CZESŁAW KOŚCIELNY
Wrocławi Műszaki Egyetem Műszaki Kibernetikai Intézet

Hibacsomó-javítás hardware megvalósítása ciklikus Reed-Solomon-kódok segítségével

ETO 681.3.041.5

Modern digitális hírközlő rendszerekben egyetlen elemi jel rendszerint egy bitnél több információt hordoz. Ezért a jelenlegi gyakorlatban elterjedt bináris hibajavító kódok, amelyekben a kódszó elemei bináris szimbólumok, nem túlzottan hatékonyak. Úgy tűnik, hogy többszintű ciklikus kódok előnyösen alkalmazhatók nagy sebességű bonyolult adatátviteli összeköttetésekben (4800, 9600 bit/s és e fölött), mágnesszalagos tároló rendszerekben stb. Ezekben a rendszerekben a hibacsomók dominálnak, és olyan kódok szükségesek, amelyek hibacsomókat képesek javítani. Szerencsére a hibacsomó-javító ciklikus kódok meglehetősen egyszerűen megvalósíthatók. Különösen a ciklikus Reed–Solomon-kódoknak van jó hibacsomó-javító képessége. Minden RS (Reed–Solomon)-kód egyúttal maximális, azaz az adott redundancia mellett a minimális Hamming-távolsága maximális. Ez a kód optimális a Reiger-korlát értelmében, ezért igen alkalmas hibacsomók javítására.

1. Ciklikus RS-kódok

Matematikai szempontból az RS-kód, mint minden ciklikus kód, ideál a $GF(q)$ test feletti, (x^n-1) niodulusú polinom-algebrában. A kód generátor-polinómja:

$$g(x)=\prod_{i=1}^{d-1}(x-\alpha^i)=x^r+\sum_{i=1}^{r}g_{r-i}x^{r-i}, \qquad (1.1)$$

ahol

d — azon kód minimális Hamming-távolsága, amelyet az (1.1) polinom generál,
α — a $GF(q)$ primitív eleme,
$GF(q)$ — a kompozit véges Galois-test

és

$$g_i\in GF(q),\ g_i\neq 0,\ i=0, 1, \ldots, r-1.$$

A kompozit megjelölés azt jelenti, hogy a testnek $q=p^n$ eleme van (ahol p prímszám), és $p^n>2$. Az (1.1) polinom által generált kód ciklikus (n, k) kód, amelyre:

a kód hossza: $n=q-1$,
az információs elemek száma: $k=n-r$,
a redundáns elemek száma: $r=n-k=d-1$,
a minimális Hamming-távolság: $d=r+1$.

Minden V kódszó tekinthető egy $GF(q)$ feletti n-dimenziós vektornak:

$$\mathbf{V}=[v_0, v_1, \ldots, v_{n-1}]. \qquad (1.2)$$

Az (1.2) vektorhoz kölcsönösen egyértelműen hozzárendelhető egy legfeljebb n–1-ed fokú, $GF(q)$ feletti polinom:

$$v(x)=\sum_{i=1}^{n}v_{n-i}x^{n-i}. \qquad (1.3)$$

Gyakorlati okokból szisztematikus kódra van szükség, azaz a $v_0, v_1, \ldots, v_{r-1}$ pozíciók a redundáns elemek, míg $v_r, v_{r+1}, \ldots, v_{n-1}$ az információs elemek.

2. A kódolási eljárás

Jelölje

$$f_i(x)=\sum_{i=1}^{k}f_{n-1}\cdot x^{n-i} \qquad (2.1)$$

az információs polinómot, amellyel ekvivalens vektor:

$$\mathbf{F}_i=[f_r, f_{r+1}, \ldots, f_{n-1}]. \qquad (2.2)$$

A (2.2) vektor komponensei az információs szimbólumok, amelyeket kódolni kell. A kódolási eljárás a kódvektor redundáns részének

$$\mathbf{F}_r=[f_0, f_1, \ldots, f_{r-1}]\text{-nek a} \qquad (2.3)$$

kiszámítását jelenti. Ez a következőképpen végezhető el:

$$f_r(x)=-R_g[f_i(x)]=\sum_{i=1}^{r}f_{r-i}\cdot x^{r-i}, \qquad (2.4)$$

ahol $R_g[y]$ y-nak a $g(x)$ polinommal való osztás során kapott maradéka. A teljes polinom:

$$f(x)=f_i(x)+f_r(x)=\sum_{i=1}^{n}f_{n-i}\cdot x^{n-i} \qquad (2.5)$$

természetesen kódpolinom, hiszen többszöröse a $g(x)$ generátorpolinomnak. Így a kódvektor a következő:

$$\mathbf{F}=[f_0, f_1, \ldots, f_{n-1}]. \qquad (2.6)$$

Egy másik kódolási eljárás az

$$\mathbf{R}=(-1)\begin{bmatrix}\mathbf{R}_{n-1}\\ \mathbf{R}_{n-2}\\ \vdots\\ \mathbf{R}_r\end{bmatrix}, \qquad (2.7)$$

$k\cdot r$ dimenziós mátrixot használja fel, ahol a mátrix sorvektorai:

$$\mathbf{R}_{n-j}=[r_{n-j,0}, r_{n-j,1}, \ldots, r_{n-j,r-1}] \qquad (2.8)$$

az

$$r_{n-j}(x)=R_g[x^{n-j}]=\sum_{i=1}^{r}r_{n-j,r-i}\cdot x^{r-i} \qquad (2.9)$$

Beérkezett: 1978. XII. 8.

összefüggésből számíthatók. Most a redundáns vektor az $\mathbf{F}_i$ k-dimenziós sorvektor és az $\mathbf{R}$ mátrix szorzatával egyenlő:

$$\mathbf{F}_r=\mathbf{F}_i\cdot\mathbf{R}. \tag{2.10}$$

Az $\mathbf{R}$ mátrix transzponáltja $r\cdot k$ mátrix:

$$\mathbf{R}^T=(-1)\begin{bmatrix}\mathbf{H}_1\\ \mathbf{H}_2\\ \vdots\\ \mathbf{H}_r\end{bmatrix}, \tag{2.11}$$

ahol

$$\mathbf{H}_i=[h_{i,0},\ h_{i,1},\ \ldots,\ h_{i,k-1}] \tag{2.12}$$

és

$$h_i(x)=R_h x^{k-1+i}. \tag{2.13}$$

Az előbbiekhez hasonlóan $R_h[y]$ y-nak a $h(x)$-szel való osztás utáni maradékát jelöli. $h(x)$-et paritáspolinomnak nevezik:

$$h(x)=(x^n-1)/g(x).$$

3. A dekódolási eljárás

A következő eljárás $\left[\frac{d-1}{2}\right]$ számú ($[x]$=entier függvény) véletlen hiba javítására szolgál feltéve, hogy a hibák a kódvektor $r=n-k$ terjedelmű részében helyezkednek el. A kódvektor 0-adik és $(n-1)$-edik pozíciója a kód ciklikus természete miatt szomszédos.

Jelölje $\mathbf{R}$ a vett kódvektort és $r(x)$ az ennek megfelelő polinomot. A vett vektor (polinom) a kódvektor (polinom) és a hibavektor (polinom) összege:

$$\mathbf{R}=\mathbf{V}+\mathrm{E}=[r_0,\ r_1,\ \ldots,\ r_{n-1}],$$

ahol

$\mathbf{V}=(v_0,\ v_1,\ \ldots,\ v_{n-1})$ adott kódvektor,
$\mathrm{E}=(e_0,\ e_1,\ \ldots,\ e_{n-1})$ hibavektor.

Tehát

$$r(x)=v(x)+e(x)=\sum_{i=1}^{n} r_{n-i}\cdot x^{n-i}$$

a vett polinom, és

$$e(x)=\sum_{i=1}^{n} e_{n-i}\cdot x^{n-i}$$

a hibapolinom.

A hibapolinomnak megkülönböztethetjük az információs szimbólumokat $e_i(x)$, illetve a redundáns szimbólumokat $e_p(x)$ értintő részeit:

$$e(x)=e_i(x)+e_p(x). \tag{3.1}$$

A hatékony dekódolási eljárás lehetővé teszi $e(x)$ meghatározását. A dekódolás során elsősorban a vett vektor szindrómájára van szükség:

$$\mathbf{S}=[s_0,\ s_1,\ \ldots,\ s_{r-1}]. \tag{3.2}$$

A szindrómapolinom

$$s(x)=R_g[r(x)]=R_g[e_i(x)]+e_p(x)=\sum_{i=1}^{r} s_{r-i}\cdot x^{r-i}, \tag{3.3}$$

mert

$$R_g[v(x)]=0 \quad \text{és} \quad R_g[e_p(x)]=e_p(x).$$

Ezután megvizsgáljuk a szindrómavektor

$$w(\mathbf{S}) \tag{3.4}$$

Hamming-súlyát, vagyis a nem nulla elemek számát. Három esetet különböztethetünk meg [2]:

1. Ha $t=\left[\frac{d-1}{2}\right]$-nél nincs több hiba a vett vektorban, és a hibák csak a rendundáns részben helyezkednek el, akkor a szindrómavektor Hamming-súlya:

$$w(\mathbf{S})\leqq t, \tag{3.5}$$

és

$$e(x)=e_p(x)=s(x). \tag{3.6}$$

2. Ha t-nél nem több hiba következik be az információs pozíciókban (és csak ott), akkor

$$e(x)=e_i(x), \tag{3.7}$$

$$s(x)=R_g[e_i(x)]. \tag{3.8}$$

Ez azt jelenti, hogy

$$e_i(x)-R_g[e_i(x)]=e_i(x)-s(x) \tag{3.9}$$

kódpolinom, és így nem nulla elémeinek száma $(2t+1)$-nél nem lehet kevesebb [3]. Ezért 1, 2, ..., t információs szimbólumhíba esetén $s(x)$ nem nulla elemeinek száma nem lehet kevesebb, mint $2t$, $2t-1$, ..., $t+1$.

3. Ha összesen nincs t-nél több hiba, és a redundáns részben bekövetkezett hibák száma z, akkor az információs pozíciókban legfeljebb $t-z$ hiba következett be. Ezek az információs részben levő hibák $t+z+1$ nem nulla elemet okozhatnak $s(x)$-ben. Ez a szám legfeljebb z-vel változhat a hibás redundáns pozíciók hatására.

Így ha a vett vektorban a hibák száma nem több t-nél, akkor

$$w(\mathbf{S})\geqq t+1. \tag{3.10}$$

Tehát a (3.4) szindróma Hamming-súlya alapján t vagy kevesebb hibát kijavíthatunk, ha a hibák a kódszó paritásrészében következtek be. (3.2)-t kivonva a vett vektorból kapjuk a javított vett vektort:

$$\mathbf{U}=\mathbf{R}-\mathbf{S}=[r_0-s_0, r_1-s_1, \ldots, r_{r-1}-s_{r-1}, r_r, \ldots, r_{n-1}], \tag{3.11}$$

A 2. és 3. esetben á hibák javítása érdekében minden információs pozíciót át kell tolni a redundáns részbe.

Így

$$r_a(x)=x^a\cdot r(x) \bmod (x^n-1)=\sum_{i=1}^{n} r_{N-i}\cdot x^{n-i}, \tag{3.12}$$

ahol $N=n-a-i \bmod (n)$. A (3.12) polinomnak megfelelő vektor:

$$\mathbf{R}_a=[r_{n-a},\ \ldots,\ r_0,\ r_1,\ \ldots,\ r_{n-a-2},\ r_{n-a-1}]. \tag{3.13}$$

Ez a vektor a vett vektorból a számú ciklikus balra léptetéssel kapható.

(3.13) szindrómája

$$s_a(x)=R_g[r_a(x)], \qquad \mathbf{S}_a=[s_0^a, s_1^a, \ldots, s_{r-1}^a]. \quad (3.14)$$

Ezért, ha a szindróma súlya t-nél nagyobb, meg kell határozni $s_a(x)$-et $a=1, 2, \ldots, k$ esetére. Ha valamilyen $m \leqq k$-ra fennáll, hogy:

$$w(\mathbf{S}_m) \leqq t, \qquad \mathbf{S}_m=[s_0^m, s_1^m, \ldots, s_{r-1}^m], \quad (3.15)$$

akkor a hibajavítás a következőképpen végezhető el:

$$\mathbf{U}_m=\mathbf{R}_m-\mathbf{S}_m=[r_{n-m}, \ldots, r_{n-m-1}]- \\ -[s_0^m, \ldots, s_{r-1}^m, 0, \ldots, 0]. \quad (3.16)$$

A dekódolási eljárás utolsó lépése a (3.16) vektor $n-m$ hellyel való jobbra léptetése, és a következő javított vektor (polinom) képzése:

$$u(x)=x^{n-m}\cdot u_m(x) \bmod (x^n-1). \quad (3.17)$$

Ha a (3.15) feltétel egyetlen $a=1, 2, \ldots, k$ esetére sem teljesül, ez azt jelenti, hogy a hibák nem tolhatók be a paritás pozíciókba, vagy pedig legalább $t+1$ hiba következett be.

4. Műveletek GF(q)-ban

Hogy a kódolási és dekódolási eljárásokat elvégezhessük, ismernünk kell a GF(q)-ban értelmezett aritmetikai műveletek végrehajtási módját.

Abban az esetben, ha $q=p$, ahol p=tetszőleges prímszám, a GF(p) műveletek a jól ismert modulo (p) aritmetikai műveletek. A feladat jóval bonyolultabb, ha q p-nek valamilyen hatványa.

Legyen $q=p^n$, n=egész szám, akkor GF(q) multiplikatív csoportja [GF(q) minden nem nulla eleme] előállítható GF(q) primitív elemeinek hatványaként:

$$\boldsymbol{\alpha}^i=[a_{i,0}, a_{i,1}, \ldots, a_{i,n-1}], \quad (4.1)$$

ahol

$$x^i \equiv \sum_{i=1}^{n} a_{i,n-i}x^{n-i} \pmod{p(x)}, \quad (4.2)$$

és

$$p(x)=x^n+\sum_{i=1}^{n} p_{n-i}\cdot x^{n-i}. \quad (4.3)$$

Ez GF(q) felett primitív, n-ed fokú polinom.

A legegyszerűbb módszer a GF(q)-beli műveletek elvégzésére az összes $q-1$ vektor (4.1) kiszámítása. Ekkor, mivel α rendje $q-$ í, a szorzás szabálya a következő:

$$\boldsymbol{\alpha}^i\cdot\boldsymbol{\alpha}^k=\boldsymbol{\alpha}^{i+k} \pmod{(q-1)}. \quad (4.4)$$

Az összeadás művelete a következőképpen végezhető el:

$$\boldsymbol{\alpha}^k+\boldsymbol{\alpha}^s=[(a_{k,0}+a_{s,0})_p, \ldots, (a_{k,n-1}+a_{s,n-1})_p], \quad (4.5)$$

ahol $(x+y)_p$ a mod p összeadást jelöli.

Vegyük észre, hogy a (4.1) vektor $a_{i,0}$ elemei a következő lineáris rekurzív relációból is kiszámíthatók:

$$a_{i,0}=-\sum_{j=0}^{n-1} p_j\cdot a_{i-j-n,0}, \quad i=n, n+1, \ldots, q-2, \quad (4.6)$$

ahol $a_{i,0}$ az $\boldsymbol{\alpha}^i$ vektor első eleme és a p_j-k a 4.3 polinom együtthatói.

(3.8)-ba behelyettesítve az $a_{0,0}=1$, $a_{1,0}=\ldots,=a_{n-1,0}$ kezdőértékeket, kiszámíthatjuk a (4.6) összefüggés által generált sorozat periódusát:

$$a_{0,0}, a_{1,0}, \ldots, a_{n-1,0}, \ldots, a_{q-2}. \quad (4.7)$$

Ezután kereséssel megállapítható, hogy GF(q) tetszőleges nem nulla $\boldsymbol{\alpha}^i$ eleméhez található-e olyan m(j), amelyre teljesül, hogy

$$a_{i,j}=a_{x,0}, \qquad i=1, 2, \ldots, q-2, \qquad j=1, 2, \ldots, n-1, \quad (4.8)$$

ahol $x=i+m(j) \pmod{(q-1)}$. Ez jelentősen egyszerűsíti GF(q) multiplikatív csoportjának kiszámítását.

1. példa: műveletek hardware megvalósítása GF(16) ban

GF(16) multiplikatív csoportja az

$$\boldsymbol{\alpha}^i=[a_{i,0}, a_{i,1}, a_{i,2}, a_{i,3}], \qquad i=0, 1, \ldots, 14$$

vektorok halmazával ábrázolható, ahol

$$x^i \equiv a_{i,0}+a_{i,1}x+a_{i,2}\cdot x^2+a_{i,3}\cdot x^3 \pmod{p(x)},$$

és $p(x)=x^4+x+1$ egy negyedfokú primitív polinom GF(2) felett.

Mivel $i<4$-re x^i mod $p(x)$ azonos x^i-vel:

$$a_{i,i}=1, \qquad a_{i,j}=0, \qquad \text{ha} \qquad i\neq j, \; i,j<4.$$

Az x^4+x+1 polinommal meghatározott rekurzív reláció:

$$a_{i,0}=a_{i-4,0}+a_{i-3,0} \pmod 2, \qquad i=4, 5, \ldots, 14,$$

így az ezen reláció által generált sorozat egy periódusa, ha $a_{0,0}=1$, $a_{1,0}=a_{2,0}=a_{3,0}=0$:

100010011010111.

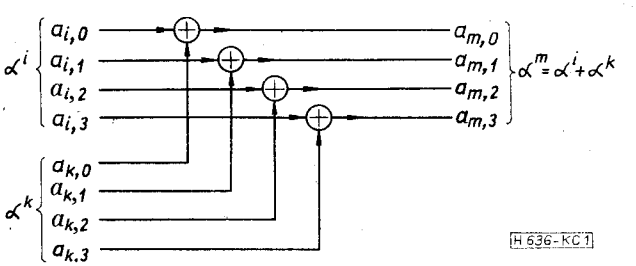
Észrevehetjük, hogy

$$a_{i,j}=a_{x,0}, \qquad x \equiv i+3-j \pmod{15},$$
$$i=0, 1, \ldots, 14, \; j=0, 1, 2, 3.$$

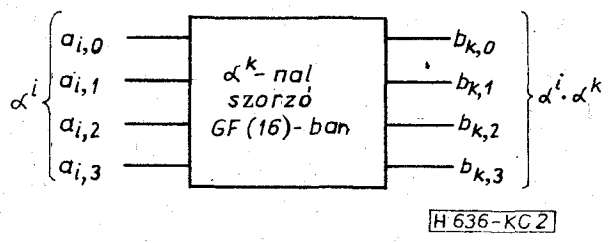
Így GF(16) minden nem nulla elemét egyszerűen felírhatjuk:

$$1=\boldsymbol{\alpha}^0 - 1000 \quad \boldsymbol{\alpha}^5 - 0110 \quad \boldsymbol{\alpha}^{10} - 1110$$
$$\boldsymbol{\alpha} - 0100 \quad \boldsymbol{\alpha}^6 - 0011 \quad \boldsymbol{\alpha}^{11} - 0111$$
$$\boldsymbol{\alpha}^2 - 0010 \quad \boldsymbol{\alpha}^7 - 1101 \quad \boldsymbol{\alpha}^{12} - 1111$$
$$\boldsymbol{\alpha}^3 - 0001 \quad \boldsymbol{\alpha}^8 - 1010 \quad \boldsymbol{\alpha}^{13} - 1011$$
$$\boldsymbol{\alpha}^4 - 1100 \quad \boldsymbol{\alpha}^9 - 0101 \quad \boldsymbol{\alpha}^{14} - 1001.$$

Mivel GF(16) karakterisztikája 2, az összeadás megegyezik a kivonással. A GF(16) feletti összeadót az 1. ábra mutatja.



1. ábra. Összeadó GF (16) -ban



2. ábra. Szorzó GF (16) -ban

A GF(16) feletti szorzás hardware megvalósítása jóval bonyolultabb. Szerencsére a hibacsomókat javító kódolási és dekódolási eljárás során csak egyetlen, GF(q)-beli konstans elemmel történő szorzásra van szükség, és ez a művelet viszonylag egyszerű. A GF(16)-beli α^i és α^k közötti szorzást megvalósító általános hálózatot a 2. ábra mutatja. A „doboz" itt modulo 2 összeadók hálózatát tartalmazza. A mod 2 összeadók száma, kapcsolásuk és csatlakozásuk a szorzó áramkör be- és kimenetéhez a kombinációs hálózatoknál használt módszerekkel határozható meg. GF(16) esetén $A(k)$ függvény kiszámítható, az eredményt az 1. táblázat tartalmazza.

Észrevehetjük, hogy:

$$b_{k,0}=A(k)$$
$$b_{k,1}=A(k+3 \pmod{15})$$
$$b_{k,2}=A(k+2 \pmod{15})$$
$$b_{k,3}=A(k+1 \pmod{15}).$$

Ha például $k=12$, akkor

$$b_{12,0}=a_{i,0}+a_{i,1}+a_{i,2}+a_{i,3}$$
$$b_{12,1}=a_{i,0}$$
$$b_{12,2}=a_{i,0}+a_{i,1}$$
$$b_{12,3}=a_{i,0}+a_{i,2}+a_{i,3}.$$

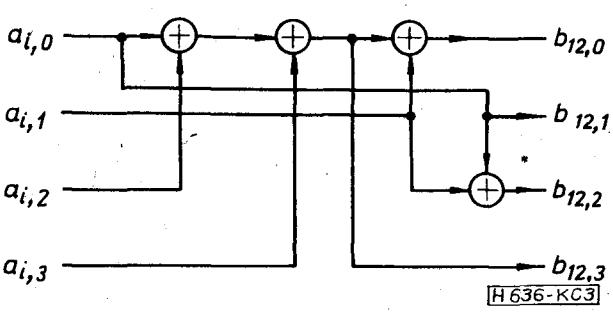
Az α^{12}-nel szorzó áramkör a 3. ábrán látható.

A GF(16) test esetében a mod 2 összeadók száma 1-től 5-ig változhat. Belátható, hogy az áramkörök optimális megvalósításakor minimalizálási probléma is fellép.

1. táblázat

Az $A(k)$ függvény GF(16)-ra

k	$A(k)$	k	$A(k)$
0	$a_{i,0}$	8	$a_{i,0}+a_{i,2}$
1	$a_{i,3}$	9	$a_{i,1}+a_{i,3}$
2	$a_{i,2}$	10	$a_{i,0}+a_{i,2}+a_{i,3}$
3	$a_{i,1}$	11	$a_{i,1}+a_{i,2}+a_{i,3}$
4	$a_{i,0}+a_{i,3}$	12	$a_{i,0}+a_{i,1}+a_{i,2}+a_{i,3}$
5	$a_{i,2}+a_{i,3}$	13	$a_{i,0}+a_{i,2}+a_{i,3}$
6	$a_{i,1}+a_{i,2}$	14	$a_{0i,}+a_{i,1}$
7	$a_{i,0}+a_{i,1}+a_{i,3}$		



3. ábra. GF (16)-ban α^{12}-nel szorzó kapcsolása

5. A kódolási és dekódolási eljárás hardware megvalósítása

Az MSI és LSI áramkörök és a mikroprocesszorok alkalmazása bizonyára kiterjeszti a hibajavító módszerek alkalmazását a digitális átviteli rendszerekben és más információs rendszerekben is.

A kódoló és hibajavító áramkör egy a GF(q) feletti véges automata, amely a 4. ábrán látható elemekből épül fel.

A $g(x)$ generátorpolinom szerinti kódolót az 5. ábra mutatja.

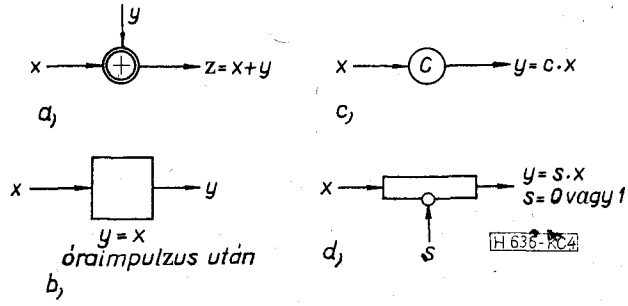
A kódoló k információs szimbólumot kap a forrástól, továbbítja azokat az átviteli csatornára és egyidejűleg a $g(x)$ generátorpolinomnak megfelelően visszacsatolt léptetőregiszterre. A k léptető jel ideje alatt a C vezérlő jel értéke 1, az 1. kapu nyit, a 2. kapu zár, az áramkör képzi $R_g[f_i(x)]$-et. k lépés után a számolás befejeződik, és a következő r óraimpulzus hatására a kódoló kimenetén a kódvektor redundáns elemei lépnek ki.

Egy $h(x)$ paritáspolinom szerinti ekvivalens kódolót mutat a 6. ábra. Itt az információs elemek beolvasása után az áramkört n-szer léptetik. A kódoló kimenetén kilép k darab információs szimbólum, és az utolsó r szimbólum alkotja a kódvektor redundáns részét.

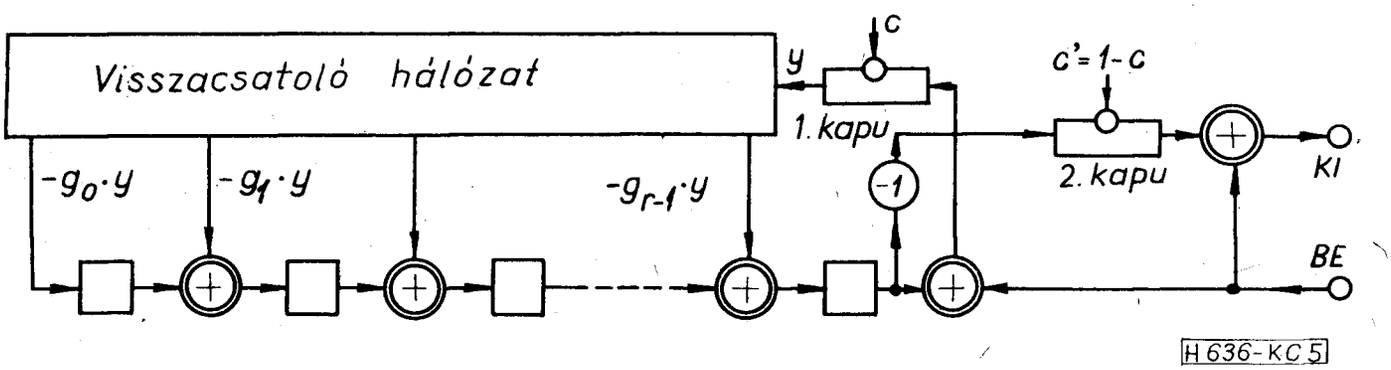
Nyilvánvaló, hogy a feltüntetett kódolóknak lehetnek egyéb, egyenértékű változatai is.

Az RS-kód általános dekódolójának vázlatát a 7. ábra mutatja. A hibajavító képességet a 3. fejezet tárgyalta, ám ez a dekódoló csak az információs elemek hibáit javítja, mivel a vevőben rendszerint nincs szükség a redundáns elemekre (természetesen a hibajavítás után).

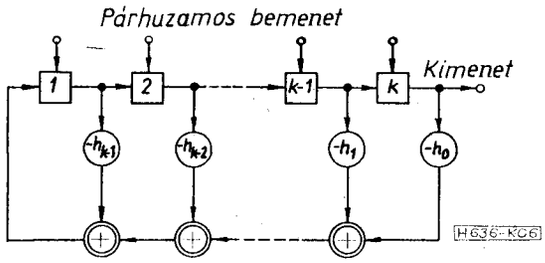
A dekódolási eljárás kétfázisú. Az első, n óraimpulzusnyi fázisban a felső regiszter n-szer, az alsó pedig k-szor lép. Így n óraimpulzus után az alsó regiszter tartalma a k információs szimbólum, a felsőé a szind-



4. ábra. RS-kódoló és -dekódoló áramkör elemei: *a)* összeadó GF *(q)* -ban, *b) c*-vel szorzó GF *(q)* -ban, *c)* memóriaelem, *d)* kapu



5. ábra. g(x) szerinti RS-kódoló vázlata



6. ábra. h(x) szerinti RS-kódoló vázlata

róma lesz. Ez idő alatt a küszöbáramkör T kimenetén 1 lesz.

A dekódolás második fázisában, amely ismét n óraimpulzusnyi, a küszöbáramkör kimenete a szindróma súlyától függ:

$$T=\begin{cases}1 & \text{ha} \quad w(S_a)>t,\\ 0 & \text{ha} \quad w(S_a)\leqq t.\end{cases}$$

A felső regiszter most is n-szer lép, de az alsó csak az utolsó k alkalommal. Így az utolsó k lépésnél

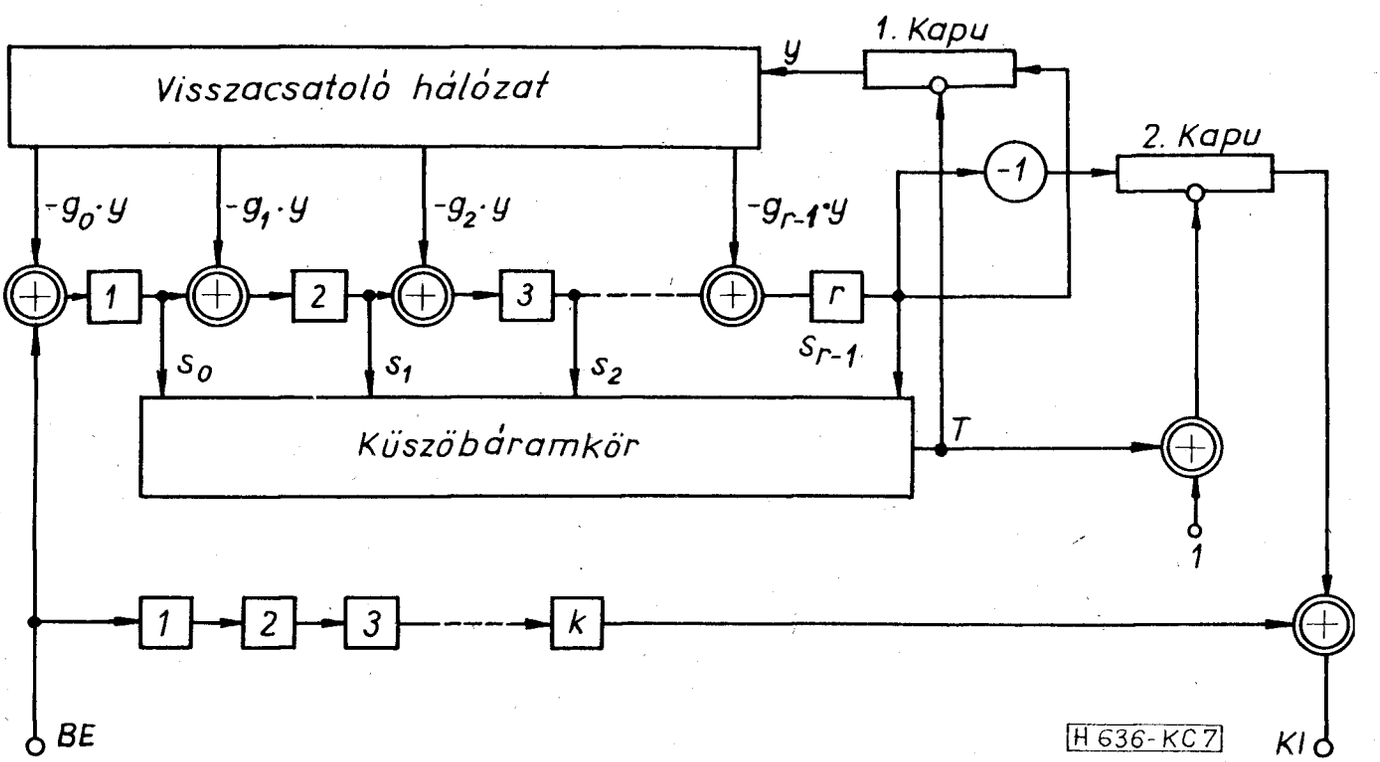
2. táblázat

A (15,9)-es HS-kód GF(16) felett ekvivalens generátorpolinomjai

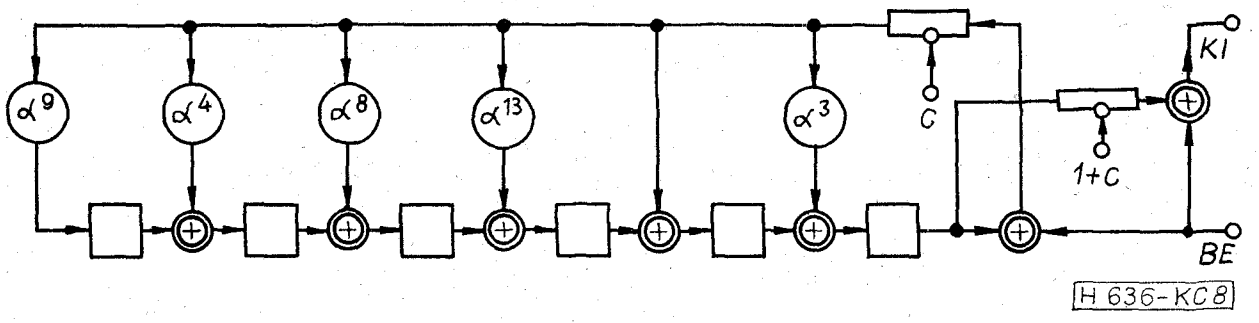
k	$g(x)$
1	$x^6+\alpha^{10}\cdot x^5+\alpha^{14}\cdot x^4+\alpha^4\cdot x^3+\alpha^6\cdot x^2+\alpha^9\cdot x+\alpha^6$
2	$x^6+\alpha^5\cdot x^5+\alpha^{13}\cdot x^4+\alpha^8\cdot x^3+\alpha^{12}\cdot x^2+\alpha^3\cdot x+\alpha^{12}$
4	$x^6+\alpha^{10}\cdot x^5+\alpha^{11}\cdot x^4+\alpha\cdot x^3+\alpha^9\cdot x^2+\alpha^8\cdot x+\alpha^9$
7	$x^6+\alpha^9\cdot x^5+\cdot x^4+\alpha^{14}\cdot x^3+\alpha^4\cdot x^2+\alpha^2\cdot x+\alpha^{12}$
8	$x^6+\alpha^5\cdot x^5+\alpha^7\cdot x^4+\alpha^2\cdot x^3+\alpha^3\cdot x^2+\alpha^{12}\cdot x+\alpha^3$
11	$x^6+\alpha^{12}\cdot x^5+\cdot x^4+\alpha^7\cdot x^3+\alpha^2\cdot x^2+\alpha\cdot x+\alpha^6$
13	$x^6+\alpha^6\cdot x^5+\cdot x^4+\alpha^{11}\cdot x^3+\alpha\cdot x^2+\alpha^8\cdot x+\alpha^3$
14	$x^6+\alpha^3\cdot x^5+\cdot x^4+\alpha^{13}\cdot x^3+\alpha^8\cdot x^2+\alpha^4\cdot x+\alpha^9$

a dekódoló kimenetén a javított információs szimbólumok jelennek meg.

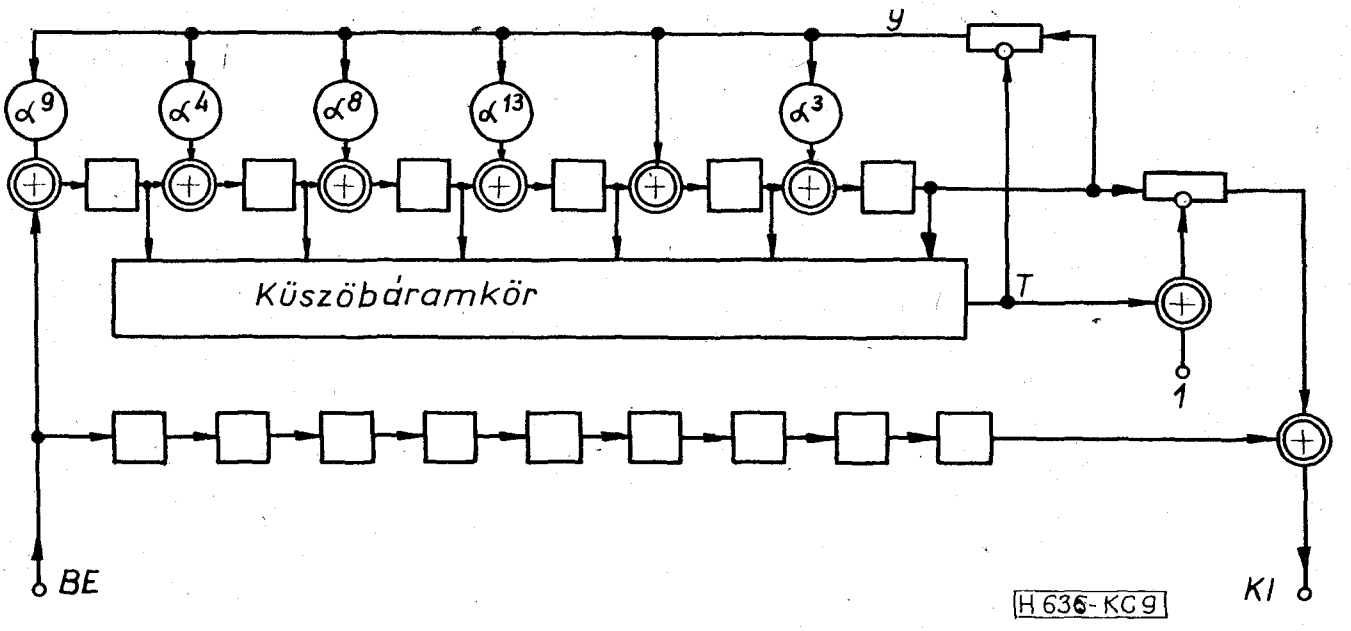
2. példa: A (15,9)-es RS-kód minimális távolsága $d=7$, kódolójának és dekódolójának megvalósítása.



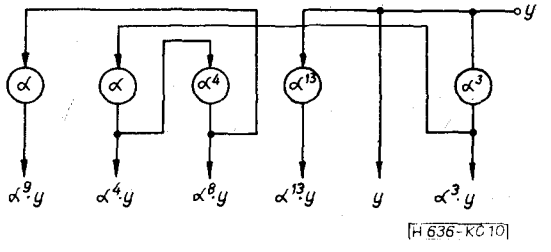
7. ábra. Hibaesomó-javító RS-kódoló általános vázlata



8. ábra. GF(16) feletti, (16,9)-es RS-kód dekódolója



9. ábra. GF(16) feletti, hibacsomó-javító, (16,9)-es RS-kód dekódolója



10. ábra. A 8. ábrán látható áramkör javasolt visszacsatoló hálózata

Mivel GF(16)-ban 8 primitív elem van, az (1.1) képletben α helyett α^k, $k=1, 2, 4, 7, 8, 11, 13, 14$ helyettesítendő. Az ily módon kiszámított generátorpolinomokat a 2. táblázat tartalmazza.

Feltételezve, hogy a generátorpolinom minden együtthatójának külön szorzó áramkör felel meg, $k=14$-re olyan kódolót és dekódolót kapunk, amelyben a mod 2 összeadók száma minimális. A kódoló és dekódoló áramköri megvalósítása a 8. és 9. ábrákon látható. Az ábrákon az összeadó és a szorzómű az 1. és 2. ábrákon láthatókkal azonosak, a memóriaszimbólum 4 db bináris tárolóként értelmezendő. A 8. ábra áramkörét a következő funkcionális elemekből lehet megépíteni: 24 flip-flop, 45 mod 2 összeadó és 8 AND kapu. A regiszter visszacsatolását a 10. ábra szerint módosítva, a mod 2 összeadók számát 6-tal csökkenthetjük.

Összefoglalás

A cikk alapján megállapítható, hogy a kódolás-elmélet gyakorlati alkalmazása új és nem könnyű elméleti feladatokat vet fel. Ezek közül a legfontosabbak:

- a kódoló és dekódoló elemszámának minimalizálása,
- a kódolási és dekódolási eljárás számítógépes szimulációja,
- a kódoló és dekódoló áramkörök számítógéppel segített tervezése.

Végül ezúton fejezem ki hálás köszönetemet dr. Osváth Lászlónak a nyelvi segítségért, valamint dr. Gordos Géza docensnek, aki a BME Híradástechnikai Elektronika Intézetében töltött 3 hónap alatt kutatómunkámat és e cikk megírását mindenben támogatta.

IRODALOM

[1] *Reiger, S. H.:* Codes for the correction of clustered errors. IRE Transactions, IT—6, 1960

[2] *Szwaja, Z.:* On step—by—step decoding of the BCH binary codes. IEEE Transactions, IT—13, 1967

[3] *Peterson, W. W.—Weldon, E. J.:* Error-correcting codes. the MIT Press, 1972.