Zaštitno kodiranje signala

Laboratorijska vježba 3

SP

Koncept kodiranja i dekodiranja blok-kodovima. Hammingov (7, 4) kod

1. dio

Student:

prezime	Ime	mat. broj
	Laboratorij	Datum

3. 3. KONCEPT KODIRANJA I DEKODIRANJA BLOK-KODOVIMA. HAMMINGOV (7, 4) KOD – 1. DIO

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:

Sadržaj:

Sazdano od:

Napomena: Cjelovit opis vježbe nalazi se na: MOODLE

Popis simulacijskih krugova:

Slika2.1.circ, Slika 2.1_1.circ, Zbroji 2. riječi od 4 bita.circ, Slika matrica H.circ, Slika Count Down.circ, Hammingov sustavan (7, 4) kod.circ, Slika 3_4.circ, Slika 3.4.circ, Slika 3.4a.circ, Slika4.1ponovljena.circ, Hamming i Pretvorba BIN2DEK.circ,

Slika2.1.circ, Slika 2.1_1.circ, Zbroji 2. riječi od 4 bita.circ, Slika matrica H.circ, Slika Count Down.circ, Hammingov sustavan (7, 4) kod.circ, Slika 3_4.circ, Slika 3.4.circ, Slika 3.4a.circ, Slika4.1ponovljena.circ, Hamming i Pretvorba BIN2DEK.circ,

3.1. 3.1. Pojam Hammingove udaljenosti

U kontinuiranim varijablama, možemo izmjeriti udaljenost Euklidovih pojmova kao što su duljina, kutovi i vektori. U binarnome svijetu, udaljenosti se mjere između dviju binarnih riječi prema takozvanoj Hammingovoj udaljenosti. Hammingova udaljenost je broj neslaganja između dvaju binarnih nizova iste dužine. To je mjera koliko su razdvojeni binarni objekti.

Hammingova udaljenost između nizova 001 i **1**01 je = 1 (nizovi se razlikuju na 1. mjestu)

Hammingova udaljenost između nizova 0011001 i **1**0**1**0**1**0**0** je = 4 (nizovi se razlikuju u 4 mjesta)

Hammingova udaljenost (*distance*) i težina (*weight*) su vrlo važni i korisni koncepti pri kodiranju. Poznavanje Hammingove udaljenosti koristi se kako bi se utvrdila sposobnost koda za otkrivanje i ispravak pogrešaka. Iako je Hammingova težina našega izabranoga kodnoga skupa 3, minimalna Hammingova udaljenost je 1. Kodna riječ 1011 i kodna riječ 1010 razlikuju za samo u jednome bitu. Jedna pogreška bita može pretvoriti niz 1011 u niz 1010. Prijemnik što primi niz 1010, prepoznat će ga kao valjanu kodnu riječ i nikada neće znati da je zapravo poslan niz 1011. Dakle, niz ima samo jedan pogrešan bit pa se jedna valjana kodna riječ pretvorila u drugu valjanu kodna riječ tako da se ne uočava pogreška. Broj 2 se onda može pretvoriti u broj 8 jednim pogrešnim bitom na prvome mjestu.

Ako sada želimo produljiti izabran kodni skup jednim bitom parnosti, evo što nam je činiti. Ako je broj jedinica neparan, tada dodavanjem jedne 1 proširimo niz, a ako je broj jedinica paran, onda ćemo dodati nulu na kraju niza. Kodni prostor se sada udvostručuje. On ide sa 16 na 32 moguće kodne riječi budući da je kodni prostor jednak 2^n, gdje je n duljina niza. Može produžiti niz opet koristeći isti proces, ali sada se kodni prostora povećava na 64. U prvome slučaju broj važećih kodnih riječi je 10 od 32 u drugome slučaju on je još uvijek 10, ali od 64, kao što prikazuje stupac 4 u tablici 3.1.

Tablica 3.1: Kodni prostor u odnosu na valjane kodne riječi, proširuje niz dodavanjem parnosti

Cjelobrojan niz od 4 bita, proširen jednim bitom pa onda još jedanput proširen jednim bitom.

0	0000	00000	000000
1	0001	00011	000110
2	0010	00101	001010
3	0011	00110	001100
4	0100	01001	010010
5	0101	01010	010100
6	0110	01100	011000
7	0111	01111	011110
8	1001	10010	100100
9	1010	10100	101000
	Ne koristi se preostalih 6	Ne koristi se preostalih 22	Ne koristi se preostalih 54

Sada imamo tri izbora za kod, s riječima od 4, 5 ili 6 bitova. Hammingova težina prvoga skupa je 3, drugoga 4, a posljednjega 5. Minimalna Hammingova udaljenost za prvi skup je 1. Minimalna Hammingova udaljenost za drugi skup je 2 odnosno 3 bita za posljednji skup.

U prvome skupu, čak i pogreška jednoga bita ne može se dopustiti (jer se ne može otkriti). U drugome skupu, jedna pogreška može se otkriti, jer pogreška jednoga bita ne pretvara kodnu riječi u neku drugu. Tako je sposobnost otkrivanja pogreške jednaka 1. U posljednjemu skupu, možemo otkriti do 2 pogreške. Dakle, njegova sposobnost otkrivanja pogreške je 2 bita.

Možemo poopćiti ovo pa je najveći broj pogrešaka bitova koje se mogu otkriti:

$$t = d_{\min} - 1$$

gdje je $d_{\min}$ Hammingova udaljenost kodnih riječi. Za kod s $d_{\min} = 3$ možemo otkriti i 1 i 2 pogrešna bita.

Dakle, želimo imati kodni skup s toliko velikom Hammingovom udaljenosti što je više moguće, jer to izravno utječe na našu sposobnost otkrivanja pogrešaka. Također se kaže da ako je cilj imati veliku Hammingovu udaljenost, moramo napraviti naše kodne riječi što dužima. To premašuje optimalnu

veličinu kodnih riječi, smanjuje brzinu informacijskih bitova za određenu propusnost i potvrđuje poslovicu "*da je slama pojela kravu*"[1], čak i pri obradi signala.

3.2. 3.2. Broj pogrešaka što ih se može ispraviti

Otkrivanje pogrešaka je jedna stvar, a ispravak je nešto sasvim drugo. Za niz od 6 bitova, postoji 6 različitih načina da biste dobili 1 pogrešan bit, a postoji 15 različitih načina da se dobiju 2 pogrešna bita.

$$\text{Broj kombinacija od 2 bita} = \frac{6!}{2!\cdot 4!} = 15$$

Što je najvjerojatniji uzrok pogreške obzirom da se radi o Gaussovome kanalu? Najvjerojatniji događaj je pogreška od 1 bita pa pogreška od 2 bita itd.. Tu je i mogućnost pogreške tri bita, ali za kod od 6 bitova, maksimalno što možemo otkriti jesu dva bita. Ovo je prihvatljiv rizik, dok je pojava od 3 pogrešna bita vrlo malo vjerojatna, jer ako je prijelazna vjerojatnost p mala (<< 1), vjerojatnost dobivanja 3 pogreške je treća potencija pogreške brzine kanala,

$$P_E(N) = p^N$$

Naravno, doista je moguće dobiti čak 6 pogrešaka u nizu od 6 bitova, ali budući da nemamo način otkrivanja ovoga scenarija, dobili smo ono što se zove *neuspjeh dekodiranja* (*decoding failure*). Ono se također zove vjerojatnost neotkrivenih pogrešaka (*probability of undetected errors*) i predstavlja granicu onoga što se može ispraviti.

Broj pogrešaka koje možemo ispraviti zadan je kao

$$t(\text{int}) = \frac{d_{\min} - 1}{2}$$

Za kod s $d_{\min} = 3$, mogućnost ispravke je samo 1 bit.

3.3. 3.3. Stvaranje blok kodova

Blok kodovi određeni su s (n, k). Kod uzima k informacijskih bitova i računa (n–k) paritetnih bitova iz generator-matrice koda. Većina blok kodova sustavni su tako da informacijski bitovi ostaju nepromijenjeni dodavanjem paritetnih bitova ili na prednjem ili stražnjem dijelu informacijskoga niz.

3.3.1. 3.3.1. HAMMINGOV KOD, JEDNOSTAVAN LINEARAN BLOK-KOD

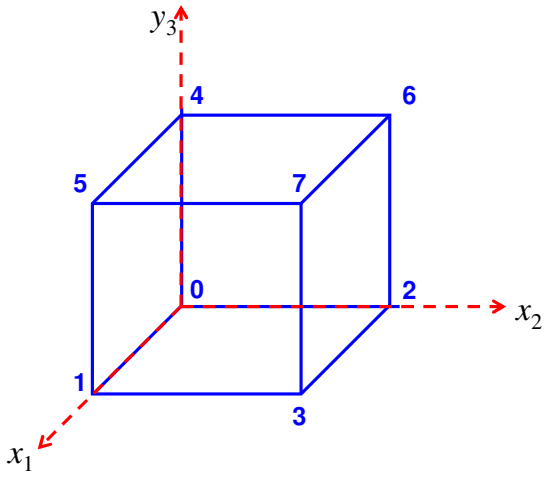
Hammingovi kodovi su najkorišteniji linearni blok kodovi. Hammingovi kodovi obično se navode kao (2^n–1, 2^n–n–1). Veličina bloka je jednaka 2^n–1. Broj informacijskih bitova u bloku jednaka je 2^n–n–1, a broj viška bitova jednaka je n. Svi Hammingovi kodovi u stanju su otkriti tri pogreške i ispraviti jednu. Uobičajene veličine Hammingovih kodova su (7, 4), (15, 11) i (31, 26). Svi ovi imaju istu Hammingovu udaljenost.

Pogledajmo (7, 4) Hammingov kod. U ovome kodu, duljina niza informacija je 4 bita pa postoji samo 16 mogućih nizova. Tablica 3.2 u nastavku sadrži popis svih tih 16 nizova.

[1] Sinonimi su: (1) Zlatno pravilo mehanike: „*Što dobiješ na sili, izgubiš na putu*!“, (2) „*Od drveća se ne vidi šuma*!“, (3) „*Što dobiješ na ćupriji, izgubiš na mostu*!“, …

Tablica 3.2: Mogućih 16 nizova iz riječi od četiri bita

Broj_Niza	i0	i1	i2	i3
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	0	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	0	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	0	0	0	0



Željeli bismo pridružiti tri paritetna bita gore navedenim informacijskim nizovima. Od 3 bita, moguće je napraviti 8 različitih kombinacija. Zanemarujemo kombinaciju „sve 0“ i pišemo preostalih sedam kombinacija u tablici 3.3.

Tablica 3.3: Stvaranje (transponirane) paritetne matrice

niz	bitovi		
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

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

Minimalna Hammingova udaljenost je: 1

Označimo redove 1, 2 i 4. Svi ostali redovi su *linearne kombinacije* (1+2, 1+4, 2+4 i 1+2+4) tih *triju osnovnih vektora*. Ovo svojstvo prikladno je kada se dokazuje zašto ovdje opisan proces uspješno radi. Ali ovdje ne idemo previše duboko u teoriju, samo razmatramo praktično motrište onoga što ova jednostavna konstrukcija može učiniti za nas.

Nazovimo ovih sedam redaka: matrica $\mathbf{H}^T$. Ona se zove (*transponirana*) *paritetna matrica* koda. Neka $\mathbf{H}^T$ ima *n* redaka i *n–k* stupaca. Za (7, 4) kod, $\mathbf{H}^T$ matrica sastoji se od 7 redaka i 3 stupca. Ako želimo, možemo preuredili retke ove matrice na bilo koji način (svojstvo linearnih kombinacija). Jedino što treba imati na umu je želja da osnovni redovi (s jednom jedinicom u retku što tvori jediničnu matricu na glavnoj dijagonali) budu zajedno ili *na vrhu* ili *pri dnu*. Svaka promjena redoslijeda dat će nam nov Hammingov kod. Ima samo dva osnovna načina na koje možemo poredati stupce matrice $\mathbf{H}^T$, svaki od njih rezultirat će *različitim* kodom.

Matrica $\mathbf{H}_1^T = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ \hline 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$, različita je od $\mathbf{H}_2^T = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ \hline 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{array}\right]$ = (zrcalna slika lijeve matrice).

Obje matrice sastoje se od istih redaka ali u drugačijemu redoslijedu. Tvorba potrebnih matrica opisana je u nastavku.

$$\mathbf{H}^T = \begin{bmatrix} \mathbf{P} \\ \hline \mathbf{I}_{n-k} \end{bmatrix} \Rightarrow \mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}] \Rightarrow \mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}] \Rightarrow \mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$$

Sada stvaramo novu matricu koja se zove **G** matrica tako da uzmemo gornji dio matrice $\mathbf{H}^T$ i dodamo joj jediničnu matricu s lijeve strane. Prema gornjim formulama, iz **G** matrice izvodi se **H** matrica.

$$\mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \Rightarrow \mathbf{H} = \left[\begin{array}{cccc:ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \Rightarrow \mathbf{H}^T = \left[\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hdashline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

Tako se dobila nova matrica veličine (4×7). Ovo je naša generator-matrica **G**. Ona nam opisuje kako će se 16 informacijskih nizova (Tablica 3.2) preslikati u 16 važećih kodnih riječi od 7 bitova. Četiri retka matrice **G** (bez redaka jedinične matrice) mogu se poredati na 24 različita načina uz istu jediničnu matricu:

$$\left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \begin{array}{l} 1.\,redak \equiv \\ 2.\,redak \equiv \\ 3.\,redak \equiv \\ 4.\,redak \equiv \end{array}$$

≡ 1.	1.	1.	1.	1.	1.	2.	2.	2.	2.	2.	2.	3.	3.	3.	3.	3.	3.	4.	4.	4.	4.	4.	4.
≡ 2.	2.	3.	3.	4.	4.	1.	1.	3.	3.	4.	4.	1.	1.	2.	2.	4.	4.	1.	1.	2.	2.	3.	3.
≡ 3.	4.	2.	4.	3.	2.	3.	4.	1.	4.	1.	3.	2.	4.	1.	4.	1.	2.	2.	3.	1.	3.	1.	2.
≡ 4.	3.	4.	2.	2.	3.	4.	3.	4.	1.	3.	1.	4.	2.	4.	1.	2.	1.	3.	2.	3.	1.	2.	1.

Napraviti C++ zadatak za usporediti sve 24 kombinacije + jedinična matrica s desna i s lijeva različito položenih dijagonala!

3.3.1.1. Zadatak

Usporediti sve 24 kombinacije + jedinična matrica s desna i s lijeva različito položenih dijagonala ($\sum = 96$)!

Matrica **G** može se prikazati stupčanim vektorom osnovnih vektora **G** matrice.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

Matrica **G** sastoji se od 2 dijela, tzv. **P** matrice $(n\text{–}k \times k)$ i jedinične matrice **I** $(k \times k)$.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \overbrace{\begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} \\ \vdots & \vdots & \vdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}}^{(n-k)\times k} \vdots \overbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}^{k\times k}$$

$\mathbf{P}$ – matrica

$\mathbf{I}_k$ – jedinična matrica

Jedinična matrica smije se i može se nalaziti i s desne strane:

$$\mathbf{G} = \left[\begin{array}{ccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right],$$

a rezultati su potpuno isti.

Napomena:

$$\mathbf{H}^T = \left[\begin{array}{c} \mathbf{I}_{n-k} \\ \hline \mathbf{P} \end{array}\right] \Rightarrow \mathbf{H} = [\mathbf{I}_{n-k} | \mathbf{P}^T] \Rightarrow \mathbf{G} = [\mathbf{P} \mid \mathbf{I}_k]$$

Kako je $\mathbf{G} = [\mathbf{P} \mid \mathbf{I}_k]$, vidi se da je $\mathbf{H} = [\mathbf{I}_{n-k} \mid \mathbf{P}^T]$, gdje je $\mathbf{P}^T$ transponirana matrica $\mathbf{P}$ pa je $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$.

Dokaz (jedinična matrica u $\mathbf{G}$ matrici je s **DESNE** strane, a jedinična matrica u $\mathbf{H}^T$ matrici je s gornje strane) :

$$\mathbf{G} \cdot \mathbf{H}^T = \left[\begin{array}{ccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \cdot \left[\begin{array}{c} 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \\ \hline 1\ 1\ 1 \\ 0\ 1\ 1 \\ 1\ 0\ 1 \\ 1\ 1\ 0 \end{array}\right] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{0}.$$

$$\mathbf{H}^T = \left[\begin{array}{c} \mathbf{P} \\ \hline \mathbf{I}_{n-k} \end{array}\right] \Rightarrow \mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}] \Rightarrow \mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$$

Kako je $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$, vidi se da je $\mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}]$, gdje je $\mathbf{P}^T$ transponirana matrica $\mathbf{P}$ pa je $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$.

Dokaz (jedinična matrica u $\mathbf{G}$ matrici je s **LIJEVE** strane, a jedinična matrica u $\mathbf{H}^T$ matrici je s donje strane) :

$$\mathbf{G} \cdot \mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \cdot \left[\begin{array}{c} 1\ 1\ 1 \\ 0\ 1\ 1 \\ 1\ 0\ 1 \\ 1\ 1\ 0 \\ \hline 1\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{array}\right] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{0}.$$

Preuređenjem (premještanjem) redaka od 4 bita matrice $\mathbf{H}$, dovest će do različite matrice $\mathbf{G}$, a time će se promijeniti preslik pri kodiranju. Bitna stvar je to, da dva blok-koda iste veličine mogu stvoriti

potpuno drugačije preslikavanje, ovisno o tome kako su poredani redovi. Svako preslikavanje ima *istu Hammingovu težinu* i ima *iste mogućnosti ispravke/otkrivanja pogrešaka*.

Sada sve što se mora uraditi je pomnožiti informacijski vektor **u** matricom **G** za dobiti kod(ira)nu riječ **v**.

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G}$$

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, \ldots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \ldots + u_{k-1}\mathbf{g}_{k-1}.$$

Dakle, generator matrica **G** služi za preslik informacijskoga niz od 4 bita, npr. **u** = [0 1 1 0] pa pomoću nje dobijemo sljedeću kod(ira)nu riječ **v** duljine 7 bitova za prijenos preko kanala sa smetnjama.

$$\mathbf{v} = \underbrace{\begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}}_{\substack{\mathbf{u}-\textit{informacijski}\\ \textit{vektor}}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0 & 0 & 1 & 1 & 1\\ 0 & 1 & 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 0\end{bmatrix}}_{\mathbf{G}} = \begin{vmatrix} 0\cdot1+1\cdot0+1\cdot0+0\cdot0=0+0+0+0=0\\ 0\cdot0+1\cdot1+1\cdot0+0\cdot0=0+1+0+0=1\\ 0\cdot0+1\cdot0+1\cdot1+0\cdot0=0+0+1+0=1\\ 0\cdot0+1\cdot0+1\cdot0+0\cdot1=0+0+0+0=0\\ 0\cdot1+1\cdot0+1\cdot1+0\cdot1=0+0+1+0=1\\ 0\cdot1+1\cdot1+1\cdot0+0\cdot1=0+1+0+0=1\\ 0\cdot1+1\cdot1+1\cdot1+0\cdot0=0+1+1+0=0 \end{vmatrix} = [0\ 1\ 1\ 0\ 1\ 1\ 0]$$

Ako se jedinična matrica stavi s desne strane, za isti ulazni informacijski vektor od 4 bita, dobiju se potpuno različiti rezultati!

$$\mathbf{v} = \underbrace{\begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}}_{\substack{\mathbf{u}-\textit{informacijski}\\ \textit{vektor}}} \cdot \underbrace{\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 0 & 1 & 0 & 0 & 1 & 0\\ 1 & 1 & 0 & 0 & 0 & 0 & 1\end{bmatrix}}_{\mathbf{G}} = \begin{bmatrix} 0\cdot1+1\cdot0+1\cdot1+0\cdot1\\ 0\cdot1+1\cdot1+1\cdot1+0\cdot1\\ 0\cdot1+1\cdot1+1\cdot1+0\cdot0\\ 0\cdot1+1\cdot0+1\cdot0+0\cdot0\\ 0\cdot0+1\cdot1+1\cdot0+0\cdot0\\ 0\cdot0+1\cdot0+1\cdot1+0\cdot0\\ 0\cdot0+1\cdot0+1\cdot0+0\cdot1 \end{bmatrix} = \begin{bmatrix} 0+0+1+0\\ 0+1+1+0\\ 0+1+1+0\\ 0+0+0+0\\ 0+1+0+1\\ 0+0+1+0\\ 0+0+0+0 \end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\\0\\1\\0\end{bmatrix} [1\ 0\ 0\ 0\ 0\ 1\ 0]^T$$

Napraviti C++ zadatak za usporediti razne kombinacije!

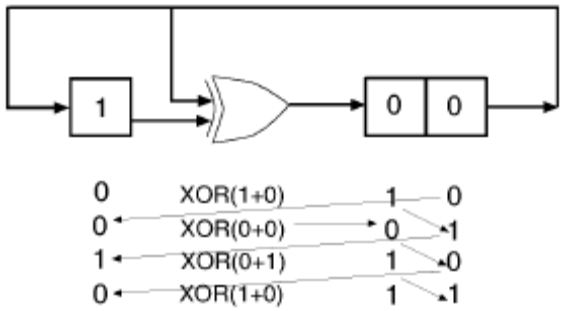
3.4. 3.4. Arhitektura kodera

Koder koristi posmični registar s linearnom povratnom vezom LFSR (*Linear Feedback Shift Register*). Korištenje LFSR, slika pokazuje kako se stvara matrica **H** korištenjem samo posmika i logike XOR.

Tablica 3.4: XOR zbrajanje modulo 2 (isto je kao i obično zbrajanje modulo 2). Pravilo je ili jedan ili drugi ali NE oba!

XOR+	**0**	**1**
0	0	1
1	1	0

Možemo proizvesti **H** matricu koja sadrži svih sedam redaka. Oni su prikazani su na slici 3.1.

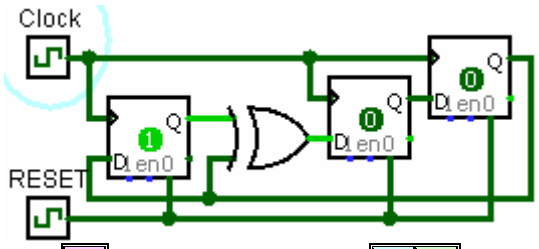


Slika 3.1: Korištenje LFSR za stvaranje matrice $\mathbf{H}^T$

i detaljnije u nastavku.

Varijacije s ponavljanjem od 2 (binarna) elementa {0, 1} ($n = 2$), 3. razreda ($r = 3$) (skupine od 3 binarne znamenke):

$$V_n^{'r} = n^r = n \cdot V_n^{'r-1} = 2^3 = 8$$

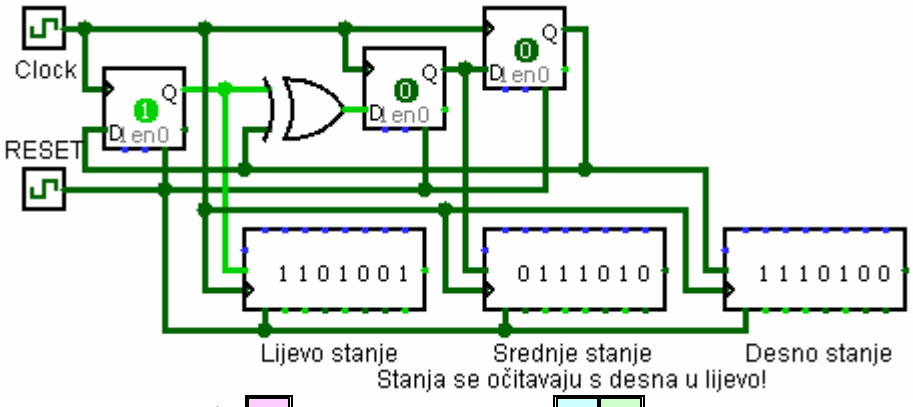


1.	1			0 0	100
2.	0	exor(1 + 0)	=	1 0	010
3.	0	exor(0 + 0)	=	0 1	001
4.	1	exor(0 + 1)	=	1 0	110
5.	0	exor(1 + 0)	=	1 1	011
6.	1	exor(0 + 1)	=	1 1	111
7.	1	exor(1 + 1)	=	0 1	101
1.	1	exor(1 + 1)	=	0 0	100

Koder koristi posmični registar s linearnom povratnom vezom LFSR (*Linear Feedback Shift Register*). Pri svakome paralelno upravljanome otkucaju (*clock*), sadržaji se prenose serijskim posmikom kroz sva stanja LFSR registara i upisuju se u $\mathbf{H}^T$ matricu.

$$\mathbf{G} = [\mathbf{P} \mid \mathbf{I}_k] \Rightarrow \mathbf{H} = [\mathbf{P}^T \mid \mathbf{I}_{n-k}] \Rightarrow \qquad \mathbf{H}^T = \begin{bmatrix} \mathbf{I}_{n-k} \\ \hline \mathbf{P} \end{bmatrix}$$

Za položaj $\mathbf{I}_k$ pod-matrice u $\mathbf{G}$ matrici s lijeve strane, također se može koristiti posmični registar s linearnom povratnom vezom LFSR (*Linear Feedback Shift Register*) ili uz drugačije konstruiran LFSR ili drugačije početne uvjete (110).



1.	1			0 0	100
2.	0	exor(1 + 0)	=	1 0	010
3.	0	exor(0 + 0)	=	0 1	001

4.	1	exor(0 + 1)	=	1	0	110
5.	0	exor(1 + 0)	=	1	1	011
6.	1	exor(0 + 1)	=	1	1	111
7.	1	exor(1 + 1)	=	0	1	101
1.	1	exor(1 + 1)	=	0	0	100

$$\mathbf{G} = [\mathbf{I}_k | \mathbf{P}] \Rightarrow \mathbf{H} = [\mathbf{P}^T | \mathbf{I}_{n-k}] \Rightarrow \mathbf{H}^T = \begin{bmatrix} \mathbf{P} \\ \hline \mathbf{I}_{n-k} \end{bmatrix}$$

Napraviti matricu za zbrajanje modulo 2 binarnih brojeva!

+	0	1
0	0	1
1	1	0

Zbrajanje modulo 2

–	0	1
0	0	1
1	1	0

Oduzimanje modulo 2

·	0	1
0	0	0
1	0	1

Množenja modulo 2

/	0	1
0		0
1		1

Dijeljenje modulo 2

1. Paritetna matrica Hammingova (7, 4) koda

```
//Paritetna matrica Hammingova (7, 4) koda
#include <iostream>
using namespace std;

void varijacija3bita();
int exili();
void ispis_koda(int);
void h_matrica();

int registar[3]={1,0,0},kod[7],exor;

void main(){
varijacija3bita(); //varijacije s
ponavljanjem (bez varijacije 000)
h_matrica();
}

void varijacija3bita(){
for(int i=0;i<7;i++){
ispis_koda(i+1);
//getchar();
exili();
registar[0]=registar[2];
registar[2]=registar[1];
registar[1]=exor;
}
}

int exili(){
if(registar[0]==registar[2])exor=0;
else exor=1;
return(exor);
}
void ispis_koda(int a){
cout<<"kod["<<a<<"] =
"<<registar[0]<<registar[1]<<registar[2]<<end
l;
```

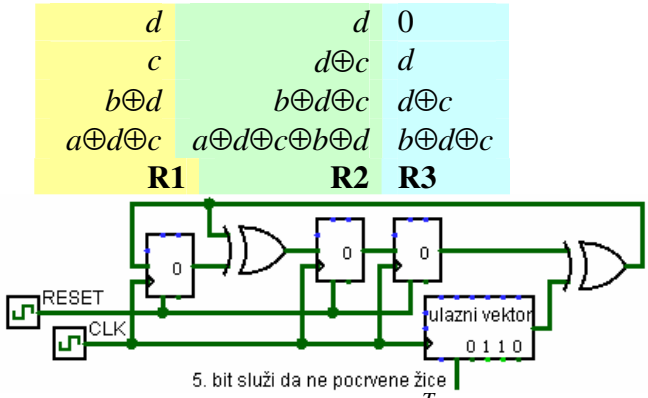
```
C:\windows\system32\cmd.exe
kod[1] = 100
kod[2] = 010
kod[3] = 001
kod[4] = 110
kod[5] = 011
kod[6] = 111
kod[7] = 101
Press any key to continue . . .
```

```
}

void h_matrica(){
// Napraviti H matricu za zbrajanje modulo 2
binarnih brojeva!
}
```

3.5. 3.5. Arhitektura kodera (B. Arazi)

Sljedeću arhitekturu kodera, predložio je B. Arazi, a na temelju navedenoga LFSR, domišljatošću i uz malo truda proizvodi se kodna riječ. Korištenjem LFSR na slici 3.1 proizvode se retci matrice $\mathbf{H}^T$, a punimo ga informacijskim vektorom, [*a b c d*]. Posmak niza tih bitova kroz ovaj LFSR daje nam sljedeće sadržaje u tri posmična registra. Slika 3.2 prikazuje kako se iz informacijskih bitova, automatski računaju paritetni bitovi.



Slika 3.2: Koder koda čija matrica pariteta je $\mathbf{H}^T$

Zadnji red daje tri paritetna bita koji se kao rep pridružuju informacijskim bitovima u nizu za stvoriti valjanu kodnu riječ.

Također možemo onda reći da su jednadžbe za paritetne bitove:

$p_1 = a + d + c$

$p_2 = a + d + c + b + d$

Napomena: *d* se zbraja 2 puta pa se odmah može skratiti (operacija EXOR).

$p_3 = b + d + c$

Za niz 0 1 1 0, dobivamo

$p_1 = 0 + 0 + 1 = 1$

$p_2 = 0 + 0 + 1 + 1 + 0 = 0$

$p_3 = 1 + 0 + 1 = 0$

Matematički okvir tvorbe kod(ira)nih riječi s paritetnim bitovima na početku kodne riječi je:

$$p_1 p_2 a p_3 b c d.$$

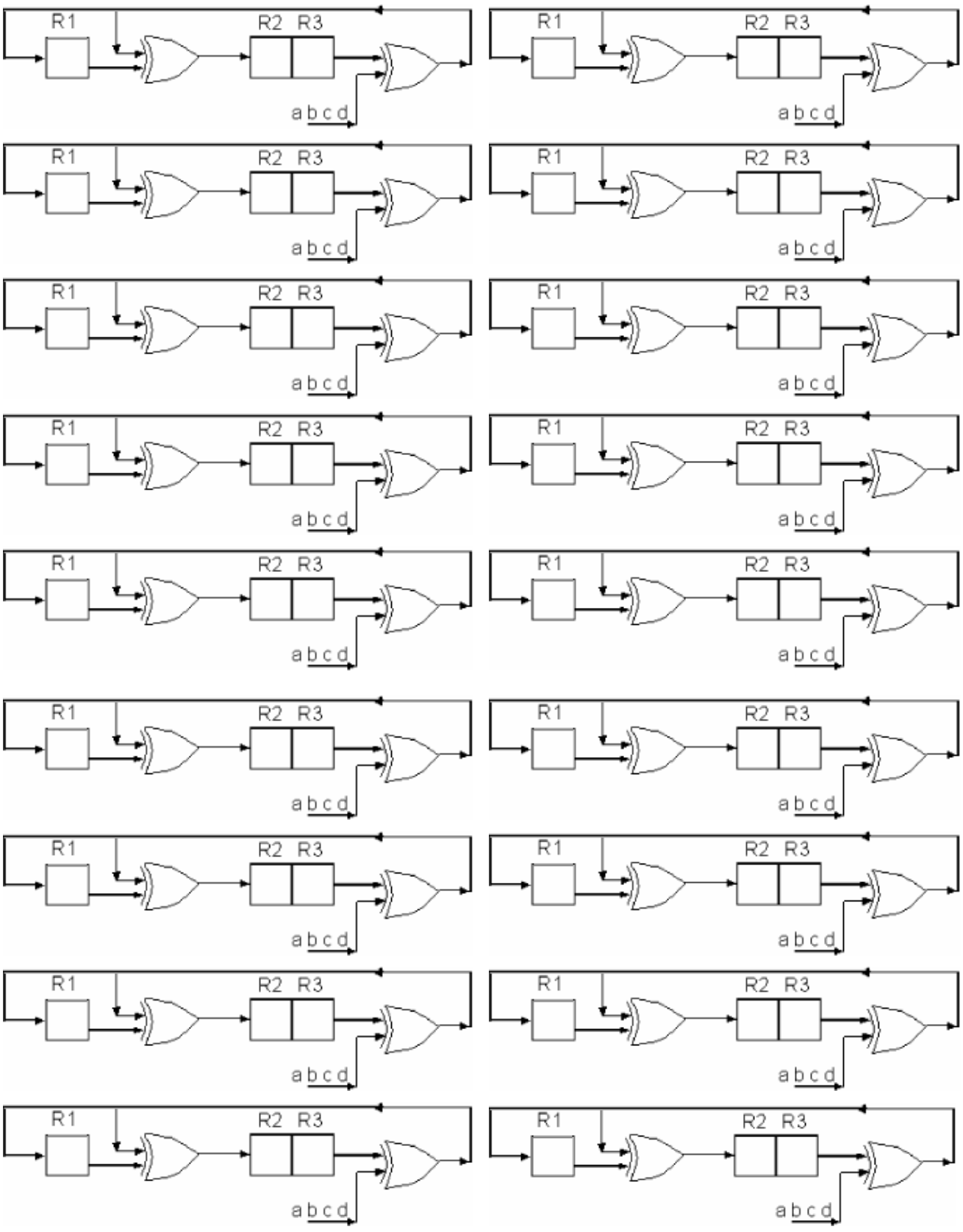
Kodna riječ je 1 0 0 0 1 1 0 i razlikuje se od one prethodno izračunate, jer je ovdje polazna točka registara bila drugačija.

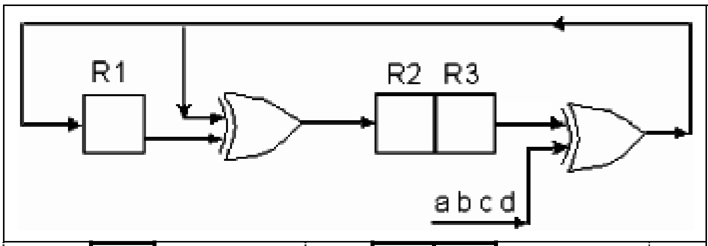
3.5.1. VJEŽBA - PUNJENJE REGISTARA INFORMACIJSKIM BITOVIMA

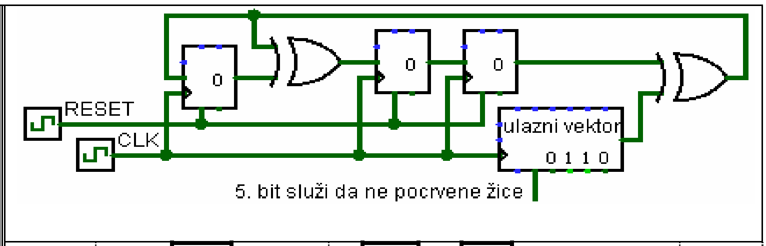
Zadatak: abcd =

d =	c =	b =	a =

Koje stanje imaju registri nakon unosa vektora [abcd]







	0	=		0	0	=				0	=		0		0	=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	
		=				=					=					=	

3.6. 3.6. Dekodiranje

Glavni koncept pri dekodiranju je odrediti koja od važećih 2^k kodnih riječi je prenesena s obzirom da se dobila jedan od 2^n riječi. Nastavit ćemo našim primjerom (7, 4) blok koda.

Prisjetimo se umnoška vektora **a** i matrice **B**. On se definira za slučaj kada je *broj stupaca vektora* jednak *broju redaka matrice*. Rezultat je vektor **c:** [**a**(1, 7) × **B**(7, 3) = **c**(1, 3) = **c**(3, 1)^T]

$$\mathbf{a}\cdot\mathbf{B} = \overset{(1,7)}{\begin{bmatrix} a_1 & a_2 & \cdots & a_7 \end{bmatrix}} \cdot \overset{(7,3)}{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \\ b_{51} & b_{52} & b_{53} \\ b_{61} & b_{62} & b_{63} \\ b_{71} & b_{72} & b_{73} \end{bmatrix}} =$$

$$= \overset{(1,3)}{\begin{bmatrix} a_1\cdot b_{11} + a_2\cdot b_{21} + \cdots + a_7\cdot b_{71} & a_1\cdot b_{12} + a_2\cdot b_{22} + \cdots + a_7\cdot b_{72} & a_1\cdot b_{13} + a_2\cdot b_{23} + \cdots + a_7\cdot b_{73} \end{bmatrix}} =$$

$$= \overset{(3,1)}{\begin{bmatrix} a_1\cdot b_{11} + a_2\cdot b_{21} + \cdots + a_7\cdot b_{71} \\ a_1\cdot b_{12} + a_2\cdot b_{22} + \cdots + a_7\cdot b_{72} \\ a_1\cdot b_{13} + a_2\cdot b_{23} + \cdots + a_7\cdot b_{73} \end{bmatrix}}^T .$$

ili za naš slučaj množimo redni vektor **r** i matricu $\mathbf{H}^T$:

$$\mathbf{s} = \mathbf{r}\cdot\mathbf{H}^T \Rightarrow [1\ redak \times 7\ stupaca]\ [7\ redaka \times 3\ stupca]\ \cdot = [1\ redak \times 3\ stupca]$$

Rezultat je redni vektor **s** (sindrom) koji ima strukturu [1 *redak* × 3 *stupca*]. Obično ga i pišemo u rednome (a ne transponiranome) obliku kao redni vektor $\mathbf{s} = [\Sigma_1\ \Sigma_2\ \Sigma_3]$ gdje su sume binarne znamenke 0 ili 1.

3.6.1. 3.6.1. SINDROM

Dekodiranje blok kodova prilično je jednostavno. Računamo nešto što se zove *sindrom* dolazne kodne riječi. To radimo množenjem dolaznoga vektora **v** i transponirane paritetne matrice $\mathbf{H}^T$. Prisjetimo se da smo na strani kodera, ulazni vektor u koder **u** množili matricom **G**, a dolazni vektor u dekoder, **r** množimo matricom $\mathbf{H}^T$ na strani dekodera.

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T$$

Veličina sindroma bit će jednaka ($n - k = 7 - 4 = 3$) bita. Za naš primjer (7, 4) koda, dužina sindroma je dakle, tri bita. Bez dokazivanja, dopustite ustvrditi kako sindrom "sve 0" znači da nije došlo do pogreške. Dakle, želimo da je *sindrom* nula.

Množimo primljen kod(ira)n vektor $\mathbf{r} = [0\ 1\ 1\ 0\ 1\ 1\ 0]$ matricu $\mathbf{H}^T$ za ustanoviti hoćemo li dobiti redni vektor **s** koji sadrži sve 3 znamenke jednake 0, jer znamo da smo onda primili ispravnu kodnu riječ.

$$\mathbf{H}^T = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ovdje se matrica **H** (a to je transponirana matrica $\mathbf{H}^T$) množi vektorom, jer je konačan rezultat isti.

$$\mathbf{s} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1\cdot0+0\cdot1+1\cdot1+1\cdot0+1\cdot1+0\cdot1+0\cdot0 \\ 1\cdot0+1\cdot1+0\cdot1+1\cdot0+0\cdot1+1\cdot1+0\cdot0 \\ 1\cdot0+1\cdot1+1\cdot1+0\cdot0+0\cdot1+0\cdot1+1\cdot0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = [0\ 0\ 0]^T$$

sindrom **s**↑ transponirana ↑ $\mathbf{H}^T$ matrica ↑ **r** dolazni vektor s kanala bez ↑ pogreške

Rezultat je uistinu $[0\ 0\ 0]^T$, sindrom sa samim nulama, a pišemo ga i ovako $\mathbf{s} = [0\ 0\ 0]$.

Pogledajmo kakav sindrom dobijemo kada se pojavi pogreška. Ako se primi vektor [1 1 1 0 1 1 0], prvi bit lijevo je primljen obrnuto (umjesto 0 dobili smo 1). Izračunat ćemo sindrome.

3.6.1.1. Zadatak 1:

Na temelju **G** matrice, simulacijskim sklopom odabrati ispravnu kodnu riječ sustavnoga (7, 4) Hammingovoga koda. Matematički ispitati sve uzorke pogrešaka te odrediti sindrome pogrešaka.

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

3.6.1.2. Zadatak 2:

Ispitati sve uzorke pogrešaka matematički te odrediti sindrome pogrešaka ako znamo da je ispravna kodna riječ jednaka [0 1 1 0 1 1 0]. Rezultate provjeriti simulacijskim sklopom ako znamo generator matricu sustavnoga Hammingovoga (7, 4) koda.

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

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [1 1 1 0 1 1 0] (pogrešan je 7. bit)

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

Sindrom je $[1\ 1\ 1]^T$, ispravljen vektor je: [1 1 1 0 1 1 0] ⊕ [1 0 0 0 0 0 0] = [0 1 1 0 1 1 0]

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 0 1 0 1 1 0] (pogrešan je 6. bit)

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

Sindrom je $[0\ 1\ 1]^T$, ispravljen vektor je: [0 0 1 0 1 1 0] ⊕ [0 1 0 0 0 0 0] = [0 1 1 0 1 1 0].

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 1 0 0 1 1 0] (pogrešan je 5. bit)

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

Sindrom je $[1\ 0\ 1]^T$, ispravljen vektor je: [0 1 0 0 1 1 0] ⊕ [0 0 1 0 0 0 0] = [0 1 1 0 1 1 0].

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 1 1 1 1 1 0] (pogrešan je 4. bit)

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

Sindrom je $[1\ 1\ 0]^T$, ispravljen vektor je: [0 1 1 1 1 1 0] ⊕ [0 0 0 1 0 0 0] = [0 1 1 0 1 1 0].

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 1 1 0 0 1 0] (pogrešan je 3. bit)

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

Sindrom je $[1\ 0\ 0]^T$, ispravljen vektor je: [0 1 1 0 0 1 0] ⊕ [0 0 0 0 1 0 0] = [0 1 1 0 1 1 0].

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 1 1 0 1 0 0] (pogrešan je 2. bit)

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

Sindrom je $[0\ 1\ 0]^T$, ispravljen vektor je: [0 1 1 0 1 0 0] ⊕ [0 0 0 0 0 1 0] = [0 1 1 0 1 1 0].

Ispravan vektor je [0 1 1 0 1 1 0], primio se vektor [0 1 1 0 1 1 1] (pogrešan je 1. bit)

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

Sindrom je $[0\ 0\ 1]^T$, ispravljen vektor je: [0 1 1 0 1 1 1] ⊕ [0 0 0 0 0 0 1] = [0 1 1 0 1 1 0]

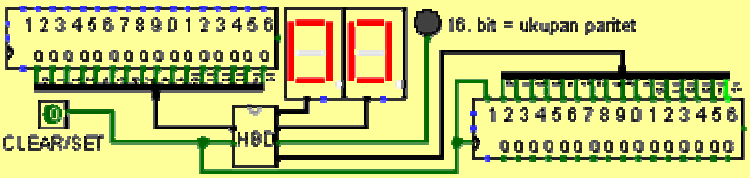
U tablici 3.5, dobivenim rezultatima popuniti stupac sindrom:

Tablica 3.5: Preslik sindroma u mjesto pogreške

uzorak pogreške								sindrom			broj bita
7.	6.	5.	4.	3.	2.	1.		2^0	2^1	2^2	
0	0	0	0	0	0	0		0	0	0	0
1	0	0	0	0	0	0		1	1	1	7
0	1	0	0	0	0	0		0	1	1	6
0	0	1	0	0	0	0		1	0	1	5
0	0	0	1	0	0	0		1	1	0	4
0	0	0	0	1	0	0		1	0	0	3
0	0	0	0	0	1	0		0	1	0	2
0	0	0	0	0	0	1		0	0	1	1

$$x^2 + s_1 x + s_1^2 + s_3 s_1^{-1} = \mathbf{0} \tag{3.1}$$

~~Sklop za računanje rednoga broja neispravnoga bita ovoga niza ($1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2$) dobije se broj 7.~~



Slika 3.3: Hamming i Pretvorba BIN2DEK.circIma pogrešku prelijeva u pod-krugu: HAM+D2D Decd+razdjelnik (*splitter*)

3.7. 3.7. Tablica sindroma

Tablica sindroma može se unaprijed stvoriti u dekoderu, tako da se može brzo pronaći mjesto na kojemu je došlo do pogreške. To se postiže promjenom ispravnoga kodnoga vektora jedan bit istovremeno i promatranjem izračunatoga sindroma. Za sindrom ovoga koda, dobili smo tablicu 3.5.

Evo još jednoga primjera, ovdje je pogrešno primljen 7. bit.

$$\mathbf{s} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1\cdot1+0\cdot1+1\cdot1+1\cdot1+1\cdot1+0\cdot1+0\cdot0 \\ 1\cdot1+1\cdot1+0\cdot1+1\cdot1+0\cdot1+1\cdot1+0\cdot0 \\ 1\cdot1+1\cdot1+1\cdot1+0\cdot1+0\cdot1+0\cdot1+1\cdot0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = [1\ 1\ 1]^T$$

Ovdje je sindrom $\mathbf{s} = [1\ 1\ 1]^T$. Ovim podatkom ulazi se u tablicu 3.5, pronađe se tzv. *vođa koskupa* (*coset leader*) pa se očita mjesto pogreške. Iz tablice dekodiranja, ovome sindromu $\mathbf{s} = [1\ 1\ 1]^T$ odgovara uzorak pogreške $\mathbf{e} = [1000000]$. Stoga je ispravna kodna riječ:

$$\mathbf{y} = \mathbf{r} + \mathbf{e} = [\mathbf{1}\ 1\ 0\ 1\ 0\ 1\ 1] \oplus [1\ 0\ 0\ 0\ 0\ 0\ 0] = [1\ \mathbf{0}\ 0\ 1\ 0\ 1\ 1]$$

Iz tablice 3.5 vidimo da je bit broj 7 pogrešno primljen. Sindrom nam govori ne samo da je došlo do pogreške, već i koji bit (njegov položaj) je pogrešno primljen tako da se može ispraviti. Sve pojedinačne bitove možemo otkriti i ispraviti na ovaj način.

Budući da je vjerojatnost jedne pogreške mnogo veća od t+1 pogreške, ova metoda zove se *dekodiranje maksimalnom vjerojatnosti* MLD (*Maximum Likelihood Decoding*). MLD je najučinkovitija metoda, iako druge metode, kao što je metoda *najbližega susjeda* (*nearest neighbor*), dovode do nešto boljih rezultata ali na račun brzine dekodiranja i memorije.

Kada se dogode dvije pogreške, kao u slučaju vektora ispod, dobili smo ne nulti sindrom. Za ne nulti sindrom, možemo ispraviti pogrešku na temelju tablice sindroma, ali to nam ostavlja mogućnost neopaženoga prolaza pogreške.

$$c_{sent} = 0110110$$

$$c_{sentrcvd} = 0110000$$

Bitovi broj 2 i 3 obrnuto su primljeni.

Sindrom za ovaj vektor je

$$\mathbf{s} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1\cdot 0+0\cdot 1+1\cdot 1+1\cdot 1+1\cdot 0+0\cdot 0+1\cdot 0 \\ 1\cdot 0+1\cdot 1+0\cdot 1+1\cdot 1+1\cdot 0+0\cdot 0+0\cdot 0 \\ 1\cdot 0+1\cdot 1+1\cdot 1+0\cdot 1+0\cdot 0+0\cdot 0+1\cdot 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = [1\ 1\ 0]^T$$

Ovdje je također sindrom $\mathbf{s} = [1\ 1\ 0]^T$, ali ovo uzrokuje promjenu primljenoga vektora u

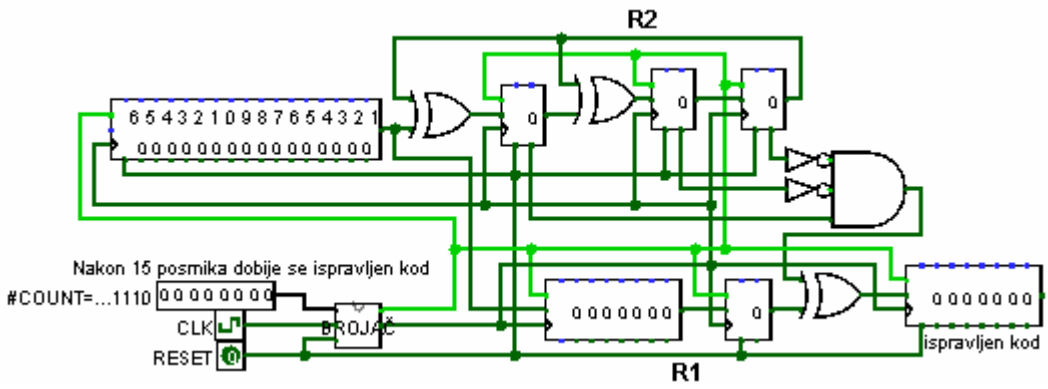
$$\mathbf{y} = \mathbf{r} + \mathbf{e} = [0\ 1\ 1\ 0\ \mathbf{0}\ \mathbf{0}\ 0] + [0\ 0\ 0\ 1\ 0\ 0\ 0] = [0\ \mathbf{1}\ \mathbf{1}\ \mathbf{1}\ 0\ 0\ 0]$$

što je očito pogrešno, jer je poslan informacijski vektor $\mathbf{u} = 0110110$.

Što se događa kada imamo tri ili četiri pogrešku. Do 3 pogreške, sindrom će biti različit od nule, ali bilo koliki broj pogrešaka veći od 3 može dati [000] sindrom, tako da blok s pogreškama prolazi neopaženo.

3.8. 3.8. Struktura dekodiranja

Sljedeći sklop automatski ispravlja pogrešku jednoga bita. On koristi LFSR krugove iz slika 2 i 3 s dodatkom I vrata. Kako se poruka pomiče, R2 registar proizvodi redove **H** matrice. Nakon *n* posmika, registar ponovno ulazi u stanje 000. Dokle god I vrata ne proizvedu 1, dolazna poruka što se posmiče registrom, ostaje neizmijenjena. Ostatak je ostavljen kao vježba vrijednome studentu za rad na detaljima.



Slika 3.4: Dekoder koji automatski ispravlja jednostruke pogrešne bitove