ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1979

Seria: AUTOMATYKA z. 49 Nr kol. 612

Zygmunt SZWAJA

Instytut

Politechnika Poznańska

O ZŁOŻONOŚCI ZWIĄZANEJ Z REALIZACJĄ KODÓW NADMIAROWYCH

Streszczenie. W pracy rozpatruje się zagadnienie lokalizacji błędów binarnych w kodach nadmiarowych. W pierwszym rzędzie przedstawiono heurystyczny sposób tworzenia kodów; stosując zaproponowany algorytm uzyskuje się zarówno kody nieliniowe jak liniowe. Dla pierwszego przypadku dekodowanie można realizować na zasadzie największego podobieństwa. Wykazano, że ta metoda - ze względu na swoją prostotę - może być użyteczna również dla kodów liniowych o małej liczbie pozycji informacyjnych, jak np. dla kodów dualnych względem kodów Hamminga.

Po przypomnieniu złożoności dekodowania opartego na związkach sum potęgowych z podstawionymi funkcjami symetrycznymi przedstawiono nową metodę dekodowania, sprowadzającą się do porównywania syndromu badanego wielomianu z syndromami kolejno generowanych wielomianów błędów.

1. HEURYSTYCZNE PODEJŚCIE DO TWORZENIA KODÓW NADMIAROWYCH

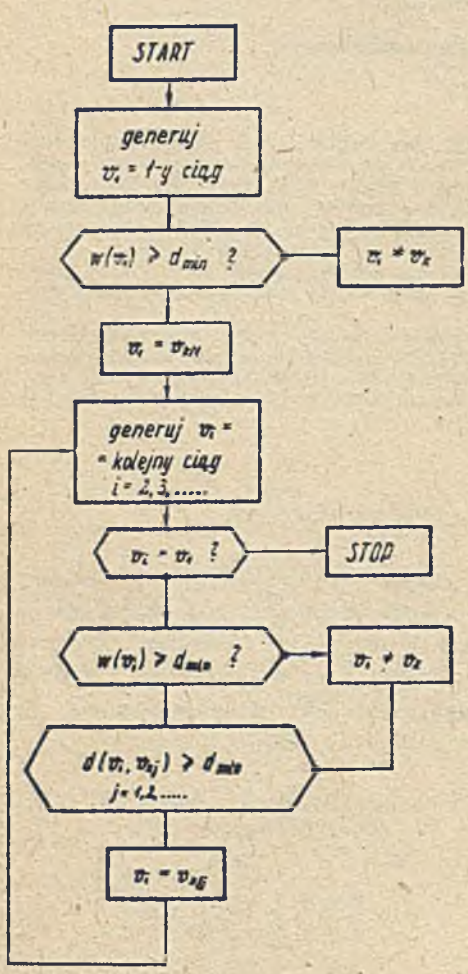
Przedstawiony poniżej zbiór sześciu ciągów - o liczbie pozycji równej 7 - charakteryzuje się tym, że każdy z ciągów różni się od pozostałych co najmniej na 3 pozycjach oraz że zawiera nie mniej niż 3 jedynki. Właściwość tę możemy wyrazić inaczej: każdy z ciągów jest odległy od pozostałych co najmniej o 3 oraz że waga każdego ciągu jest nie mniejsza niż 3, przy czym mowa tu o odległości "d", i o wadze "w" Hamminga [1].

Kod beznadmiarowy (BNM) - liczby 1-6 wyrażone w postaci binarnej	Kod nadmiarowy (NM) - zbiór 6 ciągów: n=7, $w_{min} = 3$, $d_{min} = 3$
0 0 1	0 0 1 1 1 0 0
0 1 0	0 1 1 0 0 0 1
0 1 1	1 1 1 1 1 1 1
1 0 0	1 0 0 0 1 1 0
1 0 1	0 1 0 1 1 1 0
1 1 0	1 0 1 0 0 1 1

Wymieniony zbiór ciągów - stanowiący przykład kodu nadmiarowego - może być przyporządkowany np. zbiorowi liczb 1-6 wyrażonych w postaci binarnej.

Tego rodzaju przyporządkowanie będziemy nazywać kodowaniem. Dekodowaniem będzie proces odwrotny, również w przypadku kiedy w ciągu kodu nadmiarowego wystąpiły błędy binarne, a zatem kiedy ciąg badany v_b nie jest ciągiem kodowym. liczba korygowalnych, tj. możliwych do zlokalizowania, błędów binarnych zależy od d_{min}, a mianowicie musi zachodzić:

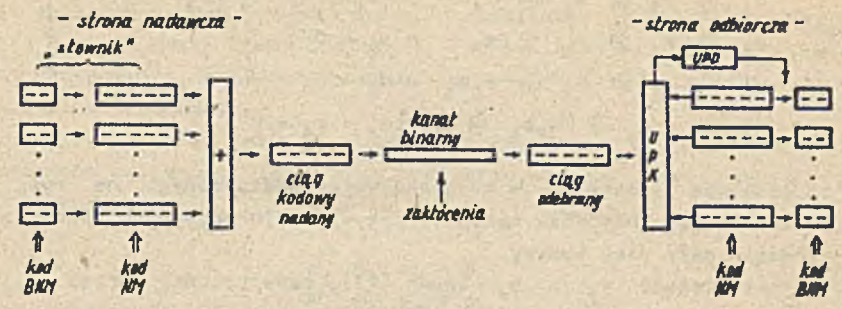
$$d_{min} \geqslant 2t + 1.$$



Rys. 1. Algorytm znajdowania ciągów kodowych v_k

Wyszczególniony kod nadmiarowy można traktować jako zbiór przypadkowo dobieranych ciągów spełniających określone wymagania. Wyszukiwanie ciągów można dokonywać opierając się na grafie jak na rys. 1. Związana z tym realizacja układowa jest dość złożona i wymaga istnienia: a) generatora ciągów binarnych, np. pseudolosowych, o długości n, b) 2 układów progowych n-wejściowych o wagach równych 1 i o wartości progu $T = d_{min}$, c) komparatora złożonego z n sumatorów mod 2, d) układu rejestrów, których liczba winna być równa liczbie ciągów kodowych, e) układu przełączającego, umożliwiającego porównywanie ciągu wygenerowanego z ciągami uznanymi uprzednio jako kodowe. Liczba koniecznych porównań jest nie większa od liczby zarejestrowanych ciągów kodowych; próba, w której badany ciąg okazał się odległy o mniej niż d_{min} implikuje odrzucenie ciągu i wygenerowanie następnego.

Na rys. 2 przedstawiono ogólny schemat przekazywania informacji, połączony z określonym wyżej kodowaniem i z dekodowaniem. Po stronie nadawczej znajduje się słownik "Kod BNM - Kod NM", a po stronie odbiorczej słownik "Kod NM - Kod BNM". Ciąg odebrany, który w wyniku zakłóceń może różnić się od ciągu nadanego, podlega porównywaniu z poszczególnymi ciągami kodu NM; przyjmuje się, że został nadany ten ciąg kodowy,



Rys. 2. Układ nadawania i odbioru informacji za pomocą kodu nadmiarowego. UPK - układ przełączników i komparatorów, UPD - układ podejmowania decyzji. Bloki UPK i UPD symbolizują, że dekodowanie oparte jest na zasadzie największego podobieństwa

który jest od ciągu odebranego najmniej odległy, czyli inaczej, do ciągu odebranego najbardziej podobny, a więc że dekodowanie odbywa się na zasadzie największego podobieństwa. Jako ciąg nadany zostaje uznany ten z porównywanych ciągów kodu NM, który jest odległy od ciągu badanego o nie więcej niż $\frac{1}{2} d_{min}$, gdzie d_{min} - minimalna odległość w kodzie NM.

"Słowniki", o których mowa wyżej, (rys. 2) można nieco uprościć przez odpowiednie przyporządkowanie ciągów nadmiarowych ciągom beznadmiarowym i na odwrót. Chodzi o to, żeby ciąg beznadmiarowy był przedrostkiem przyporządkowanego mu ciągu nadmiarowego. Zilustrowano to na przykładzie kodu ze str. 2, ze zmienionym przyporządkowaniem ciągów kodu NM i kodu BNM

$$\begin{pmatrix} 0\,0\,1 \leftrightarrow 0\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0 \leftrightarrow 0\,1\,0\,1\,1\,1\,0 \\ 0\,1\,1\,.\,0\,1\,1\,0\,0\,0\,1 \\ 1\,0\,0\,.\,1\,0\,0\,0\,1\,1\,0 \\ 1\,0\,1\,.\,1\,0\,1\,0\,0\,1\,1 \\ 1\,1\,1 \leftrightarrow 1\,1\,1\,1\,1\,1\,1 \end{pmatrix} \Longrightarrow$$

kod BNM kod NM

a_1	a_2	a_3	c_1	c_2	c_3	c_4
0	0	1	1	1	0	0
0	1	0	1	1	1	0
0	1	1	0	0	0	1
1	0	0	0	1	1	0
1	0	1	0	0	1	1
1	1	1	1	1	1	1

kod BNM

kod NM

Taka forma zapisu skłania do wyrażenia wartości pozycji nadmiarowych w postaci funkcji pozycji informacyjnych, a więc - przy oznaczeniach jak wyżej - w postaci: $c_j = f(a_i)$; $i = 1,2,3$; $j = 1,2,3,4$. Tak więc np.:

$$c_1 = \bar{a}_1\bar{a}_2a_3 \oplus \bar{a}_1a_2\bar{a}_3 \oplus a_1a_2a_3,$$

gdzie $\oplus$ - operacja sumowania mod. 2 Uwzględniając, że $\bar{a} = 1 \oplus a$ i przenosząc wszystkie wyrazy na lewą stronę, mamy dla c_1, c_2, c_3 i c_4 następujące zależności:

$$\left.\begin{array}{l} c_1 \oplus a_2 \oplus a_3 \oplus a_1a_2 \oplus a_1a_3 \oplus a_1a_2a_3 = s_1 = 0 \\ c_2 \oplus a_1 \oplus a_2 \oplus a_3 = s_2 = 0 \\ c_3 \oplus a_1 \oplus a_2 \oplus a_2a_3 = s_3 = 0 \\ c_4 \oplus a_1a_3 \oplus a_2a_3 \oplus a_1a_2a_3 = s_4 = 0 \end{array}\right\} \quad (1)$$

Daje to podstawę do innego - w stosunku do przedstawionego na rys. 2 - sposobu kodowania: znajomość wartości pozycji informacyjnych pozwala od razu określić cały ciąg kodowy.

Wyrażenia wiążące c_j z a_i (wzór (1)), oznaczyliśmy przez s_i; dla omawianych warunków są one równe zeru. Przestaje to być słuszne w przypadku występowania pojedynczych błędów w ciągu kodowym. W omawianym przykładowo kodzie wartości s_i obliczone z (1) dla przypadków wystąpienia błędu na tej samej pozycji są zależne od nadanego ciągu. Oznacza to, że wartości s_i nie są jednoznacznie związane z określoną błędną pozycją i tym samym nie stanowią podstawy do lokalizacji błędu.

2. KODY LINIOWE I ICH OPIS MACIERZOWY

Próby zbudowania kodu o $n=6$, $w_{min} = d_{min} = 3$, oparte na algorytmie według rys. 1, doprowadziły do następujących 3 przypadków:

	(a)						(b)						(c)					
	a_1	a_2	a_3	c_1	c_2	c_3	a_1	a_2	a_3	c_1	c_2	c_3						
v_1	0	0	1	0	1	1	0	0	1	1	0	0	0	0	1	1	0	1
v_2	0	1	0	1	0	1	0	1	0	1	1	0	0	1	0	0	1	1
v_3	0	1	1	1	1	0	0	1	1	0	1	1	1	0	0	1	0	1
v_4	1	0	0	1	1	0	1	0	0	0	1	1	1	0	1	0	1	0
v_5	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	0	0	1
v_6	1	1	0	0	1	1	1	1	0	1	0	1						
v_7	1	1	1	0	0	0	1	1	1	0	0	0						
	kod BNM						kod BNM						kod BNM					
	kod NM						kod NM						kod NM					

Każdy z tych kodów możemy oznaczyć krótko w postaci $(n,k)=(6,3)$, gdzie k oznacza liczbę pozycji kodu beznadmiarowego, które to pozycje możemy nazwać pozycjami informacyjnymi a n-łączną liczbę pozycji w kodzie nadmiarowym.

Jeśli chodzi o kody (a) i (b), to zauważamy, że każdy ciąg da się tu wyrazić przez sumę 2 innych ciągów tego samego kodu, przy czym sumowanie 2 ciągów sprowadza się do sumowania mod 2 poszczególnych pozycji; jest więc np. $v_3 = v_1 + v_2$, $v_1 = v_6 + v_7$, itd., a więc waga danego ciągu kodowe-

go jest odległością dwu innych ciągów; w rezultacie zachodzi tu $d_{min} = w_{min}$. Jest to cechą kodów liniowych, których kody (a) i (b) są szczególnymi przypadkami. Łatwo stwierdzić, że kod (c) i kod podany na str. 21 nie są kodami liniowymi.

Postępując podobnie jak w p. 1 możemy stwierdzić, że w przypadku kodów liniowych zależności $c_i = f(a_i)$ są prostymi funkcjami; np. dla kodu (a) zachodzi:

$$\begin{aligned} a_1 \oplus a_2 &= c_1 \\ a_1 \oplus a_3 &= c_2 \\ a_2 \oplus a_3 &= c_3 \end{aligned} \tag{2}$$

co prowadzi do prostej realizacji układowej związanej z operacją kodowania.

Dalej wprowadzając oznaczenia:

$$\begin{aligned} a_1 \oplus a_2 \oplus c_1 &= s_1 \\ a_1 \oplus a_3 \oplus c_2 &= s_2 \\ a_2 \oplus a_3 \oplus c_3 &= s_3 \end{aligned} \tag{3}$$

łatwo stwierdzimy, że wystąpienie 1 błędu na dowolnej pozycji ciągu daje inne wartości ciągów $s_1 s_2 s_3$:

błąd na pozycji ciągu kodu (a)	wartość ciągu s_1	s_2	s_3
brak błędów	0	0	0
a_1	1	1	0
a_2	1	0	1
a_3	0	1	1
c_1	1	0	0
c_2	0	1	0
c_3	0	0	1

co daje podstawę do lokalizacji pojedynczych błędów.

Zależności (2) możemy zapisać w nieco zmienionej postaci:

$$\begin{aligned} a_1.1 \oplus a_2.1 \oplus a_3.0 \oplus c_1.1 \oplus c_2.0 \oplus c_3.0 &= 0 \\ a_1.1 \oplus a_2.0 \oplus a_3.1 \oplus c_1.0 \oplus c_2.1 \oplus c_3.0 &= 0 \\ a_1.0 \oplus a_2.1 \oplus a_3.1 \oplus c_1.0 \oplus c_2.0 \oplus c_3.1 &= 0 \end{aligned} \tag{4}$$

co można traktować jako rozpisanie iloczynu macierzowego:

$$\left[a_1 a_2 a_3 \; c_1 c_2 c_3\right] \cdot \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 0, \tag{5}$$

gdzie pierwsza macierz jest ogólnym zapisem ciągu kodowego, a druga jest transpozycją tzw. macierzy kontrolnej H:

$$H = \begin{array}{c} \begin{array}{cccccc} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{6}$$

złożonej ze współczynników przy a_i w równaniach (4); zero z prawej strony zależności (5) jest macierzą wierszową złożoną z 3 zer. Zależność (5) mówi, że każdy ciąg kodowy jest ortogonalny do każdej kolumny macierzy H.

Oznaczając kolumny macierzy H przez k_i, $i = 1,2,...6$, zauważamy, że możemy utworzyć 7 kombinacji liniowych takich kolumn, których suma daje kolumnę zerową (np. $k_1+k_2+k_3=0$, $k_1+k_2+k_5+k_6=0$ itd.) i że numery tych kolumn pokrywają się z numerami pozycji, na których w ciągach kodowych znajdują się jedynki. W szczególności najmniejsza liczba liniowo zależnych kolumn macierzy H jest równa najmniejszej liczbie jedynek ciągu kodowego, a więc równa wadze kodu, decydującej o zdolności korekcyjnej.

Wyżej omówiona właściwość macierzy H ma charakter ogólny i w istocie synteza kodu o zadanej w_{min} sprowadza się do syntezy macierzy kontrolnej H takiej, że każda kombinacja $w_{min}-1$ kolumn jest liniowo niezależna.

Kolumny macierzy kontrolnej H, wzór (6), możemy traktować jako binarny zapis liczb 1-6 albo jako zbiór 6 zerojedynkowych kombinacji na 3 pozycjach. Macierz, której kolumnami są wszystkie 3-pozycyjne niezerowe kombinacje zerojedynkowe jest macierzą kontrolą kodu Hamminga $(n,k)=(7,4)$. Ogólnie, macierz H, wykorzystująca wszystkie niezerowe kombinacje na m pozycjach, jest macierzą kontrolną Hamminga $(n,k) = (2^m-1, 2^m-1-m)$ [1].

Dla kodu $(n,k) = (7,4)$ mamy:

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{7}$$

Wyszukując zależności liniowe między kolumnami tej macierzy możemy określić 15 ciągów kodowych, z których 4 liniowo niezależne tworzą macierz generującą:

$$G = \begin{array}{c} \begin{array}{ccccccc} c_1 & c_2 & c_3 & a_1 & a_2 & a_3 & a_4 \end{array} \\ \underbrace{\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}}_{n} \end{array} \qquad (8)$$

W macierzy tej ostatnie 4 pozycje są pozycjami informacyjnymi, a pierwsze 3 - nadmiarowymi; ich związek - stanowiący podstawę układowej realizacji operacji kodowania - znajdujemy na podstawie zależności:

$$v \cdot H^T = 0, \qquad (9)$$

gdzie v - wektor kodowy. Dowolny błąd można wyrazić w postaci ciągu e, o długości n, z jedynkami na błędnych pozycjach. W efekcie błędny ciąg będzie $v_b = v+e$. Ciąg ten pomnożony przez H^T daje:

$$v_b H^T = (v+e)H^T = v\,H^T + e\,H^T = e\,H^T. \qquad (10)$$

Efektem pomnożenia jw. jest tzw. syndrom, który dla przypadku kodu (7,4) jest ciągiem zerojedynkowym o długości 3. Wystąpienie pojedynczego błędu daje syndrom pokrywający się z kolumną macierzy H, o numerze równym numerowi błędnej pozycji. Daje to podstawę do dekodowania, czyli do lokalizacji błędu. Wystąpienie większej liczby błędów daje syndrom będący sumą kolumn o numerach pokrywających się z numerami błędnych pozycji.

Tego rodzaju rozważania pozwalają wyciągnąć wnioski dotyczące zdolności korekcyjnej i detekcyjnej kodu. Istotną cechą kodów Hamminga $(n,k) = (2^m-1,\ 2^m-1-m)$ jest to, że każdy z ciągów o długości n jest albo ciągiem kodowym albo ciągiem odległym od któregoś z ciągów kodowych o 1. Można to dla kodów Hamminga wyrazić w postaci:

$$2^n = 2^k \left[1 + \binom{n}{1}\right].$$

Jest to szczególny przypadek zależności wyrażającej warunek konieczny korygowalności przez kod n,k t błędów:

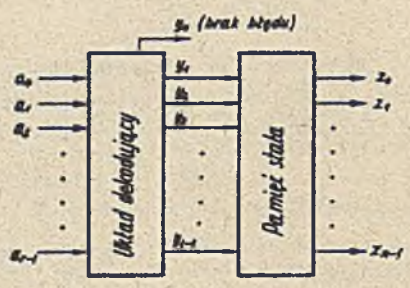
$$2^n \geqslant 2^k \left[1 + \binom{n}{1} + \binom{n}{2} = \ldots + \binom{n}{t}\right]. \qquad (11)$$

Przypadek kiedy obie strony są powiązane ze sobą znakiem równości dotyczy tzw. kodów ściśle upakowanych. Do takich kodów należą omawiane kody Hamminga oraz kod Golaya $(n,k) = (23,12)$.

Macierze kontrolne kodów Hamminga $(n,k) = (2^m-1,\ 2^m-1-m)$ można traktować jako macierze generujące kody dualne względem kodów Hamminga, a mianowicie kody $(n,k) = (2^m-1,m)$. Kody te cechują się tym, że waga wszystkich ciągów danego kodu jest taka sama i równa 2^{m-1}; wynikająca z tego duża zdolność korekcyjna, $t = 2^{m-2}$, jest opłacona dużą nadmiarowością.

Wyrażenie w nawiasach kwadratowych wzoru (11) określa liczbę wartości syndromów umożliwiających wyznaczenie położenia błędnych pozycji. Wykorzystanie tej właściwości do korygowania błędów w kodach dualnych byłoby związane z realizacją układu dekodującego o $r = n-k$ wejściach i l wyjściach, dołączonych do n-wyjściowej pamięci stałej, gdzie:

$$l = \binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t} \tag{12}$$



Rys. 3. Blokowy schemat układu logicznego lokalizującego błędy w kodzie liniowym n,k. Błędne pozycje wyznaczane na podstawie syndromu $a_0a_1\ldots a_r$ pojawiają się na odp. liniach ze zbioru $z_0, z_1, \ldots, z_{n-1}$, $r = n-k$

$$l = \left[\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}\right] \leq 2^r$$

Daje to wprawdzie prostą strukturę logiczną (rys. 3), ale liczba zaangażowanych do tego elementów jest - już od $m=4$, a więc od kodu (15,4), kiedy $l = 1926$ - bardzo duża. W tym aspekcie procedura lokalizacji błędów, oparta na zasadzie największego podobieństwa (rys. 2), okazuje się bardzo prosta.

Reasumując dotychczasowe rozważania możemy stwierdzić, że kody Hamminga i kody względem nich dualne można pod względem złożoności układowej, dotyczącej lokalizacji błędów, traktować jako dwie skrajności; w pierwszym przypadku dekodowanie, oparte na układach logicznych, jest uzasadnione nawet dla dużych n; korygowanie błędów w kodach dualnych do kodów Hamminga sensownie jest oprzeć na metodzie porównywania z ciągami kodowymi (rys. 2).

3. KODY CYKLICZNE

Począwszy od tego punktu będziemy zajmować się tylko kodami cyklicznymi; ciągi kodowe o długości n będziemy oznaczać symbolami $a_0, a_1, a_2, \ldots, a_{n-1}$ i zakładać, że pierwsze elementy ciągu, a więc $a_0, a_1, \ldots$, są pozycjami kontrolnymi, a pozostałe stanowią informację.

Nazwa "kody cykliczne" bierze się stąd, że jeżeli elementy ciągu takiego kodu rozłożymy w równych odstępach na okręgu, to dowolne cykliczne przesunięcie ciągu w lewo lub w prawo jest znowu ciągiem kodowym. Tak np. dla kodu (15,4) dualnego względem kodu Hamminga (15,11), byłoby:

$$\begin{array}{c} 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0 \\ \backslash\ \backslash\ \backslash\ .\ .\ .\ .\ \backslash\ \backslash\ \backslash \\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0 \end{array} \tag{13}$$

W efekcie do opisu kodu wystarczy znać tylko 1 jego ciąg.

Dodawanie ciągów przypomina dodawanie wielomianów; w istocie wielomiany można traktować jako pewien sposób zapisu ciągów [2]. W interesujących nas przypadkach byłyby to wielomiany nad GF(2). Tak więc ciąg kodowy o długości n $a_0 a_1 \ldots a_{n-1}$ w zapisie wielomianowym miałby postać: $a_0 + q_1 x + \ldots + a_{n-1} x^{n-1}$. Nie jest to tylko czysta formalność; wykorzystanie niektórych właściwości wielmianów prowadzi do ważnych wniosków dotyczących kodów. W szczególności kody Hamminga $(n,k) = (2^m-1,\ 2^m-1-m)$ można generować za pomocą tzw. wielomianów pierwotnych stopnia m; tak nazywamy wielomiany, które są dzielnikami dwumianu x^n+1, gdzie $n=2^m-1$ i żadnego innego dwumianu stopnia mniejszego niż n. Właściwość ta determinuje $w_{min}=3$ tych kodów. Każdy wielomian pierwotny jest nierozkładalny ale nie odwrotnie. Wielomiany niepierwotne stopnia m są dzielnikami dwumianu x^n+1, gdzie n jest dzielnikiem liczby 2^m-1. Mówi to, że kody generowane przez wielomiany niepierwotne mają wagę nie mniejszą niż 3, a więc że każdy z nich koryguje nie mniej niż 1 błąd.

Również łatwo jest wywnioskować, że kod generowany przez wielomiany:

$$g(x) = (1+x) \cdot w(x), \tag{14}$$

gdzie $w(x)$ jest wielomianem nierozkładalnym stopnia m, ma wagę nie mniejszą niż 4.

O liczbie wielomianów pierwotnych i niepierwotnych stopnia m można wnioskować na podstawie znajomości liczb m i $n=2^m-1$, i badania właściwości dwumianu x^n+1. Każdy taki wielomian jest podzielny przez $x+1$ więc suma potęg pozostałych dzielników musi być równa 2^m-2. Wśród liczb m i $n=2^{m-1}$ można wydzielić 3 grupy:

1) m i 2^m-1 są liczbami pierwszymi,
2) m jest liczbą pierwszą, a 2^m-1 jest rozkładalne,
3) m i 2^m-1 są rozkładalne.

Dla pierwszego przypadku wszystkie dzielniki wielomianu x^n+1 są wielomianami pierwotnymi stopnia m; ich liczba jest:

$$b = \frac{2^m-2}{m}. \tag{15}$$

Przypadek ten zachodzi dla $m = 3,5,7,13$ i in.

W dwu pozostałych przypadkach do określenia liczby pierwotnych i niepierwotnych wielomianów wykorzystuje się m.in. wzory Eulera i Möbiusa [3] Warto tu zaznaczyć, że wśród 10 wielomianów niepierwotnych stopnia 11 $m=11$ $n=2^{11}-1 = 2047$ są 2 takie, które generują kod Golaya $(n,k) = (23,12)$, umożliwiający korygowanie 3 błędów. Jest to obok kodów Hamminga, generowanych przez wielomiany pierwotne, jedyny przypadek kodu ściśle upakowanego [1]:

$$2^{23} = 2^{12}\left[\binom{23}{0} + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}\right]. \tag{16}$$

Jeżeli przez $p(x)$ oznaczymy wielomian pierwotny stopnia m, a więc wielomian generujący cykliczny kod Hamminga $(2^m-1, 2^m-1-m)$, to wielomian generujący kod cykliczny $(n,k) = (2^m-1,m)$, będący kodem dualnym względem kodu Hamminga, będzie:

$$h(x) = \frac{x^{2^m-1}+1}{p(x)}. \tag{17}$$

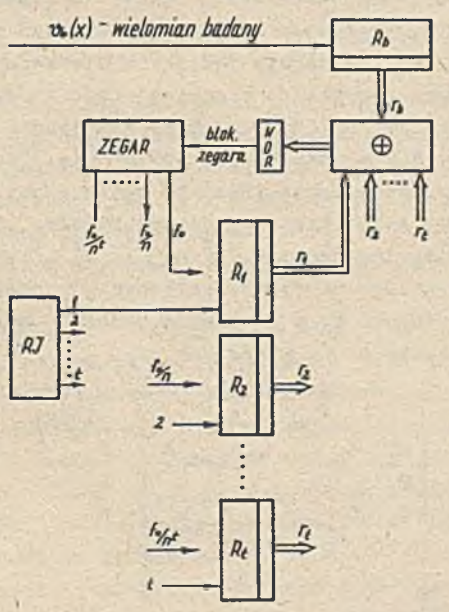
W obu przypadkach jak i we wszystkich przypadkach kiedy kod jest generowany przez dowolny iloczyn wielomianów nierozkładalnych stopnia m, wśród których znajduje się co najmniej 1 wielomian pierwotny, zachodzi $x^n=1$, $n=2^m-1$, co m.in. oznacza, że ostatni element wielomianu - w wyniku cyklicznego przesunięcia - przechodzi na pozycję pierwszą. Jeżeli kod jest generowany przez wielomian $g(x)$ (x^l+1), gdzie $l|(2^m+1)$, to zachodzi $x^l=1$.

4. ZAGADNIENIE DEKODOWANIA W KODACH BCH

Kody Hamminga $(n,k) = (2^m-1, 2^m-1-m)$, kody względem nich dualne $(n,k) = (2^m-1,m)$ i kody generowane przez wielomiany niepierwotne należą do klasy kodów BCH. Kody te opisuje się za pomocą pierwiastków wielomianów generujących; pierwiastki te można wyrazić przez potęgi elementu α, będącego pierwiastkiem dowolnego wielomianu pierwotnego stopnia m. W szczególności α i wszystkie jego potęgi aż do 2^m-1 są pierwiastkami x^n+1 gdzie $n = 2^m-1$.

Kody BCH są generowane przez $g(x)$, którego pierwiastkami są α^{m_o}, $\alpha^{m_o+1}, \ldots, \alpha^{m_o+d-2}$; są to kody o długości $n = 2^m-1$. Inną klasą kodów BCH są kody generowane przez $g(x)$ o pierwiastkach $\beta^{m_o}, \beta^{m_o+1}, \ldots, \beta^{m_o+d-2}$ gdzie β - pierwiastek wielomianu stopnia m nierozkładalnego niepierwotnego; ciągi takiego kodu mają długość n taką, że $x^n=1$, $n|(2^m-1)$. m_o może przyjmować wartości $0,1,2,\ldots$. Tak zdefiniowane kody mają $d_{min}=2t+2$ dla $m_o=0$ lub $2t+1$ dla $m_o=1,2,\ldots$; dowód tego oparty jest na właściwościach wyznacznika Vandermonde'a [4]. Proces lokalizowania błędów w ciągach kodów BCH, oparty na metodzie Petersona [1], jest dość złożony: wymaga wyznaczenia wartości badanych wielomianów dla pierwiastków wielomianu generującego, badania na osobliwość wyznacznika stopnia t, rozwiązania układu równań liniowych o t niewiadomych i rozwiązania 1 równania nieliniowego stopnia t. Modyfikacja tej metody, p. np. [5], prowadzą do pewnych uproszczeń ale zawsze konieczne jest znajdowanie wartości $v_b(x)$ dla pierwiastków $g(x)$ oraz badanie wyznaczników, często wielokrotne. W każdym przypadku konieczna jest operacja dzielenia $v_b(x)$ przez $g(x)$. Operacja ta realizowana jest w n taktach, gdzie n jest długością kodu.

Warto tu wspomnieć, że istnieje uproszczona metoda lokalizacji błędów w kodach cyklicznych [5] ale metoda ta - atrakcyjna ze względu na dużą



Rys. 4. Układ umożliwiający lokalizowanie błędów w $v_b(x)$ na podstawie porównywania generowanych w $R_1, R_2, \ldots, R_t$ reszt $r_1, r_2, \ldots, r_t$ wielomianów błędów z resztą r_b, znajdującą się w R_b. RJ - rozdzielacz jedynek, R_i - rejestr z liniowym sprzężeniem zwrotnym do wyznaczania reszty z wielomianu dzielonego przez wielomian generujący kod

szybkość działania i nieskomplikowane oprzyrządowanie - nie pozwala na pełne wykorzystanie korekcyjnych zdolności kodów BCH. Wady tej nie posiada metoda, którą można nazwać komparatorową; jej przedstawienie jest zasadniczym celem niniejszej pracy. Istota tej metody sprowadza się do porównywania syndromu badanego wielomianu, a więc $Rg[v_b(x)]$, z syndromami kolejno generowanych błędów. W hardwarowo najprostszym przypadku sprowadzałoby się to do budowy t układów dzielących przez g(x) (tj. przez wielomian generujący kod) i układu generującego impulsy taktujące o częstotliwościach fo, fo/n, fo/n^2,..., fo/n^{t-1}.

Odpowiedni układ przedstawiony jest na rys. 4. Jego praca wyglądałaby następująco: w momencie wystąpienia pierwszego taktu do pierwszej komórki układu R_1, tj. układu do wyznaczania reszty z podzielenia przez g(x), zostaje wpisana jedynka; począwszy od tego momentu rejestr pracuje autonomicznie, podając po każdym takcie stan, jaki otrzymalibyśmy dzieląc pojedynczy błąd przesuwający się po każdym takcie o 1 miejsce w prawo; jest to efektem zależności:

$$x.Rg[e(x)] = Rg[x.e(x)], \tag{18}$$

gdzie e(x) jest wielomianem błędów, a Rg jest resztą z podzielenia wielomianu zawartego w nawiasach przez g(x). W czasie odpowiadającym n taktom zostają więc wygenerowane syndromy wszystkich pojedynczych błędów. W (n+1)-ym takcie w R_1 występuje jedynka w pierwszej komórce i jednocześnie zostaje podana jedynka do pierwszej komórki rejestru R_2, taktowanego z częstotliwością n razy mniejszą w stosunku do częstotliwości taktowania rejestru R_1, a więc z częstotliwością fo/n.

Licząc od momentu wystąpienia (n+1)-go taktu - po n^2 taktach zostają wygenerowane syndromy wszystkich podwójnych błędów. Kontynuując tę myśl stwierdzimy, że w ciągu następnych $n^3, n^4, \ldots, n^t$ taktów zostają wygenerowane syndromy wszystkich błędów odpowiednio o krotności 3,4,...,t.

Błędy, których syndromy są w omawianym układzie wyznaczane, można zapisać w postaci macierzy o t wierszach

$$A = \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \text{wiersz} \\ \begin{bmatrix} A_1 & A_2 & \cdots & A_{t-1} & A_t \\ 0 & & & & \\ 0 & 0 & & & \\ \cdot & \cdot & & & \\ \cdot & \cdot & & & \\ \cdot & \cdot & & & \\ 0 & 0 & \cdots & 0 & \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ \cdot \\ \cdot \\ \cdot \\ t \end{array} \tag{19}$$

gdzie:

$$A_1 = [1 \;\; 2 \;\ldots\; n] \tag{20}$$

$$A_2 = \begin{bmatrix} A_1 & A_1 & \dots & A_1 \\ 1 & 2 & \dots & n \end{bmatrix} \tag{21}$$

$$A_3 = \begin{bmatrix} A_2 & A_2 & \dots & A_2 \\ 1 & 2 & \dots & n \end{bmatrix} \tag{22}$$

$$\dots\dots\dots$$

$$A_t = \begin{bmatrix} A_{t-1} & A_{t-1} & \dots & A_{t-1} \\ 1 & 2 & \dots & n \end{bmatrix} \tag{23}$$

przy czym w macierzy A oraz w macierzach $A_2, A_3, \dots$ symbole $0,1,2,\dots$ oznaczają macierze złożone odpowiednio wyłącznie z elementów $0,1,2,\dots$ Tak np. dla $t=2$ byłoby:

$$A = \begin{bmatrix} A_1 & A_2 \\ 0 & \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & 2 & \dots & n \end{bmatrix}$$

$$A_2 = \begin{bmatrix} A_1 & A_2 & \dots & A_1 \\ 1 & 2 & \dots & n \end{bmatrix} = \begin{bmatrix} 1\ 2\dots n & 1\ 2\dots n & \dots & 1\ 2\dots n \\ 1\ 1\dots 1 & 2\ 2\dots 2 & \dots & n\ n\dots n \end{bmatrix}$$

Każdy z elementów a_{ij} macierzy A - po jej rozpisaniu - określa numer błędnej pozycji w ciągu pojedynczych błędów. Każdy z tych ciągów - wyrażony w postaci wielomianu - jest dzielony w R_1 przez wielomian generujący $g(x)$. Odpowiednie reszty z podzielenia, por. rys. 4, są po każdym takcie sumowane mod 2 i porównywane z resztą znajdującą się w rejestrze R_b, to jest z resztą otrzymaną z podzielenia wielomianu badanego $v_b(x)$ przez $g(x)$.

Procedura dekodowania kończy się w momencie, kiedy wynik porównania jest równy zero, co wykryte przez element NOR powoduje zablokowanie generatora impulsów taktujących. Taka procedura generowania błędów jest bardzo prosta w realizacji, ale liczba taktów potrzebna do wygenerowania wszystkich kombinacji błędów o krotności $1,2,\dots,t$ jest raczej duża, bo wynosi $n^t+n^{t-1}+\dots n$. Jeśli założyć równomierny rozkład t błędów w ciągu kodowym o długości n, to można szacować, że średnia liczba taktów potrzebna do dekodowania będzie równa $^1/2\ n^t$.

Nietrudno zauważyć, że przy omówionym sposobie generowanie błędów duża ilość kombinacji błędów powtarza się, z czym związany jest zbędny czas testowania. Przy pewnej komplikacji układowej można uniknąć powtarzania co - przy założeniu jak wyżej - prowadzi do potrzeby generowania średnio $^1/2 \binom{n}{t}$ impulsów. Liczba ta może jeszcze okazać się za duża, szczególnie

w przypadkach kiedy lokalizowanie błędów ma odbywać się z niedużym opóźnieniem. W efekcie o użyteczności zaprezentowanej metody lokalizowania błędów decyduje - przy danym n i t - szybkość przekazywania informacji i organizacja jej wymiany między stacją nadawczą i odbiorczą.

LITERATURA

[1] W.W. Peterson, E.J. Weldon: Error Correcting Codes, The M.I.T. Press, 1970, ss. 9, 39-41, 117-121, 283-307.

[2] B. Gleichgewicht: Elementy algebry abstrakcyjnej. PZWS Warszawa 1966.

[3] S.W. Golomb: Shift Register Sequencies, Holden-Day, Inc. 1967, ss. 37-41.

[4] A. Mostowski, M. Stark: Elementy algebry wyższej. PWN, Warszawa 1958, ss. 102-103.

[5] Z. Szwaja: Realizacja binarnych kodów BCH, Politechnika Poznańska, Seria "Rozprawy", Poznań 1968, ss. 17-26, 48-56.

О СЛОЖНОСТИ СВЯЗАННОЙ С РЕАЛИЗАЦИЕЙ ИЗБЫТОЧНЫХ КОДОВ

Р е з ю м е

Главной целью настоящей работы является представление нового метода декодирования избыточных циклических кодов, особенно кодов БЧХ. Этот метод основан на генерировании синдромов многочленов ошибок и их сравнении с синдромом изучаемого многочлена. Метод реализуется легко, но время, необходимое для исправления ошибок при большой длине кодовых слов и при большом числе исправляемых кодом ошибок, может быть большим, а это может влиять на организацию обмена информацией.

ON COMPLEXITY CONNECTED WITH REDUNDANT CODES REALIZATION

S u m m a r y

First this paper presents an unformalized way of selecting binary sequences with a prescribed Hamming distance, i.e. of forming redundant codes. By means of this method, the nonlinear codes can be synthetized; it is shown that in such cases the redundant positions of the code sequences are nonlinear boolean functions of the information digits, this being of no practical use in correcting the erronous positions of the tested sequences. In the effect, decoding is possible only by comparing of the received sequence with the stored code words; this leads to the maximum likelihood decoding method.

It turns out that by applying the algorithm shown in Fig 1, the synthetized codes are in some cases linear. These codes are outlined in par. 2 of the paper. Two extreme classes of linear codes are shown for illustration: the one-error-correcting closely-packed Hamming codes $(n,k) = (2^m-1, 2^m-1-m)$, and their duals $(n,k) = (2^m-1,m)$ characterized, for a given m, by the largest possible, and the same, weight of all the code words. The implementation of these codes can, in principle, be based on logical elements so that decoding takes place in one step. This way of decoding is very attractive because of its simplicity but it is justifiable only for codes similar to Hamming codes; it is so because of the bulk of elements needed for the circuitry, see Fig 3, which grows rapidly with the growing number of correctable errors. In the effect, the simple method implemented as shown in Fig 2 turns out to be noteworthy for the case of rather small number of code words, this refers for inst. to the codes dual the Hamming codes.

The main part of the paper refers to the BCH codes which belong to the class of cyclic codes, and which comprise good many of codes as for inst. the above mentioned Hamming codes and their duals. One can say that the decoding procedure of these codes given by Peterson [1], and all modifications of this procedure are sohpisticated, and require a lot of algebraic operations; thus, o search for simpler methods is plausible. In the paper a new (the best, in author's conniction) decoding method is presented. It resolves itself to comparison of the syndrome of the tested polynomial with syndromes of sequentially generated error polynomials. The generation of the syndromes of errors is simple because of the relation:

$$R_g[x.e(x)] = x.R_g[e(x)],$$

where

$e(x)$ = error polynomial,

$R_g[f(x)] = f(x) \bmod g(x)$.

The number and types of circuits needed for implementation of the methed was given. The number of steps necessary for completing of the decoding procedure was estimated; as this number may be large, some hints referring to the applicability of the presented method are given.