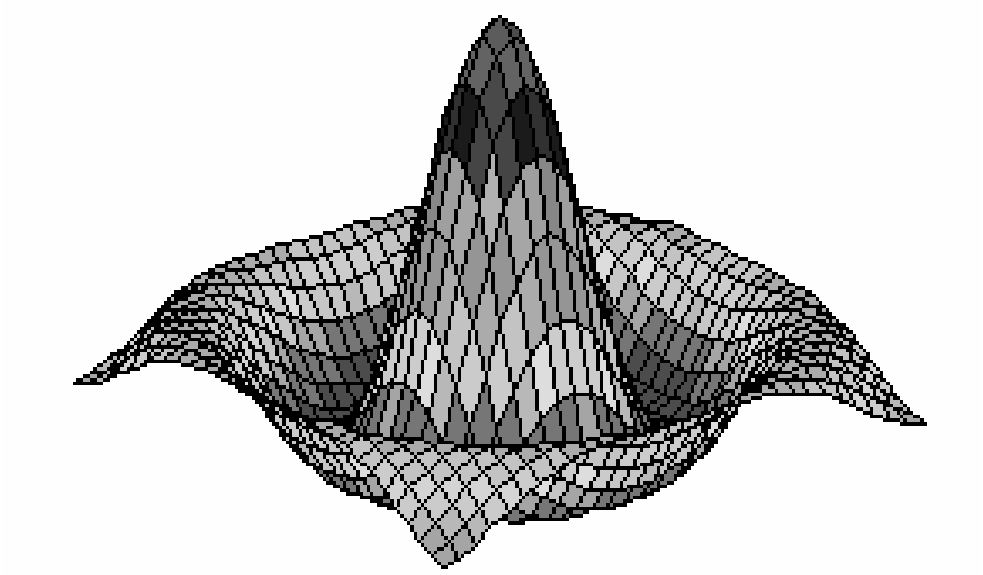
SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ZAVOD ZA TELEKOMUNIKACIJE

Matija MIKAC



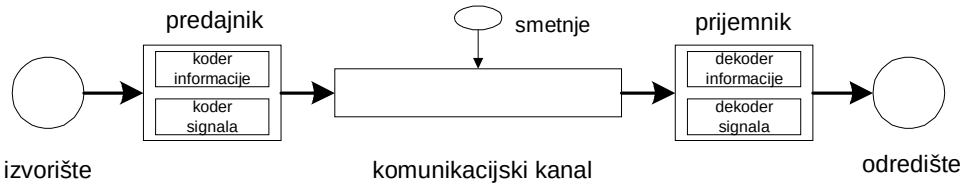
LABORATORIJSKE VJEŽBE

Teorija informacije

ZAŠTITNO KODIRANJE

Informacijski komunikacijski sustavi

Općeniti informacijski komunikacijski sustav prikazan je na slici 1. Takav sustav uključuje izvorište informacije, predajnik, komunikacijski kanal, prijemnik i odredište informacije. Izvorište generira informacije, te mu stoga pridajemo posebnu pažnju. Svaka generirana informacija sastavljena je od slijeda ulaznih simbola sadržanih u izvorišnoj abecedi. Svaki simbol izvorišne abecede javlja se unutar informacije s određenom učestalošću. Prije prijenosa, informacija se kodiraju – ulazni simboli prikazuju se korištenjem simbola iz kodne abecede (u našem slučaju, binarni simboli).
Budući da je svaki komunikacijski sustav podložan smetnjama, postoji mogućnost gubitka izvorišne informacije uslijed smetnji u komunikacijskom kanalu. Stoga je nužno omogućiti provjeru informacije na odredištu i osigurati mehanizam detekcije pogrešaka. Isto tako, ako je moguće, osigurati i ispravljanje pogreške. Da bi se to postiglo, namjerno se dodaje redundantna informacija u izvorišni slijed. Kodovi koji se koriste za to nazivaju se **zaštitnim kodovima**. Neki od zaštitnih kodova koriste se samo za otkrivanje, a neki i za otkrivanje i za ispravljanje pogrešaka.



Slika 1 : Informacijski komunikacijski sustav

Kao i optimalno kodiranje, tako se i zaštitno kodiranje obavlja se u dijelu predajnika koji se zove koder informacije. Osim kodera informacije predajnik uključuje i koder signala koji kodira prijenosni signal kako bi ga prilagodio prijenosnom mediju. Informacije koje dolaze na odredište potrebno je stoga prije prihvaćanja dekodirati, a to obavljaju dekoder signala i dekoder informacije koji se nalaze u prijemniku. Pri dekodiranju informaciju u dekoderu informacija moguće je otkriti i u određenom slučaju ispraviti pogreške.

Zaštitni kodovi

Za vrijeme prijenosa informacija postoji mogućnost njihova gubitka ili promjene sadržaja. Zbog toga je potrebno osmisliti način kako registrirati, te, ukoliko je to moguće, ispraviti pogrešku koja se na taj način može pojaviti. Za to koristimo zaštitne kodove – kodove za otkrivanje i kodove za ispravljanje pogrešaka. Često se, umjesto kodova za ispravljanje pogrešaka, u praksi koriste kodovi za otkrivanje pogrešaka u kombinaciji s tehnikama ponovnog odašiljanja (npr. ARQ – *Automatic Repeat reQuest*) pogrešno prenesenih informacija. No, u tom slučaju se ne radi o zaštitnom kodiranju s mogućnošću ispravljanja pogrešaka, već isključivo o tehnici prijenosa podataka!

Promatrajmo slučaj kad je informacija već binarno kodirana i prikazana slijedom bitova. Slijed bitova dolazi do predajnika, signal se kodira i prenosi prema prijemniku kroz prijenosni kanal. Kroz prijenosni kanal informacija se prenosi električnim vodičem (bakar) ili svjetlovodom (optička nit). Uslijed vanjskih utjecaja (smetnje, šum, kvarovi opreme..) može doći do promjene i

izobličenja signala, što na prijemniku, nakon dekodiranja signala, rezultira slijedom bitova različitim od ulaznog. Drugim riječima, pri prijenosu je došlo do promjene informacije ! Postavlja se pitanje kako komunikacijski sustav može registrirati pogrešku i da li je moguće ispraviti takvu pogrešku?! Postoji nekoliko vrsta kodova za otkrivanje i ispravljanje pogrešaka. Općenito ih možemo podijeliti na **kodove sa kontrolom pariteta** i **kodove sa ponavljanjem**. Kod kodova sa kontrolom pariteta dodaju se zaštitni bitovi koji poprimaju vrijednost ovisno o pravilu kodiranja, a određuju se primjenom *modulo* 2 aritmetike, što u stvari znači da se radi o provjeri pariteta. Kodovi s ponavljanjem višestruko uključuju informacijske bitove, čime se smanjuje mogućnost pogrešnog dekodiranja – odluka se svodi na većinsko odlučivanje.

Kodovi s kontrolom pariteta

Pri kodiranju kodom s kontrolom pariteta, na informaciju koja se sastoji od k informacijskih simbola dodaje se m paritetnih simbola prema unaprijed definiranom pravilu kodiranja. U tom slučaju se komunikacijskim kanalom prenosi $n = k + m$ simbola (bitova). Takav kôd nazivamo (n,k) blok-kôd. Naziv blok-kôd proizlazi iz dodavanja zaštitnog bloka bitova informacijskoj riječi.

Princip funkcioniranja paritetnih kodova prikazat ćemo na primjeru.
Izvorišna informacija duljine 4 bita prikazuje se vektorom informacijskih bitova $\boldsymbol{i}=[i_1\ i_2\ i_3\ i_4]$.Da bi omogućili otkrivanje pogreške i zaštitu informacije, dodajemo npr. tri zaštitna bita p_5, p_6 i p_7, koji se određuju prema pravilu kodiranja. Pravilo možemo proizvoljno definirati. Npr.

$$p_5 = i_2 + i_3 + i_4 \pmod 2$$
$$p_6 = i_1 + i_3 + i_4 \pmod 2$$
$$p_7 = i_1 + i_2 + i_4 \pmod 2$$

Preuređenjem pravila dobijemo tzv. paritetne jednadžbe :

$$0 + i_2 + i_3 + i_4 + p_5 + 0 + 0 = 0$$
$$i_1 + 0 + i_3 + i_4 + 0 + p_6 + 0 = 0$$
$$i_1 + i_2 + 0 + i_4 + 0 + 0 + p_7 = 0$$

Nakon dodavanja zaštitnih bitova kodna riječ je vektor redak $\boldsymbol{c} = [\ i_1\ i_2\ i_3\ i_4\ p_5\ p_6\ p_7\ ]$.
Paritetne jednadžbe matrično prikazujemo *matricom pariteta* **A**. *Generirajuća matrica* **G** je oblika **G =** [**I; A**]. U našem primjeru vrijedi:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

U matričnom obliku pravilo kodiranja možemo prikazati : $\boldsymbol{c} = \boldsymbol{i}\cdot \mathbf{G}$.
Osim *generirajuće matrice* **G** i *matrice pariteta* **A**, definiramo i *kontrolnu matricu* **H**. Ona je definirana kao $\mathbf{H} = [\mathbf{A}^T;\mathbf{I}]$. Transponirana kontrolna matrica $\mathbf{H}^T$ ima oblik $\mathbf{H}^T = [\mathbf{A};\mathbf{I}]^T$.

Jedno od svojstava kontrolne matrice je da za svaku ispravnu kodnu riječ $\boldsymbol{c}$ vrijedi $\boldsymbol{c}\cdot\mathbf{H}^T=[0\ 0\ 0]^T$!

Kodnu riječ **c** predajnik šalje kroz prijenosni kanal. Prijemnik prima informaciju **r** = [r_1 r_2 r_3 r_4 r_5 r_6 r_7]. Ukoliko je u prijenosu došlo do pogreške, vektor **r** će biti različit od **c**. Vektor **r** možemo prikazati kao zbroj vektora **c** i vektora pogreške **e**, **r = c + e**. Vrijednost pojedinog elementa $\mathbf{e}_i$ u vektoru pogreške **e** je 1 ukoliko je na *i*-tom bitu došlo do pogreške, a 0 ukoliko na i-tom bitu nema razlike između **c** i **r**.

Dakle, primljena kodna riječ prikazana vektorom **r**, dekodira se na odredištu. Ukoliko je došlo do pogreške u prijenosu, razlikovati će se od poslane kodne riječi **c**, što točno definira vektor pogreške **e**. Pitanje otkrivanja i ispravljanja pogreške svodi se na postupak određivanja vektora pogreške **e**. Ukoliko uspijemo točno odrediti vektor **e** na odredištu, možemo sa sigurnošću odrediti sadržaj poslane informacije.

Sindrom **S** definiramo matrično kao :

$$\mathbf{S} = \mathbf{r}\mathbf{H}^T = (\mathbf{c}+\mathbf{e})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T \qquad (4)$$

Sindrom ovisi samo o vektoru pogreške **e**. Ukoliko se u prijenosu ne dogodi pogreška sindrom će biti jednak nul-vektoru. Jednadžbu $\mathbf{S} = \mathbf{e}\mathbf{H}^T$ ne možemo analitički riješiti, već se koriste različiti pomoćni algoritmi. Jedan od algoritama dan je u knjizi prof. Sinkovića [2].

Hammingova distanca ili distanca kôda :

$d_H(\mathbf{x},\mathbf{y})$ = { broj elemenata za koje je $x_i \neq y_i$; $w_H(\mathbf{y}-\mathbf{x})$ }

gdje je w_H Hammingova težina odnosno broj ne-nul elemenata u vektoru. Distanca između dva vektora se može zapisati kao Hammingova težina njihove razlike.

Minimalna distanca d_{min} (K) blok-koda je najmanja Hammingova distanca.

Korištenjem gore navedenih pojmova možemo definirati uvjete koje kôd mora zadovoljiti kako bi bilo omogućeno ispravljanje. Kôd K = { $\mathbf{c}_i$ }, $i = 0,1,2,..,2^k-1$, može ispraviti sve oblike pogrešaka s težinom manjom ili jednakom *w* onda i samo onda ako je :

$w \leq (d_{min}(K)-1)/2$

Hammingov kôd

Poseban slučaj paritetnog zaštitnog koda je Hammingov kôd. Ukoliko koristimo *m* zaštitnih paritetnih bitova, duljina Hammingova koda je $n=2^m-1$, a broj informacijskih bita je $k=n-m=2^m-1-m$. Kontrolna matrica je dimenzija *m* x 2^m-1. Općenito, Hammingovi kôdovi mogu biti (7,4), (15,11), (31,26), (63,57), itd.
Posebnost Hammingovog koda očituje se jednostavnošću određivanja sindroma.
Ako je $w_H(\mathbf{e})=1$ (dakle, u vektoru pogreške imamo samo jednu jedinicu, tj. samo jednu pogrešku), tada je $\mathbf{S} = \mathbf{e}\mathbf{H}^T$ = *i*-ti stupac od **H**, pa treba izvesti promjenu na *i*-tom mjestu u primljenoj riječi. Ukoliko rasporedimo stupce u kontrolnoj matrici **H** prema prirodnom binarnom redoslijedu, olakšano nam je dekodiranje jer dekadski ekvivalent sindroma određuje indeks mjesta na kojem treba ispraviti pogrešku.

Hammingov kôd može ispraviti jednostruku pogrešku.

Primjer 1 – kodiranje slijeda bita Hammingovim kodom

Ulazni slijed bitova : 0101110111

Postupak kodiranja (skraćeni postupak):
Dodajemo zaštitne bitove, pozicije su redom 2^0 (1), 2^1 (2), 2^2 (4), 2^3 (8), itd. Zaštitne bitove označimo prema njihovim pozicijama sa h_1, h_2, h_4 i h_8.. Zanemarujemo bitove 'manjka' (do potpunog (15,11) koda). Slijed bitova nakon kodiranja ima oblik:

h_1	h_2	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
x	x	0	x	1	0	1	x	1	1	0	1	1	1

Zaštitne bitove h_1, h_2, h_4 i h_8 određujemo kodiranjem prema pravilu :

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
$\boldsymbol{h_1}$	h_2	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	$\boldsymbol{h_2}$	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	$\boldsymbol{h_4}$	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	h_4	b_2	b_3	b_4	$\boldsymbol{h_8}$	b_5	b_6	b_7	b_8	b_9	b_{10}

Određivanje pravila kodiranja vezano je uz binarni zapis pojedine lokacije. Npr. h_1 kontrolira sve pozicije kod kojih binarni zapis sadrži jedinicu na najmanje značajnom mjestu (...1 – sve neparne pozicije!), h_2 sve pozicije koje sadrže jedinicu na drugom najmanje značajnom mjestu (...1x – npr. pozicije 2,3,6), h_4 sve koje sadrže jedinicu na trećem mjestu (...1xx – npr. pozicije 4,5,6) itd.

Kodiranje se svodi na *modulo 2* zbrajanje, pa zaštitni bitovi poprimaju vrijednosti :

$$h_1 = b_1 \oplus b_2 \oplus b_4 \oplus b_5 \oplus b_7 \oplus b_9 \quad \text{(mod 2 zbroj)},$$
$$h_2 = b_1 \oplus b_3 \oplus b_4 \oplus b_6 \oplus b_7 \oplus b_{10} \quad \text{(mod 2 zbroj)},$$
$$h_4 = b_2 \oplus b_3 \oplus b_4 \oplus b_8 \oplus b_9 \oplus b_{10} \quad \text{(mod 2 zbroj)},$$
$$h_8 = b_5 \oplus b_6 \oplus b_7 \oplus b_8 \oplus b_9 \oplus b_{10} \quad \text{(mod 2 zbroj)},$$

a u našem primjeru :

$$h_1 = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0$$
$$h_2 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 1$$
$$h_4 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1$$
$$h_8 = 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 = 1$$

Dakle, kodirani slijed bitova je **01**0**1**101**1**110111, odnosno prikazano tablično:

h_1	h_2	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
0	**1**	0	**1**	1	0	1	**1**	1	1	0	1	1	1

Alternativno, kodiranje se može obaviti korištenjem generirajuće matrice **G** općeg oblika **G=[I;A]**, gdje je A matrica pariteta. Matrica pariteta **A** označava paritete koje 'prate' pojedini zaštitni bit. Matricu pariteta generiramo iz paritetnih jednadžbi i to tako da njeni stupci sadržavaju paritete za pojedine zaštitne bitove. U našem primjeru (u stvari se radi o skraćenom kodu koji bi trebalo nadopuniti do (15,11) jednom nulom) matrica pariteta ima četiri stupca (četiri paritetna zaštitna bita) i 10 redaka (10 informacijskih bitova). Matrica pariteta i generirajuća matrica za naš primjer su dane u nastavku.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Jedina razlika u odnosu na prije prikazani način određivanja paritetnih bitova jest da se ti bitovi smještaju na kraj riječi.
Kodirana riječ se određuje kao **c = b*G**, gdje je **b** ulazna riječ (**b = [**0 1 0 1 1 1 0 1 1 1]. U našem primjeru rezultat je **c =** [0 1 0 1 1 1 0 1 1 1 **0 1 1 1**]. Očito je da su vrijednosti kontrolnih bitova identične!

Primjer 2 – dekodiranje slijeda bita i ispravljanje pogrešaka Hammingovim kodom

Proučimo slučaj jednostruke pogreške. Neka na dekoder dolazi slijed **01**0**1**101**1**110<u>0</u>11. Greška se pojavila na bitu b_8 (12 bit).

Kao i kod kodiranja, određujemo zaštitne bitove h_1', h_2', h_4' i h_8'. Ukoliko je došlo do greške oni će se razlikovati od primljenih **0111** (h_1, h_2, h_4, h_8).

0	**1**	0	**1**	1	0	1	**1**	1	1	0	0	1	1
$\boldsymbol{h_1}$	h_2	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	$\boldsymbol{h_2}$	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	$\boldsymbol{h_4}$	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	h_4	b_2	b_3	b_4	$\boldsymbol{h_8}$	b_5	b_6	b_7	b_8	b_9	b_{10}

$h_1' = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 =$ **0** - ispravno **0**
$h_2' = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 =$ **1** - ispravno **0**
$h_4' = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 =$ **0** - pogrešno **1**
$h_8' = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 =$ **0** - pogrešno **1**

Kad uočimo razliku između poslanih i proračunatih zaštitnih bitova dobijemo vektor pogreške 1100 (od najvišeg h_8' prema h_1'!), što je decimalno 12 i odgovara rednom broju pogrešnog bita. Promjenom 12 bita u 0 dobijemo ispravan slijed ! To je primjer detekcije i ispravljanja jednostruke pogreške.

Drugi način dekodiranja bio bi određivanjem sindroma prema $\mathbf{S} = \mathbf{rH}^T$, gdje je **r** primljena riječ, a **H** kontrolna matrica definirana sa $\mathbf{H} = [\mathbf{A}^T;\mathbf{I}]$.

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

Bitno je napomenuti da se primljena riječ prema danom **H** razlikuje – zaštitni paritetni bitovi se nalaze na zadnja četiri bita, pa je **r** u stvari 0101110<u>0</u>11**0111**.
Množenjem matrica dobivamo (*modulo 2*) dobivao sindrom **S =[0 0 1 1]**. Da bi ispravili pogrešku moramo naći stupac u kontrolnoj matrici koji je jednak sindromu. To je označeni stupac – 8-stupac. Dakle, nužno je ispraviti osmi bit pa imamo 0101110<u>1</u>111**0111**, odnosno u prijašnjoj notaciji **01**0**1**1011**1**10<u>1</u>11.

Oba načina dekodiranja i ispravljanja daju isti rezultat!

Primjer 3 – Dekodiranje niza u kojem nema pogreške

U slučaju da nema pogreške u prijenosu – slučaj kad na dekoder dolazi ispravan slijed **01**0**1**1011**1**110111, proračunati zaštitni bitovi h_1', h_2', h_4' i h_8' ne razlikuju se od onih u primljenom slijedu. Dakle, vektor pogreške je 0000, što znači da nema pogreške.

Drugim postupkom dobiva se **S =** [0 0 0 0], što također naznačuje da nema pogreške!

Primjer 4 – Višestruke pogreške

Problemi mogu nastati u slučaju višestruke pogreške. Npr. neka na dekoder dolazi slijed sa dvije pogreške **01**0**1**100**1**110<u>0</u>11 010<u>0</u>11**0**0<u>1</u>1**0**1**0**. Odredimo zaštitne bitove h_1', h_2', h_4' i h_8'.

0	**1**	0	**1**	1	0	0	**1**	0	1	0	0	1	1
$\boldsymbol{h_1}$	h_2	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	$\boldsymbol{h_2}$	b_1	h_4	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	$\boldsymbol{h_4}$	b_2	b_3	b_4	h_8	b_5	b_6	b_7	b_8	b_9	b_{10}
h_1	h_2	b_1	h_4	b_2	b_3	b_4	$\boldsymbol{h_8}$	b_5	b_6	b_7	b_8	b_9	b_{10}

$h_1' = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = \mathbf{1}$ - pogrešno **1**
$h_2' = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = \mathbf{1}$ - ispravno **0**
$h_4' = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = \mathbf{1}$ - ispravno **0**
$h_8' = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = \mathbf{0}$ - pogrešno **1**

Vektor pogreške je sad 1001, što je decimalno 9. Kad bi promijenili vrijednost 9. bita (b_5), dobili bi slijed **01**0**1**100**1***1*010<u>0</u>11. Zaštitni bitovi bili bi :

$h_1' = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = \mathbf{0}$ - ispravno **0**
$h_2' = 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = \mathbf{1}$ - ispravno **0**
$h_4' = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = \mathbf{1}$ - ispravno **0**
$h_8' = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = \mathbf{1}$ - ispravno **0**

Nul-vektor pogreške bi značio da nema pogreške. Ona doista u takvom slijedu ni ne postoji, međutim ispravljeni slijed ima čak 3 pogreške! To ukazuje na nemogućnost ispravljanja višestrukih pogrešaka Hammingovim kodom! Detekcija je moguća, no i to ovisi o tome o kakvim se pogreškama radi.

Hammingov kôd može se koristiti za ispravljanje jednostrukih pogrešaka u kodnom slijedu, te za detekciju dvostrukih pogrešaka! Detekcija višestrukih (3 i više) pogrešaka nije moguća!

Hammingovi kodovi su posebna vrsta blok kodova. BCH blok kodovi, kao šira skupina blok kodova, mogu se koristiti i za ispravljanje i detekciju višestrukih pogrešaka!

Ciklički kodovi

Druga vrsta zaštitnih kodova s kontrolom pariteta koje ćemo analizirati su ciklički kodovi. Često korišten naziv je CRC (*Cyclic Redundancy Code*). Ciklički kôd je blok kôd koji dodaje zaštitni blok paritetnih bitova na kraj slijeda informacijskih bitova. Za svaki ciklički kôd vrijedi : ako je kodna riječ oblika **c** = [c_1 c_2 ... c_{n-1} c_n], onda i kodne riječi nastale cikličkim pomicanjem (npr. [c_2 c_3 ... c_n c_1] i [c_3 c_4 ... c_n c_1 c_2]) sadrže kodirane informacije. Dakle, ukoliko informacijsku riječ **i** kodiramo (dodajemo zaštitni paritetni blok **p**) i dobijemo kodnu riječ **c** oblika **c** = (**i p**), onda možemo tvrditi da će i ostale informacijske riječi iz ulazne abecede, nakon kodiranja, biti kodirane kodnim riječima koje nastaju cikličkim pomicanjem (*shift*) kodne riječi **c**. Jednostavna analiza tog osnovnog svojstva omogućava nam da zaključimo glavne karakteristike generirajuće matrice **G** za cikličke kodove.
Budući da je zaštitni blok bitova definiran prema paritetima informacijske riječi, informacija 'nula' (sadrži sve 0 bitove) će i nakon kodiranja imati oblik nul-vektora (paritetni bitovi su na 0). Kao što je prethodno opisano, generirajuća matrica **G** ima oblik [**I**;**P**], a svaka kodna riječ dobiva se vektorski kao **c** = **iG**. Posljednji redak generirajuće matrice **G** oblika [0 0 ... 1 | ... 0] je u stvari oblik kodne riječi koji pokazuje kako bi izgledalo kodiranje informacijske riječi koja sadrži sve nul bitove osim najmanje značajnog ('najdesnijeg') bita. Osnovno svojstvo cikličkog koda govori o definiranju kodnih riječi pomakom bitova postojeće. U slučaju kodne riječi prikazanog oblika, to bi značilo da pomicanjem udesno dobivamo novu kodnu riječ oblika [0 0 ... 0 | 1 0 ... 0], a to ne bi bilo u skladu s pravilima i svojstvima paritetnih kodova, koja govore da se 'informacijska nula' kodira 'kodnom nulom'. Dakle, svi zaštitni bitovi u slučaju nul-vektora također su jednaki nuli. Zbog toga zaključujemo da posljednji redak generirajuće matrice **G** mora na posljednjem bitu imati vrijednost 1. Na taj način, definiramo generirajuću matricu **G** općeg oblika :

$$\mathbf{G} = \left[\begin{array}{ccccc|ccc} 1 & 0 & 0 & \cdots & 0 & \cdots & & \\ 0 & 1 & 0 & \cdots & 0 & \cdots & & \\ 0 & 0 & 1 & & 0 & \cdots & & \\ \vdots & \vdots & \vdots & & \vdots & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & \cdots & & 1 \end{array}\right]$$

Cikličke kodove najčešće prikazujemo polinomima. Kodna riječ **c** = [c_1 c_2 ... c_n] prikazuje se polinomom oblika :

$$c(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-1} x + c_n \qquad (5)$$

U polinomu svaka potencija od *x* prikazuje bit pomaknut ulijevo. Matematički, pomak unutar kodne riječi definira se množenjem sa *x*, s tim da se koristi *modulo* $x^n + 1$ aritmetika (ciklički pomaci).

npr.

$$xc(x) \bmod(x^n + 1) = c_2 x^{n-1} + c_3 x^{n-2} \cdots + c_{n-1} x^2 + c_n x + c_1$$
$$x^2 c(x) \bmod(x^n + 1) = c_3 x^{n-1} + c_4 x^{n-2} \cdots + c_{n-1} x^3 + c_n x^2 + c_1 x + c_2$$

Na sličan način mogu se prikazati elementi generirajuće matrice – bit 1 zamjenjuje se odgovarajućom potencijom od *x*, dok se bitovi 0 ne uzimaju u obzir. Posljednji redak matrice **G**, zapisan u obliku polinoma, definira **generirajući polinom *g(x)***. Generirajući polinom u potpunosti definira karakteristike cikličkog koda. Opći oblik generirajućeg polinoma *g(x)*, što jednostavno proizlazi iz prijašnjeg opisa generirajuće matrice, je :

$$g(x) = x^{n-k} + \cdots + 1 \tag{6}$$

Bitno svojstvo generirajućeg polinoma za (*n*,*k*)-kôd je to da je on uvijek djeljitelj polinoma $x^n + 1$. To možemo zapisati kao :

$$g(x)\ h(x) = x^n + 1 \tag{7}$$

Za svaku informacijsku riječ možemo jednostavno odrediti kodnu riječ, tj. zaštitni blok, korištenjem generirajućeg polinoma *g(x)*. Postupak slijedi.

Koristimo li kôd (*n*,*k*), informacijska riječ **i** sastoji se od *k* informacijskih bitova. Nakon kodiranja, kodna riječ **c** se sastoji od *n* bitova – *k* informacijskih i *n-k* zaštitnih bitova. Karakteristična svojstva cikličkih kodova omogućuju jednostavno određivanje zaštitnog dijela koda tj. **p**. Zapišimo najprije informacijsku riječ u obliku polinoma *i(x)*.

$$i(x) = i_1 x^{k-1} + i_2 x^{k-2} + \cdots + i_{k-1} x + i_k \tag{8}$$

Može se pokazati da se svaka kodna riječ u obliku polinoma *c(x)* može prikazati kao funkcija informacijskog polinoma *i(x)*, generirajućeg polinoma *g(x)* i ostatka *r(x)*.

$$c(x) = x^{n-k} i(x) + r(x) \tag{9}$$

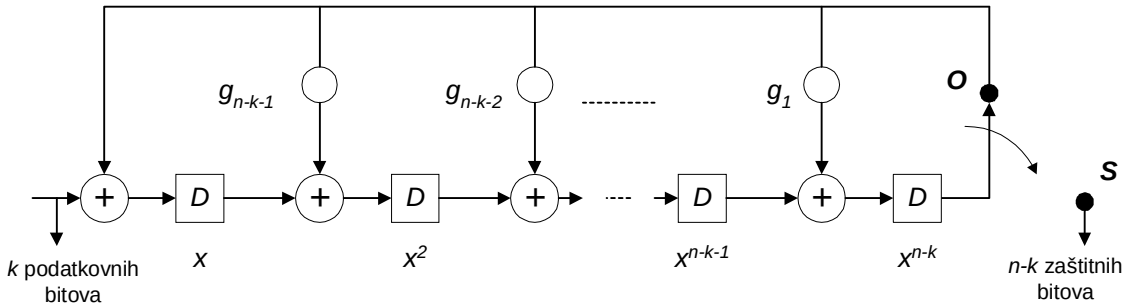
Ostatak *r(x)* prema tome predstavlja zaštitne bitove – prvi dio izraza za *c(x)* predstavlja pomak informacijskih bitova ulijevo (zauzimaju prvih *k* mjesta u kodnoj riječi). Dakle, polinom *r(x)* definira zaštitne bitove unutar kodne riječi. Općenito, *r(x)* izračunavamo iz izraza :

$$r(x) = \operatorname{rem} \frac{x^{n-k} i(x)}{g(x)} \tag{10},$$

gdje 'rem' označava ostatak dijeljenja.

Očito je da će oblik kodne riječi, odnosno njezina zaštitnog dijela *r(x)*, ovisiti o generirajućem polinomu *g(x)*.

Opisana svojstva cikličkih kodova omogućavaju relativno jednostavnu implementaciju sklopova za određivanje koda (slika 2). Koristi se $r=n-k$ posmačnih registara – svaki registar prikazan je elementom za kašnjenje *D*. Kako funkcionira sklop? *k* informacijskih bitova ulazi u sklop – sklopka je u položaju *O* i povratnom vezom sadržaj registara se mijenja. Elementi/vrata koji dovode signal u *modulo* 2 zbrajala (označeni sa g_k) ovise o strukturi generirajućeg polinoma – svaki element predstavlja određeni koeficijent u polinomu *g(x)* – vrijednosti mogu biti 0 ili 1, jer se radio o *mod* 2 operacijama. Drugim riječima, vrata će postojati samo ukoliko se pridruženi koeficijent nalazi u *g(x)*. Zbrajala koja se koriste su *modulo* 2 zbrajala i u praksi se izvode kao logička isključivo ILI (XOR) operacija. Nakon što svih *k* informacijskih bitova 'prođe' registrima, sklopka se postavlja u položaj *S*, te na izlazu dobivamo zaštitne bitove. Opisani sklop u principu obavlja, prije opisani, proračun ostatka *r(x)*.



Slika 2 : Shema sklopa za određivanje cikličkog koda

Opisani sklop služi za slijedno (serijsko) određivanje kodne riječi. Postoje i drugačije izvedbe, koje omogućuju paralelno određivanje kodne riječi, tj. istovremeno dohvaćanje svih *n-k* zaštitnih bitova.

Ciklički kodovi omogućavaju jednostavnu detekciju pogreške u informacijskom slijedu. Ispravljanje pogrešaka omogućuju posebni tipovi cikličkih kodova – npr. Reed-Solomonovi kodovi (RS kodovi).

Primjer 5. – Ciklički kod

Uzmimo kôd (7,3). Informacijska riječ u takvom kodu sadrži 3 bita, a zaštitni blok je duljine 4 bita. Kako odrediti generirajući polinom *g(x)* za taj kôd ?
Sjetimo se svojstva generirajućeg polinoma *g(x)* koje govori da je on uvijek djeljitelj polinoma x^n+1 (7). Rastavimo x^n+1 :

$$x^7+1 = (x+1)(x^3+x+1)(x^3+x^2+1)$$

Svojstvo (6) govori da je *g(x)* oblika $x^{n-k}+\ldots+1$. Dakle, u našem primjeru *g(x)* mora biti oblika $x^4+\ldots+1$. Vidimo da oba faktora x^3+x+1 i x^3+x^2+1, množena sa $x+1$ zadovoljavaju to svojstvo. Dakle, za kodove (7,3) generirajući polinomi mogu biti :

$$(x+1)(x^3+x+1)=x^4+x^2+x+x^3+x+1=x^4+x^3+x^2+1 \quad \text{i}$$

$$(x+1)(x^3+x^2+1)=x^4+x^3+x+x^3+x^2+1=x^4+x^2+x+1.$$

Uzmimo u ovom primjeru $g(x)=x^4+x^3+x^2+1$.

Jasno je da postupak određivanja karakterističnog polinoma za dulje kodove nije jednostavan. Stoga se u zadacima najčešće zadaje generirajući polinom ili neki drugi podatak koji omogućava njegovo jednostavno određivanje.

Odredimo sada kodnu riječ za informacijski riječ 001.

Informacijski polinom $i(x)=0\cdot x^2+0\cdot x+1$.

Izračunajmo *r(x)* prema (8).

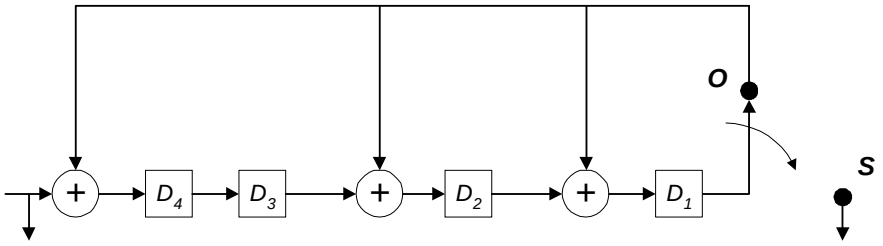
$$r(x)=\mathrm{rem}\frac{x^{n-k}i(x)}{g(x)}=rem\frac{x^4}{x^4+x^3+x^2+1}=x^3+x^2+1$$

Prikažemo li *r(x)* kao slijed bitova dobijemo zaštitni blok **r** = 1101.
Kodna riječ **c**, kojom kodiramo informaciju 001, je **c** = 001<u>1101</u>, gdje podcrtani dio predstavlja zaštitni blok.

Kako realizirati sklop koji obavlja generiranje (7,3) koda, sa definiranim generirajućim polinomom *g(x)*? Koristimo pravila opisana uz općeniti prikaz sklopa sa slike 2.

$$g(x)=x^4+x^3+x^2+1=x^4+g_1x^3+g_2x^2+g_3x+1 \rightarrow g_1=1,\ g_2=1,\ g_3=0$$

Nakon što smo odrediti parametre g_1, g_2, g_3, jednostavno ukinemo logička vrata za koje je parametar jednak nuli. U našem primjeru ostaju vrata g_1 i g_2 – vrata i *modulo* 2 zbrajala uz g_3 su nepotrebna. Sklop koji tražimo prikazan je na slici 3.



Slika 3 : Izvedba sklopa za određivanje cikličkog koda iz primjera

Provjerimo sada da li sklop doista obavlja svoju funkciju kako je opisano! Na ulaz sklopa dolazi informacijska riječ 001. Prikazat ćemo stanja registara prilikom ulaska informacija u sklop, te stanje pri preklapanju sklopke u položaj S nakon što posljednji informacijski bit dođe u registar D_1.

Registar	D_4	D_3	D_2	D_1
	0-	**0-**	**0-**	**0-**
1.	0	0-	0-	0-
2.	0	0	0-	0-
3.	**1**	0	0	0-
4.	0-	**1**	0	0
5.	0	0	**1**	0
6.	0	0	0	**1**
7.	**1**	**0**	**1**	**1**

Na izlasku iz sklopa dobijemo 1101 (prvo izlazi bit pohranjen u D_1), što je upravo sadržaj zaštitnog bloka bitova!

Dodatak – CRC standardizirani kodovi

Standardizirani ciklički kodovi CRC-16 i CRC-32 dodaju 16 odnosno 32 zaštitna bita na informaciju. Ti kodovi se često primijenjuju za detekciju promjene podataka na računalima (npr. primjena u aplikacijama koje provjeravaju status datoteke – anitvirusni programi, provjera ispravnosti arhive i informacija (*WinZip*)) Standardizirani polinomi za CRC-16 i CRC-32 kodiranje imaju oblik:

CRC-CCITT (16 bit, standardiziran po ITU-T)
$x^{16} + x^{12} + x^5 + 1$
CRC-16 (16 bit, IBM)
$x^{16} + x^{15} + x^2 + 1$
CRC-32
$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

ZADACI ZA PRIPREMU :

Prije rješavanja zadataka u Matlabu, studenti moraju riješiti zadatke za pripremu! Zadaci se predaju uz rješenja zadataka u Matlabu.

Zaštitni kodovi

Z-1. Pokažite princip funkcioniranja zaštitnog koda s ponavljanjem na proizvoljnom primjeru informacijskog komunikacijskog sustava! Objasnite na primjeru sve korake od generiranja i zaštitnog kodiranja informacije, prijenosa kroz kanal (utjecaj smetnji!), do detekcije i ispravljanja pogrešaka.

Z-2. Pokažite princip funkcioniranja zaštite informacija korištenjem paritetnih kodova – na proizvoljnoj informaciji izvedite zaštitu korištenjem Hammingovog koda i proizvoljno odabranog cikličkog koda (sami definirajte generirajući polinom $g(x)$)!

Komunikacijski sustavi

K-1. Objasnite funkcioniranje komunikacijskog sustava.Objasnite utjecaj smetnji na informaciju koja se prenosi. Na primjeru prikažite mogućnosti neispravnog dekodiranja informacije na odredištu. Diskutirajte različite mogućnosti zaštitnog kodiranja – kodiranja s ponavljanjem i paritetna kodiranja.

ZADACI – Matlab

MZ-1. Napišite Matlab funkciju zaštitnog kodiranja koristeći Hammingov kôd (7,4). Implementirati funkcije detekcije i ispravljanja jednostruke pogreške. Voditi računa o duljini unesenog niza bitova – ukoliko je niz predugačak, podijeliti ga u više informacijskih riječi i kodirati svaku zasebno!

MZ-2. Napišite Matlab funkciju zaštitnog kodiranja koristeći Hammingovi kôd (15,11). Implementirati funkcije detekcije i ispravljanja jednostruke pogreške. Ograničiti duljinu unesenog niza bitova – ukoliko je duljina veća od 11, zanemariti suvišne bitove. Ukoliko je niz kraći od 5 znakova izvesti zaštitu korištenjem (7,4) koda, te zaštitu kodom (15,11)! Komentirati razliku.

MZ-3. Implementirati funkciju koja određuje generirajuću matricu **G** za proizvoljno definiran generirajući polinom *g(x)*. Polinom unositi u binarnom obliku!

MZ-4. Implementirati funkciju za određivanje CRC zaštitnog bloka uz korištenje koda (7,3) s generirajućim polinomom $g(x)=x^4 + x^3 + x^2 + 1$.

MZ-5. Implementirati funkciju za kodiranje informacijske riječi CRC-16 (ITU-T) cikličkim kodom. Kao rezultat vratiti zaštitni blok i kompletnu kodnu riječ. Generirajuću matricu **G** definirati nakon unosa informacijske riječi.

MK-1. Simulirajte komunikacijski sustav – predajnik, kanal uz utjecaj pogreške i prijemnik. Zanemarite kodiranje i dekodiranje signala. Pretpostavite predajnik kao izvor koji generira 5 vijesti – proizvoljno definirajte vjerojatnost pojavljivanja vijesti. Definirajte vjerojatnost pogreške na kanalu P_g. Na ulazu u predajnik definirajte informaciju proizvoljne duljine.

a) Implementirajte optimalni binarni koder korištenjem Huffmanove metode. Kodiranu informaciju zaštitite Hammingovim kodom (7,4). Po prolasku kroz kanal P_g=0.1, provjerite ispravnost prijenosa i pokušajte dekodirati informaciju. Pokušajte ispraviti pogrešku.
b) Implementirajte optimalni binarni koder korištenjem Shannon-Fanove metode. Kodiranu informaciju zaštitite Hammingovim kodom (15,11). Po prolasku kroz kanal P_g=0.1, provjerite ispravnost prijenosa i pokušajte dekodirati informaciju. Pokušajte ispraviti pogrešku.
c) Optimalno kodirajte informaciju metodom po želji. Zaštitite informaciju kodom s ponavljanjem (omogućiti unos broja ponavljanja!). Po prolasku kroz kanal P_g=0.3, ispravite pogreške većinskim odlučivanjem, te dekodirajte dobivenu informaciju.

Literatura

[1] SCHWARTZ, M.: *Information, Transmission, Modulation and Noise,* 4th ed., McGraw-Hill, 1990.

[2] MATKOVIĆ, V. SINKOVIĆ, V.: *Teorija informacije*. Školska knjiga, Zagreb, 1989.

[3] SWENEY, P.: *Error control coding: An introduction,* Prentice Hall, 1991.