TEORIA INFORMAŢIEI ŞI A CODĂRII

Îndrumător de lucrări la laborator

Horia BALTA Maria KOVACI

2009

Cuprins

L1. Algoritmi pentru codarea sursei şi compresie

1.1. Codarea binară a surselor de informaţie

Lucrarea de faţă îşi propune explicitarea algoritmilor de calcul pentru a coda binar surse de informaţie întâlnite în practică.

Codarea binară a surselor de informaţie are dublu rol, de mărire a eficienţei sursei de informaţie (şi implicit de micşorare a costului transmisiei prin scăderea timpului transmisiei); de adaptare a sursei de informaţie la canalul transmisiune (canal binar, în majoritate).

Din punct de vedere al codării, la o sursă de informaţie interesează numărul de simboluri N şi probabilităţile lor de apariţie p_i, $i = \overline{1, N}$. Doar pe utilizator îl interesează ce reprezintă fiecare mesaj în parte (literă de alfabet, nivel al intensităţii pixelilor dintr-o imagine, valori ale unor mărimi fizice ce variază arbitrar[1]).

Codarea este operaţia prin care fiecărui simbol al sursei, s_i, i se alocă o succesiune binară unică:

$$s_i \longleftarrow r_i^1,\ r_i^2,\ \ldots r_i^{li} = c_i \qquad (1.1.1)$$

unde c_i reprezintă cuvântul de cod <u>asociat</u> simbolului (mesajului) sursei, iar r_i^j, cu $i = \overline{1, N}$, biţii componenţi ai cuvântului de cod (pot lua valori binare ”1” sau ”0”). Suportul fizic al valorilor binare este, de asemenea, mai puţin important pentru codare. Acest suport poate fi un semnal electric, optic, acustic, etc.

În expresia (1.1.1) l_i reprezintă lungimea cuvântului de cod c_i (număr de simboluri binare). Lungimea medie a cuvintelor de cod este dată de relaţia:

$$L = \sum_{i=1}^{N} p_i \cdot l_i \qquad (1.1.2)$$

Ieşirea codorului, pentru canalul de transmisie, reprezintă o sursă de informaţie numită secundară. Această sursă poate genera simbolurile 0 şi 1. Entropia (informaţia medie pe simbol), entropia maximă, precum şi eficienţa sursei primare, sunt date de relaţiile:

$$\begin{aligned} H(S) &= \sum_{i=1}^{N} p_i \cdot \log_2 \frac{1}{p_i}\ ; \\ H_{max}(S) &= \log_2 N\ ; \\ \eta_S &= \frac{H(S)}{H_{max}(S)}\ . \end{aligned} \qquad (1.1.3)$$

Entropia sursei secundare este egală cu informaţia medie pe un cuvânt raportată la lungimea medie a cuvântului:

$$H(X) = \frac{H(S)}{L} \qquad (1.1.4)$$

Atribuirea literelor de alfabet r_i^j pentru a forma cuvântul de cod c_i trebuie să conducă la îndeplinirea condiţiilor:

- codul să fie instantaneu (nici un cuvânt nu este prefixul altuia);

[1] Deşi, aparent, astfel de surse de informaţie nu au un număr finit de simboluri, prin eşantionare şi cuantizare, orice sursă de informaţie poate fi redusă la una cu număr finit de simboluri.

- codarea să conducă la o eficienţă maxim posibilă. Această condiţie se realizează folosind cuvinte de lungime variabilă, mai mare pentru simboluri de probabilitate de emisie (a sursei primare) mai mică. Codarea cu cuvinte de lungime variabilă este mai complicată, dar conduce la o sursă secundară cu eficienţă mai bună şi la o ieftinire a transmisiei, lungimea medie a cuvintelor de cod fiind mai mică. Codarea cu cuvinte de lungime constantă este mai simplă dar cuvintele având aceeaşi lungime oferă maleabilitate la prelucrări ulterioare.

Codarea cu cuvinte de lungime constantă se mai numeşte şi codare cu cod bloc. Cuvintele codului bloc sunt secvenţe binare diferite, de aceeaşi lungime L. Ştiind că numărul de secvenţe diferite ce pot fi constituite cu L cifre binare este 2^L, rezultă că numărul simbolurilor sursei trebuie să fie mai mic decât 2^L, sau:

$$L \geq \log_2 N = H_{max}(S) > L - 1 \qquad (1.1.5)$$

Inecuaţia a doua din (1.1.5) se datorează criteriului de eficienţă minim posibilă (sub restricţia de cod bloc).

Realizarea condiţiilor mai sus precizate se face prin codare după algoritmul Huffman (static). Acesta presupune:

1. Ordonarea probabilităţilor sursei (primare) în ordine descrescătoare;
2. Simbolurile cu probabilităţile cele mai mici (ultimele două[1] în ordonare) se reunesc formând un nou simbol cu probabilitatea sumei celor două, după care se reordonează probabilităţile, obţinându-se o sursă cu N-1 simboluri;
3. Procesul continuă până când rămân 2 simboluri. Acestora li se atribuie valorile 0 şi 1;
4. Mergând pe cale inversă, se disociază simbolurile compuse în simboluri din care s-au compus, atribuind arbitrar, la fiecare disociere, valorile 0 şi 1 pentru a găsi cuvintele de cod pentru cei doi componenţi. De reţinut că doar un singur simbol se disociază la fiecare pas, lungimea componenţilor săi crescând cu 1, celelalte simboluri păstrându-şi lungimea şi structura cuvântului de cod.

***Observaţie**: -atribuirea simbolurilor binare celor două simboluri compuse este arbitrară, codurile obţinute vor fi diferite, dar de aceeaşi eficienţă. De asemenea, ordinea simbolurilor de egală probabilitate poate fi aleasă arbitrar, diferitele moduri de atribuire conducând la coduri diferite, dar de aceeaşi eficienţă. Însă, în vederea posibilităţii de a confrunta rezultatele, se vor respecta regulile: -„0" se atribuie totdeauna simbolului de jos (ultimul în ordonare); -ordinea simbolurilor egal probabile este ordinea în care s-au citit, cu simbolul compus totdeauna ultimul.*

Programul pe calculator, asociat acestei lucrări, permite codarea surselor de informaţie prin algoritm Huffman static.

Precizarea sursei de informaţie se poate face simplu, introducând valorile probabilităţilor sursei şi numărul lor. Probabilităţile trebuie să îndeplinească condiţia:

$$\sum_{i=1}^{N} p_i = 1 \quad ; \quad 0 \leq p_i \leq 1 ; \quad \forall\ i = \overline{1, N} \qquad (1.1.6)$$

Probabilităţile pot fi introduse indirect, prin numere întregi, pozitive k_i, calculatorul urmând să facă transformarea:

$$p_i = \frac{k_i}{\sum_{k=1}^{N} k_i} \qquad (1.1.7)$$

[1] Algoritmul se poate aplica şi pentru obţinerea codurilor nebinare (având un alfabet cu m simboluri). În acest caz se grupează câte m simboluri.

ceea ce asigură îndeplinirea relaţiei (1.1.6).

Sursa de informaţie poate fi şi un text scris (în caractere ASCII), cu cel puţin 3 caractere diferite. Calculatorul va înţelege că simbolurile sursei (caracterele prezente în text) au probabilităţi precizate prin ponderea lor în text (număr de prezenţe ale caracterului respectiv raportat la întregul număr de caractere ale textului).

Programul permite şi o a treia sursă de informaţie. Simbolurile acesteia reprezintă intervale de lungime egală, şi în număr finit, de pe axă reală. În acest caz datele vor consta în valorile eşantioanelor unui semnal discret (Fig.1.1.1) , mărginit în timp cu suportul finit precizat ca dată de intrare M. Tot ca dată de intrare calculatorul va cere şi q-cuanta. Având aceste mărimi q (cuanta), M (numărul de eşantioane) şi b(i) (valorile eşantioanelor), pentru găsirea simbolurilor sursei şi implicit a numărului lor – N, calculatorul va proceda astfel:

- va căuta eşantioanele de valoare minimă şi maximă : min; max;
- va calcula $N = [\frac{\max - \min}{2}] + 1$ unde [ ] semnifică partea întreagă;

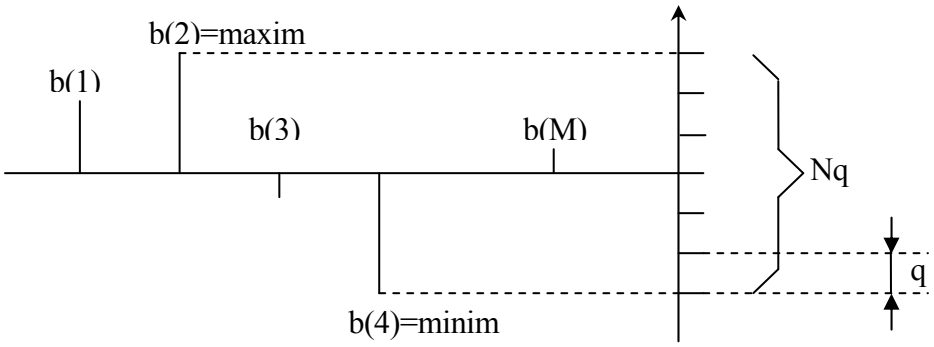


Fig.1.1.1 Sursă de informaţie „semnal discret”

- va calcula $N = [\frac{\max - \min}{q}] + 1$ unde [ ] semnifică partea întreagă;
- va atribui sursei de informaţie primare simbolurile: [min, min + q); [min+q, min+2q); ... ;[min+(N-1)q,min+Nq)
- va calcula probabilitatea simbolurilor ca fiind egale cu fracţiunea din numărul eşantioanelor M, ce au valori cuprinse în intervalele ce definesc simbolurile. Este posibil desigur să rămână simboluri cu probabilitate 0.

În toate cele 3 cazuri se vor calcula şi se vor putea vizualiza:

a) probabilităţile sursei primare;
b) datele asociate sursei şi codării:
   - entropia sursei primare H(S);
   - entropia maximă a sursei primare $H_{max}(S)$;
   - eficienţa sursei primare η_S;
   - entropia sursei secundare (eficienţa codării) H(X);
   - lungimea medie a cuvintelor de cod L;
   - redundanţa sursei primare $R(S)=H_{max}(S)-H(S)$;
   - redundanţa relativă a sursei primare $\rho_S = 1-\eta_S$;
   - redundanţa sursei secundare $\rho_X = 1-H(X)$.
c) cuvintele de cod;
d) graful codării;

e) schema algoritmului Huffman (numai pentru cazul codării cu cuvinte de lungime variabilă);
f) în cazul sursei „semnal discret" se pot vedea şi graficul semnalului precum şi valorile eşantioanelor.

Desfăşurarea lucrării

a). Rulaţi programul cobsinf.exe selectând opţiunea „Demonstraţie". Selectaţi pe rând cele trei feluri de surse de informaţie (tablou, text şi semnal discret), şi pentru fiecare sursă solicitaţi o codare bloc şi o codare prin algoritmul Huffman static. Urmăriţi aplicarea algoritmului.
b). Alcătuiţi surse de informaţie de forma celor prezentate şi codaţi-le bloc şi prin algoritmul Huffman static, calculând în fiecare caz entropia sursei, eficienţa sursei, lungimea medie a cuvintelor codului obţinut, precum şi eficienţa codării. Porniţi pentru început cu surse tablou mai simple, apoi măriţi numărul de simboluri. Utilizaţi ulterior şi surse „text" sau „semnal discret".
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate. Durata acestuia este aproximativ constantă şi independentă de numărul de studenţi ce efectuează simultan testul (calculatorul poate testa până la 4 studenţi simultan – în sensul că afişează pe rând date pentru cei $n \leq 4$ studenţi şi apoi întreabă pe rând, în aceeaşi ordine studenţii, timpii de afişare şi de chestionare rezervaţi fiecărui student sunt constanţi şi nu influenţează pe cei rezervaţi altuia). Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.2. Algoritmul Huffman dinamic

Compresia este procesul de minimizare a spaţiului ocupat sau a timpului necesar transmiterii unei anumite cantităţi de informaţie.

Metodele de compresie pot fi împărţite în:

- Metode de compresie cu pierdere;
- Metode de compresie fără pierdere.

Metodele de compresie cu pierdere de informaţie sunt folosite în special în transmiterea semnalului audio şi video, unde pierderea de informaţie are ca rezultat o scădere a calităţii sunetului, respectiv imaginii.

Compresia de date fără pierdere, prezentă în programele de arhivare, în sistemele de transmisiune a datelor, a evoluat de-a lungul timpului pornind de la algoritmi simpli (suprimarea zerourilor, codarea pe şiruri) şi ajungând la algoritmii complecşi folosiţi în prezent.

Metodele de compresie fără pierderi au la bază ideea că în general cantitatea de informaţie prelucrată (transmisă, depozitată) conţine o anumită redundanţă ce se datorează:

- Distribuţiei caracterelor (unele caractere au o frecvenţă de apariţie mult mai mare decât altele);
- Repetării consecutive a unor caractere;
- Distribuţiei grupurilor de caractere (unele grupuri sunt mult mai frecvente decât altele şi în plus există grupuri care nu apar deloc);
- Distribuţiei poziţiei (unele caractere sau grupuri ocupă poziţii preferenţiale, predictibile în anumite blocuri de date).

Având în vedere toate aceste tipuri se poate înţelege de ce o anumită tehnică de compresie poate da un rezultat bun pentru un anumit tip de surse, pentru altele însă rezultatul fiind dezastruos. În studiul compresiei se urmăreşte obţinerea unui algoritm care să ofere o compresie cât mai bună pentru tipuri de surse cât mai diferite.

Aprecierea cantitativă a compresiei realizate se face utilizând *factorul de compresie*, definit ca:

$$F_c = \frac{n_u}{n_c} \qquad (1.2.1)$$

unde n_u este lungimea în biţi a mesajului iniţial şi n_c lungimea de compresie.

Algoritmul Huffman dinamic

Algoritmii de tip Huffman static au dezavantajul că necesită cunoaşterea prealabilă a statisticii sursei. Acest dezavantaj poate fi înlăturat utilizând un algoritm dinamic.

Algoritmul Huffman dinamic este un algoritm de compresie. Mesajele au fost deja codate în prealabil, dar neeficient, cu un cod bloc, de lungime *n* biţi/cuvânt. Ideea de bază în această codare este folosirea pentru codarea unui simbol s_{i+1} din mesaj a unui graf de codare, ce este un arbore care creşte, dintr-un punct iniţial (numit sursă) şi care este construit pe baza primilor *i* simboluri din mesaj. După transmiterea simbolului s_{i+1} se va revizui arborele de codare în vederea codării simbolului s_{i+2}.

Codarea presupune la fiecare pas transmiterea codului aferent simbolului de intrare şi modificarea corespunzătoare a grafului şi a codului. Dacă în mesajul transmis este un simbol nou, adică un simbol care n-a mai apărut în mesaj, se transmite mai întâi codul frunză goală şi apoi cei *n* biţi aferenţi simbolului nou apărut. Frunza goală are semnificaţia că urmează un nou simbol. Frunza goală se codifică ca un mesaj oarecare, dar ponderea sa întodeauna va rămâne nulă.

Exemplu:
În continuare se prezintă evoluţia arborelui (grafului) asociat codului în timpul codării mesajului "M_i = aaabccc":

Obs: *Semnificaţia notaţiilor este:*

-pentru nod: x | p |

- x: număr de ordine (creşte de la stânga spre dreapta şi de jos în sus pe linii);
- p: ponderea cumulată din nodurile aferente.

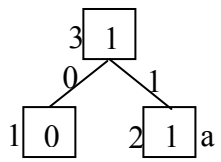
- pentru frunză (nod terminal): x | p | y

- x: număr de ordine;
- p: pondere (număr de apariţii ale respectivului simbol);
- y: simbolul.

Frunza goală, notată cu "0", este de pondere 0.

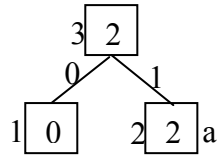
- pentru ramură: - spre stânga ≡ codare cu zero;
- spre dreapta ≡ codare (atribuire) cu unu.

1. M_i = "a"



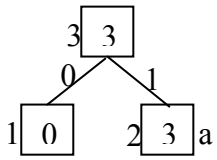
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a"

2. M_i = "aa"



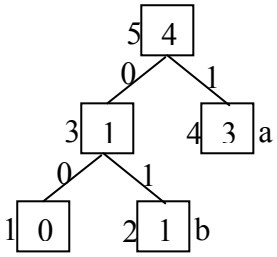
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: "a1"

3. M_i = “aaa”



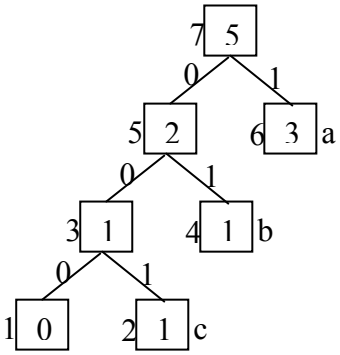
Codurile sunt: 0 - 0
a - 1
Mesajul transmis este: “a11”

4. M_i = “aaab”



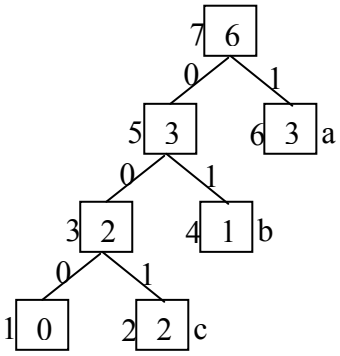
Codurile sunt: 0 - 00
a - 1
b - 01
Mesajul transmis este: “a110b”

5. M_i = “aaabc”



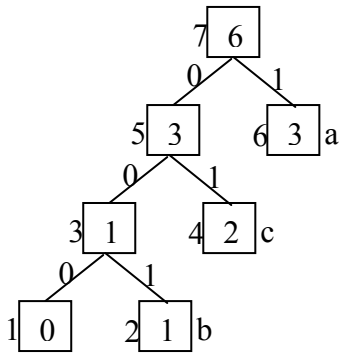
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c”

6. M_i = “aaabcc”



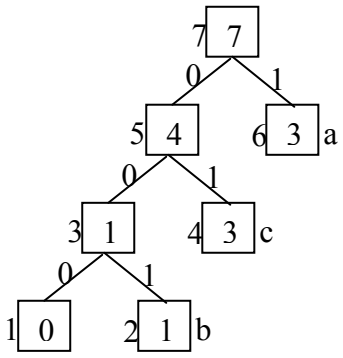
Codurile sunt: 0 - 000
a - 1
b - 01
c - 001
Mesajul transmis este: “a110b00c001”

Deoarece ponderea nodului “c” este mai mare decât ponderea nodului “b” se va face interschimbarea între noduri, realizându-se o rearanjare a arborelui:



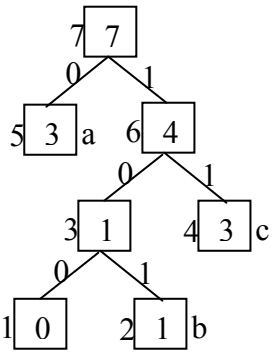
Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c001”

7. M_i = “aaabccc”



Codurile sunt: 0 - 000
a - 1
b - 001
c - 01
Mesajul transmis este: “a110b00c00101”

În urma rearanjării arborelui se obţine:



Codurile sunt: 0 - 100
a - 0
b - 101
c - 11
Mesajul transmis este: “a110b00c00101”

Dacă se presupune că simbolurile mesajului de la intrare au fost codate în prealabil cu un cod bloc, cu n=8 biţi/cuvânt vom avea lungimea mesajului iniţial:
Considerând caracterele în mesajul iniţial codate pe 8 biţi, lungimea acestuia este:

$$n_u = 7 \cdot 8 = 56 \text{ biţi} \qquad (1.2.2)$$

Mesajul comprimat (a110b00c00101) are:

$$n_c = 8 \cdot 3 + 10 = 34 \text{ biţi} \qquad (1.2.3)$$

Rezultă un factor de compresie:

$$F = \frac{n_u}{n_c} \cong 1,7$$

Desfăşurarea lucrării:

a) Se lansează executabilul dHuffman.exe din directorul Huffman Dinamic, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru codare şi decodare.
b) Se verifică, cu programul, exemplul luat în această lucrare, selectând pe rând compresia, respectiv decompresia prezentate.
c) Pentru compresie şi decompresie se alege câte un exemplu oarecare asemănător cu cel prezentat în lucrare. Se realizează compresia/decompresia, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

1.3. Algoritmi de compresie de tip LZ

Cu toate că algoritmii de compresie Huffman, static sau dinamic, sunt cei mai performanţi (din punct de vedere al factorului de compresie) relativ la sursele fără memorie, această concluzie nu este valabilă şi pentru sursele cu memorie, surse frecvent întâlnite în practică.

Algoritmii de compresie de tip LZ fac parte din categoria tehnicilor de dicţionar adaptive. Ele servesc (în special) la compresia fişierelor de tip text, fişiere ce se caracterizează prin repetarea frecventă a unor subşiruri.

Ideea de bază în algoritmii LZ este înlocuirea unor subşiruri ale mesajului cu cuvinte de cod, astfel încât, la o nouă apariţie a unui subşir să se transmită doar codul asociat lui.

Algoritmul LZ –77

Mesajul de intrare este trecut printr-un buffer de lungime N. Bufferul este divizat în două blocuri distincte numite Lempel şi Ziv, ca în Fig. 1.3.1.

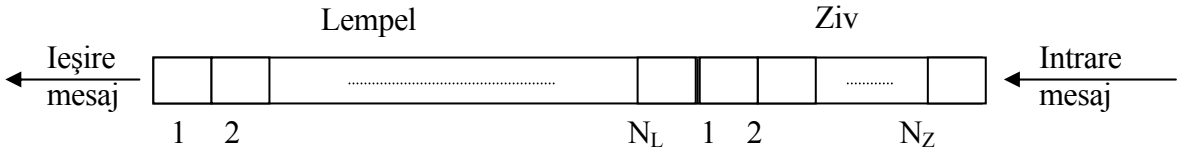


Fig. 1.3.1. Buffer–fereastră utilizat în algoritmul LZ–77

Mesajul de ieşire constă într-o succesiune de triplete de forma „P, L, A”. Compresia decurge astfel:

-se încarcă primele N_Z caractere din mesajul de intrare în blocul Ziv;

-se transmite la ieşire tripletul „0, 0, x”, unde x reprezintă primul caracter (literă) din mesajul de intrare (aflat pe poziţia 1 în blocul Ziv);

se deplasează cu o poziţie mesajul de intrare în blocul Ziv, astfel încât x ajunge acum în poziţia N_L. Din acest moment:

-se caută în blocul Lempel un subşir, S', care este identic cu cel mai lung subşir, S, care începe în poziţia 1 a blocului Ziv şi este conţinut în întregime în blocul Ziv. Subşirul S' trebuie neapărat să înceapă în blocul Lempel, dar poate să sew sfârşească în blocul Ziv. Dacă un astfel de subşir, S', există, atunci se va transmite tripletul „P, L, A”, unde P = poziţia de la care începe subşirul S'; L = lungimea subşirurilor S şi S'; A = litera ce succede subşirul S în mesajul de intrare.

Dacă nu există nici un subşir S' care să coincidă cu cel mai scurt subşir S, atunci se va transmite tripletul „0, 0, x”, unde x are aceeaşi semnificaţie ca şi mai sus;

-se deplasează mesajul de intrare până când, după caz, A sau x ajung pe poziţia N_L şi procesul de căutare se reia până când tot mesajul a trecut prin blocul Ziv.

Decompresia presupune, în principal, aceleaşi etape. Utilizând tripleţii recepţionaţi, se adaugă subşiruri în blocul Ziv mesajului deja decomprimat, subşiruri aflate deja în blocul Lempel urmând ca, mai apoi, toate să fie deplasate în blocul Lempel.

Observaţie: *este posibil să se renunţe la cel de-al doilea zero din tripletul „0, 0, x", el fiind redundant.*

Cei trei componenţi ai tripletului „P, L, A" sunt codaţi binar bloc, separat. Astfel, codul pentru P are lungimea:

$$k_P = \log_2 (N_L + 1) \qquad (1.3.1)$$

(trebuie codat şi 0 din situaţia „0, 0, x" sau „0,x"), iar pentru L:

$$k_L = \log_2 (N_Z - 1) \qquad (1.3.2)$$

considerând doar varianta „0, x". Caracterele de tipul „A" pot fi codate, de exemplu, ASCII.

Din relaţiile (1.3.1) şi (1.3.2) rezultă că, pentru o bună compresie, N_L+1 şi N_Z-1 trebuie să fie puteri întregi ale lui 2. În relaţia (1.3.2) s-a luat N_Z-1 şi nu doar N_Z deoarece, în situaţia extremă a lungimii maxime pentru subşirul S, N_Z va fi egal cu lungimea subşirului S plus unu; acest „unu" rezervă literei A un loc în blocul Ziv.

Exemplu *Fie parametrii N_L =7 şi N_Z =4, iar mesajul de intrare „aaababacacab…". Aplicarea algoritmului LZ –77 asupra acestui mesaj de intrare este ilustrată în Fig. 1.3.2 a).*
Mesajul comprimat este, aşadar: „0a72b63c73b…", sau în binar:
„000a11110b11011c11111b…". Fig. 1.3.2. b) prezintă un exemplu de decompresie.

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a	a	a	b	0		a
						a	a	a	b	a	7	2	b
			a	a	a	b	a	b	a	c	6	3	c
a	a	b	a	b	a	c	a	c	a	b	7	3	b

a) compresia mesajului „aaababacacab…"

1	2	3	4	5	6	7	1	2	3	4	P	L	A
							a				0		a
						a	a	a	a	b	7	3	b
		a	a	a	a	b	a	a	c		3	2	c
a	a	a	b	a	a	c	b	a	c		4	2	c
b	a	a	c	b	a	c	a	c	b	c	3	3	c

b) decompresia mesajului „0aa73b32c42c33c"

Fig. 1.3.2. Exemplu de aplicare a algoritmul LZ–77

Algoritmul LZ–78

Spre deosebire de varianta anterioară, la LZ–78 blocul Lempel este un dicţionar în continuă creştere. De asemenea, nu există teoretic nici o limitare a blocului Ziv.

Algoritmul LZ–78 este, în esenţă, următorul:

-se iniţializează dicţionarul (blocul Lempel) cu un şir nul;

-se încarcă mesajul de intrare în blocul Ziv;

-se transmite în clar primul caracter şi se introduce şi în dicţionar la poziţia 1. Cele două poziţii ale dicţionarului (notate cu 0 şi 1) sunt, în acest moment codate prin „0" şi „1". Din acest moment:

-se caută în dicţionar un subşir, S', identic cu cel mai lung subşir, S, din blocul Ziv. Binenţeles, S începe din prima poziţie a blocului Ziv. Dacă există un astfel de subşir, S', atunci se transmite codul

aferent urmat de caracterul A ce urmează lui S în mesajul de intrare. În plus, dicţionarul se îmbogăţeşte cu o poziţie, poziţie unde se trece subşirul SA. Dacă nici măcar primul caracter, x, din Ziv nu se regăseşte ăn dicţionar, atunci se transmite 0x, unde 0 semnifică subşirul din prima poziţie, codat prin cuvântul de cod „00…0”.

Observaţie: *Atunci când dicţionarul ajunge la un număr de elemente egal cu 2^k, tuturor codurilor elementelor anterioare li se adaugă ca şi prefix „0”, iar ultimul subşir va fi codat prin „10…0”.*

Exemplu: *În Fig. 1.3.3 a) se prezintă un exemplu de compresie utilizând algoritmul LZ–78. Mesajului de intrare este cel din exemplul anterior, „aaababacacab…”. Mesajul comprimat rezultat este „a1a0b1b1c6a3”, sau în binar „a1a0b01c110a011”. În Fig. 1.3.3 b) se prezintă un exemplu de decompresie, utilizând acelaşi algoritm. Mesajul comprimat (de intrare) este „a1b3b5a3a”. Rezultatul decompresiei este „aababcabcaaba”.*

Blocul Lempel (dicţionarul)

0	
1	a
2	aa
3	b
4	ab
5	c
6	ac
7	aca
8	

Blocul Ziv	Mesajul de ieşire
a a a b a b a c a c a b	a
a a b a b a c a c a b	1a
b a b a c a c a b	0b
a b a c a c a b	1b
a c a c a b	1c
a c a b	6a
b	3

b) compresia mesajului „aaababacacab…”;

Blocul Lempel (dicţionarul)

0	
1	a
2	b
3	ab
4	c
5	abc
6	abca
7	

Mesaj decomprimat	Mesaj comprimat
a	a
a a b	1b
a a b a b c	3b
a a b a b c a b c a	5a
a a b a b c a b c a a b a3a	

b) decompresia mesajului „a1b3b5a3a”

Fig. 1.3.3 Exemplu de aplicare a algoritmul LZ–78.

Decompresia presupune următorul algoritm:
-se citeşte primul caracter (codul ASCII aferent); acest caracter se introduce în dicţionar la poziţia 1 şi se generează şi la ieşire; în continuare procedura este:
-se citeşta codul poziţiei la care se găseşte subşirul transmis. Acest cod conţine:

$$k = \sup(\log_2 n) \quad \text{biţi} \tag{1.3.3}$$

unde sup(x) reprezintă aproximarea prin adaos a lui x, iar n dimensiunea momentană a dicţionarului;
-se citeşte din dicţionar subşirul S aflat la poziţia citită anterior şi se generează la ieşire;

-se citeşte din mesajul recepţionat următorul caracter, A (în cod ASCII), şi se generează şi acesta la ieşire. Dacă A este un nou caracter, atunci se introduce în dicţionar la o nouă poziţie;
-se introduce în dicţionar subşirul SA la următoarea poziţie;
Procesul se repetă până la epuizarea mesajului recepţionat.

Algoritmul LZW (Lempel–Ziv–Welch)

Modificarea esenţială adusă de algoritmul LZW este faptul că atât compresorul cât şi decompresorul cunosc simbolurile (caracterele) componente ale mesajului. Cu alte cuvinte, înainte de începerea propriu-zisă a compresiei (decompresiei) dicţionarul este iniţializat cu toate caracterele posibil a fi emise.(În exemplul următor, ilustrat în Figura 3.4a, dicţionarul este iniţializat cu simbolurile a–00, b–01, c–10.) În acest fel nu se mai trimite informaţie despre următorul caracter la fiecare pas. O altă modificare o constituie existenţa unui prefix, P, care se memorează de la un pas la următorul.

Algoritmul compresiei este:
-se iniţializează dicţionarul (aşa cum s-a descris anterior);
-se citeşte primul caracter şi se memorează ca şi prefix, P. Din acest moment:
-se citeşte următorul caracter, notat E. Se caută în dicţionar subşirul PE. Dacă acest subşir există, atunci se trece la pasul următor, fără a emite nimic şi fără a adăuga nimic în dicţionar. Singura modificare este că acum PE devine prefix pentru pasul următor. Dacă subşirul PE nu există în dicţionar, atunci se execută următoarele operaţii:
-se trimite la ieşire codul aferent subşirului P;
-se trece pe post de prefix caracterul E (E→P);
-se adaugă în dicţionar subşirul PE la următoarea poziţie liberă.
Procesul continuă în acest fel până la epuizarea întregului mesaj de intrare.

Observaţie: *La fel ca şi la LZ-78, pe vreme ce dicţionarul creşte trebuie modificat corespunzător codul binar loc asociat.*

Algoritmul decompresiei este asemănător compresiei. Diferenţa este că simbolul E nu se citeşte direct de la intrare (ca şi la compresie) ci doar după ce s-a decodat codul recepţionat. Mesajul decomprimat se poate citi fie de la E (exceptând primul simbol, care se citeşte de la P), fie prin compunerea şirului decodat.

Exemplu: *Fig. 1.3.4. prezintă câte un exemplu de compresie şi decompresie utilizând algoritmul LZW. Pentru compresie s-a ales ca mesaj de intrare: „aaababacacab”, acelaşi ca şi în exemplele anterioare. Mesajul comprimat este: „031052081”, sau în binar „001100100010101000001000001”. Pentru decompresie s-a ales mesajul „013205”, car în binar se exprimă: „0001011010000101”. Mesajul decomprimat este: „ababcaabc”.*

<u>Desfăşurarea lucrării:</u>

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă, urmărind aplicarea celor trei algoritmi la compresie şi decompresie. Aflaţi pentru fiecare exemplu şi factorul de compresie.
b). Alcătuiţi, utilizând trei sau patru litere, mesaje de circa 10 litere lungime. Comprimaţi mesajele independent de program şi verificaţi rezultatele cu ajutorul programului. Pentru decompresie utilizaţi mesajele comprimate de colegi, fără a cunoaşte originalul. Confruntaţi rezultatele decomprimării cu cele originale. Calculaţi în fiecare caz factorul de compresie, considerând că orice literă (caracter) se scrie în binar pe 8 biţi.
c) La sfârşitul lucrării de laborator se va efectua test asupra cunoştinţelor acumulate, prin intermediul programului pe calculator. Testul constă din: 1–o compresie şi 2–o decompresie, a unui mesaj ales aleatoriu de către calculator printr-unul dintre cei trei algoritmi (de asemenea ales aleatoriu de program) şi 3 –cinci întrebări teoretice fiecare cu un răspuns corect din cinci propuse, având ca temă algoritmii de compresie.

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir transmis	Cod transmis	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
	a					
a	a	aa	a	0=00	aa	3=11
a	a	aa				
aa	b	aab	aa	3=11	aab	4=100
b	a	ba	b	1=001	ba	5=101
a	b	ab	a	0=000	ab	6=110
b	a	ba				
ba	c	bac	ba	5=101	bac	7=111
c	a	ca	c	2=010	ca	8=1000
a	c	ac	a	0=0000	ac	9=1001
c	a	ca				
ca	b	cab	ca	8=1000	cab	10=1010
b		b	b	1=0001		

a) compresia mesajului „aaababacacab”;

Prefix	Mesaj de intrare	Şir căutat în dicţionar	Şir recepţionat decodat	Cod recepţionat	Dicţionarul a 0=00 b 1=01 c 2=10	
P	E	PE			Şir adăugat în dicţionar	Cod adăugat în dicţionar
			a	0		
a	b	ab	b	1	ab	3=11
b	a	ba	ab	3	ba	4=100
a	b	ab				
ab	c	abc	c	2	abc	5=101
c	a	ca	a	0	ca	6=110
a	a	aa	abc	5	aa	7=111
a	b	ab				
ab	c	abc				

b) decompresia mesajului „013205”;
Fig. 1.3.4. Exemplu de aplicare a algoritmului LZW

L2. Coduri simple corectoare de o eroare

2.1. Codul Hamming

2.1.1. Cod Hamming corector de o eroare

Codurile Hamming constiruie prima clasă de coduri bloc liniare corectoare de erori şi au fost propuse de R. Hamming în 1950.

Codul Hamming este un cod protector. Scopul codării este acela de a adapta sursa de informaţie la canalul de transmisie. În cazul de faţă sursa este deja codată şi se face o codare pentru protecţia informaţiei. Acest lucru se realizează prin mărirea redundanţei (prin creşterea suportului binar; biţilor de informaţie adăugânduse biţii de control). Biţii de control au ca scop de a crea legături, relaţii între biţii de informaţie. Aceste relaţii folosesc codorului pentru a calcula biţii de control şi folosesc decodorului pentru a verifica corectitudinea transmisiei.

Codul Hamming este un cod nesistematic (simbolurile de control sunt intercalate printre simbolurile informaţionale, situându-se pe poziţii puteri ale lui 2).

2.1.1.1. Codarea codului Hamminmg corector de o eroare

Particularitatea codului liniar Hamming corector de o eroare constă în forma matricii de control, H, care are fiecare coloană structurată prin reprezentarea binară a numărului zecimal de ordine al coloanei respective. Din acest motiv, corectorul Z, calculat cu relaţia $Z = H \cdot V^T$, decodat din binar în zecimal, indică numărul de ordine zecimal al bitului eronat din cuvântul recepţionat.

Distanţa minimă a acestui cod este:

$$d \geq 2e_c + 1 = 3,$$

unde e_c reprezintă numărul de erori corectabile.

Se consideră codul cu parametrii:

- n=7, numărul de simboluri în cuvânt;
- m=3, numărul de simboluri de control în cuvânt;
- k=4, numărul de simboluri de informaţie în cuvânt.

Secvenţa de informaţie este:

$$i = i_3 i_5 i_6 i_7$$

şi secvenţa de control:

$$C = C_1 C_2 C_4$$

astfel încât cuvântul de cod rezultat va fi:

$$V = C_1 C_2 i_3 C_4 i_5 i_6 i_7$$

Codarea înseamnă calculul simbolurilor de control C_1, C_2, şi C_4 când se dau simbolurile de informaţie i_3, i_5, i_6, i_7.

Relaţia de codare:

$$H \cdot V^T = 0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.1)$$

unde *H* reprezintă matricea de control, ce este de dimensiune *m×n*.

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.2)$$

Suma făcându-se modulo doi.

Exemplul 1:

Pentru n=7, k=4, m=3 şi secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod,V, se poate scrie sub formă literară: $V = C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 7 simboluri ce alcătuiesc cuvântul de cod şi 3 biţi de control. Rezultă că matricea de control, H, va conţine 3 linii şi 7 coloane.
Din relaţia de codare (2.1.1), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \end{aligned}$$

Rezultă cuvântul de cod:

$$V=1011010$$

2.1.1.2. Decodarea codului Hamminmg corector de o eroare

Având cuvântul recepţionat V', compus din 7 imboluri:

$$V' = C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului Hamming cu relaţia:

$$z = H \cdot V'^{T} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \end{bmatrix} \qquad (2.1.3)$$

Matricea de control:

$$H_{[m,n]} = \begin{bmatrix} h_1 & \cdots & h_i & \cdots & h_n \end{bmatrix} \qquad (2.1.4)$$

unde, pentru acest caz se observă că m=3 şi n=7. Coloana h_i exprimă în cod binar natural numărul de ordine al coloanei respective, cu bitul cel mai puţin semnificativ în linia *m*.

Din relaţia (2.1.3) rezultă:

$$\begin{aligned} z_4 &= C_4' + i_5' + i_6' + i_7' \\ z_2 &= C_2' + i_3' + i_6' + i_7' \\ z_1 &= C_1' + i_3' + i_5' + i_7' \end{aligned} \qquad (2.1.5)$$

Decodarea zecimală a corectorului z, indică poziţia erorii, dacă este una singură, în cuvântul V'. Va fi eronat simbolul cu indicele r, dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 \qquad (2.1.6)$$

Cuvântul recepţuionat se mai poate scrie ca fiind:

$$V' = V + \varepsilon,$$

unde ε reprezintă cuvântul eroare.
Relaţia (2.1.3) va deveni:

$$z = H \cdot V'^{T} = H \cdot (V + \varepsilon)^{T} = H \cdot \varepsilon^{T} = h_r$$

unde poziţia erorii se calculează cu relaţia (2.1.6).

Obs: *În cazul în care există două erori, sunt cazuri în care nu numai că nu se corectează nici o eroare, dar se mai eronează un al treilea simbol, rezultând la recepţie trei simboluri eronate.*

Exemplul 2:

Decodaţi cuvântul recepţionat V'=1001001

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_1' C_2' i_3' C_4' i_5' i_6' i_7'$$

$$V' = 1\ 0\ 0\ 1\ 0\ 0\ 1$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4' + i_5' + i_6' + i_7' = 1 + 0 + 0 + 1 = 0$$
$$z_2 = C_2' + i_3' + i_6' + i_7' = 0 + 0 + 0 + 1 = 1$$
$$z_1 = C_1' + i_3' + i_5' + i_7' = 1 + 0 + 0 + 1 = 0$$

Aşadar, poziţia eronată este:

$$r = 4z_4 + 2z_2 + z_1 = 4 \cdot 0 + 2 \cdot 1 + 1 \cdot 0 = 2$$

Rezultă cuvântul eroare:

$$\varepsilon = 0100000$$

Se poate scrie:

$$V = V' + \varepsilon = 1001001 + 0100000 = 1101001$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 0001$$

2.1.2. Cod Hamming corector de o eroare şi detector de două erori

Condiţia necesară şi suficientă pentru ca un cod să poată simultan corecta maxim *t* erori şi a detecta maxim *e* erori este ca distanţa minimă a codului să fie:

$$d \geq t + e + 1 = 1 + 2 + 1 = 4, \qquad e > t$$

2.1.2.1. Codarea codului Hamminmg corector de o eroare şi detector de două erori

Pentru a înlătura dezavantajul codului Hamming corector de o eroare (acela de a erona suplimentar, la depăşirea capacităţii de corecţie a codului, t>1) şi de al face mai util în aplicaţii

practice, codul a fost modificat în sensul creşterii distanţei minime de la d=3 la d=4, ceea ce permite detecţia erorilor duble.

Creşterea distanţei de cod de la 3 la 4 s-a făcut prin adăugarea unui simbol de control suplimentar, numit simbol de control al parităţii, C_0, structura cuvîntului de cod devenind:

$$V = C_0C_1C_2i_3C_4i_5i_6i_7$$

Matricea de control se modifică şi are structura:

$$H' = \begin{bmatrix} 0 & H \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Relaţia de codare va fi:

$$H' \cdot V^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ i_3 \\ C_4 \\ i_5 \\ i_6 \\ i_7 \end{bmatrix} = 0 \qquad (2.1.7)$$

Sau în formă explicită:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 \\ C_2 &= i_3 + i_6 + i_7 \\ C_4 &= i_5 + i_6 + i_7 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 \end{aligned} \qquad (2.1.8)$$

Exemplul 3:

Fie secvenţa de informaţie i=1010, găsiţi cuvântul V de cod.

Rezolvare:

Se observă că există 4 simboluri de informaţie: i_3, i_5, i_6, i_7. Cuvântul de cod, V, se poate scrie sub formă literară: $V = C_0C_1C_2i_3C_4i_5i_6i_7$. Avem aşadar 8 simboluri ce alcătuiesc cuvântul de cod şi 4 biţi de control. Rezultă că matricea de control, H', va conţine 4 linii şi 8 coloane.
Din relaţia de codare (2.1.8), rezultă relaţiile de calcul a biţilor de control:

$$\begin{aligned} C_1 &= i_3 + i_5 + i_7 = 1+0+0 = 1 \\ C_2 &= i_3 + i_6 + i_7 = 1+1+0 = 0 \\ C_4 &= i_5 + i_6 + i_7 = 0+1+0 = 1 \\ C_0 &= C_1 + C_2 + i_3 + C_4 + i_5 + i_6 + i_7 = 1+0+1+1+0+1+0 = 0 \end{aligned}$$

Rezultă cuvântul de cod:

V=01011010

2.1.2.2. Decodarea codului Hamminmg corector de o eroare şi detector de două erori

Având cuvântul recepţionat V', compus din 8 imboluri:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$

se poate calcula corectorul codului cu relaţia:

$$Z = H' \cdot V'^{T} = \begin{bmatrix} z \\ z_0 \end{bmatrix} = \begin{bmatrix} z_4 \\ z_2 \\ z_1 \\ z_0 \end{bmatrix} \qquad (2.1.9)$$

Unde:

- z, reprezintă corectorul de la codul Hamming corector de o eroare;
- z_0, reprezintă simbolul binar, 0 sau 1, cu ajutorul căruia se pot detecta erori pare (z_0=0).

Există patru cazuri:

Cazul 1:

$$\begin{cases} z = 0 \\ z_0 = 0 \end{cases}$$

nu există erori sau nu există erori detectabile prin cod.

Cazul 2:

$$\begin{cases} z \neq 1 \\ z_0 = 0 \end{cases}$$

se face detecţia erorilor duble.

Cazul 3:

$$\begin{cases} z = 0 \\ z_0 = 1 \end{cases}$$

simbolul C_0 este eronat.

Cazul 4:

$$\begin{cases} z \neq 0 \\ z_0 = 1 \end{cases}$$

există o eroare corectabilă.

Va fi eronat simbolul cu indicele r-1, unde r este dat de relaţia:

$$r = 4z_4 + 2z_2 + z_1 + z_0 \qquad (2.1.10)$$

Exemplul 4:

Decodaţi cuvântul recepţionat V'=01001010

Rezolvare:

Cuvântul de cod recepţionat se poate scrie ca fiind:

$$V' = C_0'C_1'C_2'i_3'C_4'i_5'i_6'i_7'$$
$$V' = 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0$$

Relaţia (2.1.5) va deveni:

$$z_4 = C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 1+0+1+0 = 0$$
$$z_2 = C_2^{'} + i_3^{'} + i_6^{'} + i_7^{'} = 0+0+1+0 = 1$$
$$z_1 = C_1^{'} + i_3^{'} + i_5^{'} + i_7^{'} = 1+0+0+0 = 1$$
$$z_0 = C_0^{'} + C_1^{'} + C_2^{'} + i_3^{'} + C_4^{'} + i_5^{'} + i_6^{'} + i_7^{'} = 0+1+0+0+1+0+1+0 = 1$$

Rezultă că avem situaţia cazului 4, când există o eroare corectabilă.
Aşadar, rezultă r:

$$r = 4z_4 + 2z_2 + z_1 + z_0 = 4\cdot 0 + 2\cdot 1 + 1\cdot 1 + 1 = 4$$

Deci va fi eronat simbolul cu indicele r-1, adică 4-1=3, i_3.
Rezultă cuvântul eroare:

$$\varepsilon = 00010000$$

Se poate scrie:

$$V = V^{'} + \varepsilon = 01001010 + 00010000 = 01011010$$

Rezultă secvenţa de informaţie:

$$i = i_3 i_5 i_6 i_7 = 1010$$

Desfăşurarea lucrării:
a) Se lansează executabilul Hamming.exe din directorul Hamming, după care se introduce parola TTI. Rulaţi programul, selectând opţiunea Demonstraţie, pentru toate cele patru variante prezentate.
b) Se verifică, cu programul, toate exemplele luate în această lucrare, selectând pe rând codurile prezentate.
c) Pentru fiecare cod în parte, se alege câte un exemplu de codare/decodare, asemănător cu cele prezentate în lucrare, cu observaţia ca secvenţa de informaţie să conţină 11 biţi de informaţie, la codare, respectiv cuvîntul recepţionat să conţină 15/16 biţi, la decodare. Se realizează codarea/decodare, după care, cu programul, se verifică rezultatele obţinute.
d) La sfârşitul lucrării de laborator se va efectua un test asupra cunoştinţelor acumulate. Rezultatul testului se va concretiza printr-o notă, calculată automat de calculator.

2.2. Codul ciclic

Codurile ciclice sunt utilizate pentru protejarea informaţiei împotriva perturbaţiilor.

Codurile ciclice sunt coduri bloc (toate cuvintele au aceeaşi lungime, codarea şi decodarea unui bloc este independentă de a celorlalte).

Denumirea de "ciclice" provine de la proprietatea pe care o au cuvintele de cod şi anume dacă v_i = 1001001 este cuvânt de cod atunci şi v_j = 0010011 şi v_k = 1100100 sunt cuvinte de cod . Adică orice permutare ciclică a unui cuvânt de cod este un alt cuvânt de cod .

Parametrii codului sunt n (biţi) lungimea cuvântului de cod, k numărul de biţi de informaţie per cuvânt, m numărul de biţi de control per cuvânt şi polinomul generator g(x). Deşi poate fi facută şi o codare nesistematică vom considera codul sistematic cei k biţi de informaţie fiind situaţi pe poziţiile cele mai semnificative, iar cei m biţi de control pe poziţiile mai puţin semnificative.
Fiecărui cuvânt de cod i se poate ataşa un polinom de grad maxim n-1:

$$v = v_{n-1}v_{n-2}v_{n-3}.....v_1v_0,$$

coeficienţii v_i fiind coeficienţi binari (din câmpul binar {0,1}).
Polinomul asociat cuvântului va fi :

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + v_{n-3}x^{n-3} + ...+ v_1x + v_0 \qquad (2.2.1)$$

Ponderea unui cuvânt reprezintă numărul de 1 din cadrul cuvântului.
Polinomul de informaţie este :

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + i_{k-3}x^{k-3} + ...+ i_1x + i_0 \qquad (2.2.2)$$

de grad k-1.

Pentru orice cuvânt de cod este valabilă relaţia :

$$\text{rest}\frac{v(x)}{g(x)} = 0 \qquad (2.2.3)$$

Deci cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului g(x) numit polinomul generator al carui grad va fi deci m = n-k, ($g_m = 1$):

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + ...+ g_1x + g_0 \qquad (2.2.4)$$

Operaţiile se fac modulo $(x^n + 1)$.
Dacă, codul este corector de o singură eroare atunci:

$$n = 2^m - 1. \qquad (2.2.5)$$

Codurile ciclice corectoare de o eroare, având distanţa de cod (distanţa Hamming minimă între cuvintele codului) $d_{Hmin} = 3$, sunt capabile să corecteze o eroare sau să detecteze două.

Codarea codului ciclic corector de o eroare

Codarea va fi prezentată în două moduri nesistematică sau sistematică.

Cu relaţia $v(x) = i(x)\cdot g(x)$ se poate face codarea nesistematică, dar pentru că nu este utilă în practică nu va fi luată în considerare .

În continuare va fi deci prezentată codarea sistematică unde corespondenţa dintre i(x) şi v(x) este dată prin relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\frac{i(x)\cdot x^m}{g(x)} \qquad (2.2.6)$$

unde $\text{rest}\frac{i(x)\cdot x^m}{g(x)}$ semnifică restul împărţirii polinomului $i(x)\cdot x^m$ la g(x).

Operaţia de adunare din ecuaţia (2.2.6) este sumă modulo 2 (SAU exclusiv), iar coeficienţii polinoamelor sunt din câmpul binar {0,1}.
Matematic codarea poate fi facută polinomial sau matricial.

a) Polinomial

Codarea va fi prezentată printr-un exemplu, putând fi uşor generalizată.

Fie $g(x) = x^3 + x + 1$. Deci m = 3 şi cu relaţia $2^m - 1 = n$ rezultă că n = 7 de unde k = n – m = 4. Vom avea ca atare 4 biţi de informaţie $i = v_{n-1}v_{n-2}v_{n-3}v_{n-4}$ cu polinomul asociat :

$$i(x) = v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x + v_{n-4} \qquad (2.2.7)$$

şi deci $i(x)\cdot x^m = (v_{n-1}x^3 + v_{n-2}x^2 + v_{n-3}x^3 + v_{n-1}x + v_{n-4})\cdot x^3 = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3$

Biţii de control sunt coeficienţii restului împărţirii lui $i(x)\cdot x^m/g(x)$.

$$\begin{array}{l|l} v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 & \underline{x^3 + x + 1} \\ \underline{v_{n-1}x^6 \quad + v_{n-1}x^4 + v_{n-1}x^3} & v_{n-1}x^3 + v_{n-2}x^2 + (v_{n-3} + v_{n-1})x + (v_{n-4} + v_{n-2} + v_{n-1}) \end{array}$$

$$/\ \ v_{n-2}x^5 + (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-1})x^3$$
$$\underline{v_{n-2}x^5 \qquad\qquad + v_{n-2}x^3 \qquad + v_{n-2}x^2}$$

$$/\ \ (v_{n-3} + v_{n-1})x^4 + (v_{n-4} + v_{n-2} + v_{n-1})x^3 + v_{n-2}x^2$$
$$\underline{(v_{n-3} + v_{n-1})x^4 \qquad\qquad + (v_{n-3} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x}$$

$$/\ \ (v_{n-4} + v_{n-2} + v_{n-1})x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-3} + v_{n-1})x$$
$$\underline{(v_{n-4} + v_{n-2} + v_{n-1})x^3 \qquad\qquad + (v_{n-4} + v_{n-2} + v_{n-1})x + v_{n-4} + v_{n-2} + v_{n-1}}$$

$$(v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

$$\Rightarrow r(x) = (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}, \qquad (2.2.8)$$

unde r(x) este restul împărţirii .
Conform relaţiei (2.2.6) se obţine :

$$v(x) = v_{n-1}x^6 + v_{n-2}x^5 + v_{n-3}x^4 + v_{n-4}x^3 + (v_{n-3} + v_{n-2} + v_{n-1})x^2 + (v_{n-4} + v_{n-3} + v_{n-2})x + v_{n-4} + v_{n-2} + v_{n-1}$$

Pentru că n = 7 şi v(x) este de forma:

$$v(x) = v_6x^6 + v_5x^5 + v_4x^4 + v_3x^3 + v_2x^2 + v_1x + v_0 \qquad (2.2.9)$$

şi ţinând cont de relaţia (2.2.8) simbolurile de control vor fi date de:

$$\begin{aligned} v_2 &= v_4 + v_5 + v_6 \\ v_1 &= v_3 + v_4 + v_5 \\ v_0 &= v_3 + v_5 + v_6 \end{aligned} \qquad (2.2.10)$$

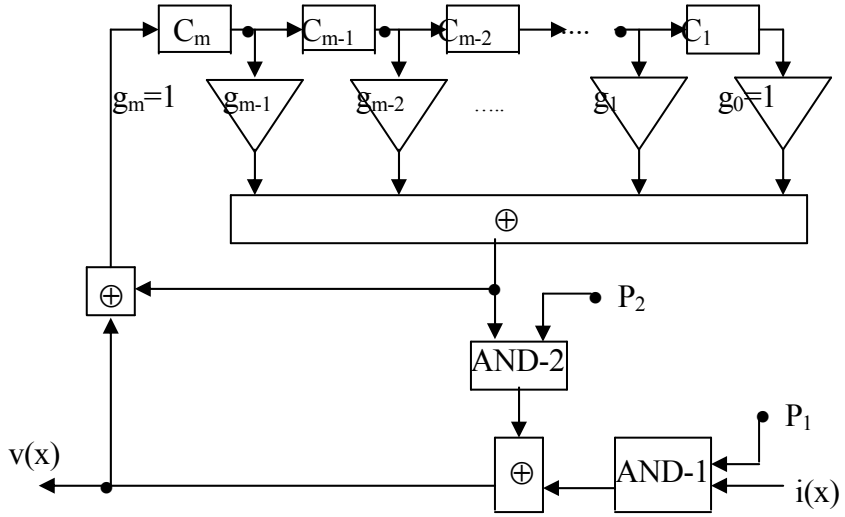
Aşadar cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$ unde primii 4 sunt biţii de informaţie, iar ultimii 3 cei de control.

b) Codor ciclic cu RDR (codarea matricială)

În Fig. 2.2.1 este prezentat un codor ciclic care implementează relaţia de codare (2.2.6). Secvenţa de informaţie, i(x), intră în codor în primele k tacte, primul bit fiind cel mai semnificativ şi, de asemenea, este conectată şi la ieşire. Pentru aceasta, întrerupătorul 1, format din poarta AND-1 este închis iar întrerupătorul 2, format din poarta AND-2 este deschis (vezi semnalele de comandă P_1 şi P_2).

În următoarele m tacte întrerupătorul 1 este deschis iar întrerupătorul 2 este închis, astfel că secvenţa r, generată de RDR, este livrată la ieşirea v.

a) Codor ciclic RDR ;



b) Semnalele de validare a porţilor „AND”

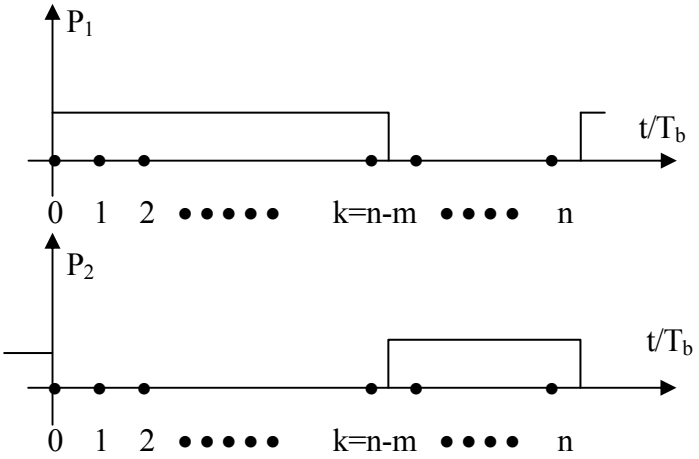


Fig. 2.2.1 Codor ciclic cu RDR şi semnalele de comandă

Obs. *–Se observă că, în ultimele k tacte, la intrările sumatorului (care adună intrarea RDR-lui cu reacţia sa) se regăseşte acelaşi semnal, astfel că, în aceste k tacte, în celula C_k se va introduce o secvenţă nulă care „curăţă” registrul pentru un nou cuvânt de cod.*

<u>Din funcţionarea registrului de deplasare cu reacţie rezultă relaţia:</u>

$$S_j = TS_{j-1} + v_{n-j}U \tag{2.2.11}$$

unde S reprezintă starea registrului, U este o matrice coloană, iar T este matricea caracteristică a registrului de deplasare cu reacţie. Ele sunt de forma:

$$S=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}, \qquad U=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, \qquad T=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ g_0 & g_1 & g_2 & \cdots & g_{m-1} \end{bmatrix}. \tag{2.2.12}$$

Ţinând cont de observaţia făcută mai sus rezultă că $S_n = 0$.
Considerând acelaş exemplu

Matricea caracteristică, T, este:

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \tag{2.2.13}$$

Utilizând relaţiile (2.2.11) şi (2.2.13) pentru cazul n = 7 vom avea:

$$S_7 = v_6T^6U + v_5T^5U + v_4T^4U + v_3T^3U + v_2T^2U + v_1TU + v_0U = 0 \tag{2.2.14}$$

Efectuând calculele rezultă:

$$T \cdot U = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},\ T^2 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix},\ T^3 \cdot U = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},\ T^4 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$
$$T^5 \cdot U = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix},\ T^6 \cdot U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.2.15}$$

Introducând (2.2.15) în (2.2.14) şi efectuând calculele obţinem:

$$v_2 = v_4 + v_5 + v_6$$
$$v_1 = v_3 + v_4 + v_5$$
$$v_0 = v_4 + v_3 + v_2 = v_4 + v_3 + v_4 + v_5 + v_6 = v_3 + v_5 + v_6$$

relaţii identice cu relaţiile (2.2.10) deci cele două proceduri de calcul ne-au dus la aceleaşi rezultate .

Cuvântul de cod va fi $v = v_6v_5v_4v_3v_2v_1v_0$.

Decodarea codului ciclic corector de o eroare

Decodarea codului ciclic cuprinde verificarea corectitudinii cuvântului recepţionat:

$$w(x) = v'_{n-1}x^{n-1} + v'_{n-2}x^{n-2} + ... + v'_1x + v'_0, \tag{2.2.16}$$

corectarea sau detectarea erorilor, după destinaţia codului şi apoi selecţia biţilor de informaţie.

Verificarea presupune calculul restului împărţirii lui w(x) la g(x). Dacă restul este 0 se decide: cuvânt corect recepţionat. Dacă nu, atunci se decide: cuvânt eronat.(Prima decizie poate fi falsă, a doua nu!) Pentru codurile detectoare decodarea ia sfârşit aici. Pentru codurile corectoare analiza restului permite determinarea poziţiei erorilor. Acest lucru presupune existenţa unei corespondenţe biunivoce între resturile posibile şi cuvintele eroare corectabile de către codul dat.

Dacă codul este corector de o eroare, atunci decodarea decurge astfel:

1°- se calculează corectorul z_n cu formula:

$$z = \sum_{j=0}^{n-1} v'_j T^j U = \sum_{j=0}^{n-1} \varepsilon_j T^j U \tag{2.2.17}$$

unde ε_j reprezintă coeficienţii polinomului eroare:

$$\varepsilon(x) = w(x) + v(x) \tag{2.2.18}$$

2°- dacă z = 0 se decide că w(x) este corect w(x) = v(x)

- dacă $z \neq 0$ atunci $z = T^rU$, unde r este indicele coeficientului eronat. Comparăm z cu T^jU şi găsim r. Corecţia presupune schimbarea valorii lui v'_r.

Dacă codul ciclic corectează mai multe erori, atunci decodarea se poate face pe seama corespondenţei amintite anterior.

a) Decodarea polinomială

Metoda constă în împărţirea lui w(x) la g(x). Va rezulta un polinom rest, r(x). Dacă acesta este diferit de 0, cuvântul recepţionat este eronat şi prin identificarea restului r(x) cu valorile din tabelul cuvinte eroare – corectori (vezi Anexe) se determină poziţia erorii şi se realizează corecţia.

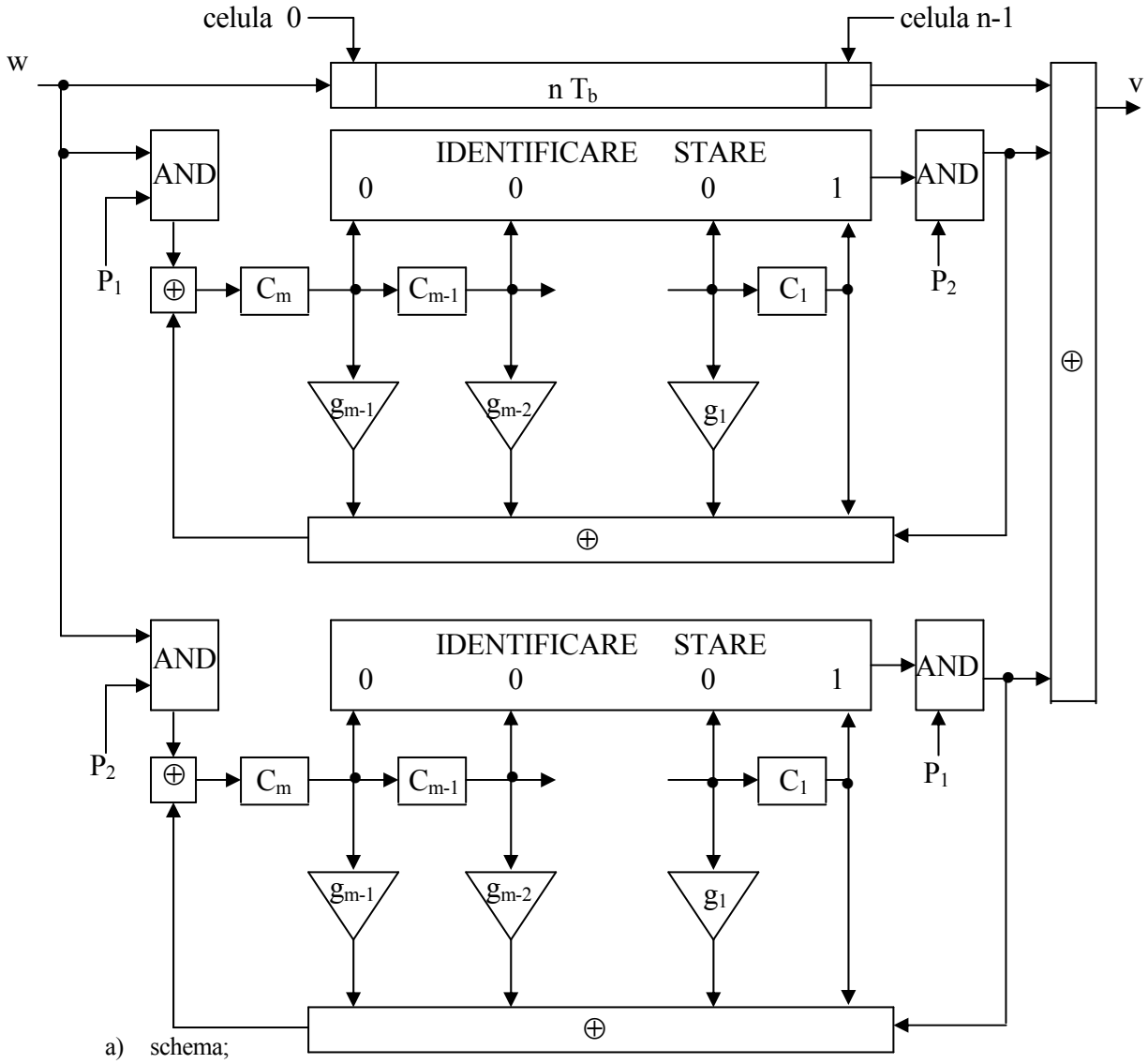
Spre exemplu fie : $w(x) = x^6 + x^5 + x^3 + x^2 + 1$. Calculăm rest[$w(x) / g(x)$]:

$$
\begin{array}{l|l}
x^6 + x^5 + x^3 + x^2 + 1 & x^3 + x + 1 \\
\underline{x^6 + x^4 + x^3} & \underline{x^3 + x^2 + x + 1} \\
x^5 + x^4 + x^2 + 1 & \\
\underline{x^5 + x^3 + x^2} & \\
x^4 + x^3 + 1 & \\
\underline{x^4 + x^2 + x} & \\
x^3 + x^2 + x + 1 & \\
\underline{x^3 + x + 1} & \\
x^2 &
\end{array}
$$

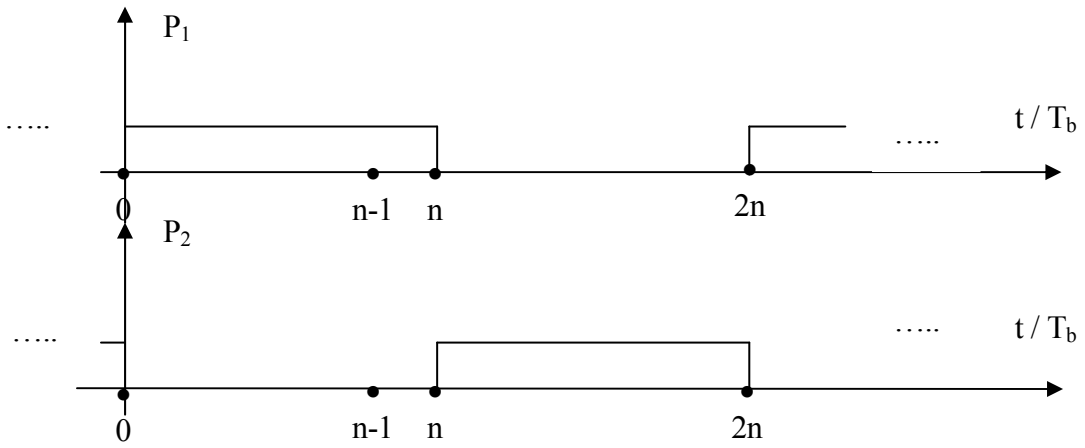
deci conform tabelului din anexă este eronat w_2. Schimbând valoarea bitului w_2 rezultă :

$$w(x) = x^6 + x^5 + x^3 + 1.$$

b) Decodor ciclic cu RDR (decodarea matricială)



a) schema;



b)Semnalele de validare a porţilor AND ;

Fig. 2.2.2. Decodor ciclic corector de o eroare

În Fig. 2.2.2. este prezentată structura unui decodor ciclic corector de o eroare. Biţii cuvântului recepţionat vor intra în schemă unul după altul în ordine, începând cu bitul v'_{n-1}.

Schema de decodare conţine un registru de deplasare (blocul „nT_b" este un registru de întârziere cu n celule) cu rol de memorie, care păstrează cuvântul recepţionat care intră pas cu pas şi două subscheme cu RDR pentru corecţie care funcţionează în contratimp. Procedura de decodare durează 2n tacte. În primele n tacte în memorie intră cuvântul 1 care prin poarta P_1 intră şi în primul decodor, ieşirea acestuia fiind 0 pentru că P_2 este închis. În următoarele n tacte cuvântul 2 va intra prin poarta P_2 în al doilea decodor care are ieşirea 0 deoarece P_1 este închis, în acest timp cuvântul 1 părăseşte memoria trecând prin sumator şi este corectat de primul decodor care identifică starea 00.....1 şi care are acces la sumator prin P_2.

După n = 7 tacte starea registrului va fi:

$$S'_7 = v'_6T^6U + v'_5T^5U + v'_4T^4U + v'_3T^3U + v'_2T^2U + v'_1TU + v'_0U \qquad (2.2.19)$$

care nu va mai fi 0 decât în cazul în care nu avem eroare.

Dacă eroarea afectează bitul v_r starea registrului de deplasare cu reacţie după intrarea întregului cuvânt este:

$$S'_7 = v'_6T^6U + v'_5T^5U + ... + v'_rT^rU + ... + v'_1TU + v'_0U \qquad (2.2.20)$$

Deoarece la emisie $S_7 = 0$ şi $v'_j = v_j$ pentru $j \neq r$ şi $v'_r = v_r + 1$ rezultă:

$$S'_7 + S_7 = S'_7 = T^rU = z \qquad (2.2.21)$$

Pe durata următoarelor n = 7 tacte RDR din prima subschemă va evolua numai sub acţiunea semnalului de reacţie şi după n-r-1 tacte se va ajunge în starea:

$$S'_{7+n-r-1} = T^{n-r-1}\, S'_7 = T^{n-1}U = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad (2.2.22)$$

În acelaş timp printr-o deplasare cu n-r-1 tacte eroarea care se afla în celula r va ajunge în celula n-1. Detectorul va sesiza starea $S'_{7+n-r-1}$ fixă şi prin poarta P_2 care este închisă va emite semnalul de corecţie (blocul IDENTIFICĂ generează un 1^L) care se însumează cu bitul eronat aflat în celula n-1, astfel la ieşirea sumatorului final se va obţine cuvântul corectat.

Desfăşurarea lucrării:

a) Rulaţi programul de pe calculator alegând butonul “Demonstraţie” de la codul ciclic şi se urmăreşte o codare şi o decodare.

b) Alegeţivă un polinom generator şi secvenţa de informaţie şi efectuaţi o codare, apoi alegând polinomul generator şi biţii recepţionaţi o decodare. Verificaţi apoi rezultatul cu ajutorul calculatorului.

c) La sfârşitul lucrării de laborator se efectuează testul de verificare a cunoştinţelor acumulate prin intermediul calculatorului. Testul constă în a răspunde la 5 întrebări teoretice, fiecare având un răspuns corect din cele cinci propuse, şi efectuarea unei codări şi a unei decodări pornind de la polinomul generator şi secvenţa de informaţie respectiv secvenţa recepţionată generate aleator de calculator.

L3. Coduri ciclice corectoare de erori multiple

3.1 Codul BCH

Codurile BCH fac parte din categoria codurilor ciclice.
Cuvintele de cod BCH vor avea deci structura ca şi cele de cod ciclic:

$$v = v_{n-1}\, v_{n-2} \ldots v_1\, v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.1.1}$$

unde coeficienţii u_j sunt coeficienţi binari (fac parte din câmpul binar {0, 1}).
Polinomul corespunzător acestui cuvânt fiind:

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1x + v_0 \tag{3.1.2}$$

adică un polinom de grad n - 1.

Polinomul de informaţie este:

$$i(x) = i_{k-1}x^{k-1} + i_{k-2}x^{k-2} + \ldots + i_1x + i_0 \tag{3.1.3}$$

care este un polinom de gradul k - 1.

Cuvintele de cod se aleg astfel încât să fie multiplii ai polinomului generator g(x) şi deci acesta va trebui sa fie un polinom de gradul m = n – k:

$$g(x) = g_mx^m + g_{m-1}x^{m-1} + \ldots + g_1x + g_0 \tag{3.1.4}$$

Coeficienţii polinomului g(x) sunt de asemenea coeficienţi binari.
Deoarece polinomul trebuie să aibă gradul m rezultă că $g_m = 1$, iar g_0 trebuie să fie şi el egal cu 1 pentru că altfel putem da factor pe x şi gradul polinomului va fi mai mic decât k.

Polinomul g(x) este deci de forma:

$$g(x) = x^m + g_{m-1}x^{m-1} + \ldots + g_1x + 1 \tag{3.1.5}$$

Cuvintele de cod sunt elemente ale câmpului Galois $GF(2^q)$ generat de un polinom primitiv p(x) de grad q (sunt clase de resturi modulo p(x)), câmp obţinut aşa cum este prezentat în Anexa.

Codarea BCH

Codarea codurilor BCH poate fi făcută în două moduri:

a) Cu relaţia

$$v(x) = i(x)\cdot g(x) \tag{3.1.6}$$

care conduce la obţinerea unui cod nesistematic (relaţie mai puţin utilizată).

b) Pentru obţinerea unei structuri sistematice, la care informaţia să se găsească nemodificată pe poziţiile cele mai semnificative se foloseşte relaţia:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m/g(x)) \tag{3.1.7}$$

Această relaţie se mai poate scrie:

$$v(x) = q(x)\cdot g(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1 x + v_0 \qquad (3.1.8)$$

deci se obţine un cuvânt de cod ciclic care este multiplu a lui g(x) şi este format din simbolurile de informaţie aflate pe poziţiile cele mai semnificative (primele k) şi m simboluri de control determinate de restul împărţirii lui $i(x)\cdot x^m$ la g(x).
Relaţia (3.1.8) poate fi adusă sub forma $H\cdot v^T = 0$.

$$\begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \qquad (3.1.9)$$

Acesta este de forma $H\cdot v^T = 0$ deci îl putem determina pe H prin identificare ca fiind:

$$H = \begin{bmatrix} h_0 & h_1 & \ldots & h_{m-1} & h_m & 0 & \ldots & 0 & 0 \\ 0 & h_0 & \ldots & h_{m-2} & h_{m-1} & h_m & \ldots & 0 & 0 \\ \vdots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & h_0 & \ldots & h_{m-1} & h_m \end{bmatrix} \qquad (3.1.10)$$

Pentru a corecta t erori independente g(x) trebuie să îndeplinească anumite condiţii. Ştiim că v(x) trebuie să fie multiplu al lui g(x) şi că g(x) trebuie să fie divizor a lui $x^n + 1 = 0$.
Pentru aceasta alegem un număr de r rădăcini ale lui $x^n + 1 = 0$ pe care le notăm cu $\beta_i = \alpha^i \in GF(2^q)$. Aceste rădăcini β_i sunt elemente primitive ale extensiei de ordinul q al câmpului Galois binar $GF(2^q)$. Acest ordin poate fi determinat cu relaţia:

$$n = 2^q - 1 \qquad (3.1.11)$$

de aici rezultând q şi extensia în care se lucrează $GF(2^q)$.
Pe g(x) îl vom determina ca cel mai mic multiplu comun al polinoamelor minimale a rădăcinilor β_i.

$$g(x) = c.m.m.m.c.\{m_1(x),\ldots m_r(x)\} \qquad (3.1.12)$$

Iar dacă toate aceste r polinoame sunt relativ prime între ele atunci:

$$g(x) = \prod_{i=1}^{r} m_i(x) \qquad (3.1.13)$$

Expresia polinomului minimal care este definit ca polinomul $m_i(x)$ ireductibil de grad minim pentru care $m_i(\beta_i) = 0$ este:

$$m_i(x) = (x+\beta_i)\left(x+\beta_i^{2}\right)\ldots\left(x+\beta_i^{2^{q-1}}\right) \qquad (3.1.14)$$

Deşi în calculul lui $m_i(x)$ folosim coeficienţi $GF(2^q)$ coeficienţii lui $m_i(x)$ vor fi binari, fapt care iese în evidenţă şi din calculul acestor coeficienţi în cazul exemplului 1 prezentat mai jos.
Pentru corecţia unui număr de maxim t erori se impune alegerea unui număr de r = 2t rădăcini $\beta_i \in GF(2^q)$ ceea ce determină obţinerea unui cuvânt de cod având distanţa minimă $d \geq 2t + 1$.

În cazul codurilor ciclice binare, dacă α este o rădăcină şi α^2, $\alpha^{2\cdot 2}$, ... $\alpha^{2\cdot(q-1)}$ sunt tot rădăcini şi deci pentru a forma polinomul generator este suficient să luăm rădăcinile impare:

$$\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}.$$

Structura matricii de control H în cazul codurilor BCH se determină deci impunând ca v(x) să aibă rădăcini pe $\beta_1 = \alpha,\ \beta_3 = \alpha^3,\ \dots,\ \beta_{2t-1} = \alpha^{2t-1}$.
Faptul că β_i este rădăcină a cuvântului de cod v(x) se exprimă prin $v(\beta_i) = 0$ sau mai explicit cu relaţia:

$$v(\beta_i) = v_{n-1}\beta_i^{n-1} + \dots + v_1\beta_i^1 + v_0\beta_i^0 = 0 \quad \text{cu } i = 1, 3, \dots, 2t-1 \tag{3.1.15}$$

Pentru fiecare i putem interpreta această relaţie ca pe un produs scalar al cuvântului format prin puterile lui α^i sau (β_i) şi cuvântul de cod:

$$\begin{bmatrix} \alpha^{i\cdot(n-1)} & \cdots & \alpha^{i\cdot 1} & \alpha^{i\cdot 0} \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = 0 \tag{3.1.16}$$

Înlocuind cu valorile corespunzătoare pentru i (i = 1, 3, ..., 2t-1) obţinem:

$$\begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^3 & \alpha^0 \\ \vdots & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \begin{bmatrix} v_{n-1} \\ \vdots \\ v_1 \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{3.1.17}$$

Deci matricea de control H în cazul codurilor BCH este de forma:

$$H = \begin{bmatrix} \alpha^{n-1} & \cdots & \alpha^2 & \alpha^1 & \alpha^0 \\ \alpha^{3\cdot(n-1)} & \cdots & \alpha^6 & \alpha^3 & \alpha^0 \\ \vdots & & & & \\ \alpha^{(2t-1)\cdot(n-1)} & \cdots & \alpha^{(2t-1)\cdot 2} & \alpha^{2t-1} & \alpha^0 \end{bmatrix} \tag{3.1.18}$$

Această matrice este formată din elementele GF(2^q) şi are t linii şi n coloane. Fiecare element poate fi exprimat printr-un cuvânt binar din q biţi şi deci vom putea scrie matricea H şi sub forma binară dispunând vertical cuvintele binare care înlocuiesc puterile lui α. Forma binară a matricii H va avea astfel q·t linii şi tot n coloane.

Exemplul 1:

Proiectarea unui cod BCH de lungime n = 15 şi corectând 2 erori cu determinarea polinomului generator şi a relaţiilor de codare.

$$n = 2^q - 1 = 15 \Rightarrow q = 4 \Rightarrow \text{extensia GF}(2^4)$$

Elementele lui GF(2^q) sunt clase de resturi modulo p(x) un polinom primitiv de gradul q = 4.

Un polinom p(x) este primitiv dacă este ireductibil şi binomul $x^n + 1$ pe care p(x) îl divide are cel puţin gradul $n = 2^q - 1$. Fie acest polinom primitiv $p(x) = x^4 + x + 1$. Putem să constatăm că $x^4 + x + 1$ divide pe $x^{15} + 1$ ($n = 2^q - 1 = 2^4 - 1 = 15$), dar nu divide nici un $x^n + 1$ cu $1 \le n < 15$ deci este primitiv. Se ştie că pentru orice câmp Galois $\alpha^n = 1$, deci la noi $\alpha^{15} = 1$.
Câmpul GF(2^4) generat de $p(x) = x^4 + x + 1$ obţinut printr-o rădăcină primitivă α a lui $x^4 + x + 1$ cu relaţia $\alpha^4 = \alpha + 1$ este cel dat în Anexa:

Pentru t = 2 se aleg rădăcinile $\beta_1 = \alpha$ si $\beta_3 = \alpha^3$, suntem în cazul binar şi alegem doar rădăcinile impare până la ordinul 2t-1 = 4-1 = 3).
Polinoamele minimale vor fi conform relaţiei (3.1.14):

$$m_1(x) = (x + \alpha)\cdot(x + \alpha^2)\cdot(x + \alpha^4)\cdot(x + \alpha^8) = x^4 + x^3(1 + \alpha^2 + \alpha^2 + 1 + \alpha + \alpha) + x^2(1+ \alpha + \alpha^2 + 1+ \alpha + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^3 +\alpha + \alpha^3 + \alpha^3 + \alpha + \alpha^2) + x(1 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3) + \alpha^{15} = x^4 + x + 1$$

$$m_3(x) = (x + \alpha^3)\cdot(x + \alpha^6)\cdot(x + \alpha^{12})\cdot(x + \alpha^{24}) = x^4 + x^3(\alpha^{24} + \alpha^{12} + \alpha^6 + \alpha^3) + x^2(\alpha^{36} + \alpha^{30} + \alpha^{18} + \alpha^{27} + \alpha^{15} + \alpha^9) + x(\alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{21}) + \alpha^{45} = x^4 + x^3 + x^2 + x + 1$$

În aceste două relaţii puterile lui α mai mari decât 15 s-au exprimat în funcţie de α^{15}, iar suma s-a efectuat modulo doi.
Cele două polinoame fiind prime între ele şi folosind relaţia (3.1.13), g(x) se obţine:

$$g(x) = m_1(x)\cdot m_3(x) = (x^4 + x + 1)\cdot(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^5 + x^4 + x^5 + x^4 + x^3 + x^2 + x + x^4 + x^3 + x^2 + x + 1 = x^8 + x^7 + x^6 + x^4 + 1$$

$\Rightarrow$ m = 8 simboluri de control $\Rightarrow$ k = 15 - 8 = 7 simboluri de informaţie, adică g = 111010001. Cuvântul de cod va fi:

$$v = v_{14}v_{13}v_{12}v_{11}v_{10}v_9v_8v_7v_6v_5v_4v_3v_2v_1v_0$$

unde primii 7 sunt biţii de informaţie şi ultimii 8 cei de control.
Matricea de control H va avea 2 linii (t = 2) şi 15 coloane (n = 15):

$$H = \begin{bmatrix} \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^9 & \alpha^8 & \alpha^7 & \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 & \alpha^2 & \alpha^1 & 1 \\ \alpha^{42} & \alpha^{39} & \alpha^{36} & \alpha^{33} & \alpha^{30} & \alpha^{27} & \alpha^{24} & \alpha^{21} & \alpha^{18} & \alpha^{15} & \alpha^{12} & \alpha^9 & \alpha^6 & \alpha^3 & 1 \end{bmatrix}$$

Luând puterile lui α modulo 15 şi ţinând cont că $\alpha^{15} = 1$ cu ajutorul tabelului câmpului GF(2^4) obţinem exprimarea binară a lui H:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

adică q·t = 4·2 = 8 linii şi n = 15 coloane.

Folosind relaţia (3.1.9) se pot obţine relaţiile de codare.
Pentru o secvenţă informaţională i = 0000001 cuvântul de cod sistematic se poate determina cu relaţia (3.1.7):

$i(x) = 1$
$x^m\cdot i(x) = x^8$
$i(x)\cdot x^m/g(x) = x^8 / x^8 + x^7 + x^6 + x^4 + 1 \Rightarrow$
$\Rightarrow rest(i(x)\cdot x^m/g(x)) = x^7 + x^6 + x^4 + 1$
$\Rightarrow v(x) = x^8 + x^7 + x^6 + x^4 + 1$
adică v = 000000111010001

Decodarea BCH

În cazul codurilor ciclice binare pentru a putea corecta erori este suficient să determinăm poziţia acestora din expresia sindromului. Modul în care poziţia erorilor poate fi găsită cu ajutorul sindromului rezultă din prezentarea următoare.
Un cod ciclic corector de t erori are cuvinte de cod care au un număr r = 2t de rădăcini $\beta_i \in GF(2^q)$, $\beta_i=\alpha^i$:

$$v(\beta_i)= v(\alpha^i) = 0 \qquad (3.1.19)$$

La recepţie trebuie să verificăm legea de codare dată de relaţia (3.1.15). În cazul în care avem erori de tip aditiv putem exprima această lege sub forma:

$$r(\beta_i) = u(\beta_i) + e(\beta_i) = e(\beta_i) = e(\alpha^i) = S_i \qquad (3.1.20)$$

unde $e(\beta_i)$ este eroarea şi S_i este sindromul, iar i = 1, 3, ..., 2t-1 în cazul codurilor binare.

Pentru a vedea cum putem găsi poziţia erorii cu ajutorul sindromului luăm un mic exemplu:

Presupunem un cuvânt oarecare care are erori pe două poziţii de exemplu 1 şi 4 deci $e(x)=x^4+x$. Ţinând cont de relaţia (3.1.20) vom avea sindromul:

$$S_i = e(\alpha^i) = (\alpha^i)^1 + (\alpha^i)^4 = (\alpha^1)^i + (\alpha^4)^i = X_1^i + X_2^i$$

Expresia care indică poziţia erorii s-a notat cu X_k - **locatorul erorii**:

$$X_k = \alpha^j \text{ , } k = 1, \ldots t \text{ şi } j = 0, \ldots n-1 \qquad (3.1.21)$$

Deci avem atâţia locatori câte erori (în număr de t), iar erorile pot apărea pe orice poziţie în cadrul cuvântului (de la 0 la n-1).
Deci putem exprima sindromul S_i sub forma:

$$S_i = \sum_{k=1}^{t} X_k^i \qquad (3.1.22)$$

Această relaţie ne indică un sistem de ecuaţii neliniare care are ca necunoscute pe X_k (adică locatorii).

Pentru realizarea decodării va fi prezentat algoritmul Peterson cu căutare Chien care constă în:
1° Calcularea sindromului erorii

$$S_i = r\left(\alpha^i\right) = \sum_{k=1}^{t} X_k^i \quad \text{cu} \quad i=1, 3, \ldots 2t-1$$

2° Cu ajutorul locatorilor putem forma un polinom al erorilor care are ca rădăcini locatorii:

$$\sigma(x) = \prod_{k=1}^{t} (x + X_k) = x^t + \sigma_1 x^{t-1} + \ldots + \sigma_t \qquad (3.1.23)$$

Dacă punem condiţia ca X_k să fie rădăcină a lui $\sigma(x)$ obţinem:

$$X_k^t + \sigma_1 X_k^{t-1} + \ldots + \sigma_t = 0 \qquad (3.1.24)$$

Dacă înmulţim (3.1.24) cu X_k^i şi însumăm în funcţie de k:

$$\sum_{k=1}^{t} X_k^{t+i} + \sigma_1 \sum_{k=1}^{t} X_k^{t+i-1} + ... + \sigma_t \sum_{k=1}^{t} X_k^{i} = 0 \qquad (3.1.25)$$

Ţinând cont de (3.1.22) relaţia (3.1.25) devine:

$$S_{t+i} + \sigma_1 S_{t+i-1} + ... + \sigma_t S_i = 0 \qquad (3.1.26)$$

Această ecuaţie este un sistem liniar de t ecuaţii cu t necunoscute care se poate rezolva aplicând regula lui Cramer.

Pentru codurile BCH folosind şi relaţia $S_{2k} = S_k^2$ expresiile coeficienţilor σ_i în funcţie de S_i pentru un număr de 1, 2 sau 3 erori sunt:

Tabelul 3.1.1

t	σ_i
1	$\sigma_1 = S_1$
2	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_3 + S_1^3}{S_1}$
3	$\sigma_1 = S_1$ $\sigma_2 = \dfrac{S_1^2 S_3 + S_5}{S_1^3 + S_3}$ $\sigma_3 = S_1^3 + S_3 + S_1 \sigma_2$

3° Căutarea poziţiei eronate.

Cunoscând σ_i putem determina poziţia erorii.

Dacă împărţim relaţia (3.1.24) cu X_k^t rezultă relaţia:

$$\sum_{i=1}^{t} \sigma_i X_k^{-i} = 1 \qquad (3.1.27)$$

i dându-ne poziţia eronată.

Numărul maxim de erori corectabile este t eroarea putându-se găsi pe oricare din cele n poziţii ale cuvântului.

În cazul unei căutări Chien se începe căutarea simbolului eronat de la r_{n-1}. Pentru o poziţie oarecare n-j ţinând cont de (3.1.21) vom avea $X_k = \alpha^{n-j}$

$$\begin{aligned} &\sum_{i=1}^{t} \sigma_i \alpha^{-i(n-j)} = 1 \Rightarrow \sum_{i=1}^{t} \sigma_i \alpha^{-i \cdot n} \cdot \alpha^{i \cdot j} = 1 \\ &\Rightarrow \sum_{i=1}^{t} \sigma_i \left(\frac{1}{\alpha^n} \right)^i \cdot \alpha^{i \cdot j} = 1 \end{aligned} \qquad (3.1.28)$$

Deoarece $\alpha^n = 1$ obţinem ecuaţia de căutare Chien:

$$\sum_{i=1}^{t} \sigma_i \alpha^{i \cdot j} = 1 \quad \text{cu} \quad j = 1,...n \tag{3.1.29}$$

lui j = 1 corespunzându-i simbolul r_{n-1}, iar lui j = n simbolul r_0.
Deci conform ecuaţiei (3.1.29) eroarea apare când suma respectivă este egală cu 1.

Exemplul 2:

Presupunem că recepţionăm un cuvânt BCH cu n = 15 şi k = 7 şi că avem t = 2 erori.

$$r = 110101011010011$$
$$\Rightarrow r(x) = x^{14} + x^{13} + x^{11} + x^9 + x^7 + x^6 + x^4 + x^1 + 1$$

Calculăm sindromul. Pentru t = 2 avem de calculat până la S_3 inclusiv (2t-1 = 2·2-1 = 3):

$$S_1 = r(\alpha) = \alpha^{14} + \alpha^{13} + \alpha^{11} + \alpha^9 + \alpha^7 + \alpha^6 + \alpha^4 + \alpha + 1=$$
$$= 1 + \alpha^3 + 1 + \alpha^2 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 + \alpha + \alpha^3 + 1 + \alpha +$$
$$+ \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha + 1 = \alpha^2 + \alpha + 1 = \alpha^{10}$$

$$S_3 = r(\alpha^3) = \alpha^{42} + \alpha^{39} + \alpha^{33} + \alpha^{27} + \alpha^{21} + \alpha^{18} + \alpha^{12} + \alpha^3$$
$$+ 1= \alpha^{12} + \alpha^9 + \alpha^3 + \alpha^{12} + \alpha^6 + \alpha^3 + \alpha^{12} + \alpha^3 + 1 = \alpha^9 +$$
$$+ \alpha^6 + \alpha^{12} + \alpha^3 + 1 = \alpha + \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^2 + \alpha^3 +$$
$$+ \alpha^3 + 1 = 0$$

Din tabelul 3.1.1 avem pentru t = 2:

$$\sigma_1 = S_1 = \alpha^{10}$$
$$\sigma_2 = \frac{S_3 + S_1^3}{S_1} = \alpha^{20} = \alpha^5$$

În calculele anterioare am utilizat valorile pentru puterile lui α din tabelul 1 şi puterile care depăşesc α^{15} le-am exprimat in funcţie de aceasta ţinând cont că $\alpha^{15} = 1$.
Folosim relaţia de căutare (3.1.29) şi avem:

$j = 1 \Rightarrow \sigma_1\alpha^{1\cdot1} + \sigma_2\alpha^{2\cdot1} = \alpha^{10}\cdot\alpha + \alpha^5\cdot\alpha^2 = \alpha + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^3 = 1 + \alpha^2 = \alpha^8$
$j = 2 \Rightarrow \sigma_1\alpha^{1\cdot2} + \sigma_2\alpha^{2\cdot2} = \alpha^{10}\cdot\alpha^2 + \alpha^5\cdot\alpha^4 = \alpha^8$
$j = 3 \Rightarrow \sigma_1\alpha^{1\cdot3} + \sigma_2\alpha^{2\cdot3} = \alpha^{10}\cdot\alpha^3 + \alpha^5\cdot\alpha^6 = \alpha^4$
$j = 4 \Rightarrow \sigma_1\alpha^{1\cdot4} + \sigma_2\alpha^{2\cdot4} = \alpha^{10}\cdot\alpha^4 + \alpha^5\cdot\alpha^8 = \alpha^2$
$j = 5 \Rightarrow \sigma_1\alpha^{1\cdot5} + \sigma_2\alpha^{2\cdot5} = \alpha^{10}\cdot\alpha^5 + \alpha^5\cdot\alpha^{10} = 0$
$j = 6 \Rightarrow \sigma_1\alpha^{1\cdot6} + \sigma_2\alpha^{2\cdot6} = \alpha^{10}\cdot\alpha^6 + \alpha^5\cdot\alpha^{12} = \alpha^5$
$j = 7 \Rightarrow \sigma_1\alpha^{1\cdot7} + \sigma_2\alpha^{2\cdot7} = \alpha^{10}\cdot\alpha^7 + \alpha^5\cdot\alpha^{14} = \alpha^{10}$
$j = 8 \Rightarrow \sigma_1\alpha^{1\cdot8} + \sigma_2\alpha^{2\cdot8} = \alpha^{10}\cdot\alpha^8 + \alpha^5\cdot\alpha^{16} = \alpha^2$
$j = 9 \Rightarrow \sigma_1\alpha^{1\cdot9} + \sigma_2\alpha^{2\cdot9} = \alpha^{10}\cdot\alpha^9 + \alpha^5\cdot\alpha^{18} = \alpha^5$
$j = 10 \Rightarrow \sigma_1\alpha^{1\cdot10} + \sigma_2\alpha^{2\cdot10} = \alpha^{10}\cdot\alpha^{10} + \alpha^5\cdot\alpha^{20} = 1$ deci este eronat simbolul n - j = 15 – 10 = 5 (r_5)
$j = 11 \Rightarrow \sigma_1\alpha^{1\cdot11} + \sigma_2\alpha^{2\cdot11} = \alpha^{10}\cdot\alpha^{11} + \alpha^5\cdot\alpha^{22} = \alpha^4$
$j = 12 \Rightarrow \sigma_1\alpha^{1\cdot12} + \sigma_2\alpha^{2\cdot12} = \alpha^{10}\cdot\alpha^{12} + \alpha^5\cdot\alpha^{24} = \alpha$
$j = 13 \Rightarrow \sigma_1\alpha^{1\cdot13} + \sigma_2\alpha^{2\cdot13} = \alpha^{10}\cdot\alpha^{13} + \alpha^5\cdot\alpha^{26} = \alpha^{10}$
$j = 14 \Rightarrow \sigma_1\alpha^{1\cdot14} + \sigma_2\alpha^{2\cdot14} = \alpha^{10}\cdot\alpha^{14} + \alpha^5\cdot\alpha^{28} = \alpha$
$j = 15 \Rightarrow \sigma_1\alpha^{1\cdot15} + \sigma_2\alpha^{2\cdot15} = \alpha^{10}\cdot\alpha^{15} + \alpha^5\cdot\alpha^{30} = 1$ deci este eronat simbolul n - j = 15 – 15 = 0 (r_0)

Cuvântul decodat va fi deci 11010101***1***1001***0*** .
Secvenţa de informaţie de la ieşirea decodorului va fi i = 1101010.

Desfăşurarea lucrării:
a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, 2, sau 3 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.
b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, 2, sau 3 erori alegând şirul de biţi de informaţie de lungimea corespunzătoare. Pentru cazul t = 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.
c) La sfârşitul lucrării de laborator se va efectua cu ajutorul programului un test asupra cunoştinţelor acumulate. Testul cuprinde 5 întrebări teoretice fiecare cu un răspuns corect din cinci propuse, o codare şi o decodare a unui mesaj pe care calculatorul îl generează în mod aleator.

3.2 Codul Reed-Solomon

Codurile Reed-Solomon (RS) fac parte din categoria codurilor ciclice, însă sunt coduri nebinare. Spre deosebire de celelalte coduri ciclice, alfabetul codului RS nu este câmpul binar {0, 1}, ci un câmp finit de ordin superior, numit câmp Galois şi care va fi descris în Anexa. În acest fel, cuvintele codului RS nu sunt secvenţe (succesiuni) de biţi, ci de caractere. Aceste caractere pot fi reprezentate, la rândul lor, prin secvenţe binare, însă sunt indivizibile din punct de vedere al codării şi decodării Reed-Solomon.

Structural, cuvintele de cod RS au aceeaşi alcătuire ca şi cele de cod ciclic:

$$v = v_{n-1}v_{n-2} \dots v_1v_0 \qquad v_j \in GF(2^q, p(x)) \quad j = 0 \div n-1 \tag{3.2.1}$$

unde: v – cuvântul de cod, format din n caractere;
$v_{n-1}v_{n-2} \dots v_k$ – caracterele de informaţie, în număr de k;
$v_{k-1}v_{k-2} \dots v_0$ – caracterele de control, în număr de m;
q – ordinul câmpului;
p(x) – polinomul generator al câmpului GF.
Relaţia de codare are aceeaşi formă ca şi la codurile ciclice:

$$v(x) = i(x)\cdot x^m + \text{rest}\,(i(x)\cdot x^m / g(x)) \tag{3.2.2}$$

unde g(x) este polinomul generator al codului, al cărui construcţie este prezentată mai jos, iar

$$i(x) = v_{n-1}\cdot x^{k-1} + v_{n-2}\cdot x^{k-2} + \dots + v_{k+1}\cdot x + v_k \tag{3.2.3}$$

este polinomul de informaţie.

Prin relaţia de codare (3.2.2) se obţine polinomul ataşat cuvântului de cod, polinom ai cărui coeficienţi sunt tocmai caracterele ce alcătuiesc cuvântul de cod dat de (3.2.1). Relaţia (3.2.2) indică, deasemenea, că v(x) este un multiplu al lui g(x).

Codul RS, având parametrii n, k şi m, construit după relaţia (3.2.2), este capabil să corecteze un număr e_c de caractere eronate, unde:

$$2\cdot e_c = m = n-k \tag{3.2.4}$$

La decodare, spre deosebire de codurile ciclice, într-un cuvânt de cod RS recepţionat, în vederea corecţiei, este necesară atât localizarea erorii, cât şi stabilirea valorii ei.

Polinomul generator, g(x), al codului

Pentru a se corecta t erori dintr-un cuvânt este necesar a se preciza poziţia fiecăreia precum şi valoarea ei. Dacă ne referim la un cuvânt de lungime n, unde:

$$n = 2^q - 1 \tag{3.2.5}$$

atunci informaţia necesară pentru a preciza un caracter eronat între cele n este:

$$i_{p1} = -\log_2(1/n) \tag{3.2.6}$$

Pentru a include şi varianta "cuvânt fără eroare", considerăm că în acest caz se "eronează" un al n+1–lea caracter, fictiv, astfel încât informaţia necesară pentru a preciza poziţia unui caracter eronat (dintre n+1 caractere) este:

$$i_{p1} = \log_2(n+1) \tag{3.2.7}$$

Această informaţie poate fi conţinută de un caracter din $GF(2^q)$. Deasemenea şi valoarea erorii, ε, poate să fie orice caracter din $GF(2^q)$:

$$w = v + \varepsilon \tag{3.2.8}$$

unde: w –caracterul recepţionat $\in GF(2^q)$;

v –caracterul emis $\in GF(2^q)$;

ε -valoarea erorii $\in GF(2^q)\backslash\{0\}$.

Incluzând şi cazul "eronare a caracterului fictiv", rezultă că ε poate lua orice valoare din $GF(2^q)$, adică 2^q valori posibile. Informaţia necesară pentru a preciza valoarea ei este identică cu cea dată de (3.2.7).

În concluzie, pentru fiecare eroare ce se doreşte a fi corectată este necesară o informaţie egală cu 2q biţi, adică două caractere din $GF(2^q)$. La t erori sunt necesare 2t caractere (cantitate de informaţie).

Obs. *În fapt condiţia anterioară este una suficientă, cea necesară implică mai puţină informaţie deoarece nu se poate erona un caracter de două ori. De exemplu în cazul a două erori:*

$$i_{p2} = -\log_2 \frac{1}{C_{n+1}^2} = \log_2 \frac{(n+1)\cdot n}{2} = \log_2(n+1) + \log_2 n - 1 \tag{3.2.9}$$

iar pentru n suficient de mare $i_{p2} \approx 2i_{p1} - 1$.

Cele 2t caractere de informaţie necesare soluţionării problemei corecţiei se află din 2t ecuaţii, care înseamnă tot atâtea legături (proprietăţi) pentru cuvântul receptionat. Aceste 2t proprietăţi pentru cuvintele de cod RS sunt generate prin relaţia de codare (3.2.2). Prin această relaţie v(x) devine multiplul lui g(x), ceea ce înseamnă că rădăcinile lui g vor fi şi rădăcini pentru v. Rezultă necesitatea ca g să aibă 2t rădăcini. Aceste rădăcini pot fi oricare dintre cele 2^q elemente ale câmpului $GF(2^q)$. Vom alege pentru g rădăcinile $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$, datorită simplităţii şi simetriei:

$$g(x) = (x+\alpha)(x+\alpha^2)\ldots(x+\alpha^{2t}) = x^m + g_{m-1}x^{m-1} + \ldots + g_1 x + g_0 \tag{3.2.10}$$

Aşadar cuvântul de cod RS, v, rezultat prin codarea cu ajutorul relaţiei (3.2.2), în care g este dat de (3.2.10), are proprietatea că elementele $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$ sunt rădăcini pentru polinomul ataşat, v(x).

Codarea codului Reed– Solomon

Codarea se poate face utilizând relaţia (3.2.2).
Spre exemplu în cazul codului RS cu n = 7, k = 5, t = 1 având polinomul generator:

$$g(x) = (x + \alpha)(x + \alpha^2) = x^2 + \alpha^4 x + \alpha^3 = x^2 + 5x + 4 \qquad (3.2.11)$$

Ecuaţia împărţirii este:

$$i(x)\cdot x^2 = (2x^4 + 5x^3 + x^2 + x + 5)(x^2 + 5x + 4) + x + 1 \qquad (3.2.12)$$

unde: $-i(x)\cdot x^2$ este deîmpărţitul ($i = [2\ 1\ 7\ 5\ 4] \Rightarrow i(x) = 2x^4 + x^3 + 7x^2 + 5x + 4$);
$-2x^4 + 5x^3 + x^2 + x + 5$ este câtul;
$-g(x) = x^2 + 5x + 4$ este împărţitorul, iar
$- x + 1$ este restul.

Împărţirea polinomului $i(x)\cdot x^m = i(x)\cdot x^2$ la g(x), cerută pentru codare de relaţia (3.2.2), este prezentată mai jos:

$$\begin{array}{l|l} 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 & x^2 + 5x + 4 \\ \underline{2x^6 + 6x^5 + 5x^4} & \hline 2x^4 + 5x^3 + x^2 + x + 5 \\ / \quad 5x^5 + 4x^4 + 5x^3 + 4x^2 & \\ \quad \underline{5x^5 + 2x^4 + x^3} & \\ \quad / \quad 2x^4 + 6x^3 + 4x^2 & \\ \quad\quad \underline{x^4 + 5x^3 + 4x^2} & \\ \quad\quad / \quad x^3 & \\ \quad\quad\quad \underline{x^3 + 5x^2 + 4x} & \\ \quad\quad\quad / \quad 5x^2 + 4x & \\ \quad\quad\quad\quad \underline{5x^2 + 2x + 1} & \\ \quad\quad\quad\quad / \quad\quad x + 1 & \end{array}$$

Cuvântul de cod, conform relaţiei (3.2.2) rezultă:

$$v(x) = i(x)\cdot x^2 + x + 1 = 2x^6 + x^5 + 7x^4 + 5x^3 + 4x^2 + x + 1 \qquad (3.2.13)$$

Decodarea codului Reed-Solomon

Decodarea codului RS poate fi făcută atât în timp cât şi în frecvenţă

Decodarea codului RS-corector de erori multiple

Decodarea va fi sistematizată sub forma unor paşi algoritmici. La fiecare pas se va prezenta operaţia necesară a fi executată, scopul urmărit cât şi argumentările necesare. Fie aşadar codul RS corector de t erori.

Pasul I

Conform relaţiei (3.2.10) g(x) este un polinom de grad $k = 2t > 2$. Prin construcţie, cuvintele de cod RS au proprietatea că polinoamele ataşate lor sunt divizibile cu g(x), adică elementele câmpului $GF(2^2)$ $\alpha, \alpha^2, \ldots, \alpha^{2t}$ sunt rădăcini atât pentru g(x) cât şi pentru orice cuvânt de cod v(x):

$$\begin{cases} g(\alpha^j) = 0 \\ v(\alpha^j) = 0 \qquad j = 1 \div 2t \end{cases} \qquad (3.2.14)$$

Aceste proprietăţi constituie şi punctul de plecare în decodare. Presupunând că w este un cuvânt recepţionat:

$$w = v + \varepsilon \quad \text{sau} \quad w(x) = v(x) + \varepsilon(x) \tag{3.2.15}$$

vom calcula 2t coeficienţi, numiţi coeficienţi sindrom, Sj, în forma:

$$S_j = w(\alpha^j) = v(\alpha^j) + \varepsilon(\alpha^j) = \varepsilon(\alpha^j) \qquad j=1\div 2t \tag{3.2.16}$$

Evident că dacă nu există erori $S_j = 0$. În acest caz se trece la pasul VI. Evident concluzia poate fi eronată. Un exemplu în argumentarea acestei afirmaţii este situaţia: ε = cuvânt de cod. Dar în acest caz numărul erorilor depăşeşte puterea de corecţie de t erori.

Pasul II

Dacă există erori în limitele corectabile (numărul erorilor este mai mic sau egal cu t) atunci există coeficienţi sindrom diferiţi de zero. Fie cuvântul eroare în forma:

$$\varepsilon(x) = \sum_{i=1}^{t} {}^{r_i} \cdot x^{k_i} \tag{3.2.17}$$

unde:

$$Y_i = \alpha^{r_i} \tag{3.2.18}$$

reprezintă valoarea erorii $r_i \in \{0, 1, 2, \ldots n-1\}$, iar:

$$X_i = \alpha^{k_i} \tag{3.2.19}$$

reprezintă locatorul erorii $k_i \in \{0, 1, 2, \ldots, n-1\}$. Cu aceste notaţii coeficienţii sindrom au expresiile:

$$S_j = \sum_{i=1}^{t} Y_i \cdot \left(\alpha^j\right)^{k_i} = \sum_{i=1}^{t} Y_i \cdot X_i^j, \qquad j = 1\div 2t \tag{3.2.20}$$

Ecuaţiile (3.2.20) reprezintă un sistem de 2t ecuaţii cu 2t necunoscute: t locatori ai erorilor X_i şi t valori pentru respectivele erori Y_i.

Rezolvarea acestui sistem de ecuaţii se va face în mai multe etape. La pasul prezent se vor calcula coeficienţii polinomului σ(x) ai cărui rădăcini sunt locatorii erorilor:

$$\sigma(x) = \sum_{i=1}^{t} (x + X_i) = x^t + \sigma_{\ 1} \cdot x^{t-1} + \ldots + \sigma_{t-1} \cdot x + \sigma_t \tag{3.2.21}$$

Pentru că Xi , $1 \le i \le t$ este o rădăcină a lui σ(x) putem scrie:

$$X_i^t + \sigma_1 \cdot X_i^{t-1} + \ldots + \sigma_{t-1} \cdot X_i + \sigma_t = 0, \quad i = 1\div t \tag{3.2.22}$$

Înmulţind ecuaţiile (3.2.22) pe rând cu $X_i^k Y_i$ şi sumându-le obţinem ecuaţia:

$$\sum_{i=1}^{t} Y_i \cdot X_i^{t+k} + \sigma_1 \cdot \sum_{i=1}^{t} Y_i X_i^{t+k-1} + \ldots + \sigma_{t-1} \cdot \sum_{i=1}^{t} Y_i \cdot X_i^{k+1} + \\ + \sigma_t \cdot \sum_{i=1}^{t} Y_i \cdot X_i^k = 0 \tag{3.2.23}$$

sau, ţinând cont de (3.2.20) pentru k luând valorile 1, 2, ..., t:

$$S_{t+k} + \sigma_1 \cdot S_{t+k-1} + \ldots + \sigma_{t-1} \cdot S_{k-1} + \sigma_t \cdot S_k = 0 , \quad k = 1 \div t \tag{3.2.24}$$

Ecuaţiile (3.2.24) reprezintă un sistem de t ecuaţii cu t necunoscute (coeficienţii σ_i) a cărui rezolvare constituie obiectivul acestui pas algoritmic. Ecuaţiile (3.2.24) pot fi puse sub forma compactă:

$$A_s \cdot \sigma = B_s \tag{3.2.25}$$

unde:

$$A_s = \begin{bmatrix} S_t & S_{t-1} & \ldots & S_1 \\ S_{t+1} & S_t & \ldots & S_2 \\ \ldots & \ldots & \ldots & \ldots \\ S_{2t-1} & S_{2t-2} & \ldots & S_t \end{bmatrix}; \quad \sigma = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \ldots \\ \sigma_t \end{bmatrix}; \quad B_s = \begin{bmatrix} S_{t+1} \\ S_{t+2} \\ \ldots \\ S_{2t} \end{bmatrix} \tag{3.2.26}$$

Calculând inversa matricii A_s găsim soluţia sistemului (3.2.24) în forma:

$$\sigma = A_s^{-1} \cdot B_s \tag{3.2.27}$$

Obs: *Toate calculele trebuiesc făcute în câmpul GF(2²), atât coeficienţii sindrom, S_j, cât şi coeficienţii σ fiind elemente ale respectivului câmp.*

În rezolvarea ecuaţiei (3.2.25) pot apărea trei situaţii:

1° rangul matricii As este e < t şi este egal cu al matricii [AsBs]. În acest caz numărul de erori este e şi din rezolvarea ec (3.2.25) rezultă un număr e de coeficienţi σ_l nenuli Rezolvarea ecuaţiei (3.2.25) presupune restrângerea sistemului (10.24) la un număr e < t de ecuaţii cu e necunoscute, rezolvabil.

2° rangul matricii As este t. În acest caz există As^{-1} iar ecuaţia (3.2.25) are soluţie dată prin relaţia (3.2.27).Se vor găsi t erori în acest caz.

3° rangul matricii As este e` <t şi este mai mic decât al matricii [AsBs]. O astfel de situaţie este posibil să apară dacă numărul erorilor depăşeşte t. În acest caz se semnalează prezenţa erorilor în număr necorectabil. Funcţie de aplicaţie se va abandona cuvântul în cauză sau se va cere retransmisia sa.

Pasul III

Ecuaţia (3.2.22) se poate rescrie în forma:

$$\sum_{j=1}^{t} \sigma_j \cdot X_i^{-j} = 1 \tag{3.2.28}$$

Ştiind că X_i este de forma $X_i = \alpha^{k_i}$ unde k_i indică rangul pe care îl ocupă eroarea (ex: k_i = n-1 este prima poziţie) se vor putea afla locatorii erorilor printr-o operaţie de căutare:

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot j} = 1 \quad ? \qquad k = 1, 2, \ldots, n \tag{3.2.29}$$

Acei k pentru care (3.2.29) este o identitate, indică prezenţa erorii pe poziţia:

$$r = n - k \tag{3.2.30}$$

Obs: *Înlocuind pe:*

$$X_r = \alpha^r \qquad (3.2.31)$$

în (3.2.28) şi utilizând identitatea $\alpha^n = 1$ *obţinem:*

$$\sum_{j=1}^{t} \sigma_j \cdot \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{n \cdot j} \alpha^{-j \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{(n-j) \cdot r} = \sum_{j=1}^{t} \sigma_j \cdot \alpha^{k \cdot jr} = 1$$

Pasul IV

Dispunând de poziţiile erorilor dispunem implicit de numărul lor. Reţinem că problema are soluţie doar dacă e < t. Cunoscând aşadar e locatori ai erorilor în forma $X_i = \alpha^{k_i}$, i = 1÷e, din sistemul de ecuaţii (3.2.20) se reţin e ecuaţii în vederea aflării valorilor Y_i pentru cele e erori. Acest sistem este compatibil unic determinat. Rezolvarea sa conduce la aflarea celor e valori necesare Y_i.

Pasul V

Cunoscând atât poziţiile erorilor $X_i = \alpha^i$, i = 1÷e, cât şi valorile lor $Y_i = \alpha^{r_i}$, i=1÷e putem face corecţia caracterelor eronate:

$$v_{k_i} = w_{k_i} + Y_i, \qquad i = 1 \div e \qquad (3.2.32)$$

Pasul VI

Se face selecţia caracterelor de informaţie şi livrarea lor la ieşire.

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare. Se urmăreşte pas cu pas algoritmul de codare existând posibilitatea de a alege un cod corector de t = 1, sau 2 erori. Urmăriţi cu atenţie la fiecare pas mesajele calculatorului care va vor ghida în rularea corectă a programului. De asemenea se urmăreşte pas cu pas algoritmul de decodare.

b) Realizaţi o codare a unui cuvânt în cazul unui cod corector de t = 1, sau 2 erori alegând şirul de caractere de informaţie de lungimea corespunzătoare. Pentru cazul t = 1 şi 2 efectuaţi apoi decodarea unui cuvântului de cod. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

L4. Coduri convoluţionale

Codurile convoluţionale au fost introduse în 1954 de P. Elias şi reprezintă o clasă de coduri corectoare având o mare aplicabilitate practică.

În cazul codurilor convoluţionale, fiecare bloc de *n* simboluri binare de la ieşirea codorului depinde atât de blocul de *k* simboluri binare prezent la intrarea sa, la momentul considerat, cât şi de *m* blocuri precedente. În consecinţă codurile convoluţionale introduc un efect de memorie de ordinul *m*. Cantitatea $K=m+1$ se numeşte lungimea de constrângere a codului.

Codorul este constituit dintr-un sistem de *m* registre de întârziere, fiecare având o capacitate de *k* biţi, care memorează cele *m* blocuri de *k* simboluri de informaţie, dintr-o mulţime de funcţii liniare, ce generează blocurile de *n* simboluri de la ieşire şi dintr-un convertor paralel-serie. Mărimea $R=k/n$ se numeşte randamentul codului.

Un cod convoluţional de randament *R* este o aplicaţie de la mulţimea matricilor (binare) cu un număr de *k* linii şi număr infinit de coloane către mulţimea matricilor (binare) cu un număr de *n* linii şi număr infinit de coloane, unde $n > k$:

$$\mathbf{C} : \mathbf{M}_{k\times\infty} \rightarrow \mathbf{M}_{n\times\infty} \tag{4.1}$$

Astfel, prin transformarea $\mathbf{C}$, fiecărei matrici $\mathbf{I} \in \mathbf{M}_{k\times\infty}$, de forma :

$$\mathbf{I} = \begin{bmatrix} i_{01} & i_{11} & i_{21} & \cdots & i_{j1} & \cdots \\ i_{02} & i_{12} & i_{22} & \cdots & i_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ i_{0k} & i_{1k} & i_{2k} & \cdots & i_{jk} & \cdots \end{bmatrix}, \qquad i_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{k}} \tag{4.2}$$

i se ataşează o matrice $\mathbf{V} \in \mathbf{M}_{n\times\infty}$, de forma:

$$\mathbf{V} = \begin{bmatrix} a_{01} & a_{11} & a_{21} & \cdots & a_{j1} & \cdots \\ a_{02} & a_{12} & a_{22} & \cdots & a_{j2} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0k} & a_{1k} & a_{2k} & \cdots & a_{jk} & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{0n} & a_{1n} & a_{2n} & \cdots & a_{jn} & \cdots \end{bmatrix}, \quad a_{js} \in \{0,1\} \;\; \forall\, \mathrm{j} = \overline{0,\infty}, \;\; \mathrm{s} = \overline{1,\mathrm{n}} \tag{4.3}$$

Matricea **I** conţine practic biţii de informaţie, în ordinea $i_{01}\, i_{02} \ldots \mathrm{i}_{0k}\, \mathrm{i}_{11} \ldots$, iar matricea **V** secvenţa codată : $a_{01}\, a_{02} \ldots \mathrm{a}_{0n}\, \mathrm{a}_{11} \ldots$. Dacă $a_{js} \equiv i_{js}$ pentru orice *j* pozitiv şi pentru orice $s = 1 \div k$, atunci codul se numeşte sistematic. Făcând apel la o reprezentare polinomială, matricile **I** şi **V** se pot scrie ca :

$$\mathbf{I}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{i}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{i}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{i}_{sk} \mathrm{D}^s \end{bmatrix}, \quad \mathbf{V}(\mathrm{D}) = \begin{bmatrix} \sum_{s=0}^{\infty} \mathrm{a}_{s1} \mathrm{D}^s \\ \sum_{s=0}^{\infty} \mathrm{a}_{s2} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sk} \mathrm{D}^s \\ \vdots \\ \sum_{s=0}^{\infty} \mathrm{a}_{sn} \mathrm{D}^s \end{bmatrix} \tag{4.4}$$

Cu aceste notaţii, relaţia de codare, poate fi scrisă astfel :

$$\mathbf{V}(\mathrm{D}) = \mathbf{G}(\mathrm{D}) \cdot \mathbf{I}(\mathrm{D}) \tag{4.5}$$

unde **G**(D) se numeşte matricea generatoare a codului şi are forma:

$$\mathbf{G}(\mathrm{D}) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1k}(D) \\ \cdots & \cdots & \cdots & \cdots \\ g_{n1}(D) & g_{n1}(D) & \cdots & g_{nk}(D) \end{bmatrix} \tag{4.6}$$

În funcţie de felul polinoamelor gebneratoare $g_{js}(D)$, codurile convoluţionale pot fi clasificate ca şi:

a) Coduri *sistematice* sau *nesistematice*. Dacă primele k linii din **G**(D) formează matricea unitate de ordinul k, I_k, atunci codurile sunt sistematice. În acest caz biţii din **I** se vor regăsi printre biţii din **V**. Astfel, dacă cele k simboluri de informaţie, prezente sunt efectiv emise, adică se găsesc în mod explicit în blocul de n simboluri de la ieşirea codorului pe primele k poziţii, codul se numeşte cod sistematic. Pentru codurile nesistematice, biţii din **V** sunt combinaţii liniare ale biţilor din **I**, neexistând biţi de informaţie şi de control ca şi în cazul precedent.

b) Coduri *recursive* sau *nerecursive*. Dacă toate polinoamele generatoare care compun **G**(D) sunt finite, atunci codul rezultat este nerecursiv. În caz contrar polinoamele generatoare $g_{js}(D)$ pot fi scrise sub forma:

$$g_{js}(D) = \frac{a_{js}(D)}{b_{js}(D)} \tag{4.7}$$

unde polinoamele a_{js} şi b_{js} sunt finite. Deci, dacă există cel puţin un polinom $b_{js}(D) \neq 1$, atunci codul este recursiv.

Lungimea de constrângere este unul dintre parametri importanţi ai codurilor convoluţionale. O altă definiţie a sa este dată în relaţia de mai jos:

$$K = 1 + \max_{j,s} grad\{a_{j,s}(D), b_{j,s}(D)\} \tag{4.8}$$

Pentru a ilustra codurile convoluţionale nerecursive şi nesistematice, în Fig. 4.1. se prezintă un exemplu de codor convoluţional de randament R=1/2 şi de lungime de constrângere K=(m+1)=3. Intrarea sa este constituită din blocurile de k=1 simbol şi ieşirea sa de blocurile de n=2 simboluri.

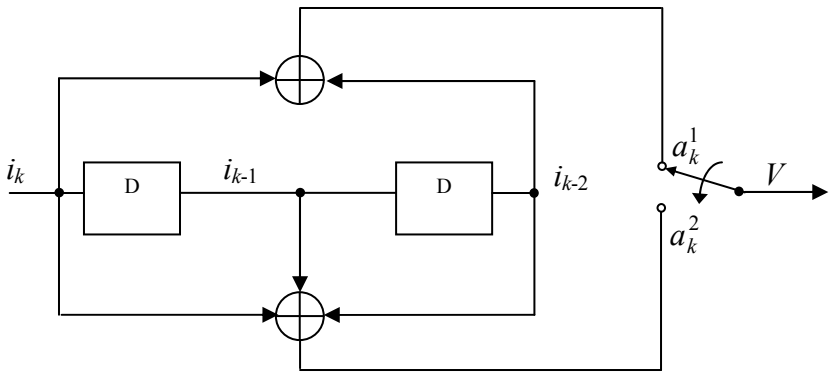


Fig. 4.1. Exemplu de codor convoluţional nerecursiv, nesistematic (R = 1/2, K = 3).

Relaţiile de calcul ale secvenţelor de la ieşire sunt:

$$a_k^1 = \sum_{j=0}^{2} i_{k-j} g_j^1 \ , \ a_k^2 = \sum_{j=0}^{2} i_{k-j} g_j^2 \tag{4.9}$$

Cele două secvenţe generatoare sunt:

$$\begin{aligned} g^1 &= \left[g_0^1, g_1^1, g_2^1\right] = [1,0,1] \\ g^2 &= \left[g_0^2, g_1^2, g_2^2\right] = [1,1,1] \end{aligned} \tag{4.10}$$

Observăm că ieşirile codorului fiind egale cu o combinaţie liniară a simbolurilor de informaţie, codul este liniar. Codurile convoluţionale sunt de asemenea definite pornind de la polinoamele lor generatoare exprimate în funcţie de variabila D (întârziere) echivalentă cu variabila Z^{-1} a transformatei Z. Considerând tot exemplul din Fig. 4.1, polinoamele generatoare ale acestui cod au expresiile:

$$\begin{aligned} g^1 &\rightarrow G^1(D) = g_0^1 + g_1^1 D + g_2^1 D^2 \\ g^2 &\rightarrow G^2(D) = g_0^2 + g_1^2 D + g_2^2 D^2 \end{aligned} \tag{2.11}$$

şi:

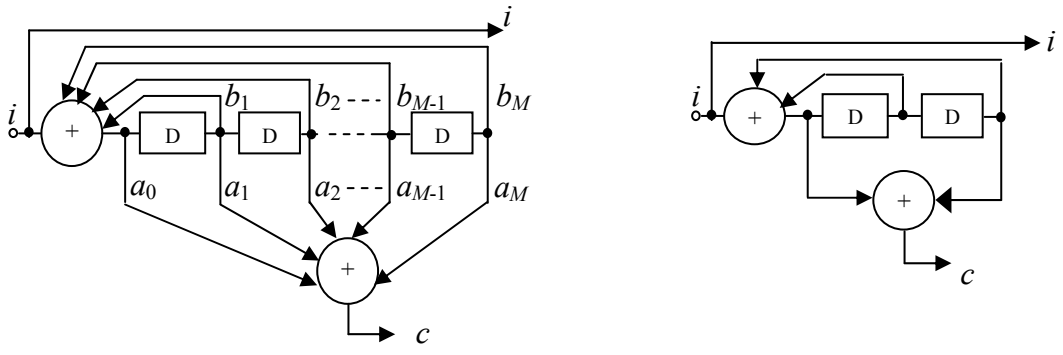
$$\begin{aligned} G^1(D) &= 1 + D^2 \\ G^2(D) &= 1 + D + D^2 \end{aligned} \tag{2.12}$$

rezultând matricea generatoare $G=[1+D^2,\ 1+D+D^2]$.

În general polinoamele generatoare ale codorului se exprimă în octal şi astfel, pentru cazul din Fig. 4.1, avem:

$$\begin{aligned} G^1 &= [1,0,1] = 5(\text{în octal}) \\ G^2 &= [1,1,1] = 7(\text{în octal}) \end{aligned} \tag{2.13}$$

În Fig.4.2 a) se prezintă schema generală a unui codor recursiv sistematic, RSC (*Recursive Systematic Code*), iar în figura Fig. 4.2 b) un caz particular al acesteia.



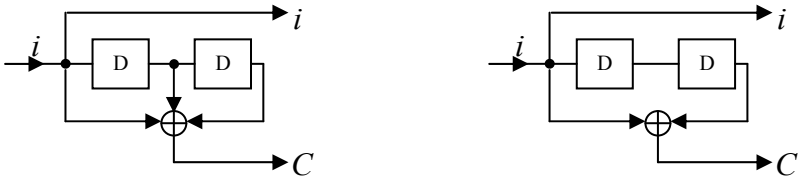
a) b)

Fig. 4.2. Cod convoluţional recursiv sistematic: a) schema generală, b) exemplu (R=1/2, K=3).

În exemplul considerat matricea generatoare este de forma:

$$G = \left[1, \frac{1+D^2}{1+D+D^2}\right] \tag{2.14}$$

În figura următoare sunt prezentate două exemple de codoare sistematice şi nerecursive, NRSC (*Non-Recursive Systematic Code*), considerându-se randamentul R=1/2 şi lungimea de constrângere K=3:



a) b)

Fig. 4.3 Coduri convoluţionale sistematic şi nerecursiv, R=1/2, K=3, G=1+D+D^2.

Diagrame de stări

Diagrama de stări este o reprezentare a funcţionării unui codor convoluţional, în care timpul nu apare în mod explicit.

În Fig.4.4 s-a reprezentat diagrama de stări asociată codorului convoluţional din Fig. 4.1. Diagrama de stări permite evaluarea funcţiei de transfer a codorului, care va fi utilizată pentru calculul performanţelor codului.

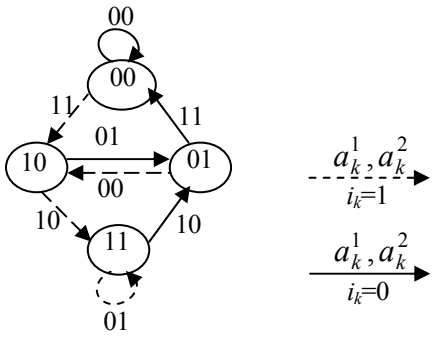


Fig. 4.4: Diagrama de stări a codorului din Fig. 4.1.

Diagramele de stări corespunzătoare codurilor convoluţionale din Fig.4.2 b) şi Fig.4.3 a) şi b) sunt prezentate în Fig.4.5.

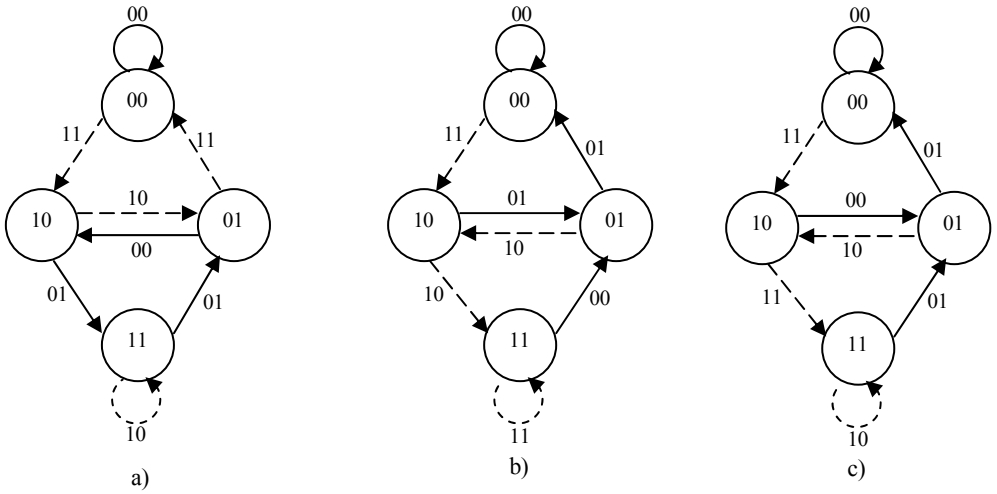


Fig. 4.5 Diagramele de stare pentru codoarele convoluţionale: a) RSC; b) şi c) NRSC.

Diagrama trellis

Fiecare bloc de $n = 2$ simboluri de la ieşirea acestui codor depinde de blocul de $k = 1$ simbol prezent la intrarea sa dar şi de $m = 2$ blocuri de k simboluri conţinute în memoria sa. Aceste $mk = 2$ simboluri definesc starea codorului. Se notează cu $S_0 = (00)$, $S_1 = (01)$, $S_2 = (10)$ şi $S_3 = (11)$, cele patru stări posibile ale codorului din Fig. 4.1. Oricare ar fi starea iniţială a codorului, după $m+1=3$ întârzieri la intrarea codorului, toate stările au fost atinse.

Funcţionarea codorului poate fi explicată ţinând seama doar de stările sale şi de tranziţiile dintre acestea, numite ramuri (ramificaţii, braţe). Diagrama *trellis* astfel obţinută este reprezentată în Fig. 4.6, pentru codorul convoluţional din Fig. 4.1, presupunând ipoteza că starea sa iniţială era $S_0 = (00)$.

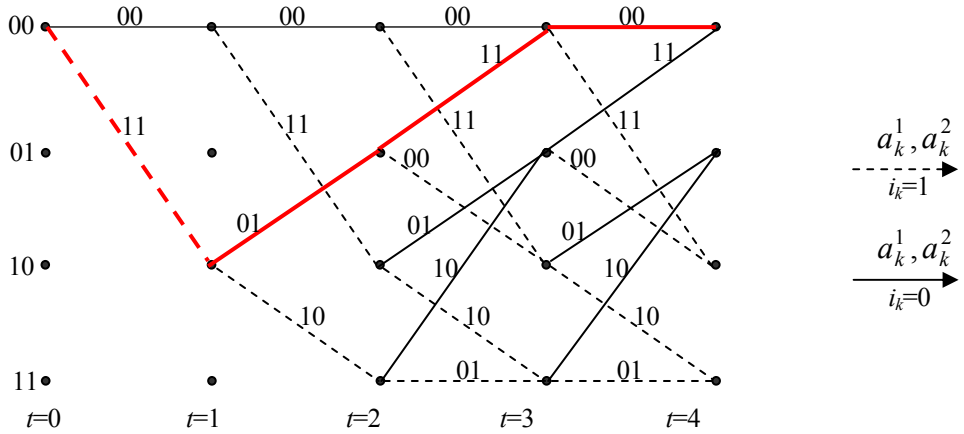


Fig. 4.6. *Trellis*-ul codorului convoluţional din Fig. 4.1.

Ramurile reprezentate prin linii punctate corespund prezenţei unui simbol de informaţie egal cu 1, la intrarea codorului, şi ramurile reprezentate prin linii pline, unui simbol de informaţie egal cu 0. Fiecărei ramuri i s-a asociat valoarea cuplului binar disponibil la ieşirea codorului.

După $m+1$ întârzieri, oricare ar fi starea iniţială a codorului, *trellis*-ul se repetă. Din fiecare nod pleacă 2^k ramuri (în cazul de faţă sunt două ramuri) şi în fiecare nod converg 2^k ramuri.

Pornind de la starea $S_0=(00)$ în momentul $t=0$, de exemplu, vedem că există patru căi care permit atingerea stării $S_0=(00)$ în momentul $t=4$.

00 00 00 00 → calea 1
00 11 01 11 → calea 2
11 10 10 11 → calea 3
11 01 11 00 → calea 4

Codarea codului convoluţional

Folosind o codare convoluţională, se codează următoarea secvenţă de informaţie: i=100111, considerând polinomul generator al codului: $g(D)=1+D^2$.
În acest scop se foloseşte Fig. 4.7.

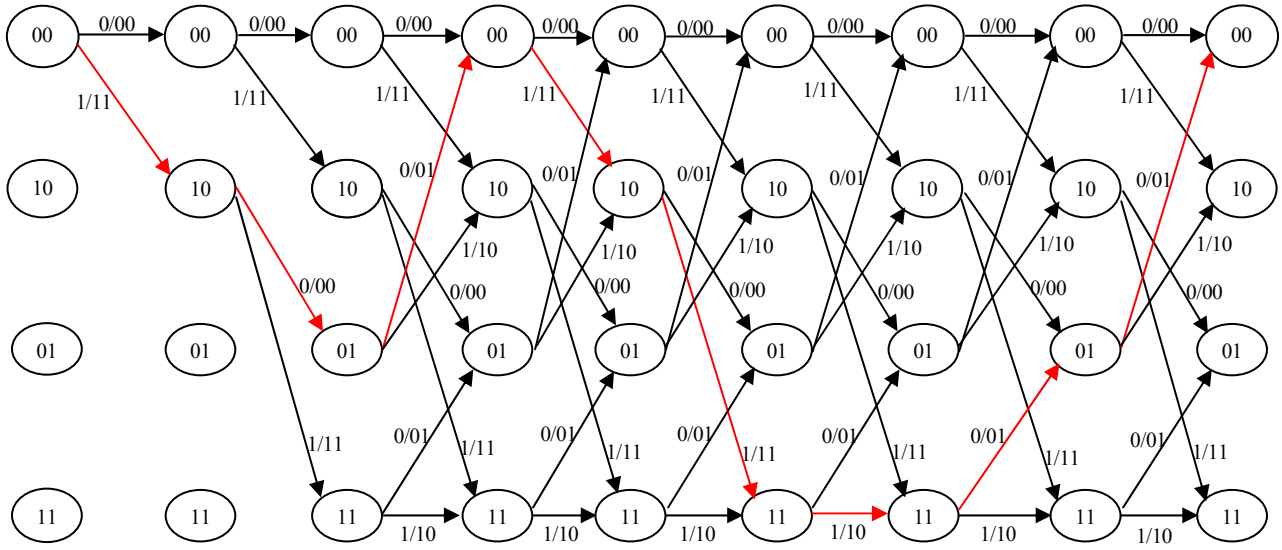


Fig. 4.7. Utilizarea diagramei trellis în cazul codării secvenţei de informaţie i=100111, considerând diagrama de stări din Fig. 4.5. c).

Cuvântul de cod obţinut este v=11 00 01 11 11 10 01 01. Trellis-ul codorului convoluţional se închide la zero, astfel, datorită acestui fapt au apărut două grupe suplimentare de biţi.

Calea pe trellis care ne generează cuvântul de cod, v, este cea marcată cu roşu.

Decodarea codului convoluţional, algoritmul Viterbi cu decizie hard

Pentru prezentarea acestui algoritm se consideră un canal binar simetric (fără memorie), intrarea decodorului fiind alcătuită dintr-o secvenţă de simboluri binare.

În fiecare moment două ramuri, aparţinând la două căi diferite, converg spre fiecare nod al *trellis*-lui (Fig.4.6). Din aceste două căi una este mai probabilă, altfel spus, se găseşte la cea mai mică distanţă Hamming faţă de secvenţa recepţionată, decât cealaltă cale. Distanţa fiind o funcţională aditivă, în fiecare nod se păstrează calea cea mai probabilă numită cale supravieţuitoare. Dacă se obţin două căi cu aceeaşi distanţă Hamming, doar o singură cale este pastrată, alegându-se în mod arbitrar una din cele două căi posibile.

În figura următoare se prezintă un exemplu de decodare pentru codorul reprezentat în Fig. 4.3. b), cu diagrama de stări reprezentată în Fig. 4.5. c). Calea rezultantă este marcată cu culoare roşie.

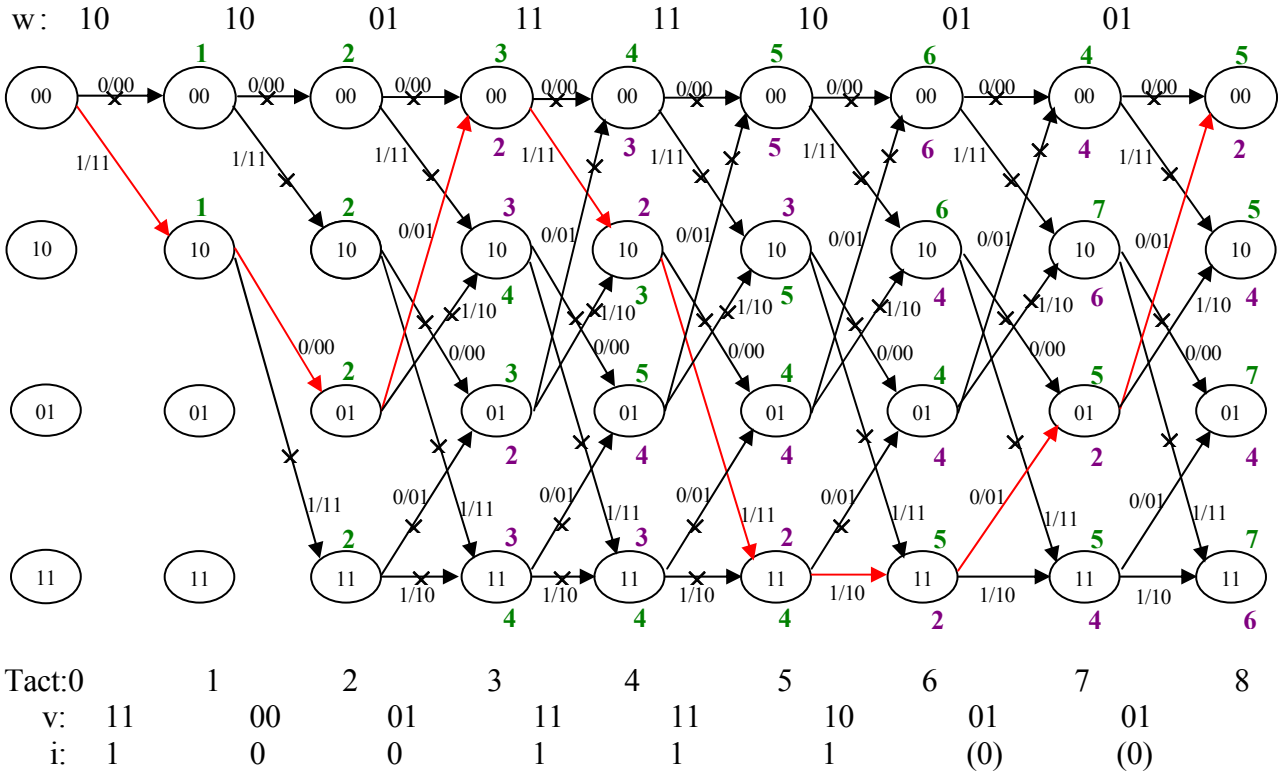


Fig. 4.8. Utilizarea diagramei trellis în cazul decodării secvenţei w=A7E5$_H$, considerând diagrama de stări din Fig. 4.5. c).

Desfăşurarea lucrării:

a) Rulaţi programul pe calculator, utilizând opţiunea demonstrativă atât pentru algoritmul de codare cât şi pentru cel de decodare.

b) Realizaţi o codare şi o decodare, folosind algoritmul de decodare Viterbi. Verificaţi apoi calculele cu ajutorul programului de pe calculator.

Bibliografie

[1] P. Elias, *"Error-free Coding"*, IRE Trans. Inform. Theory, vol IT-4, pp.29-37, 1954,

[2] Chien, R, "*Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes",* IEEE Transactions on Information Theory, Volume 10, Issue 4, Page(s): 357 - 363, Oct 1964,

[3] A. Glavieux, M. Joindot, *"Communications numériques. Introduction"*, Masson, Paris, 1996,

[4] M. E. Borda, *"Teoria transmiterii informaţiei"*, Editura Dacia, Cluj-Napoca, 1999,

[5] G. Wade, *"Coding Techniques-An Introduction to Compresion and Error Control"*, Creative Print and Design, Ebbw Vale, Geat Britain, 2000,

[6] J. Proakis, *"Digital Communications"*, 4th Ed., McGraw Hill, New York, 2000,

[7] H. Baltă, M. Kovaci, *"A Comparasion Between Weight Spectrum of Different Convolutional Code Types"*, conferinţă Oradea, 2004.