Zaštitno kodiranje signala

Laboratorijska vježba 6

SP

Standardno polje za (8, 2) kôd

Student:

prezime	Ime	mat. broj
	Laboratorij	Datum

Sadržaj:

6. 6. STANDARDNO POLJE ZA (8, 2) KOD

Koristiti:

1. "Upute za rad simulacijskim alatom Logisim 2.7.1." (Moodle)
2. "Simulacijski primjeri" (Moodle)

Priprema za vježbu

Iz skripte "*Zaštitno kodiranje signala*" pročitati poglavlja:

Sazdano od:

Napomena: Cjelovit opis vježbe nalazi se na: MOODLE

Koristan alat za razumijevanje i analizu linearnih blok-kodova

6.1. 6.1. Uvod

Standardno polje može se zamisliti kao organizacijski alat, napunjen ormarić koji sadrži sve moguće 2^n binarne n-torke (nazvane vektori) - ništa ne nedostaje i ništa se ne ponavlja. Cio prostor n-torki naziva se vektorski prostor, V_n. Na prvi pogled, prednost ovoga alata čini se ograničena na malene blok-kodove, jer za duljine kodova veće od $n = 20$ postoje milijuni n-torki u V_n. Međutim, čak za velike kodove, standardno polje omogućuje vizualizaciju važnih pitanja nastupa, kao što su granice sposobnosti za ispravak pogrešaka i mogući ustupci između ispravke i otkrivanja pogrešaka.

6.2. 6.2. Standardno polje

Za (n, k) linearan blok-kod, svi mogući 2^n primljeni vektori raspoređuju se u polje, nazvano *standardno polje*, tako da prvi redak polja sadrži skup svih 2^k kodnih riječi $\{\mathbf{u}\}$, počevši kodnom riječju "sve 0" (niz "sve 0" mora biti član skupa kodnih riječi). Izraz *kodna-riječ* isključivo se koristi za naznačiti *ispravnu* kodnu riječ u prvome retku standardnoga polja. Izraz *vektor* koristi se za označiti bilo kakav poredan niza bitova (na primjer, bilo koja n-torka iz V_n). Prvi stupac standardnoga polja sadrži sve ispravljive uzorke pogrešaka.

Izraz *uzorak pogreške* označava binarnu n-torku $\mathbf{e}$, koja, kada se pribroji ispravnoj kodnoj riječi $\mathbf{u}$ što se prenosi, rezultira prijemom n-torke ili vektora $\mathbf{r} = \mathbf{u} + \mathbf{e}$, a to se može nazvati *oštećena* kodna riječ. U *standardnome polju*, svaki redak, zove se *koskup*. On se sastoji od *čelnika koskupa* (uzorak u lijevome stupcu), a u ostalim stupcima smješteni su popravljivi uzorci pogrešaka, a to su oštećene kodne riječi (oštećene upravo tim čelnikom koskupa). Struktura standardnoga polja za (n, k) kod prikazuje se u nastavku:

$$\begin{array}{ccccccc}
\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_i & \cdots & \mathbf{u}_{2^k} \\
\mathbf{e}_2 & \mathbf{u}_2 + \mathbf{e}_2 & \cdots & \mathbf{u}_i + \mathbf{e}_2 & \cdots & \mathbf{u}_{2^k} + \mathbf{e}_2 \\
\mathbf{e}_3 & \mathbf{u}_2 + \mathbf{e}_3 & \cdots & \mathbf{u}_i + \mathbf{e}_3 & \cdots & \mathbf{u}_{2^k} + \mathbf{e}_3 \\
\vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
\mathbf{e}_j & \mathbf{u}_2 + \mathbf{e}_j & & \mathbf{u}_i + \mathbf{e}_j & \cdots & \mathbf{u}_{2^k} + \mathbf{e}_l \\
\vdots & \vdots & \vdots & \vdots & \cdots & \\
\mathbf{e}_{2^{n-k}} & \mathbf{u}_2 + \mathbf{e}_{2^{n-k}} & \cdots & \mathbf{u}_i + \mathbf{e}_{2^{n-k}} & \cdots & \mathbf{u}_{2^k} + \mathbf{e}_{2^{n-k}}
\end{array} \qquad (6.1)$$

Kodna riječ $\mathbf{u}_1$ sa samim nulama, ima dvostruku ulogu. Ona je prije svega jedna od ispravnih kodnih riječi. Također, $\mathbf{u}_1$ može se smatrati uzorkom pogreške $\mathbf{e}_1$ - uzorak koji naznačuje da pogreške nema, tako da je $\mathbf{r} = \mathbf{u}$. Niz sadrži sve 2^n n-torke u prostoru (svaka n-torka pojavljuje se samo jednom). Svaki koskup ili redak sadrži 2^k n-torki. Stoga, postoji $2^n/2^k = 2^{n-k}$ koskupova (ili redaka).

Algoritam dekodiranja zamjenjuje oštećenu kodnu riječ, $\mathbf{u} + \mathbf{e}$, ispravnom kodnom riječju $\mathbf{u}$, koja se nalazi na vrhu stupca u kojemu se nalazi $\mathbf{u} + \mathbf{e}$. Pretpostavimo da se kodnih riječi $\mathbf{u}_i$ prenosi preko kanala sa šumom. Ako je čelnik koskupa uzorak pogreška uzrokovane kanalom, dobiven vektor

dekodirat će se točno u poslanu kodnu riječi $\mathbf{u}_i$. Ako uzorak greška nije čelnik koskupa, rezultirat je pogrešno dekodiranje. Postoji nekoliko ograničenja sposobnosti ispravke pogrešaka linearnih blok kodova. Bilo koji izvediv kodni sustav mora zadovoljiti sve granice. Jedno takvo ograničenje, zove se Hammingovo ograničenje i opisuje se u nastavku.

Broj paritetnih bitova:

$$n - k \geq \log_2\left[1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}\right] \tag{6.2}$$

ili broj koskupova:

$$2^{n-k} \geq \left[1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}\right] \tag{6.3}$$

gdje binomni faktor $\binom{n}{j}$ predstavlja broj načina na koji j bitova od mogućih n, može imati pogrešku. Imajte na umu da zbroj članova unutar pravokutnih zagrada vodi minimalnome broju redova potrebnih u standardnome polju za ispraviti sve kombinacije pogrešaka od t bitova. Nejednadžba daje donju granicu za $n - k$, broja paritetnih bitova (ili broj 2^{n-k} koskupova) kao funkciju sposobnosti koda za ispravak pogrešaka od t bitova. Isto tako, nejednadžba se može opisati kao zadavanje gornje granice za sposobnost ispravke pogrešaka od t bitova, kao funkciju broja $n - k$ paritetnih bitova (ili 2^{n-k} koskupova). Za bilo koji (n, k) linearan blok-kod, za omogućiti sposobnosti ispravke pogrešaka od t bitova, nužan uvjet jest ispunjenje Hammingovoga ograničenja.

Da bi se pokazalo koliko standardan niz omogućuje prikaz ovih ograničenja, kao primjer uzmimo BCH (127, 106) kod. Niz sadrži sve $2^n = 2^{127} \approx 1{,}7 \times 10^{38}$ n-torki u prostoru. Prvi redak polja sadrži $2^k = 2^{106} \approx 8{,}1 \times 10^{31}$ kodnih riječi pa je to broj stupaca polja. U lijevome stupcu nalazi se $2^{n-k} = 2^{21} =$ 2097152 čelnika koskupova (ili ispravljivih uzoraka pogrešaka), dakle, ovo je broj redaka polja. Iako je broj n-torki i kodnih riječi ogroman, problem nije pojedinačan unos. Prvenstveno je zanimljiv broj koskupova, jer ih ima 2097152 pa stoga postoji barem 2097151 mogućih uzoraka s pogreškama što ih može ispraviti ovaj kod. Nadalje, pokazalo se kako ovoliki broj koskupova određuje gornju granicu sposobnosti koda pri ispravci pogrešaka od t bitova.

Budući da svaka kodna riječ sadrži 127 bitova, postoji 127 načina pojave pojedinačne pogreške. Izračunat ćemo koliko kombinacija ima za pojavu dvostruke pogreške, a to je $\binom{127}{2}$ = 8001. Za broj trostrukih pogrešaka (jer smo do sada mali samo dio od ukupnoga broja uzoraka od 2097151 popravljivih pogreška, imamo $\binom{127}{3}$ = 333375 načina kako bi kombinirali trostruke pogreške. Tablica 6.1 navodi ove proračune, pokazujući da uzorak pogreške "sve 0" zahtijeva prisutnost prvoga koskupa.

Tablica 6.1: Ograničenje ispravaka pogrešaka za (*127*, *106*) *kod*

Broj pogrešnih bitova	Broj zahtijevanih koskupova	Ukupan broj zahtijevanih koskupova
0	1	1
1	127	128
2	8001	8129
3	333375	341504
4	10334625	10676129

U tablici se također pokazuje da je osim velikoga broj koskupova potrebnih za svaku vrstu pojedine četverostruke pogreške, potreban je i ukupan broj koskupova za tu vrstu pogrešaka.

Ova tablica pokazuje da je (127, 106) kod može ispraviti sve jednostruke, dvostruke i trostruke uzorke pogrešaka, a neiskorištenim recima je svojstvena mogućnost ispravke još više pogrešaka. Moglo bi biti primamljivo pokušati prilagoditi sve moguće uzorke pogrešaka od 4 bita u nizu. Međutim, tablica 6.1 pokazuje da to nije moguće, jer je broj preostalih koskupova u nizu mnogo manji od kumulacijskoga broja potrebnih koskupova, a kao što se navodi u posljednjemu retku tablice. Dakle, ovaj (127, 106) kod ima Hammingovo ograničenje koje jamči ispravak svih pogrešaka od maksimalno 3 bita.

6.3. 6.3. Savršeni kodovi

Prethodno razmatranje (127, 106) koda s prikazanom sposobnošću ispravke *jednostrukih*, *dvostrukih* i *trostrukih* pogrešaka dokazuje ono što je istina za mnoge kodova. To jest, često postoji preostala sposobnost ispravke pogrešaka iznad vrijednosti t. Kod za ispravak t pogreška naziva se *savršen kod*, ako njegovo standardno polje ima sve uzorke do t pogrešaka i bez drugih uzoraka kao što su čelnici koskupova tj., nema preostalih (*residual*) sposobnosti ispravaka pogrešaka. Hammingovi kodovi su savršeni kodovi, jer mogu ispraviti *samo jednostruke pogreške*. Struktura standardnoga polja može se koristiti za potvrdu ovoga svojstva. Hammingove kodove odlikuju (n, k) dimenzije kako slijedi:

$$(n, k) = (2^m–1, 2^m–1– m)$$

gdje je m = 3, 4. Tako je broj koskupova jednak $2^{n–k} = 2^m$ dok je $n – k = m$. Zbog toga što je $n = 2^{m–1}$, postoji $2^{m–1}$ načina stvaranja pojedinačne pogreške. Prema tome, broj koskupova, 2^m, iznosi točno 1 (za slučaj bez pogreške) plus nekoliko načina da se pojavi jedna pogreška u n bitova. Dakle, svi Hammingovi kodovi koji mogu ispraviti samo jednostruke pogreške, doista su savršeni kodovi.

Donekle slična situacija događa se za trostruki ispravak pogrešaka Golayevim (23, 12) kodom, koji je savršen kod. Obratite pažnju da je za ovaj kod, broj koskupova u standardnome polju jednak $2^{n–k} = 2^{11}$ = 2048. Praćenjem formata iz tablice 1, razvili smo tablicu 2 za Golayev (23, 12) kod, kao što se prikazuje u nastavku:

Tablica 6.2: Ograničenje ispravke pogrešaka za Golay (*23*, *11*) *kod*

Broj pogrešnih bitova	Broj zahtijevanih koskupova	Ukupan broj zahtijevanih koskupova
0	1	1
1	23	24
2	253	277
3	1771	2048

Tablica 6.2 pokazuje da je Golayev (23, 11), doista pogodan kod, jer nema sposobnost ispravke preostalih pogrešaka iznad t = 3.

6.4. 6.4. Primjer (n, k)

Standardan niz pruža uvid u moguće ustupke između ispravke i otkrivanja pogrešaka. Razmotrite primjer nekoga (n, k) koda i čimbenike koji utječu na vrijednosti (n, k) što ih treba izabrati.

1. Za netrivijalan ustupak između ispravke i otkrivanja pogreške, poželjna je sposobnost ispravke pogrešaka od najmanje t = 2.
2. Hammingova udaljenost između dviju kodnih riječi predstavlja broj položaja bitova u kojima se razlikuju dvije kodne riječi. Najmanja Hammingova udaljenost između svih kodnih riječi što ih kod obuhvaća, naziva se minimalna udaljenost, $d_{\min}$ koda. Kao sposobnost ispravke pogrešaka od t = 2, koristimo sljedeći temeljni odnos za pronalazak najmanje udaljenosti:
$$d_{\min} = 2t + 1 = 5$$
3. Za svaki značajniji sustav kodiranja, poželjno je imati najmanje k = 2 podatkovnih bitova. Dakle, on će imati $2^2 = 4$ kodne riječi. Sada se kod može označiti kao (n, 2) kod.
4. Tražimo minimalnu vrijednost za n, što će omogućiti ispravak svih mogućih jednostrukih i dvostrukih pogrešaka. U ovome primjeru, svaka od 2^n n-torki u nizu nalazit će se u tablici. Minimalna vrijednost n je poželjna, jer kada god n poraste za samo jedan cio broj, broj n-torki u standardnim poljima se udvostručuje. Naravno, poželjno je da se tim popisom može rukovati. Za kodove iz "stvarnoga svijeta", želimo što manji n zbog različitih razloga - učinkovitost propusnosti i jednostavnost. Ako se koristi Hammingovo ograničenje izborom n, tada se može odabrati n = 7. Međutim, dimenzije takvoga (7, 2) koda neće zadovoljiti naše početne zahtjeve za sposobnošću ispravke pogrešaka od t = 2 bita uz $d_{\min}$ = 5. Za uvjeriti se u ovo, potrebno je uvesti još jednu gornju granicu na sposobnost ispravke od t bitova pogrešaka (ili $d_{\min}$). Ova granica, naziva se Plotkinova granica i opisuje se u nastavku:
$$d_{\min} \le \frac{n \times 2^{k-1}}{2^k - 1} \tag{6.4}$$

Općenito, linearan (n, k) kod mora ispuniti sva prethodna ograničenja uključujući sposobnost ispravke pogrešaka (ili minimalnu udaljenost). Za kodove velikih brzina, ako se dosegne Hammingova granica, Plotkinova granica također će se doseći, a to je i prije bio slučaj za primjer (127, 106) koda. Za kodove malih brzina, to je zaobilazan (*around*) drugi način. Dok ovaj primjer podrazumijeva kod male brzine, važno je testirati mogućnost ispravke pogrešaka Plotkinovom granicom. Zbog $d_{\min} = 5$, iz jednadžbe (6.4) trebalo bi biti jasno da n mora biti 8 pa stoga i (8, 2) kod mora imati minimalne dimenzije kako bi se ispunili uvjeti za ovaj primjer.

6.5. 6.5. Oblikovanje (8, 2) koda

6.5.1. 6.5.1. UKRATKO PONOVIMO

Kako se odabire kodna riječ koja je izvan prostora od 2^8 8-morki? Elementi koji su u središtu rješenja toga problema.

1. Broj kodnih riječi je $2^k = 2^2 = 4$.
2. Mora se primijeniti svojstvo zatvorenosti.
3. Vektor "sve 0" mora biti jedna od kodnih riječi.
4. Svaka kodna riječ ima 8 bitova.
5. Budući $d_{\min} = 5$, težina svake kodne riječi (osim kodne riječi sve-nule) također mora biti najmanje 5 (na temelju svojstva zatvorenosti). Težina vektora je definirana kao broj komponenata vektora različitih od nule.
6. Pretpostavimo da je kod sustavan, pa 2 bita sasvim desno u svakoj kodnoj riječi predstavljaju odgovarajuću poruku.

Slijedeći zadatak je pridružiti kandidate kodnih riječi porukama koje ispunjavaju sve prethodno navedene uvjete.

poruke	kodne riječi
00	00000000
01	11110001
10	00111110
11	11001111

(6.5)

Pri oblikovanju skupa kodnih riječi treba se pridržavati svojstava težine i sustavnoga oblika koda. Izbor prvih nekoliko kodnih riječi jednostavan je. Kako se proces nastavlja, izbor postaje teži i ograničen zbog svojstva zatvorenosti.

6.6. 6.6. Kodiranje, dekodiranje i ispravak pogrešaka

Stvaranje kodne riječi $\mathbf{u}_i$ u nekome (n, k) kodu uključuje oblikovanje umnoška vektora poruke od k bitova $\mathbf{m}_i$ i $k \times n$ generator-matrice $\mathbf{G}$. Za sustav kodiranja u jednadžbi (6.5), $\mathbf{G}$ se može napisati kao što prikazuje jednadžba (6.6):

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

Oblikovanjem umnoška $\mathbf{m} \cdot \mathbf{G}$ za sve poruke u jednadžbi (6.5) dat će sve kodne riječi prikazane u toj jednadžbi.

Dekodiranje započinje računanjem sindroma, što se može smatrati učenje "simptoma" pojedine pogreške. Za (n, k) kod, sindrom $\mathbf{s}$, od ($n - k$) bitova, umnožak je primljenoga vektora, $\mathbf{r}$, od n bitova i transponirane ($n - k$) $\times$ n matrica provjere paritet, $\mathbf{H}$, gdje se $\mathbf{H}$ konstruira tako da su redovi matrice $\mathbf{G}$ okomiti na redove matrice $\mathbf{H}$, to jest, $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$. Za ovaj (8, 2), primjer, $\mathbf{s}$ je vektor od 6 bitova, a $\mathbf{H}^T$ je 6×8 matrica, napisana kao što prikazuje jednadžba (6.7):

$$\mathbf{H}^T = \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{array}\right] \tag{6.7}$$

Sindrom za svaki uzorak pogreške može se izračunati kao što prikazuje jednadžba (6.8):

$$\mathbf{s}_i = \mathbf{e}_i \, \mathbf{H}^T \qquad i = 1, \ldots 2^{n-k} \tag{6.8}$$

Tablica 6.3 prikazuje tablicu svih $2^{n-k} = 64$ sindroma kao standardan niz za (8, 2) kod.

Tablica 6.3: Sindromi i standardan niz za (8, 2) kod.

	sindromi	uzorak pogreške	standardno polje		
1.	**000000**	**00000000**	**11110001**	**00111110**	**11001111**
2.	111100	00000001	11110000	00111111	11001110
3.	001111	00000010	11110011	00111100	11001101
4.	000001	00000100	11110101	00111010	11001011
5.	000010	00001000	11111001	00110110	11000111
6.	000100	00010000	11100001	00101110	11011111
7.	001000	00100000	11010001	00011110	11101111
8.	010000	01000000	10110001	01111110	10001111
9.	100000	10000000	01110001	10111110	01001111
10.	110011	00000011	11110010	00111101	11001100
11.	111101	00000101	11110100	00111011	11001010
12.	111110	00001001	11111000	00110111	11000110
13.	111000	00010001	11100000	00101111	11011110
14.	110100	00100001	11010000	00011111	11101110
15.	101100	01000001	10110000	01111111	10001110
16.	011100	10000001	01110000	10111111	01001110
17.	001110	00000110	11110111	00111000	11001001
18.	001101	00001010	11111011	00110100	11000101
19.	001011	00010010	11100011	00101100	11011101
20.	000111	00100010	11010011	00011100	11101101
21.	011111	01000010	10110011	01111100	10001101
22.	101111	10000010	01110011	10111100	01001101
23.	000011	00001100	11111101	00110010	11000011
24.	000101	00010100	11100101	00101010	11011011
25.	001001	00100100	11010101	00011010	11101011
26.	010001	01000100	10110101	01111010	10001011
27.	100001	10000100	01110101	10111010	01001011
28.	000110	00011000	11101111	00100110	11010111
29.	001010	00101000	11011001	00010110	11100111
30.	010010	01001000	10111001	01110110	10000111
31.	100010	10001000	01111001	10110110	01000111
32.	001100	00110000	11000001	00001110	11111111
33.	010100	01010000	10100001	01101110	10011111
34.	100100	10010000	01100001	10101110	01011111
35.	011000	01100000	10010001	01011110	10101111
36.	101000	10100000	01010001	10011110	01101111
37.	110000	11000000	00110001	11111110	00001111
38.	110010	00000111	11110110	00111001	11001000
39.	110111	00010011	11100010	00101101	11011100
40.	111011	00100011	11010010	00011101	11101100
41.	100011	01000011	10110010	01111101	10001100
42.	010011	10000011	01110010	10111101	01001100
43.	111111	00001101	11111100	00110011	11000010
44.	111001	00010101	11100100	00101011	11011010
45.	110101	00100101	11010100	00011011	11101010
46.	101101	01000101	10110100	01111011	10001010
47.	011101	10000101	01110100	10111011	01001010
48.	011110	01000110	10110111	01111000	10001001
49.	101110	10000110	01110111	10111000	01001001
50.	100101	10010100	01100101	10101010	01011011

	sindromi	uzorak pogreške	standardno polje		
1.	**000000**	**00000000**	**11110001**	**00111110**	**11001111**
51.	011001	01100100	10010101	01011010	10101011
52.	110001	11000100	00110101	11111010	00001011
53.	011010	01101000	10011001	01010110	10100111
54.	010110	01011000	10101001	01100110	10010111
55.	100110	10011000	01101001	10100110	01010111
56.	101010	10101000	01011001	10010110	01100111
57.	101001	10100100	01010101	10011010	01101011
58.	100111	10100010	01010011	10011100	01101101
59.	010111	01100010	10010011	01011100	10101101
60.	010101	01010100	10100101	01101010	10011011
61.	011011	01010010	10100011	01101100	10011101
62.	110110	00101001	11011000	00010111	11100110
63.	111010	00011001	11101000	00100111	11010110
64.	101011	10010010	01100011	10101100	01011101

6.1 Za (8, 2) kod opisan u poglavlju 6.6.3, potvrdite da su vrijednosti za generator-matricu, matricu provjere pariteta i vektora sindroma, ispravne za svaki od koskupova od 1 do 10.

Provjeravamo ispravnost generator matrice generiranjem 4 kodne riječi formulom: $\mathbf{u} = \mathbf{m} \cdot \mathbf{G}$

$$\mathbf{G} = \left[\begin{array}{cccccc|cc} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$\mathbf{u}_1 = [00] \cdot \mathbf{G} = [00000000]$$
$$\mathbf{u}_2 = [01] \cdot \mathbf{G} = [11110001]$$
$$\mathbf{u}_3 = [10] \cdot \mathbf{G} = [00111110]$$
$$\mathbf{u}_4 = [11] \cdot \mathbf{G} = [11001111]$$

Sindrom $\mathbf{s}$ jednak je:

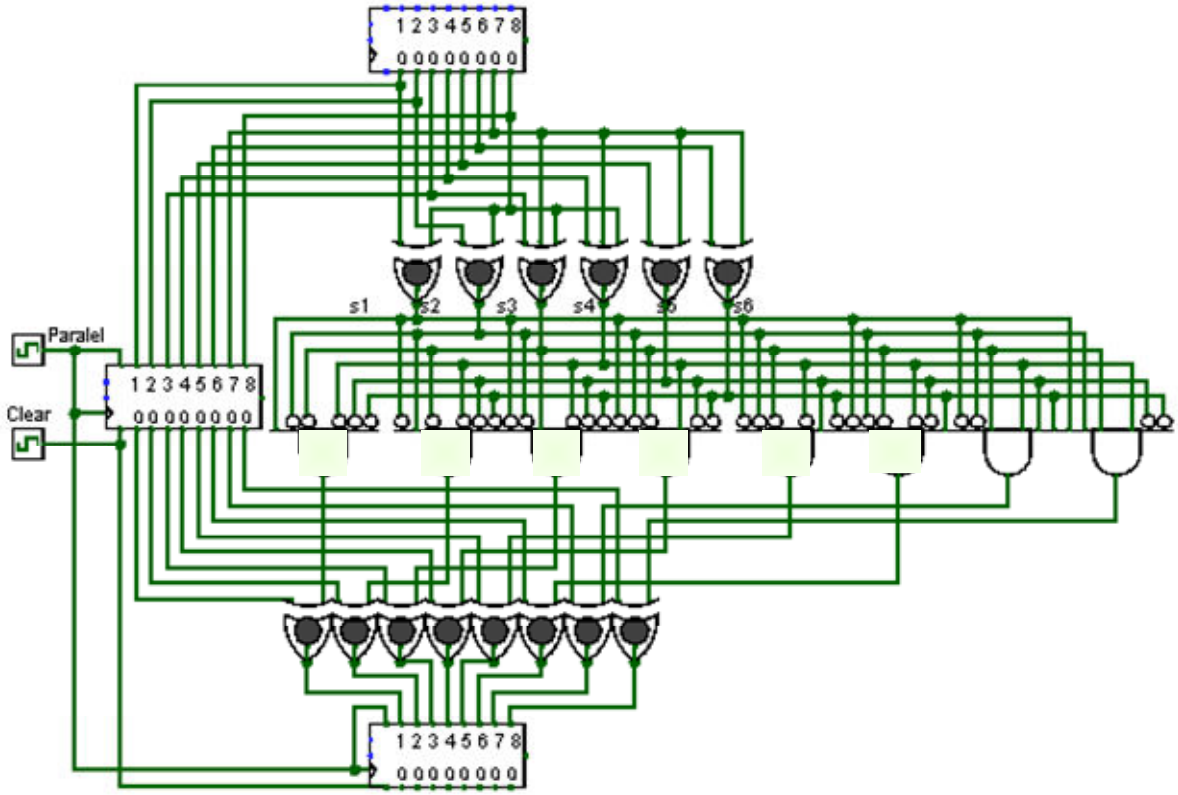
$$\mathbf{s} = \mathbf{e} \cdot \mathbf{H}^T = \mathbf{e} \cdot \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hdashline 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{array}\right]$$

Stvaranje paritetne $\mathbf{H}$ matrice iz generator matrice $\mathbf{G}$:

$$\mathbf{H} = [\mathbf{I}_{n-k} \mid \mathbf{P}^T] = \left[\begin{array}{cccccc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

$$\mathbf{s}_1 = [00000000] \cdot \mathbf{H}^T = [000000]$$
$$\mathbf{s}_2 = [00000010] \cdot \mathbf{H}^T = [111100]$$
$$\mathbf{s}_3 = [00000100] \cdot \mathbf{H}^T = [001111]$$
$$\mathbf{s}_4 = [00001000] \cdot \mathbf{H}^T = [000001]$$
$$\mathbf{s}_5 = [00010000] \cdot \mathbf{H}^T = [000010]$$
$$\mathbf{s}_6 = [00010000] \cdot \mathbf{H}^T = [000100]$$
$$\mathbf{s}_7 = [00100000] \cdot \mathbf{H}^T = [001000]$$
$$\mathbf{s}_8 = [01000000] \cdot \mathbf{H}^T = [010000]$$
$$\mathbf{s}_9 = [10000000] \cdot \mathbf{H}^T = [100000]$$
$$\mathbf{s}_{10} = [00000011] \cdot \mathbf{H}^T = [110011]$$

6.2 Korištenjem isključivo (*exclusive*) ILI i I vrata, primijenite krug za dekodiranje, sličan onome što ga prikazuje slika 6.12, a koji će ispraviti pogrešku za sve jednostruke uzorke pogrešaka (8, 2) koda opisan čelnicima koskupova od 2 do 9 na slici 6.1.



Slika: 6.1

Za kod u sustavnome obliku, dekoder samo isporučuje bitove poruke (u_7 i u_8). Stoga se mogu ukloniti osjenčana vrata.

6.3 Detaljno objasnite kako se mogu iskoristiti vrata EXOR i AND u krugu za dekodiranje, slično onima što su prikazana na slici 6.1, a ispravljaju sve jednostruke i dvostruke uzorke pogrešaka (8, 2) koda te otkrivaju pogreške za uzorke trostrukih pogrešaka (čelnici koskupa ili redci od 38 do 64).

Za ispravak svih jednostrukih i dvostrukih pogrešaka (8, 2) kodom, čije je standardno polje prikazuje slika 6.2, trebamo sklopovlje (za ispravak jednostrukih pogrešaka) plus dodatna vrata (za ispravak dvostrukih pogrešaka) kao što slijedi.

Pretpostavimo da je kod u sustavnome obliku pa dekoder treba dekodirati samo 2 bita sasvim desno (informacijski podaci) svake kodne riječi od 8 bitova.

Za ispravak dvostrukih pogrešaka koji oštećuju podatke, znači da sklopovlje mora dodatno isporučiti odgovarajući uzorak pogreške sve dok se pogreška podudara s jednom od podatkovnih pogrešaka među pogreškama od 10 do 22 (tablica 6.3 i slika 6.2) standardnoga polja.

Jedna od mogućih primjena sklopa za upotpuniti ovo, sastoji se od kruga prikazanoga u rješenju zadatka , s dodatnim vratima.

~~Za ove slučajeve, potrebno je isprobati još više vrata, gdje nemamo ne-nulte sindrome, ali nije uključen ni ispravak pogrešaka. Ovo je korisno za otkrivanje pogrešaka za bilo koji od sindroma označenih brojevima od 38 do 64 (u ovome primjeru).(vidi kraj ovoga zadatka)~~

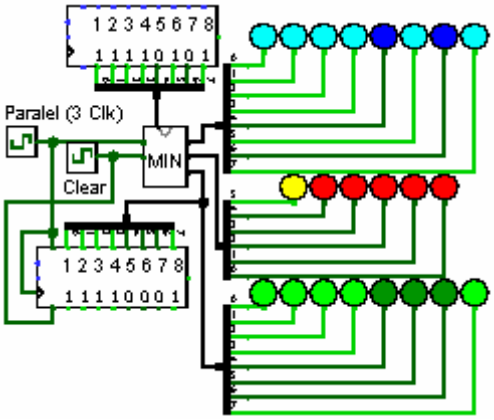
Zahtijevaju se dodatna logička vrata

gdje  predstavljaju AND vrata. Žice označene 1, …, 6, spojene su na znamenke sindroma s_1, …, s_6, a male kružnice na žicama znače "komplement podatka".

Broj u svakome od AND vrata predstavlja broj uzorka pogreške iz tablice na slici 6.15 (brojevi od 10 do 22) i ublažene (smanjene) su izlazom tih vrata.

Izlazi iz logičkih AND vrata vode se na EXOR vrata sa 7 ulaza (atribut: *When an odd number are ON*). Na jedan od ulaza EXOR vrata B (od 11 do 16) dovede se izlaz iz EXOR vrata A (od 16 do 22).

Onda se izlazi iz A i B EXOR vrata vode na ulaze postojećih vrata br. 7 odnosno br. 8. I to dvoje vrata imaju isti atribut: (*When an odd number are ON*). Ovo upotpunjuje sklop za ispravak i *dvostrukih* pogrešaka.



sindrom	čelnik	kodne riječ za jediničnu matricu-desno		
000000	00000000	11110001	00111110	11001111
111100	00000001	11110000	00111111	11001110
001111	00000010	11110011	00111100	11001101
000001	00000100	11110101	00111010	11001011
000010	00001000	11111001	00110110	11000111
000100	00010000	11100001	00101110	11011111
001000	00100000	11010001	00011110	11101111
010000	01000000	10110001	01111110	10001111
100000	10000000	01110001	10111110	01001111

Slika 6.2

Za ove slučajeve, potrebno je isprobati još više vrata, gdje nemamo ne-nulte sindrome, ali nije uključen ni ispravak pogrešaka. Ovo je korisno za otkrivanje pogrešaka za bilo koji od sindroma označenih brojevima od 38 do 64 (u ovome primjeru).

Svaki redak standardnoga polja (osim prvoga retka) predstavlja skup oštećenih kodnih riječi uz nešto zajedničko, otuda i ime koskup. Što je to zajedničko svakome unosu bilo kojega koskupa? Imaju isti sindrom. Nakon izračunana sindroma, ispravak oštećene kodne riječi nastavlja se pronalaskom uzorka pogreške koji odgovara tome sindromu. Konačno, uzorak pogreške oduzima se (oduzimanje = zbrajanje modulo-2) od oštećene kodne riječi, što ispravlja izlaz.

Iz ovoga se može vidjeti da su operacije *zbrajanja* i *oduzimanja* ustvari *jednake* te se mogu zamijeniti binarnom operacijom XOR (isključivo-ILI), a operacija množenja može se zamijeniti binarnom operacijom AND (&, I).

Budući da svaki koskup ima isti sindrom, jednadžba (6.8) može se napisati u smislu primljenoga vektora, **r**, kao što slijedi:

$$\mathbf{s}_i = \mathbf{r}_i \, \mathbf{H}^T \tag{6.9}$$

Svaki od vektora $\mathbf{u}_i$, $\mathbf{e}_i$, $\mathbf{r}_i$ i $\mathbf{s}_i$ može se opisati u sljedećem općem obliku:

$$\mathbf{x}_i = \{x_1, x_2, \ldots, x_j, \ldots\}$$

Za kodnu riječ $\mathbf{u}_i$ u ovome primjeru, indeks $i = 1, \ldots, 2^k$ pokazuje da postoje 4 različite kodne riječi, a indeks $j = 1, \ldots, n$ pokazuje da ima 8 bita po kodnoj riječi. Za primljen vektor $\mathbf{r}_i$, indeks $i = 1, \ldots, 2^n$ pokazuje da postoje 256 različita vektori, a indeks $j = 1, \ldots, n$ pokazuje da ima 8 znamenaka u vektoru. Za uzorak pogreške $\mathbf{e}_i$, indeks $i = 1, \ldots, 2^{n-k}$ pokazuje da postoje 64 različita popravljiva uzorka pogrešaka, a indeks $j = 1, \ldots, n$ pokazuje da postoje 8 znamenaka po uzorku pogreške. Za sindrom $\mathbf{s}_i$, indeks $i = 1, \ldots, 2^{n-k}$ ukazuje na to da postoje 64 različita sindroma, a indeks $j = 1, \ldots, n-k$ pokazuje da postoji 6 znamenaka po sindromu. Radi jednostavnosti, indeks i se odbacuje pa će se vektori $\mathbf{u}_i$, $\mathbf{e}_i$, $\mathbf{r}_i$, i $\mathbf{s}_i$ označit kao **u**, **e**, **r**, odnosno **s**, gdje se u svakome slučaju podrazumijeva neki i-ti vektor.

6.6.1. 6.6.1. USTUPCI IZMEĐU OTKRIVANJA I ISPRAVKE POGREŠAKA

Korištenjem skupa kodnih riječi u jednadžbi (6.5), izgradilo se standardno polje (vidi tablicu 6.3). Sposobnošću otkrivanja i ispravak pogrešaka može se baratati, pod uvjetom da prevladava sljedeći odnos za udaljenost:

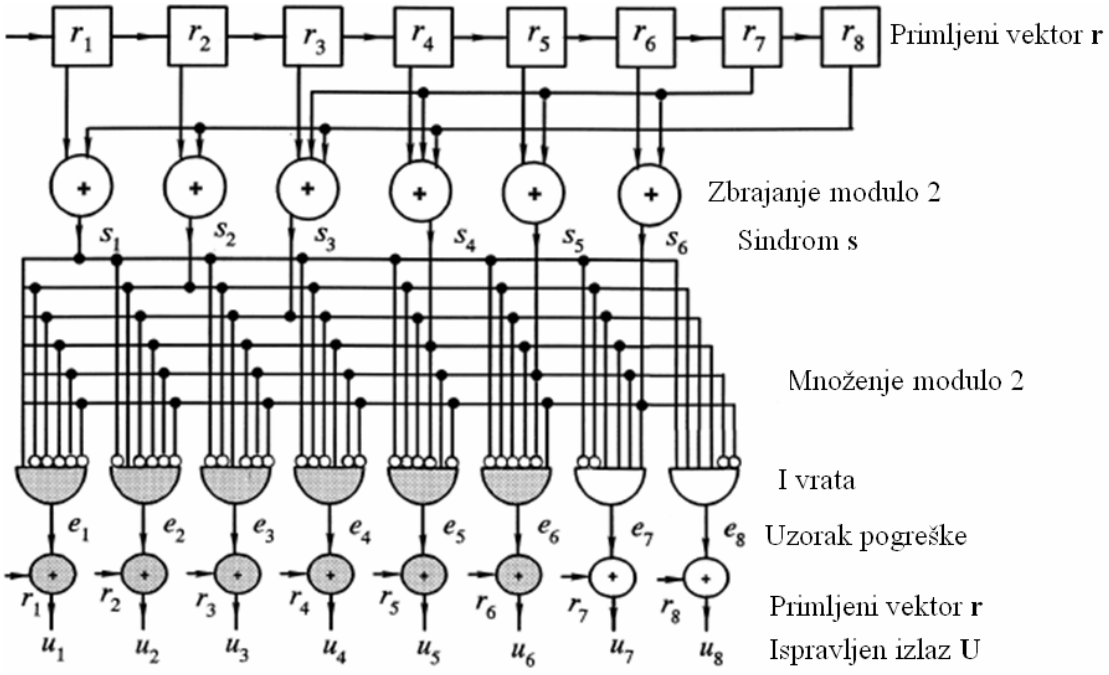
$$d_{\min} \geq \alpha + \beta + 1 \tag{6.10}$$

gdje α predstavlja broj pogrešnih bitova što ih treba ispraviti, β predstavlja broj pogrešnih bitova što ih treba otkriti, uz $\beta \geq \alpha$. Za primjer (8, 2) koda, na raspolaganju su izbori ustupaka kako slijedi:

otkrivanje (β)	Ispravak (α)
2	2
3	1
4	0

Ova tablica pokazuje da se (8, 2) kod može primijeniti samo za popravak pogrešaka, što znači da on najprije otkrije $\beta = 2$ pogrešaka, a onda ih ispravlja. Ako se poneki ispravak pogrešaka žrtvuje, tako da kod ispravlja samo jednostruke pogreške, onda se povećava sposobnost otkrivanja pa se mogu otkriti sve $\beta = 3$ pogreške. I na kraju, ako se ispravak pogrešaka u potpunosti žrtvuje, dekoder se može primijeniti za otkrivanje $\beta = 4$ pogreške. U slučaju samo otkrivanja pogrešaka, sklop je vrlo jednostavan. Računa se sindrom i otkriva se pogreška dok se ne dođe do sindroma "sve 0".

Sklop dekodera za ispravak jedne pogreške može se napraviti logičkim vratima, kao što prikazuje slika 3.



Slika 3: Strujni krug za dekodiranje (*8*, *2*) *koda.*

Vrata isključivo (*exclusive*) ILI (EX-OR) obavljaju istu operaciju kao aritmetika modulo 2 pa se stoga koristi isti simbol. Vrata I prikazana su kao polu-kružnice. Male kružnice na završetcima svakoga ulaznoga priključka i vrata I, ukazuju na komplementarnu logiku binarnoga stanja. Na ovoj slici, primljeni vektor, **r** ulazi u dekoder na dva mjesta istovremeno. U gornjemu dijelu na slici, 8 znamenaka primljenoga vektora učitava se u posmični registar čiji stupnjevi su povezani sa 6 EX-ILI vrata, od kojih svaka daju 1 bit sindromu $\mathbf{s}_j$, gdje je $j = 1, ..., 6$. Ožičenje strujnoga kruga između primljenoga vektora, **r** i EX-ILI vrata diktiraju jednadžbe (6.9), kako slijedi:

$$\mathbf{s} = [r_1\ r_2\ r_3\ r_4\ r_5\ r_6\ r_7\ r_8] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{6.11}$$

Stoga se svaka od $\mathbf{s}_j$ znamenki što ih sadrži sindrom **s**, može opisati jednadžbom (6.11) prema $\mathbf{r}_j$ znamenkama primljenoga vektora na sljedeći način:

$$s_1 = r_1 + r_8 \qquad s_2 = r_2 + r_8 \qquad s_3 = r_3 + r_7 + r_8$$

$$s_4 = r_4 + r_7 + r_8 \qquad s_5 = r_5 + r_8 \qquad s_6 = r_6 + r_7$$

Ako se dekoder primijeni za ispravak samo jednostruke pogreške, tj. $\alpha = 1$ i $\beta = 3$, onda nas to vodi na podcrtavanje ispod koskupa 9 u tablici 3, a ispravak pogrešaka događa se samo kad je jedan od 8 sindroma povezan pojavom jedne pogreške. Lako je provjeriti da ulazi na slici 1 pretvaraju bilo koji

sindrom, označen brojevima od 1 do 9, u odgovarajući uzorak $\mathbf{e}_i$ pogreške. Uzorak pogreške potom se oduzima (zbrajanje modulo-2) od "potencijalno" oštećenoga dobivenoga vektora, čime se ispravlja izlaz, **u**. Dodatna vrata potrebna su za testiranje slučaja, kada je sindrom različit od nule, a izlazi I vrata su same nule - takav događaj, događa se za bilo koji sindrom označen brojevima od 10 do 64.

Takav ishod onda se koristi za naznačiti otkrivenu pogrešku. Imajte na umu da je slika 1, zbog udžbeničkih razloga, nacrtana za naglasiti korake algebarskoga dekodiranja - izračun sindroma, dobivanje uzorka pogreške i na kraju, ispravak izlaza. U "stvarnome svijetu", (*n*, *k*) kod obično se konfigurira u sustavnome obliku, što znači da znamenke krajnje desne kodne riječi imaju *k* bitova podataka, a ravnoteža kodne riječi sastoji se od $n - k$ paritetnih bitova. Dekoder ne treba isporučiti cijelu kodnu riječ, njegov izlaz može se sastojati samo od podatkovnih bitova. Dakle, sklopovlje na slici 1 postaje jednostavnije uklanjanjem vrata koja su prikazana zasjenjeno.

Uočite da se proces dekodiranja oštećene kodne riječi prvo otkrivanjem, a zatim ispravkom pogreške može usporediti s poznatom medicinskom analogijom. Bolesnik (potencijalno oštećena kodna riječ) ulazi u medicinsku ustanovu (*dekoder*). Liječnik ispituje fizičko stanje pacijenta testiranjem (množenje $\mathbf{r}\cdot\mathbf{H}^T$), kako bi odredio simptom (*sindrom* **s**). Zamislite da liječnik utvrdi karakteristične točke na pacijentu *x*-zrakama. Iskusan liječnik ne bi odmah prepoznao povezanost između simptoma i bolesti (uzorak pogreške), recimo za tuberkulozu.

Liječnik-početnik možda bi morao koristiti priručnik za pridružiti bolesti simptom (tj., sindrom u odnosu na uzorak pogreške kao što je navedeno u tablici 3, ili oblikovano I vratima kao na slici 6.1). Završnim korakom osigurava se odgovarajući lijek pacijentu, čime se izbjegava bolest (drugim riječima, metodom modulo 2 dodaje se (pribraja se) uzorak pogreške oštećenoj kodnoj riječi, čime se ispravlja manjkava kodna riječ). U kontekstu binarnih kodova, ovdje se koristi neobična vrsta medicine. Pacijent se (iz)liječi ponavljanjem primjene izvorne bolesti (homeopatija).

Ako je dekoder napravljen za popravak pogrešaka, onda je $\alpha = 2$ i $\beta = 2$. Za ovaj slučaj, otkrivanje i ispravak svih jednostrukih i dvostrukih pogreške mogu se zamisliti kao podcrtavanje koskupa 37 u standardnome polju u tablici 3. Čak iako je (8, 2) kod sposoban ispraviti neke kombinacije trostrukih pogrešaka što odgovara čelnicima koskupa od 38 do 64, dekoder se najčešće primjenjuje kao dekoder ograničene udaljenosti. Znači da ispravlja sve kombinacije pogrešaka do *t*, uključujući i *t* pogrešaka, ali ne i kombinacije pogrešaka veći od *t*. Dekoder se može ostvariti logičkim vratima, koristeći primjenu koja je slična sklopu na slici 1.

Iako se za opis ovih ustupaka koristi malen kod, primjer se može proširiti (bez ulaska u pojedinosti u standardnome polju) na bilo koju veličinu koda. Strujni krug na slici 1 dekodira na paralelan način, što znači da se sve znamenke kodnih riječi dekodiraju istovremeno. Takvi dekoderi korisni su samo za relativno male kodova. Kad je kod (pre)velik, takvo paralelno dekodiranje postaje vrlo složeno pa se uglavnom bira jednostavniji serijski pristup (koji će zahtijevati više vremena za obradu, nego paralelno sklopovlje).

Ako je *k* prevelik, primjena dekodera pomoću pregledne tablice postaje preteška. Kod (127, 92) ima 2^{92} ili približno 5×10^{27} kodnih vektora. Ako se postupak dekodiranja sastoji od jednostavne pregledne tablice, zamislite veličinu memorije koja je neophodna za prihvatiti toliki broj kodnih riječi. Srećom, moguće je smanjiti složenost dekodiranja stvaranjem onoliko kodnih riječi koliko ih je potrebno, umjesto da ih se sve pohranjuje u tablicu.

6.6.2. 6.6.2. STANDARDNO POLJE JE PREGLEDNO

U kontekstu tablice 3, (8, 2) kod zadovoljava Hammingovo ograničenje. To jest, iz standardnoga polja vidi se da (8, 2) kod može ispraviti sve kombinacije jednostrukih i dvostrukih pogrešaka. Razmotrite sljedeće pitanje: "Pretpostavimo da se prijenos odvija preko kanala koji uvijek uvodi pogreške u obliku praska pogrešaka od 3 bita, tako da ne postoji interes za ispravak jednostruke ili dvostruke pogreške, je li moguće postaviti čelnika koskupa da se odziva samo na trostruke pogreške"? Jednostavno je vidjeti da u nizu od 8 bitova postoje $\binom{8}{3} = 56$ načina za napraviti trostruke pogreške.

Ako samo želimo ispraviti svi tih 56 kombinacija trostrukih pogreške, postoji dovoljno prostora (dovoljan broj koskupova) u standardnome polju, jer postoje 64 retka. Zar to ne radi? Odgovor je

„Ne!“ Za bilo koji kod, parametar preskakanja za određivanje sposobnosti ispravke pogrešaka je d_{min}. Za (8, 2) kod, $d_{min} = 5$ diktira mogućnost ispravke pogrešaka u samo 2 bita.

Kako standardno polje daje neki uvid o tome zašto ova shema ne radi? Za skupinu uzoraka pogrešaka koja bi omogućila ispravak pogrešaka od x bitova, cijela skupina vektora težine x mora biti čelnik koskupa, to jest, vektori moraju zauzeti samo krajnji lijevi stupac. U tablici 3, svi vektori težina 1 i 2 pojavljuju se u krajnjemu lijevome stupcu standardnoga polja i nigdje drugdje. Čak, iako prisilimo sve vektore težine 3 s brojevima redova od 2 do 57, naći ćemo da će se neki od tih vektora morati pojaviti na drugome mjestu u polju (što krši temeljno svojstvo standardnoga polja). U tablici 6.3, zasjenjena polja nacrtana su oko svakoga od 56 vektora težine 3. Pogledajte čelnike koskupova što predstavljaju uzorke pogrešaka od 3 bita, u redovima 38, 41-43, 46-49 i 52 standardnoga polja.

Sada pogledajte unose istih brojeva redova u krajnjem desnom stupcu, gdje osjenčani pravokutnici ukazuju na druge vektore težine 3. Uočavate li dvosmislenost koja postoji za svakoga od prije navedenih redova te zašto nije moguće ispraviti sve uzorke pogrešaka od 3 bita s ovim (8, 2) kodom? Pretpostavimo da dekoder prima vektor težine 3 [1 1 0 0 1 0 0 0], što se nalazi u retku 38 u krajnjem desnom stupcu. Ova manjkava kodna riječ mogla je nastati na jedan od dva načina. Jedan način bio bi da se poslala kodna riječi [1 1 0 0 1 1 1 1], a ometa je uzorak pogreške [0 0 0 0 0 1 1 1] od 3 bita. Druga mogućnost bila bi da je poslana kodna riječ [0 0 0 0 0 0 0 0], a ometa je uzorak pogreške [1 1 0 0 1 0 0 0] od 3 bita.

6.7. 6.7. Zaključak

U ovoj laboratorijskoj vježbi, prikazana su osnovna načela blok kodova, naglašavajući strukturu standardnoga polja. Koriste se primjeri što uključuju granice, savršene kodove i provedbene ustupke kako bi se dobio uvid u neke algebarske strukture linearnih blok kodova. Također se pokazalo kako standardno polje nudi slutnju o tome zašto neka željena svojstva ispravke pogrešaka određenoga koda možda nisu moguća.

6.8. 6.8. Zadatak

1. Na temelju primjera na slici 1, u simulacijskome alatu nacrtajte dekoder za zadanu matricu (8, 2) koda.

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

Nacrtajte koder (8, 2) koda!

*Pomoć: Potražite odgovarajuću datoteku među *.circ datotekama na MOODLE.*

2. Izračunajte pripadne matrice **H** i $\mathbf{H}^T$.

H =	$\mathbf{H}^T$ =

3. Na temelju izračunatih matrica nacrtajte pripadni koder i dekoder (sindrom generator te sklop za ispravak jednostruke pogreške).

Nacrtajte sindrom generator i dekoder (8, 2) koda!

4. Ako ste za koder koristili simulacijsku datoteku na sustavu MOODEL, nacrtajte je racionalnije.

Nacrtajte koder (8, 2) koda!

5. Koliko ima informacijskih vektora i koji su?

6. Koliko ima kodiranih riječi i koje su?

7. Koliko maksimalno pogrešaka može ispraviti ovaj kod (vidi Tablicu 6.3)?

8. Koliko bitova ima paritetan dio kodirane riječi i prikažite/označite ih?

9. Koliko se ukupno riječi može napraviti od broja bitova paritetnoga dijela matrice **G** i kojom formulom?

10. Gdje je smještena jedinična matrica u generator matrici?

11. Koje riječi iz pogrešaka ne može ispraviti ovaj kod (vidi opis i Tablicu 6.3)?

12. Koliki je ukupan broj mogućih kodnih riječi? Koliko ostaje neiskorištenih riječi u ovome kodu?