9, 10 şi 11

CODURI DE ERORI ÎN SISTEMELE DE CALCUL

I. SCOPUL LUCRĂRILOR

Lucrările prezintă codurile detectoare şi corectoare de erori în sistemele de calcul. Scopul lucrării constă în a familiariza studenţii cu modalităţile de codificare a informaţiilor la emisie, în vederea efectuării transmisiei la distanţă, decodificarea informaţiei la recepţie, detectând erorile de transmisie şi, în anumite situaţii corectarea acestor erori.

II. NOŢIUNI TEORETICE

1 Coduri de erori

Informaţiile pot suferi modificări involuntare în timpul transmisiei sau stocării lor în memoria calculatorului. Trebuie deci utilizate anumite **coduri** care permit detectarea sau chiar corectarea erorilor datorate acestor modificări.

Aceste coduri se constituie pe un număr de biţi superior celui strict necesar pentru a codifica informaţia. Astfel, celor m biţi de date li se adaugă k biţi de control. Deci, în memorie vor fi stocaţi $n = m + k$ biţi. Asemenea configuraţii definesc **codurile redundante**.

Unele coduri nu permit decât detectarea erorilor (coduri **autoverificatoare**), altele permit detectarea şi corectarea uneia sau mai multor erori (coduri **autocorectoare**).

2. Controlul de paritate

Este codul autoverificator cel mai simplu. El se compune din $m + 1$ biţi: cei m biţi de informaţie la care se adaugă al ($m + 1$) - lea bit numit **bit de paritate**. Valoarea sa este aleasă astfel ca numărul total de biţi egali cu 1, calculat pe $n + 1$ biţi să fie par (în cazul unei **parităţi pare**), sau impar (**paritate impară**).

Exemplu: Transmiterea de caractere codificate în **ASCII** pe 7 biţi plus un bit de paritate între un calculator şi un terminal.

A → 1 1000001
B → 1 1000010
E → 0 1000101
↑ bit de paritate impară

Dacă un bit este schimbat din eroare în timpul transferului, paritatea nu mai este verificată. Dacă eroarea este detectată trebuie retransmisă informaţia deoarece eroarea nu poate fi localizată pentru a putea fi corectată.

În general, controlul de paritate nu permite decât detectarea unui număr impar de erori, în cazul unui număr par efectele se pot anula.

Controlul de paritate nu poate fi utilizat decât pentru transmisii în care posibilitatea apariţiei erorilor este scăzută (de exemplu, în interiorul unui calculator sau între calculator şi perifericele sale).

3. Codul dublei parităţi

Codificare: considerăm exemplul unui cod ASCII (7 biţi);

- fiecare caracter este codificat pe o linie a unui tablou;
- un cod de paritate impară este efectuat pe fiecare linie (transversal);
- un cod de paritate impară este efectuat pe fiecare coloană (longitudinal);

Decodificare

- controlul transversal permite detectarea erorilor pe linie;
- controlul longitudinal permite detectarea erorilor pe coloană;

Exemplu: Se doreşte transmiterea mesajului 1968; se detectează o eroare la intersecţia primei linii cu coloana a 4-a.

	1	2	3	4	5	6	7	bit paritate	control transversal
1→	0	1	1	$\boxed{1}$	0	0	1	0	F
9→	0	1	1	1	0	0	1	1	A
6→	0	1	1	0	1	1	0	1	A
8→	0	1	1	1	0	0	0	0	A
bit paritate	1	1	1	1	0	0	1		
contr. longit.	A	A	A	F	A	A	A		

Dubla paritate permite corectarea unei erori, sau în anumite cazuri a unui număr impar de erori (ca de exemplu un număr impar de biţi eronaţi pe aceeaşi linie sau coloană). Totuşi, în majoritatea cazurilor este posibilă numai detectarea unui număr impar de erori (ca de exemplu trei biţi repartizaţi pe linii şi coloane diferite).

Principiul dublei parităţi este adesea utilizat în stocarea pe bandă magnetică a informaţiilor. Astfel, pentru o bandă cu n piste:

- fiecare caracter este stocat transversal pe n -1 piste;
- un bit de paritate transversal este stocat pe a n-a pistă;
- toate cele m caractere (un bloc) suportă un control longitudinal de paritate.

4. Codul lui Hamming

Codul lui Hamming este un cod autocorector bazat pe teste de paritate. Versiunea cea mai simplă permite corectarea unui bit eronat. Celor m biţi de informaţie li se adaugă k biţi de control al parităţii. Deci avem $n = m + k$ biţi necesari pentru transmiterea informaţiei.

Deoarece trebuie indicate $n + 1$ posibilităţi de eroare (inclusiv absenţa erorii) prin cei k biţi de control, trebuie ca $2^k \geq n + 1$. Cele 2^k posibilităţi de de codificare pe k biţi servesc la determinarea poziţiei erorii, apoi se poate corecta bitul eronat.

Tabelul următor permite determinarea lui k când se cunoaşte n:

m	0	0	1	1	2	3	4	4	5	6	7	8	9	10	...	120
k	1	2	2	3	3	3	3	4	4	4	4	4	4	4	...	8
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...	128

De obicei se ia $n = 2^k - 1$ în loc de $n < 2^k - 1$.

Dacă se numerotează biţii de la dreapta spre stânga pornind de la 1, biţii de control (sau de paritate) sunt plasaţi pe poziţia puterilor lui 2 (biţii cu numărul 1, 2, 4, 8, 16, ...). Fiecare bit de control efectuează control de paritate (pară sau impară) asupra unui anumit număr de biţi de date. Se determină astfel cei n biţi de transmis sau de stocat.

Exemplu: Dacă m = 4 se poate construi un cod Hamming (CH) pe 7 biţi (n = 7), adăugând 3 biţi de control (k = 3).

7	6	5	4	3	2	1
m_4	m_3	m_2	k_3	m_1	k_2	k_1

Cei trei biţi de control sunt plasaţi pe poziţia puterilor lui 2: $k_1 \rightarrow 1$; $k_2 \rightarrow 2$; $k_3 \rightarrow 4$.

Vom vedea acum, pentru fiecare bit al mesajului care sunt biţii de control care permit verificarea parităţii sale.

$7 = (0111)_2 = 4 + 2 + 1$ $\rightarrow$ 7 este controlat de k_3, k_2, k_1;
$6 = (0110)_2 = 4 + 2$ $\rightarrow$ 6 este controlat de k_3, k_2;
$5 = (0101)_2 = 4 \quad + 1$ $\rightarrow$ 5 este controlat de k_3, k_1;
$4 = (0100)_2 = 4$ $\rightarrow$ 4 este controlat de k_3;
$3 = (0011)_2 = \quad 2 + 1$ $\rightarrow$ 3 este controlat de k_2, k_1;
$2 = (0010)_2 = \quad 2$ $\rightarrow$ 2 este controlat de k_2;
$1 = (0001)_2 = \quad\quad 1$ $\rightarrow$ 1 este controlat de k_1;

Problema se pune şi invers: care sunt poziţiile binare controlate de către fiecare cod?

k_1 controlează biţii cu numerele 1, 3, 5, 7;
k_2 controlează biţii cu numerele 2, 3, 6, 7;
k_3 controlează biţii cu numerele 4, 5, 6, 7.

Când se recepţionează informaţia, se efectuează controlul de paritate. Pentru fiecare bit de control se compară valoarea transmisă cu cea recalculată. Dacă cele două valori sunt identice, se atribuie valoarea 0 unei variabile binare A_i asociată bitului de control k_i, altfel, A_i primeşte valoarea 1.

Valoarea zecimală a configuraţiei binare formată din variabilele A_k, A_{k-1}, ..., A_1 furnizează poziţia bitului eronat, care se poate corecta.

Reluăm exemplul precedent:

Presupunem că: pentru k_1, $A_1 = 1$, pentru k_2, $A_2 = 1$, iar pentru k_3, $A_3 = 0$. Eroarea se găseşte în poziţia $(A_3A_2A_1)_2 = (011)_2 = 3$.

Într-adevăr, k_1 poate detecta o eroare în poziţiile 1, 3, 5, 7, k_2 poate detecta o eroare pe poziţiile 2, 3, 6, 7, iar k_3 poate specifica o eroare pe poziţiile 4, 5, 6, 7. O eroare detectată de k_1 şi k_2 dar nu şi de k_3 nu poate proveni decât din bitul 3.

Exemple:

$(A_3A_2A_1)_2 = (000)_2 \rightarrow$ indică absenţa unei erori;
$(A_3A_2A_1)_2 = (001)_2 \rightarrow$ indică eroare pe bitul 1;
$(A_3A_2A_1)_2 = (110)_2 \rightarrow$ indică eroare pe bitul 6.

Exemplu de recepţionare a unui mesaj: $(1011100)_2$.

}tiind că s-a utilizat un CH cu paritate pară, să se reconstituie mesajul iniţial.

n = 7, deci k = 3, m = 4.

număr	7	6	5	4	3	2	1
tip	m_4	m_3	m_2	k_3	m_1	k_2	k_1
valoare	1	0	1	1	1	0	0

$k_1 = 0$ controlează poziţiile 1, 3, 5, 7, nu se verifică, deci $A_1 = 1$;
$k_2 = 0$ controlează poziţiile 2, 3, 6, 7, se verifică, deci $A_2 = 0$;
$k_3 = 0$ controlează poziţiile 4, 5, 6, 7, nu se verifică, deci $A_3 = 1$;

Adresa binară a erorii $(A_3A_2A_1)_2 = (101)_2 = 5$. Bitul cu numărul 5, care este egal cu 1 este eronat. Mesajul iniţial corectat şi fără biţii de control este: $(1001)_2$.

5. Calculul simplificat al codului lui Hamming (CHS)

Prin metoda lui Hamming, pentru detectarea şi corectarea unei singure erori se poate simplifica calculul biţilor de control, astfel:

- se transformă poziţiile din mesaj care conţin valori egale cu 1 în echivalentul lor binar ;
- se însumează modulo 2, astfel:
    - pentru paritate pară: un număr par de 1→ 0, un număr impar de 1→1 (sumă modulo 2 directă);
    - pentru paritate impară: un număr par de 1→1, un număr impar de 1→0 (sumă modulo 2 inversată).

Exemplu: Să codificăm mesajul $(10101011001)_2$ cu paritate pară: $m = 11$, $k = 4$ şi $n = 15$.

Număr	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
tip	m_{11}	m_{10}	m_9	m_8	m_7	m_6	m_5	k_4	m_4	m_3	m_2	k_3	m_1	k_2	k_1
valoare	1	0	1	0	1	0	1	?	1	0	0	?	1	?	?

Biţii cu valoarea 1 se găsesc pe poziţiile 15, 13, 11, 9, 7, 3, deci:

15 = 1 1 1 1
13 = 1 1 0 1
11 = 1 0 1 1
9 = 1 0 0 1
7 = 0 1 1 1
3 = 0 0 1 1
----
0 1 0 0 → biţi de paritate
$k_4\,k_3\,k_2\,k_1$

Mesajul codificat este deci $(101010101001100)_2$.

Exemplu de recepţie a unui mesaj:

S-a primit mesajul următor: $(101000101001100)_2$, codificat cu paritate impară. Biţii cu valoarea 1 se găsesc pe poziţiile 15, 13, 9, 7, 4, 3.

15 = 1 1 1 1
13 = 1 1 0 1
9 = 1 0 0 1
7 = 0 1 1 1
4 = 0 1 0 0
3 = 0 0 1 1
---------- → adunare modulo 2 inversată.
0 1 0 0
$A_4A_3A_2A_1$ → eroare pe poziţia 4.

După corectarea erorii şi eliminarea codurilor de control, mesajul iniţial este: $(10100011001)_2$.

6. Codul lui Hamming şi erorile grupate

Metoda lui Hamming poate corecta în general doar un bit eronat, dar se poate utiliza pentru detectarea şi corectarea erorilor multiple pe o secvenţă de biţi aranjând mesajul sub formă matricială, codificând pe linie după metoda HC şi transmiţând mesajul pe coloane.

De exemplu, pentru transmiterea mesajului **"**Hamming**"** se codifică pe o linie fiecare caracter în cod ASCII, completând cu biţii de control după metoda CH.

ASCII	Cod Hamming (pentru fiecare literă)											
	11	10	9	8	7	6	5	4	3	2	1	(număr)
H → 1001000	1	0	0	1	1	0	0	1	0	0	0	
a → 1100001	1	1	0	0	0	0	0	0	1	1	0	
m → 1101101	1	1	0	0	1	1	0	0	1	1	1	
m → 1101101	1	1	0	0	1	1	0	0	1	1	1	
i → 1101001	1	1	0	0	1	0	0	1	1	0	1	
n → 1101110	1	1	0	0	1	1	1	1	0	0	1	
g → 1100111	1	1	0	0	0	1	1	0	1	0	1	

Dacă se produc erori grupate, pentru o secvenţă de biţi suficient de scurtă, (≤ 7 pentru acest caz) atunci, efectuând transmisia pe coloană vom avea un singur bit eronat pe linie, pe care-l putem corecta datorită biţilor de control adăugaţi potrivit metodei lui Hamming.

În ultimii ani, codurile autocorectoare sunt din ce în ce mai utilizate pentru a asigura integritatea informaţiilor stocate în memorie.

Un cod autocorector permite creşterea considerabilă a timpului mediu între două defecţiuni care pot să apară: erorile nu apar decât atunci când numărul lor depăşeşte capacitatea de corectare a codului respectiv. Erorile care nu sunt corectate, cel puţin se pot detecta.

Detectarea erorilor grupate

În comunicaţiile la distanţă, erorile sunt mult mai frecvente decât în interiorul calculatorului. Erorile consecutive pot fi extinse adesea la un bloc întreg de biţi de informaţie.

Se vor utiliza în acest sens coduri care permit detectarea erorilor grupate, corectarea acestora fiind adesea prea costisitoare.

7. Metoda codurilor polinomiale (CRC)

CRC [Cyclic Redondant Coding] este metoda cea mai folosită pentru detectarea erorilor grupate. Înaintea transmiterii, informaţiei i se adaugă biţi de control, iar pe baza acestora, dacă la recepţionarea mesajului se detectează erori, atunci acesta trebuie retransmis.

O informaţie pe n biţi poate fi considerată ca lista coeficienţilor binari ai unui polinom cu n termeni, deci de grad n-1.

Exemplu: 110001 $\rightarrow x^5 + x^4 + 1$

Pentru a calcula biţii de control se va efectua un anumit număr de operaţii cu aceste polinoame cu coeficienţi binari. Operaţiile se vor efectua modulo 2, adunarea şi scăderea nu va ţine seama de cifra de transport, deci toate operaţiile de adunare şi scădere sunt identice cu operaţia logică XOR.

Pentru generarea şi verificarea biţilor de control atât sursa cât şi ddestinaţia mesajului utilizează un polinom generator G (x).

Dacă M (x) este polinomul corespunzător mesajului iniţial (de transmis), iar r este gradul polinomului generator G (x), atunci algoritmul de construire şi verificare a codurilor care se incorporează în mesajul de transmis este următorul:

1) se înmulţeşte M (x) cu x^r (se adaugă *r* zerouri la sfârşitul mesajului iniţial);
2) se efectuează împărţirea modulo 2:
   $(M (x) * x^r) / G (x) = Q (x) + R (x) / G (x)$;
3) câtul Q (x) se ignoră, iar restul R (x) conţine *r* biţi. Se efectuează scăderea modulo 2: $M (x) * x^r - R (x) = T (x)$, iar T (x) este polinomul care reprezintă mesajul de

transmis. Polinomul ciclic T (x) = Q (x) * G (x) este un multiplu al polinomului generator.

4) La recepţionarea mesajului se efectuează împărţirea T (x) /G (x):

a) dacă restul = 0 nu sunt erori de transmisie;

b) altfel, s-au produs erori, deci mesajul trebuie retransmis.

Exemplu de transmitere a unui mesaj.

Se doreşte transmiterea mesajului 101101 (6 biţi) → M (x) $=x^5 + x^3 + x^2 + 1$.

Polinomul generator este: 1011→ G (x) = $x^3 + x + 1$ de grad r = 3.

1) Efectuăm înmulţirea: M (x) x^r = 101101000 (se adaugă r = 3 zerouri la M (x).

2) Realizăm împărţirea modulo 2: M (x) x^r/G (x):

```
1 0 1 1 0 1 0 0 0 | 1 0 1 1
1 0 1 1             ---------
--------              1 0 0 0 0 1 → câtul Q(x)
        1 0 0 0
        1 0 1 1
        --------
        0 0 1 1 → R (x) = 0 1 1
```

3) Câtul Q (x) este ignorat. Pentru a realiza diferenţa modulo 2 M (x) x^r - R (x) este suficientă adăugarea celor r biţi din R (x) la sfârşitul mesajului M (x) → mesajul de transmis este T (x) = 101101011.

Exemplu de recepţionare a unui mesaj. S-a primit mesajul următor: 11010101. G (x) = 1011 (4 biţi) → G (x) = $x^3 + x + 1$ de grad r = 3.

4) Se efectuează împărţirea T (x) / G (x).

```
1 1 0 1 0 1 0 1 | 1 0 1 1
1 0 1 1           ---------
--------            1 1 1 1 0
 1 1 0 0
 1 0 1 0
 --------
   1 1 1 1
   1 0 1 1
   --------
        1 0 0 0
        1 0 1 1
        --------
           1 1 1 →R (x) = 1 1 1
```

R (x) ≠ 0, s-au detectat erori de transmisie, mesajul se retransmite.

Cele mai utilizate polinoame generatoare G (x) sunt:

- CRC - 12 $= x^{12} + x^{11} + x^3 + x^2 + x + 1$;
- CRC - 16 $= x^{16} + x^{15} + x^2 + 1$;
- CRC - CCITT $= x^{16} + x^{12} + x^5 + 1$.

III. MODUL DE LUCRU

Se realizează aplicaţii care codifică informaţiile la emisie şi decodifică informaţiile la recepţie. Se exemplifică codul de paritate memorie, codul dublei parităţi, codul lui Hamming şi metoda codurilor polinomiale.

IV. CONŢINUTUL REFERATULUI

1. Sumarul noţiunilor întâlnite.
2. Rezolvaţi următoarele probleme:
   - 2.1 Fie o transmisie utilizând codul lui Hamming cu paritate impară. Găsiţi mesajul transmis (reprezentare octală) ştiind că reprezentarea octală a datelor de transmis (compuse din pe 16 biţi) este 116570_8.
   - 2.2 Fie o transmisie utilizând codul lui Hamming cu paritate impară. Regăsiţi mesajul iniţial având grijă să corectaţi erorile eventuale dacă mesajul primit (pe 21 biţi) este 6130014_8.
   - 2.3 Fie o transmisie utilizând codul lui Hamming cu paritate pară. Găsiţi mesajul transmis (reprezentare octală) ştiind că reprezentarea octală a datelor de transmis (compuse din 4 biţi) este 13_8.
   - 2.4 Fie o transmisie utilizând codul lui Hamming cu paritate pară. Regăsiţi mesajul iniţial având grijă să corectaţi erorile eventuale dacă mesajul primit (pe 7 biţi) este 134_8.
   - 2.5 Presupunem că se transmit date codificate prin metoda CRC al cărei polinom generator este $G(x) = x^5+x+1$. Dacă se doreşte transmiterea datelor (9 biţi) cu reprezentarea octală 456, care va fi mesajul transmis?
   - 2.6 Presupunem că se transmit date codificate prin metoda CRC al cărei polinom generator este $G(x) = x^5+x+1$. Dacă se primeşte pe 15 biţi mesajul octal 76343 ce putem zice de mesajul iniţial?
   - 2.7 Dacă $G(x) = x^2+x+1$, mesajul primit (date + CRC) este $T(x) = x^{10}+x^7+x^6+x^4+x^3+x$. S-au produs erori de transmisie? Dacă nu, daţi sub forma polinomului M (x) datele iniţiale transmise.
   - 2.8 Să se scrie un program prin care să se adauge bitul de paritate pară pentru un mesaj dat sub forma unei configuraţii binare.
   - 2.9 Să se scrie un program prin care să se testeze bitul de paritate pară pentru un mesaj dat sub forma unei configuraţii binare.
3. Observaţii şi concluzii personale.