Marijo Nižetić

Zaštitno kodiranje signala

Kodiranje i dekodiranje konvolucijskim kodovima

Laboratorijske vježbe
(radni materijal)

Sadržaj

Izvor: N. Rožić, *Informacije i komunikacije-kodiranje s primjenama, Fakultet elektrotehnike, strojarstva i brodogradnje, Sveučilište u Splitu*, NIP "Alinea", Zagreb, 1992. (CH 12)

1. 1. KODIRANJE I DEKODIRANJE KONVOLUCIJSKIM KODOVIMA

1.1. Uvod

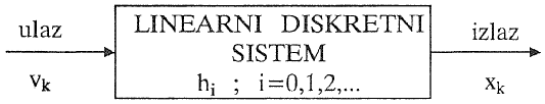
U ***blok-kodovima***, svaka *m*-torka informacijskih simbola definira, preko 1:1 preslikavanja (injekcije), kontrolnu skupinu od *n–m* simbola u kôdnoj riječi sustavnoga, odnosno definira kompletnu kôdnu riječ (*n*-torku) nesustavnoga kôda. Ovakva operacija preslikavanja odgovara sustavu ***bez memorije*** tako da je izlaz iz sustava (kôdna riječ) određen isključivo trenutnim ulazom (*m*-torkom informacijskih simbola).

Za razliku od ovoga, ***konvolucijski kôd*** predstavlja izlaz iz ***sustava s memorijom***. Kodna riječ, tj. *n*-torka, ne ovisi dakle samo o trenutnoj ulaznoj *m*-torci, nego i o prethodnim ulaznim *m*-torkama. Na taj način ne postoji jednostavno 1:1 preslikavanje između ulazne skupine od *m* informacijskih simbola i izlazne skupine od *n* kôdnih simbola, nego se struktura kôda proteže na cijeli kodirani niz. Prisutnost memorije čini konvolucijske kodove matematički složenijima. Međutim, praktička rješenja ograničavaju se na linearne, vremenski invarijantne sustave, pa takvi kodovi mogu biti opisani relacijom ***konvolucije***. Posebno za binarne sustave, konvolucija se može implementirati pomoću jednostavnih posmičnih registara (SR) u kombinaciji sa zbrajalima po modulu 2.

Zanimljivo je također naglasiti, da su praktička rješenja konvolucijskih kodova ograničena na male vrijednosti *n* i *m* (do 4), za razliku od rješenja u blok-kodovima kod kojih *n* i *m* mogu biti reda 100 i više.

1.2. 1.2 KONVOLUCIJSKO KODIRANJE

Ovdje se matematički opisuju konvolucijski (*n*, *m*) kodovi što se temelje na teoriji diskretnih linearnih sustava s binarnim ulazima. Općenitiji pristup za višestruke ulaze jednostavan je, ali ne veliko praktičko značenje. Matematički opis konvolucijskoga kodiranja u potpunosti se temelji na *linearnoj algebri* (*teorija linearnih sustava*). Kauzalan linearan sustav s jednim ulazom i jednim izlazom ili tzv. SISO (*Single Input - Single Output*) sustav (slika 2).



Slika 2 Model linearnog SISO sustava

Sustav na slici 2, opisan je diskretnom konvolucijom:

$$x_k = \sum_{i=0}^{\infty} h_i v_{k-i} \qquad k = 0, \pm 1, \pm 2, \cdots \tag{1}$$

gdje su h_i; i = 0, 1, 2, …, vrijednosti odziva sustava na jedinični Diracov impuls uzeti u diskretnim trenucima $i \cdot \Delta t$. Relacija (12.1) može se pisati u kompaktnijem obliku preko operatora kašnjenja L, tj.

$$x_k = h(L)v_k, \tag{2}$$

gdje je:

$$h(L) = \sum_{i=0}^{\infty} h_i L^i = h_0 + h_1 L + h_2 L + \cdots \tag{3}$$

uz:

$$L^i v_k = v_{k-i}$$

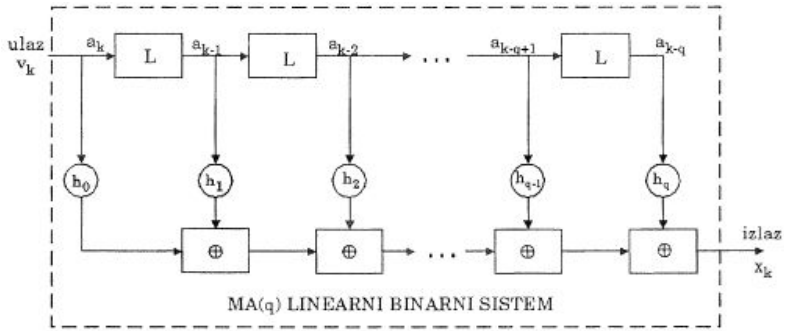
Za konvolucijske kôdere zanimljivi su linearni sustavi s konačnom memorijom, tako da $h(L)$ u (3) sadrži samo $q + 1$ element, tj. vrijedi:

$$h(L) = h_0 + h_1 L + h_2 L^2 + \ldots + h_q L^q \tag{4}$$

Relacija (4) predstavlja prijenosnu funkciju tzv. *MA*(*q*) modela. Za binarne sustave relacija (2) će predstavljati konvoluciju po modulu 2:

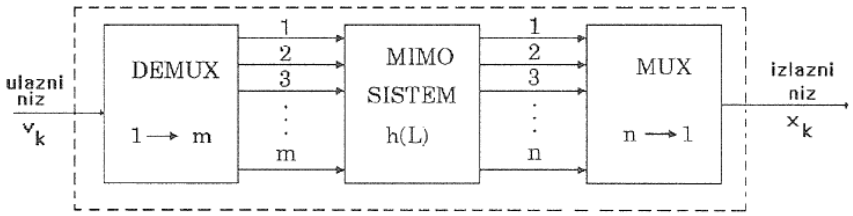
$$x_k = h(L) v_k. \tag{5}$$

Slika 12.3 ilustrira primjenu konvolucije (12.5) preko D bistabila sa zbrajalima po modulu 2.



Slika 12.3 Posmični registar SR kao MA(*q*) *linearni binarni sustav*

Unošenje redundancije zahtijeva povećanu brzinu signalizacije na izlazu iz kôdera. Ako je interval signalizacije ulaza Δt, izlazni interval signalizacije je $\rho \cdot \Delta D = m/n \cdot \Delta t$. Ovo je slično kao i za blok-kodove. Međutim, povećana brzina signalizacije, gledajući sa stajališta konvolucijskoga kôdera kao linearnoga sustava, zahtijeva da sustav ima više izlaza. Ustvari, neki (*n*, *m*) konvolucijski kôder, odgovara linearnome sustavu s *m* ulaza i *n* izlaza, pa je prikladno koristiti MIMO (*Multi Input-Multi Output*) modele kao na slici 12.4.

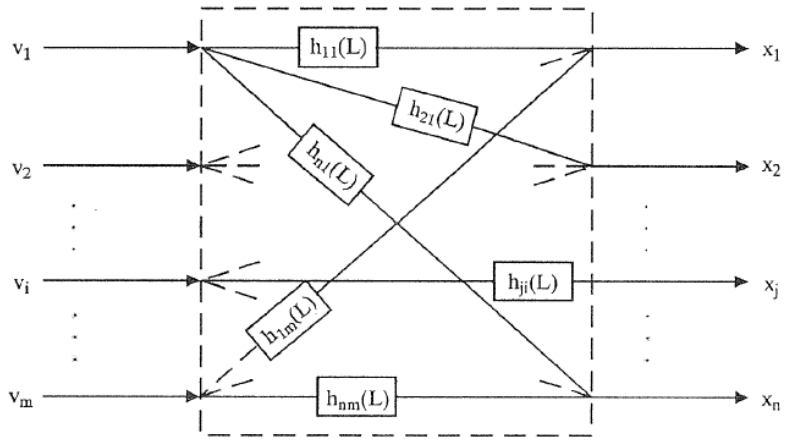


Slika 4 Blok-shema konvolucijskog (*n*, *m*) *kôdera*

Slično kao i u blok-kodovima, ulazni blokovi dužine *m* demultipleksiraju se u *m* kanala koji predstavljaju elemente *m*-komponentnoga vektora ulaza $\mathbf{v} = (v_1 v_2 \ldots v_m)$ u MIMO sustav čija je matrica odziva *h*(*L*). Vektor ulaza **v** mijenja se s intervalom $m \cdot \Delta t$ (tzv. ulazni okvir). Izlazni *n*-komponentni vektor $\mathbf{x} = (x_1 x_2 \ldots x_n)$, čiji je interval također $m \cdot \Delta t$, multipleksira se, a dobiveni izlazni niz predstavlja kodiranu skupinu, odnosno kôdnu riječ. Brzina signalizacije izlaznih simbola je, dakle, *n*/*m* puta veća od brzine signalizacije ulaznih simbola. Treba napomenuti da blok-shema na slici 12.4 vrijedi i za blok-kodove uz uvjet da MIMO sustav nema memoriju. MIMO sustav s *m* ulaza i *n* izlaza opisan je ($n \times m$) polinomskom matricom odziva *h*(*L*):

$$h(L) = \begin{bmatrix} h_{11}(L) & h_{12}(L) & \cdots & h_{1m}(L) \\ h_{21}(L) & h_{22}(L) & \cdots & h_{2m}(L) \\ \vdots & \vdots & & \vdots \\ h_{n1}(L) & h_{n2}(L) & \cdots & h_{nm}(L) \end{bmatrix} \tag{12.6}$$

gdje su elementi $h_{ji}(L)$ polinomi po *L*, stupnja do najviše *q* i predstavljaju odzive pojedinačnih podsustava koji povezuju *j*-ti izlaz s *i*-tim ulazom (slika 12.5).



Slika 5 Opća blok-shema MIMO sustava sa m ulaza i n izlaza

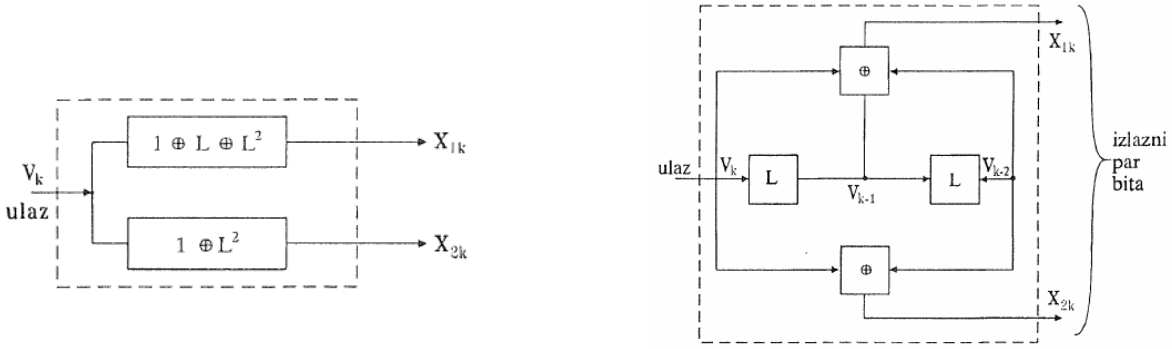
Općenito, ako su svi $h_{ji}(L)$ bez nazivnika, odnosno nazivnik je skalar, MIMO sustav opisan relacijom (12.6) je *MA* tipa reda q, tj. $MA(q)$. Ako neki element $h_{ji}(L)$ ima nazivnik u obliku polinoma, MIMO sustav je ARMA tipa. Maksimalan stupanj brojnika u (12.6) određuje vrijednost q, a maksimalan stupanj determinante od $h(L)$ određuje red sustava, tj. vrijednost p. Odgovarajući MIMO sustav u tome slučaju odgovara ARMA (p, q) modelu.

Važno je napomenuti da struktura $h(L)$ koja je zanimljiva općenito pa tako i za konvolucijske kodove, nije proizvoljna. Prikladne su one strukture koje omogućuju jednoznačno određivanje ulaza na osnovi poznatoga izlaza. Sa stajališta identifikacije sustava, riječ je o tzv. ***osmotrivim*** strukturama koje se mogu različito definirati. Jedan jednostavan uvjet je da polinomska matrica $h(L)$ sadrži elemente koji nemaju zajedničkih faktora izuzev skalarnih vrijednosti. Opširnija analiza može se naći npr. u [1], [2]. Jedan pristup ovome problemu sa stajališta prognoziranja, a posebno zanimljiv za ARMA strukture, definira tzv. prediktorski kanonski oblik [3] koji definira osmotrive strukture preko odnosa između stupnjeva pojedinih elemenata u autoregresijskome dijelu od $h(L)$.

Ono što je ovdje od osnovne važnosti, to je da neprikladna struktura $h(L)$ vodi u tzv. katastrofične konvolucijske kodove, o čemu će se nešto više pisati u poglavlju 12.4.

1. PRIMJER 1: MIMO sustav za (2, 1) kôder

Za primjenu je od osnovnoga interesa binarni konvolucijski kôd u kojemu se svaki informacijski simbol kodira u dva izlazna simbola, tako da je redundancija kôda $R = 1/2$. Izlazni par simbola ovisi o trenutnome ulaznom simbolu, ali i o q prethodnih ulaznih simbola. Jedno zanimljivo rješenje je određeno s $q = 2$. Odgovarajući MIMO sustav ima dakle 1 ulaz i 2 izlaza kao na slici 12.6.



a) blok-shema *b) sklopovska realizacija*

Slika 6 MIMO sustav za (2, 1) kôd

Vrijedi:

$$h(L) = \begin{bmatrix} h_{11}(L) \\ h_{21}(L) \end{bmatrix} = \begin{bmatrix} 1 \oplus L \oplus L^2 \\ 1 \oplus L^2 \end{bmatrix}$$

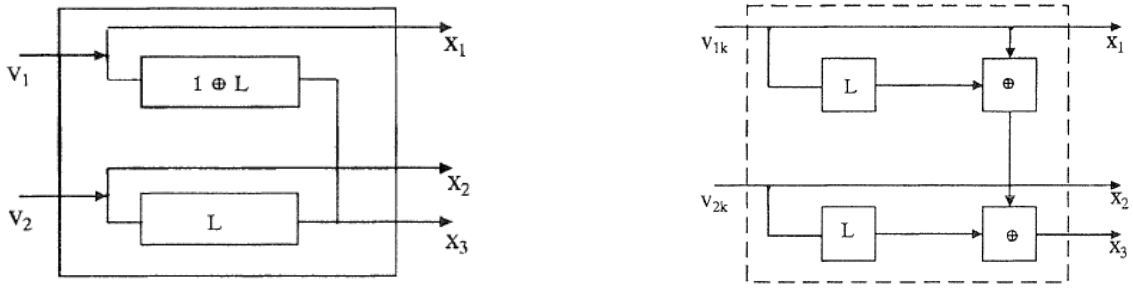
2. PRIMJER 2: MIMO sustav za (3, 2) kôder

Konvolucijski (3, 2) kôd nastaje kada se za svaki ulazni par simbola definiraju tri izlazna simbola koji ovise o trenutnome ulaznome paru v_1v_2 i o prethodnome paru, tj. Lv_1Lv_2. Redundancija kôda je 1/3, odnosno koeficijent prijenosa je $\rho = 2/3$.

Neka je dana matrica odziva:

$$h(L) = \begin{bmatrix} h_{11}(L) & h_{12}(L) \\ h_{21}(L) & h_{22}(L) \\ h_{31}(L) & h_{32}(L) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 \oplus L & L \end{bmatrix}$$

Slika 7a ilustrira blok-shemu odgovarajućeg MIMO sustava s 2 ulaza i 3 izlaza, a slika 7b odgovarajuću realizaciju pomoću D bistabila i zbrajala.



a) blok-shema b) sklopovska realizacija

Slika 7 MIMO sustav za (3, 2) konvolucijski kôd

U nekome trenutku k, izlaz iz MIMO sustava definira relacija konvolucije kao (2), tj.:

$$\mathbf{x}_k = h(L)\mathbf{v}_k \tag{7}$$

gdje je $\mathbf{x}_k$ (stupčasti) vektor od n komponenti, $\mathbf{v}_k$ (stupčasti) vektor m od komponenti, a $h(L)$ je $(n \times m)$ polinomska matrica definirana relacijom (12.6).

3. PRIMJER 3: Konvolucijsko (2, 1) kodiranje

Za $h(L)$ iz primjera 1, relacija (7) postaje:

$$\begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} = \begin{bmatrix} 1 \oplus L \oplus L^2 \\ 1 \oplus L^2 \end{bmatrix}$$

odnosno vrijedi:

$$x_{1,k} = v_k \oplus v_{k-1} \oplus v_{k-2}$$

$$x_{2,k} = v_k \oplus v_{k-2}$$

$$x_{2,k} = v_k \oplus v_{k-2}.$$

Neka je ulazni niz $\mathbf{v}_k$ = (00110111001). Budući da je $q = 2$, izlazni niz se računa od $k = 3$ pa nadalje, tako da uz početne vrijednosti $\mathbf{v}_1 = \mathbf{v}_2 = 0$ vrijedi tablica 1:

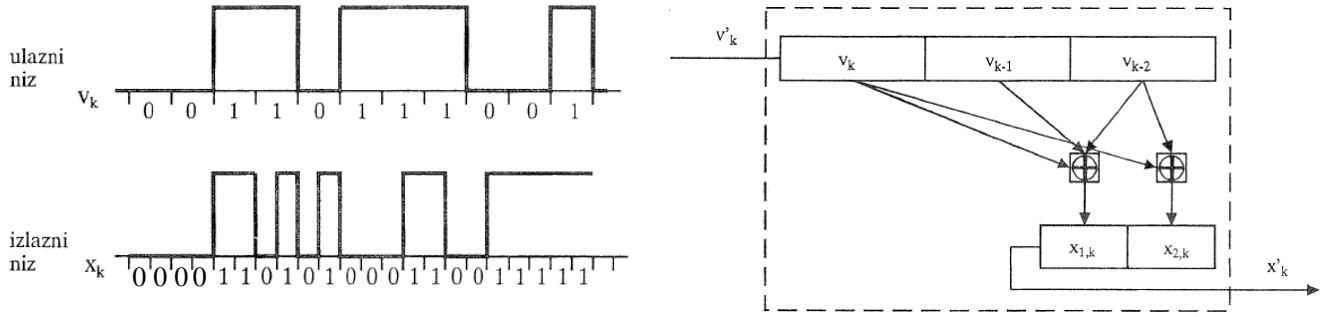
Tablica 1 Ulazni i izlazni digiti (2, 1) kôdera

k	1	2	3	4	5	6	7	8	9	10	11
$\mathbf{v}_k$	0	0	1	1	0	1	1	1	0	0	1
$\mathbf{x}_{1k}$	1	0	1	0	0	0	0	1	0	1	1
$\mathbf{x}_{2k}$	1	1	1	1	1	0	1	0	1	1	1

Zbog serijskoga prijenosa, izlaz iz konvolucijskog kôdera je niz tj.:

$$\mathbf{x}_k = 110101000110011111 \ldots; k = 3, \ldots, 11$$

Odgovarajući ulazni i izlazni binarni signali prikazani su na slici 8a, a odgovarajuća shema (2, 1) kôdera može se, na osnovi slike 1 i slike 6, prikazati kao na slici 8b.



a) ulazni i izlazni signal *b) blok-shema*

Sl. 8 Binarni (2, 1) kôder.

Za MIMO sustave s više ulaza i izlaza, polinomsku matricu $h(L)$ prikladnije je izraziti sljedećom relacijom:

$$h(L) = H_0 + H_1 L + H_2 L^2 + \ldots + H_q L^q \tag{8}$$

gdje su $\mathbf{H}_i$ $(n \times m)$ skalarne matrice čiji elementi predstavljaju koeficijente uz operator i-toga stupnja unutar elemenata od $h(L)$. Relacija (7) postaje:

$$x_k = \sum_{i=0}^{q} H_i L^i v_k \tag{9}$$

odnosno ako se nizovi $\mathbf{v}_k$ i $\mathbf{x}_k$ označe preko stupčastih vektora $\mathbf{v}$ i $\mathbf{x}$ čije su komponente simboli u nizu, vrijedi:

$$\mathbf{x} = \mathbf{H} \cdot \mathbf{v} \tag{10}$$

gdje je matrica $\mathbf{H}$ općenito polu-beskonačna blok-matrica dana s:

$$\mathbf{H} = \begin{bmatrix} H_0 & 0 & 0 & 0 & \cdots \\ H_1 & H_0 & & & \cdots \\ H_2 & H_1 & H_0 & & \cdots \\ \vdots & H_2 & H_1 & H_0 & \\ H_q & \vdots & H_2 & H_1 & \ddots \\ 0 & H_q & \vdots & H_2 & \ddots \\ 0 & 0 & H_q & \vdots & \ddots \\ 0 & 0 & 0 & H_q & \\ 0 & 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \tag{11}$$

Uz $q = \infty$, elementi matrice $\mathbf{H}$ u donjem trokutu jednaki su nuli.

4. PRIMJER 4: Konvolucijsko (3, 2) kodiranje

Primjer (3, 2) polinomske matrice $h(L)$ dan je u primjeru 2. Skalarne matrice $\mathbf{H}$, u (8) dane su sa:

$$\mathbf{H}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{H}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}.$$

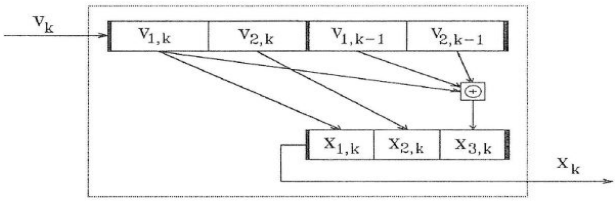
Neka je ulazni niz $\mathbf{v}$ = (11011011…) pa u skladu s relacijom (10) slijedi izlazni vektor:

$$x = \begin{bmatrix} x_{1,1} \\ x_{2,1} \\ x_{3,1} \\ x_{1,2} \\ x_{2,2} \\ x_{3,2} \\ x_{1,3} \\ x_{2,3} \\ x_{3,3} \\ x_{1,4} \\ x_{2,4} \\ x_{3,4} \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0 & & & & & & & & & & & & & \\ 0 & 1 & & 0 & & 0 & & 0 & & 0 & \cdots & & & & \\ 1 & 0 & & & & & & & & & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & 1 & 0 & & & & & & & & & & \\ 0 & 0 & & 0 & 1 & & 0 & & 0 & & 0 & \cdots & & & \\ 1 & 1 & & 1 & 0 & & & & & & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & 0 & 0 & & 1 & 0 & & & & & & & \\ & 0 & & 0 & 0 & & 0 & 1 & & 0 & & 0 & \cdots & & \\ & & & 1 & 1 & & 1 & 0 & & & & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & & & 0 & 0 & & 1 & 0 & & & & \\ & 0 & & 0 & & & 0 & 0 & & 0 & 1 & & 0 & \cdots & \\ & & & & & & 1 & 1 & & 1 & 0 & & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & & & & & 0 & 0 & & 1 & 0 & & \\ & 0 & & 0 & & 0 & & & 0 & 0 & & 0 & 1 & \cdots & \\ & & & & & & & & 1 & 1 & & 1 & 0 & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & & & \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ \vdots \end{bmatrix}$$

što daje:

$$\mathbf{x} = (111\ 010\ 100\ 110\ \ldots)$$

Slika 9 ilustrira realizaciju odgovarajućeg (3, 2) kôdera koji se temelji na općoj shemi sa slike 1 i MIMO sustavu na slici 7b.



Slika 9 Binarni (3, 2) kôder

Treba napomenuti da se relacija (10), koja definira konvolucijski (*n*, *m*) kôd i koja je proizašla kao rezultat tretiranja ovoga kôda preko MIMO linearnih sustava, ponešto razlikuje od standardne notacije uobičajene za blok-kodove. Za blok-kodove vrijedi relacija oblika:

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

gdje su **x** i **v** redni vektori s *n* i *m* komponenti, a **G** je ($m \times n$) generator-matrica. Proizlazi da je generator-matrica konvolucijskoga kôda jednaka transponiranoj matrici **H** definiranoj relacijom (11). Vrijedi dakle:

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & \ldots & G_q & & \\ & G_0 & G_1 & G_2 & \ldots & G_q & 0 \\ & & G_0 & G_1 & G_2 & \ldots & G_q \\ & 0 & & G_0 & G_1 & G_2 & \ldots & G_q \\ & & & & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \qquad (12.12)$$

gdje su podmatrice $\mathbf{G}_i = \mathbf{H}_i^T$ ($m \times n$) matrice koeficijenata i-te potencije u $h(L)$. Ekvivalentno, ***polinomska generator-matrica*** $\mathbf{G}(\mathbf{L})$ bit će transponirana matrica odziva $h(L)$, tj.:

$$G(L) = h^T(L) \tag{12.13}$$

Obzirom na strukturu matrice **G** u (12), dužina kôdnih riječi u konvolucijskome kôdu ne može se jednoznačno definirati. Dužina kôdnih riječi (n, m) konvolucijskoga kôda može se interpretirati kao onaj broj izlaznih bitova (multipleksiranih u niz) koji je rezultat utjecaja trenutnoga ulaznoga bĩta. Budući da kôder sadrži ukupno $q + 1$ stupanj (slika 1), može se definirati tzv. ***izlazna ograničena dužina***:

$$N = n\left[1 + max\left\{stupanj\ \ h_{ij}(L)\right\}\right]$$

tj.:

$$N = n(1 + q) \tag{12.14}$$

U literaturi se, pored navedene definicije, javljaju i druge, nedovoljno jasne i neujednačene. Spomenimo još definiciju koja polazi jednostavno od broja stupnjeva u SR-u, tj.:

$$N = q + 1$$

koja se može nazvati ***memorijska ograničena dužina***.

Konvolucijski kodovi stoga se često označavaju kao (n, m, N) kodovi, gdje je N definiran na jedan od spomenutih načina, što često stvara zbrku. U tome smislu možda je najprikladnije označavati kodove kao (n,m,q), a N tretirati odvojeno u skladu s primjenom.

Nadalje, slično blok-kodovima, mogu se definirati ***sustavan*** konvolucijski (n, m) kodovi. Generator-matrica definirana relacijom (12), za sustavne kodove sadrži elemente **G**, koji su ($m \times n$) matrice strukture:

$$G_0 = \left[I_m \vdots D_0\right]$$
$$G_i = \left[0 \vdots D_i\right] \qquad i = 0, 1, 2, \ldots, q \tag{15}$$

odnosno polinomska generator-matrica $\mathbf{G}(\mathbf{L}) = \mathbf{h}^T(\mathbf{L})$ je strukture:

$$G(L) = \left[I_m \vdots D(L)\right] \tag{16}$$

5. PRIMJER 5: Sustavan (3, 2) kôd

Za (3, 2) konvolucijski kôd definiran u primjeru 4, (2×3) podmatrice $\mathbf{G}_0$ i $\mathbf{G}_1$ su definirane s:

$$G_0 = H_0^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad G_1 = H_1^T = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

pa proizlazi da $\mathbf{G}_0$ i $\mathbf{G}_1$ zadovoljavaju strukturu (15), odnosno odgovarajući (3, 2) kôd je sustavan. Ova je činjenica uočljiva i na slici 9, jer su prva 2 digita na izlazu iz kôdera, tj. $\mathbf{x}_{1,k}$ i $\mathbf{x}_{2,k}$ jednaka ulaznome paru informacijskih digita $\mathbf{v}_{1,k}$ i $\mathbf{v}_{2,k}$. Treći redundantni simbol $\mathbf{x}_{3,k}$ je dobiven kao linearna kombinacija $\mathbf{v}_{1,k}$, $\mathbf{v}_{1,k-1}$ i $\mathbf{v}_{2,k-1}$.

Također, polinomska ($m \times n$) generator-matrica je uz (13) dana s (primjer 2):

$$G(L) = h^T(L) = \begin{bmatrix} 1 & 0 & 1 \oplus L \\ 0 & 1 & L \end{bmatrix}$$

što prema (16) odgovara strukturi sustavnoga kôda.

6. PRIMJER 6: Sustavan (2, 1) kôd

Konvolucijski (2, 1) kôd definiran sa $h(L)$ u primjeru 12.1 nije sustavan kôd, jer

$$G(L) = h^T(L) = \left[1 \oplus L \oplus L^2 \quad 1 \oplus L^2\right]$$

nema strukturu (16). Međutim, lako je definirati ekvivalentan sustavan (2, 1) kôd. Prvi element u **G**(**L**) treba biti jednak 1, pa se može nakon jednoga koraka dijagonalizacije pisati:

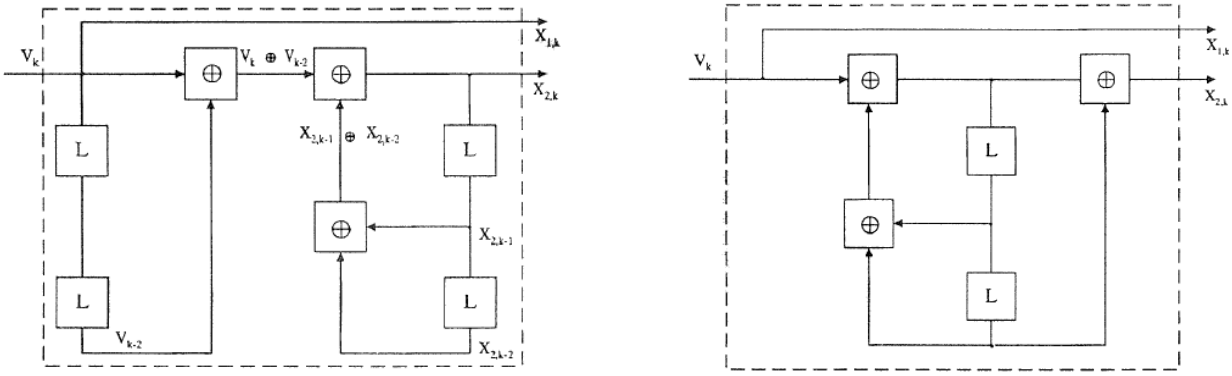
$$G(L) = \left[1 \quad \frac{1 \oplus L^2}{1 \oplus L \oplus L^2} \right]$$

Polinom u nazivniku drugoga elementa znači da je riječ o ARMA sustavu drugoga reda, odnosno radi se o ARMA (2, 2) modelu. Iz (7) slijedi:

$$x_{1,k} = v_k$$

$$x_{2,k} = \frac{1 \oplus L^2}{1 \oplus L \oplus L^2} v_k = x_{2,k-1} \oplus x_{2,k-2} \oplus v_k \oplus v_{k-2} \qquad (17)$$

Izlaz $x_{2,k}$ sadrži autoregresiju drugoga reda. Odgovarajući MIMO sustav s jednim ulazom i dva izlaza ilustriran je na slici 10a. Međutim, broj elemenata za kašnjenje može biti minimiziran jer se redoslijed kaskadno spojenih podsustava može zamijeniti, pa vrijedi shema kao na slici 10b.



a) prema relaciji (12.17) *b) minimizirano rješenje*

Sl. 10 Hardverska realizacija sustavnoga (2, 1) konvolucijskog kôdera.

1.3. Parametri konvolucijskih kodova

Izlazna kôdna riječ iz konvolucijskoga kôdera definirana je trenutnom informacijskom *m*-torkom, ali i prethodnim *m*-torkama. Koliko prethodnih *m*-torki ima utjecaj na trenutnu kôdnu riječ ovisi o broju stupnjeva D bistabila u kôderu, odnosno o trajanju memorije sustava $q + 1$. Za svaku novo učitanu *m*-torku simbola na ulazu, kôder daje na izlazu odgovarajuću *n*-torku, tako da je koeficijent prijenosa definiran kao i u blok-kodovima $\rho = m/n$.

Konvolucijskim kodovima obično su pridružena tri parametra (n, k, m).

n ... broj izlaznih bitova

k ... broj ulaznih bitova

m ... broj memorijskih registara

Iznos k/n naziva se *brzina koda.* To je mjera učinkovitosti koda. Obično su k i n parametri u rasponu od 1 do 8, m od 2 do 10, a brzina koda od 1/8 do 7/8, osim aplikacije napravljenih za dubok svemir gdje je brzina koda malena (koristi se 1/100 ili čak i veći omjer).

Pored toga što k definira minimalnu dužinu kôdne riječi, kôd se može ograničiti i na maksimalan broj bĩta L, čime su potpuno definirani tzv. (k, L) kodovi ograničene dužine niza, ili RLL (*run-length-limited*) kodovi.

Često proizvođači čipova za konvolucijske kodove navode parametre koda (n, k, L), količina L označava *duljinu ograničenja* (*constraint length*)[1] koda i definira se izrazom,

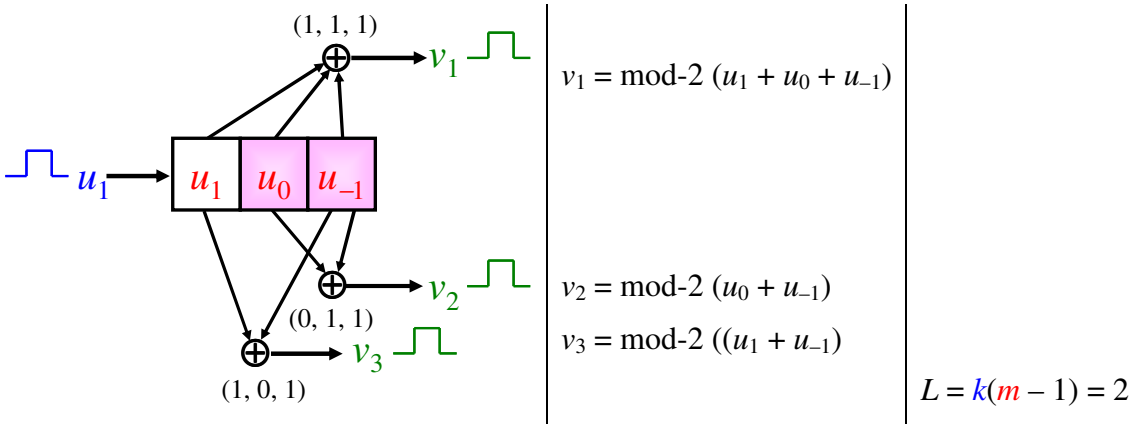
$$L = k(m - 1)$$

[1] Constraint Length.doc

a broj stanja konvolucijskoga koda jednak je $2^{k(L-1)}$. Duljina ograničenja L, predstavlja *broj bitova* u *memoriji kodera* koji utječu na *stvaranje* n izlaznih bitova. Duljina ograničenja L također se označava velikim slovom K, što se može pobrkati s malim slovima k, a on predstavlja broj ulaznih bitova. U nekim knjigama K je jednak umnošku k i m. Često, u komercijalnim specifikacijama, kodovi su određen s (r, K), gdje je r brzina koda k/n, a K je *duljina ograničenja.* Duljina ograničenja K, također je jednaka L–1, kao što je definirano u ovome radu. Za konvolucijske kodove to će se prikazivati kao (n, k, m), a ne kao (r, K).

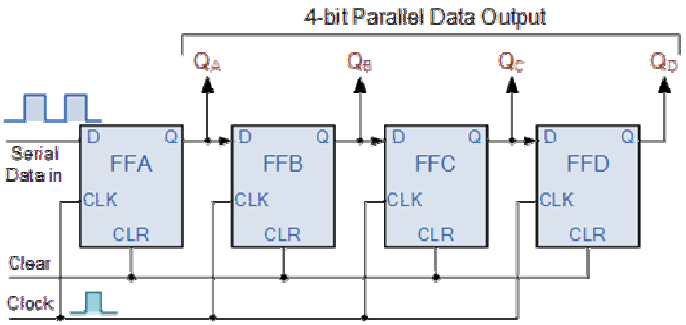
1.4. 1.4. Parametri koda i struktura konvolucijskoga koda

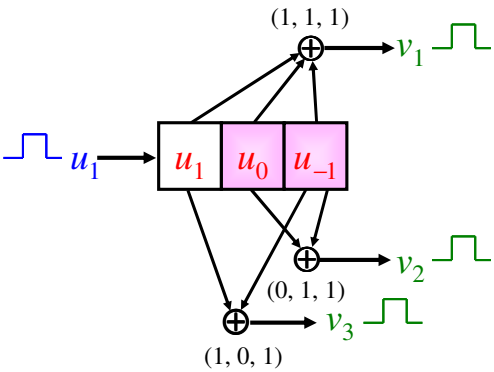
Strukturu konvolucijskoga koda lako je nacrtati iz njegovih parametara. Prvo nacrtamo m okvira koji predstavljaju m registara. Zatim nacrtamo n zbrajala modulo-2 za predstavljanje n izlaznih bitova. Sada spojimo registre na zbrajala pomoću *generator-polinoma* kao što je prikazano na slici 1.

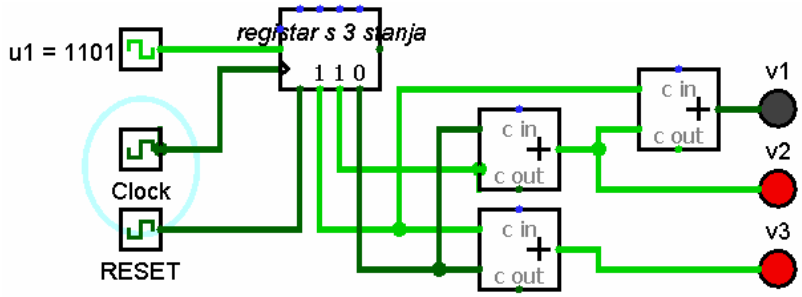


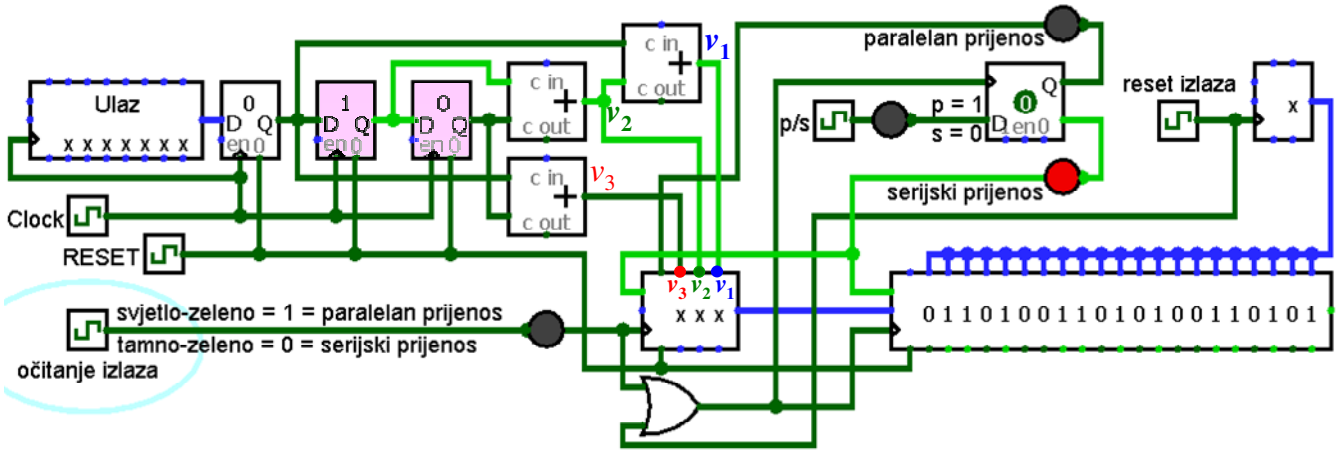
Slika 1: Ovaj (3, 1, 3) konvolucijski kod ima 3 registara, 1 ulazni bitni i 3 izlazna bita.

Posmični registar









Na slikama je prikazana brzina koda 1/3. Svaki *ulazni* bit kodira se u 3 *izlazna* bita. *Duljina ograničenja koda* je 2. Tri izlazna bita proizvode 3 zbrajala modulo-2 zbrajanjem odgovarajućih bitova u memorijskim registrima. *Izbor*, koji će se bitovi zbrojiti da bi proizveli izlazni bit, zove se *polinom-generator* (*g*) za taj izlazni bit. Na primjer, prvi izlazni bit ima polinom-generator (1, 1, 1). Izlazni bit 2 ima polinom-generator (0, 1, 1), a treći izlazni bit ima polinom (1, 0, 1). Izlazni bitovi samo su zbroj ovih bitova.

v_1 = mod-2 $(u_1 + u_0 + u_{-1})$

v_2 = mod-2 $(u_0 + u_{-1})$

v_3 = mod-2 $((u_1 + u_{-1})$

Polinomi rezultiraju *kodom* prema svojoj *jedinstvenoj kakvoći zaštite od pogreške*. Jedan (3, 1, 4) kod može imati sasvim različita svojstva u odnosu na drugi kod, ovisno o odabranim polinomima.

1.5. 1.5 Kako se odabiru polinomi?

Postoji mnogo izbora za polinome koda bilo kojega reda *m*. Oni svi ne moraju rezultirati proizvodnjom nizova koji imaju dobre osobine zaštite od pogrešaka. Knjiga Petersena i Weldona sadrži cjelovit popis tih polinoma. Dobri polinomi što se nalaze na ovome popisu, pronađeni su računalnom simulacijom. Popis dobrih polinoma za kodove brzine 1/2 dan je u nastavku.

Tablica 1 - *Polinom-generatori* što ih je pronašao Busgang za *dobre* kodove brzine 1/2

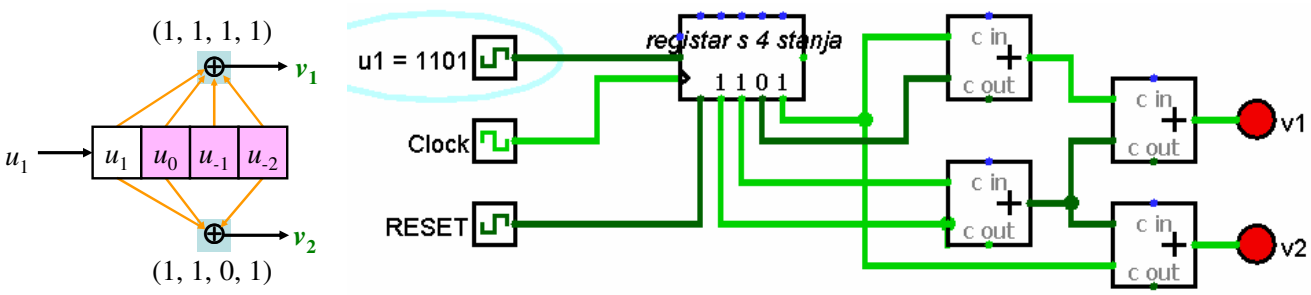
Duljina ograničenja	Zbrajalo G1	Zbrajalo G2
3	110	111
4	1101	1110
5	11010	11101
6	110101	111011
7	110101	110101
8	110111	1110011
9	110111	111001101
10	110111001	1110011001

1.6. 1.6. Stanja koda

Imamo stanja uma pa to preslikavamo i na kodere. Jedan dan smo depresivni, a već sljedećega dana možda sretni što su različita stanja u kojima možemo biti. Naš izlaz ovisi o našim stanjima uma pa „prema izrazu lica“ (*tongue-in-cheek*)[3] možemo kazati da i koderi rade na ovaj način. Ono što im je izlaz ovisi o tome kakvo je stanje njihova „uma“. Naša stanja su složena, ali stanja kodera su samo niz

[3] Aludira se na izraz lica što se stvara „plaženjem“ nečijega jezika prema tuđem obrazu. Zajedljivac (*tongue in cheek*) je antiteza ove druge fraze - *ravnodušno lice* (*with a straight face*).

bitova. Sofisticirani koderi imaju duge duljine ograničenja, a drugi jednostavno imaju kratke, pokazujući broj stanja u kojima mogu biti. Kod (2, 1, 4) na slici 2 ima ograničenu duljinu 3.



Slika 2 - Stanja koda pokazuju sadržaj registara

Zasjenjeni registri drže ove bitove. Ne zasjenjeni registar drži dolazni bit. To znači da 3 bita ili 8 različitih kombinacija ovih bitova može biti prisutno u tim memorijama registara. Ovih 8 različitih kombinacija određuju kakav ćemo izlaz dobiti za kodirane nizove v_2 i v_2.

Broj kombinacija bitova u zasjenjenim registrima zove se *stanja koda* i definirano je

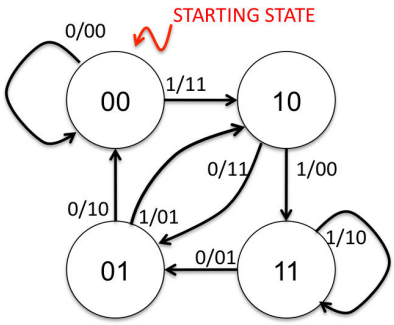
$$\text{Broj stanja} = 2^L$$

gdje je L ... duljinu ograničenja koda i jednaka je

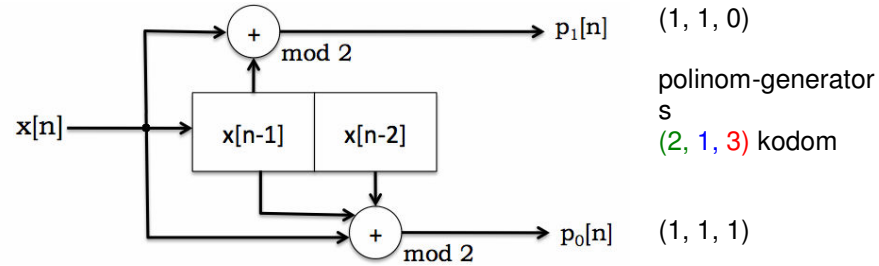
$$L = k(m - 1).$$

Promišljajte o stanjima kao svojevrsnome početnom uvjetu. Izlazni bit ovisi o tome početnom stanju što se *mijenja u svakome vremenskom odsječku*.

Ispitajmo stanja koda (2, 1, 4) prikazanoga prije. Ovaj kod šalje 2 bita na izlaz, za svaki ulazni bit. On je kod brzine 1/2. Njegova duljina je ograničenja na 3. Ukupan broj stanja iznosi 8. Osam stanja ovoga (2, 1, 4) koda su: 000, 001, 010, 011, 100, 101, 110, 111.



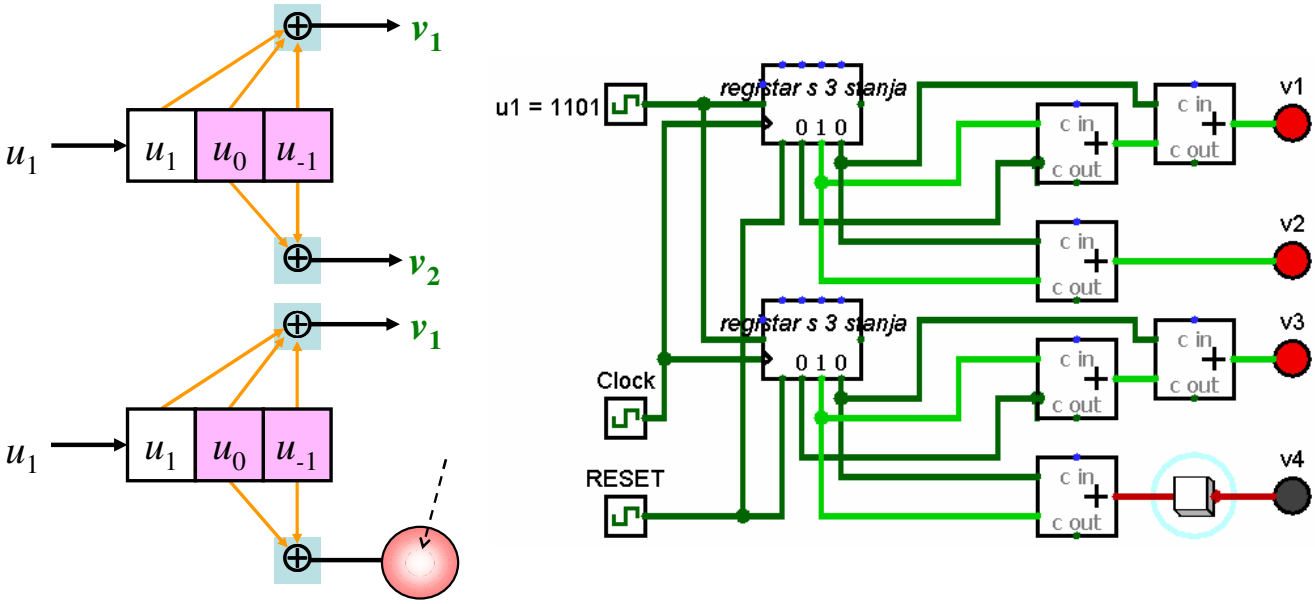
Slika X Dijagram stanja za (2, 1, 2) kod.



Slika Y: Pogled na blok dijagram konvolucijskoga kodiranja (2, 1, 3) s posmičnim registrima.

1.7. 1.7. Probušen kodovi

U posebnome slučaju kada je k = 1, brzine kodova 1/2, 1/3, 1/4, 1/5, 1/7 ponekad se nazivaju *matični kodovi.* Možemo kombinirati pojedine ulazne kodne bitove za proizvesti *probušenih kodova* koji nam daju brzina koda, različitu od 1/*n.* Korištenjem zajedno dvo-brzinskih kodova 1/2 kao što je prikazano na slici 3, a zatim jednostavno ne slanjem jednoga od izlaznih bitova možemo pretvoriti primjenu ove brzine od 1/2 u brzinu od 2/3 koda; 2 dolazna i 3 odlazna bita (a ne 4, koliko ih je raspoloživih).



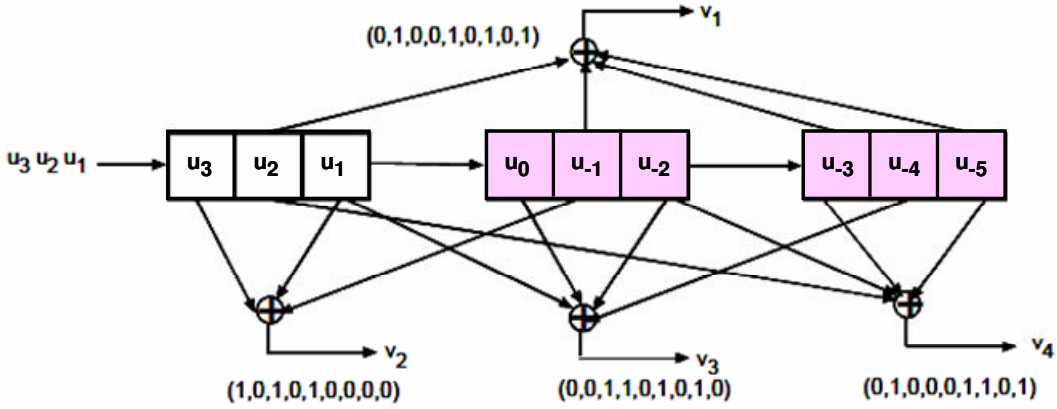
Slika 3 - Dva (2, 1, 3) konvolucijska koda proizvode 4 izlazna bita. Bit broj v_2 je "probušen", tako da je kombinacija učinkovit (3, 2, 3) kod.

Ovaj koncept zove se *bušenje*. Na prijemnoj strani, dodatni bitovi koji ne utječu mjeru dekodiranja umeću se na odgovarajuća mjesta prije dekodiranja. Ova tehnika omogućuje nam proizvesti kodove mnogih različitih brzina korištenjem samo jednoga jednostavnog sklopa. Iako također možemo izravno konstruirati kod brzine 2/3, kao što ćemo kasnije vidjeti, prednost probušenoga koda je da se brzine mogu dinamički mijenjati (programski), ovisno o stanju kanala, kao što je utjecaj atmosferskih smetnji, itd. Čvrsta provedba, iako jednostavnija, ne dopušta raznolikost.

1.8. 1.8. Struktura koda za k > 1

Naizmjence možemo stvoriti kodove gdje je k više od 1 bita kao što je (4, 3, 3) kod. Ovaj kod ima 3 ulazna i 4 izlazna bita. Broj registara je 3. Duljina ograničenja je $3 \times 2 = 6$. Kod ima 64 stanja. I ovaj kod zahtjeva polinome 9-toga reda. Zasjenjeni kvadrati predstavljaju duljinu ograničenja.

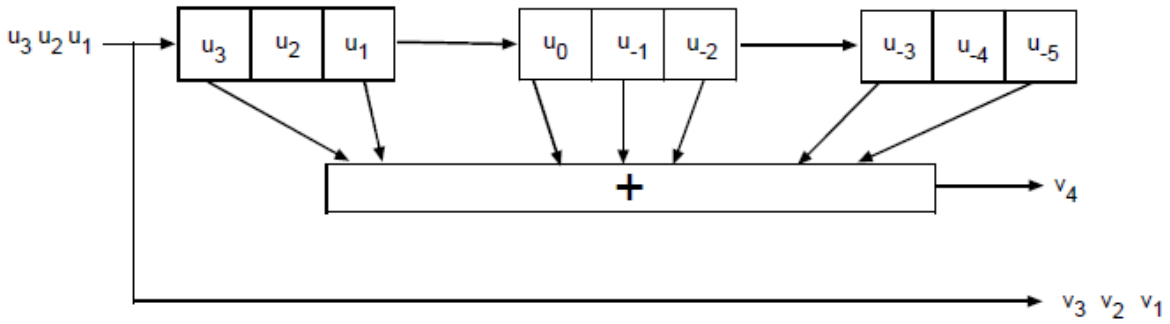
Postupak za izradu strukture (n, k, m) gdje je k broj veći od 1 je kao što slijedi. Prvo nacrtamo k skupova od m kvadrata. Zatim nacrtamo n zbrajala. Sada spojimo n zbrajala na memorijske registre koristeći koeficijente odgovarajuće polinoma n-toga $(k \cdot m)$ stupanja. Ono što ćemo dobiti je struktura kao što je ona na slici 4 za kod (4, 3, 3).



Slika 4 - Ovaj (4, 3, 3) konvolucijski kod ima 3 ×3 = 9 memorijskih registara, 3 ulazna i 4 izlazna bita. Osjenčani registri sadrže "stare" bitove predstavljajući trenutno stanje [duljina ograničenja je $L = k\,(m–1) = 3 \cdot (3–1) = 6$].

1.9. 1.6. Sustavan u odnosu na nesustavan kod

Poseban oblik konvolucijskoga koda u kojemu izlazni bitovi sadrže lako prepoznatljivi niz ulaznih bitova naziva se *sustavan oblik*. Sustavnu inačicu gornjega (4, 3, 3) koda prikazuje slika 5.



Slika 5 - Sustavna inačica konvolucijskoga (4, 3, 3) koda. Inačica ima isti broj memorijskih registara, 3 ulazna bita i 4 izlazna bita. Izlazni bitovi sastoje se od izvorna 3 bita i 4-toga "paritetnoga" bita.

Od 4 izlazna bita, 3 su ista kao i 3 ulazna bita. Četvrti bit je vrsta paritetnoga bita što je napravljen od kombinacije 3 bita koristeći jedan polinom. Sustavni kodovi radije se koriste u odnosu na ne-sustavne kodove, jer omogućuju brz pogled. Oni također zahtijevaju manje opreme za kodiranje. Drugo važno svojstvo sustavnih koda je da nisu "katastrofični", što znači da se pogreške ne mogu prostirati na katastrofičan način. Sva ta svojstva čine ih vrlo poželjnima. Sustavni kodovi također se koriste u *rešetkasto kodiranoj modulaciji* TCM (*Trellis Coded Modulation*). Svojstva zaštite od pogrešaka sustavnih kodova, međutim ista su kao ona za ne.sustavne kodove.

1.10. 1.10. Kodiranje dolaznoga niza

Izlazni niz **v**, može se izračunati konvolucijom ulaznoga niza **u** i impulsnoga odziva g. To se može izraziti kao

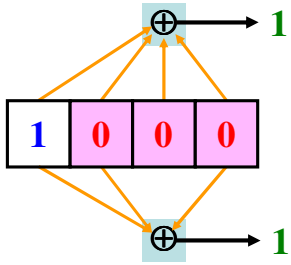
$$\mathbf{v} = \mathbf{u} * g$$

ili u općenitijem obliku

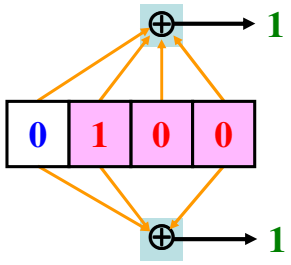
$$v_l^j = \sum_{i=0}^{m} u_{l-i} g_i^j$$

gdje je v_l^j izlazni bit l kodera j, u_{l-i}je ulazni bit, a g_i^j je i-ti izraz u polinomu j.

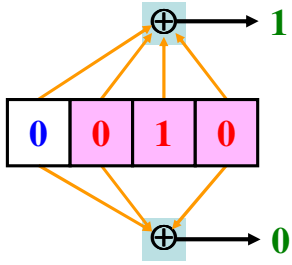
Kodirajmo niz od dva bita 1 i 0 (2, 1, 4) kodom te pogledajmo kako proces radi. Prvo ćemo propustiti jedan bit „1“ kroz ovaj koder kao što prikazuje slika 6.



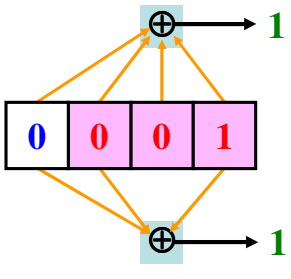
a) t = 0, stanje kodera = 000
Ulazni bit = 1, izlazni bitovi = 11



b) t = 1, stanje kodera = 100
Ulazni bit = 0, izlazni bitovi = 11



c) t = 2, stanje kodera = 010
Ulazni bit = 0, izlazni bitovi = 01



d) t = 3, stanje kodera = 001
Ulazni bit = 0, izlazni bitovi = 11

Slika 6 - Prolazi kroz koder niza od samo 1 bita. Jedan bit stvara izlaz od osam bitova.

a. U trenutku t = 0, vidimo da je početno stanje kodera sve nule (bitovi u desnome dijelu L = 3 registarskih pozicija). Ulazni bit 1 uzrokuje pojavu dva bita 11 na izlazu. Kako smo to

izračunali? Zbroj svih bitova u registrima po modulu 2 za prvi bit (gornji izlaz) i zbroj tri bita po modulu 2 za drugi izlazni bit (donji izlaz) prema koeficijentima polinoma.

b. U trenutku $t = 1$, ulazni bit 1 pomiče se udesno u registar. ulazni registar je sada prazan i puni se bitom 0 za ispiranje. Koder je sada u stanju 100. Izlazni bitovi su sada opet 11 prema istome izračunu.

c. Ulazni bit 1 pomiče se opet udesno. Sada je koder u stanju 010 pa se drugi bit za ispiranje seli u ulazni registar. Izlazni bitovi su sada 10.

d. U trenutku $t = 3$, ulazni bit seli se u posljednji registar i stanje ulaza je 001. Sada su izlazni bitovi 11.

e. U trenutku $t = 4$, ulazni bit 1 potpuno je prošao kroz koder i koder se isprao u stanje „sve nule“ te je spreman za prijem sljedećega niza.

Imajte na umu da jedan bit proizvodi izlaz od 8 bitova, iako je nominalna brzina koda 1/2. Ovo pokazuje da je za male nizove prekoračenje znatno veće od nominalne brzine, što je vrijedi samo za duge nizove.

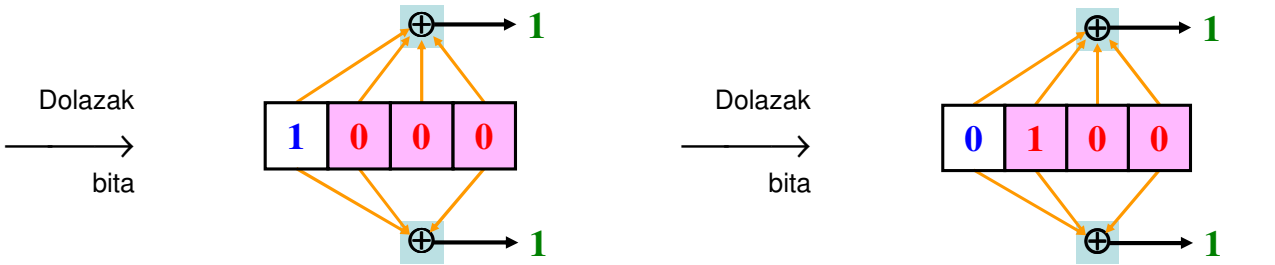
Ako učinimo istu stvar s bitom vrijednosti 0, dobili bi niz od 8 bitova sa svim nulama. Ono što smo upravo proizveli zove se *impulsni odziv* ovoga kodera. Odziv na bit 1 je niz 11 11 10 11 i zove se *impulsni odziv*. Slično, odziv na ulazni bit 0 ima *impulsni odziv* 00 00 00 00 (nije prikazano, ali to je očigledno).

Konvolucija[5] ulaznoga niza s kodnim polinomom proizvodi ova dva izlazna niza, što je razlog zašto se ti kodovi zovu *konvolucijski kodovi.* Prema principu *linearne superpozicije*, sada možemo proizvesti kodiran niz iz navedenih dviju impulsnih odziva kao što slijedi. Pretpostavimo da imamo ulazni niz 1011 i želimo znati što će biti kodiran niz. Možemo *izračunati izlaz* samo dodavanjem *pomaknute verzije pojedinih impulsnih odziva*.

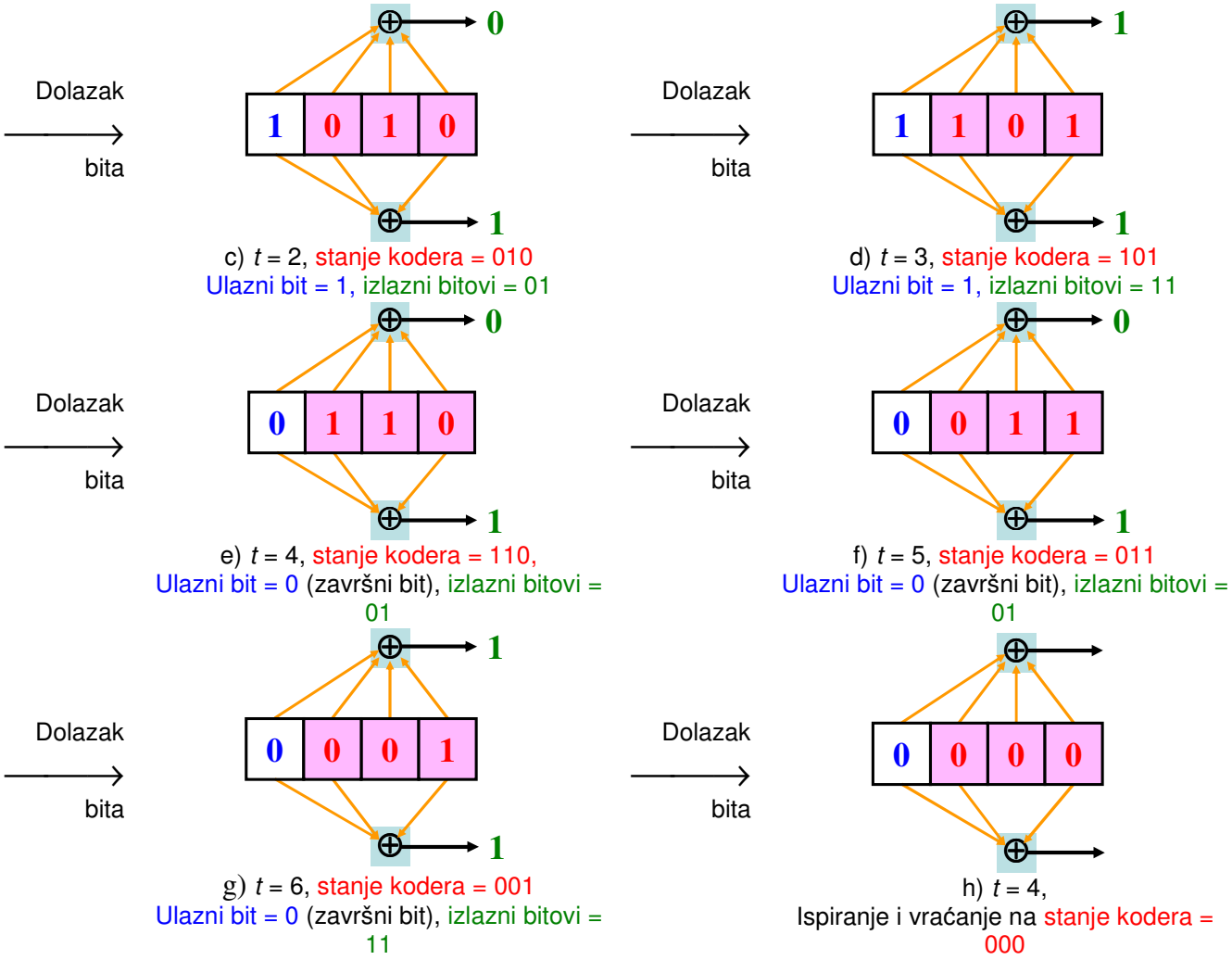
Ulazni bitovi				Njihov impulsni odziv						
1				11	11	10	11			
	0				00	00	00	00		
		1				11	11	10	11	
			1				11	11	10	11
Zbrojiti za dobiti odziv → :										
1	0	1	1	11	11	01	11	01	01	11

Dobili smo odziv na niz 1011 dodavanjem pomaknute verziju odziva za 1 i 0.

Na slici 7, ručno smo postavili niz 1011 pomoću kodera za provjeru gornjega i za ne povjerovati, dobit ćemo isti odziv.



[5] Konvolucija je integral što opisuje veličinu preklapanja jedne krivulje g drugom krivuljom f dok se jedna pomiče preko druge (Vidi http://mathworld.wolfram.com/Convolution.html).



Slika 7: Kodiranje (2, 1, 4) kodom

Ovo pokazuje da je konvolucijski model ispravan. Prethodan rezultat kodiranja, u svakome vremenskom odsječku prikazuje tablica 2.

Tablica 2 - izlazni bitovi i bitovi kodera za (2, 1, 4) kod - ulazni bitovi su: 1011000

Vrijeme	Ulazni bit	Izlazni bitovi	Stanje registara
0	1	11	000
1	0	11	100
2	1	01	010
3	1	11	101
4	0	01	110
5	0	01	011
6	0	11	001

Kodiran niz je **11 11 01 11 01 01 11.**

1.11. 1.11. Oblikovanje kodera

Dvije metode u prethodnome poglavlju pokazuju matematički što se događa u koderu. Sklop kodera je puno jednostavniji, jer koder nije matematika. Koder za konvolucijski kôd koristi preglednu tablicu za obaviti kodiranje. Pregledna tablica sastoji se od četiri stavke.

1. Ulazni bit
2. Stanje kodera. To je jedno od 8 mogućih stanja za primjer (2, 1, 4) koda
3. Izlazni bitovi. Za kod (2, 1, 4), dok su 2 bita izlazi, izbori su 00, 01, 10, 11.
4. *Stanje izlaza*, To će biti *ulazno stanja* za sljedeći bit.

Za (2, 1, 4) kod zadan polinomima slici 7, stvorena je sljedeća pregledna tablica u *Excel* obliku.

Tablica 3 - pregledna tablica za koder (2, 1, 4) koda

Ulazni bit	Stanje Ulaza				Izlazni bitovi			Stanje izlaza		
I1	s1	s2	s3		O1	O2		s1	s2	s3
0	0	0	0		0	0		0	0	0
1	0	0	0		1	1		1	0	0
0	0	0	1		1	1		0	0	0
1	0	0	1		0	0		1	0	0
0	0	1	0		1	0		0	0	1
1	0	1	0		0	1		1	0	1
0	0	1	1		0	1		0	0	1
1	0	1	1		1	0		1	0	1
0	1	0	0		1	1		0	1	0
1	1	0	0		0	0		1	1	0
0	1	0	1		0	0		0	1	0
1	1	0	1		1	1		1	1	0
0	1	1	0		0	1		0	1	1
1	1	1	0		1	0		1	1	1
0	1	1	1		1	0		0	1	1
1	1	1	1		0	1		1	1	1

Ovaj pregledna tablica u potpunosti opisuju (2, 1, 4) kod. Ona je različita za svaki kod, ovisno o parametrima i korištenim polinomima.

Grafički, postoje tri načina na koje možemo analizirati koder da se bolje razumije njegov rad. To su:

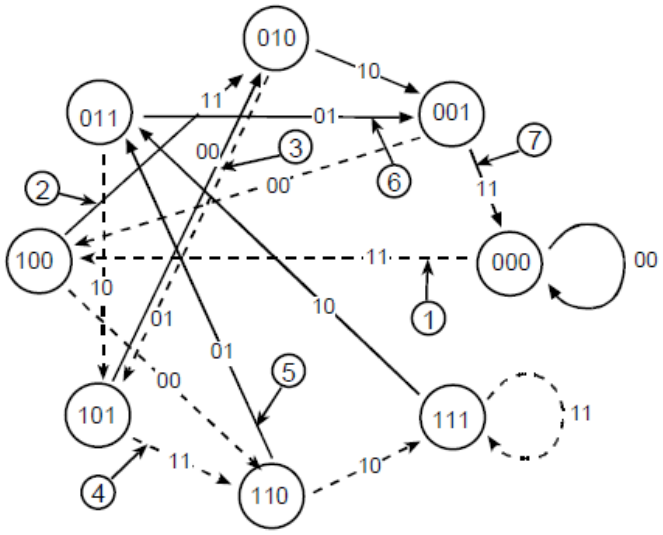
2. Dijagram stanja
4. Dijagram stabla
6. Dijagram rešetke.

1.12. 1.12. Dijagram stanja

Dijagram stanja za (2, 1, 4) kod prikazuje slika 8.

U pune crte označavaju dolazak 0
Crtkane linije ukazuju na dolazak 1.
Svaka kružnica predstavlja stanje.
U bilo kojemu trenutku, koder boravi u jednome od tih stanja.
Crte prema i od njega pokazuju stanja prijelaza što su moguća kako bitovi stižu.
Samo dva događaja mogu se dogoditi u svakome trenutku, dolazak bita 1 ili bita 0.
Svaki od tih dvaju događaja omogućuje koderu skok u različito stanje.
Dijagram stanja nema vrijeme kao dimenziju i stoga stremi intuitivnome razmatranju.
Izlazni bitovi za svaki slučaj su prikazani na crti

Strelica označava stanje prijelaza



Slika 8 - Dijagram stanja (2, 1, 4) koda

Svaka kružnica predstavlja stanje. U bilo kojemu trenutku, koder boravi u jednome od tih stanja. Crte prema i iz njega pokazuju stanje prijelaza koje je moguće kako bitovi pristižu. Samo se dva događaja mogu dogoditi u svakome trenutku, dolazak bita 1 ili dolazak bita 0. Svaki od tih dvaju događaja omogućuje koderu skok u drugo stanje. Dijagram stanja nema vrijeme kao dimenziji i stoga stremi intuitivnome razmatranju.

Usporedimo gornji dijagram stanja prema preglednoj tablici kodera. Dijagram stanja sadrži iste informacije kao i tablica pretraživanja, ali se prikazuju grafički. Čvrste crte pokazuju dolazak 0, a isprekidane crte prikazuju dolazak 1. Izlazni bitovi, za svaki slučaj, prikazani su uz crtu, a strelica označava stanje prijelaza.

Još jednom vratimo se ideji stanja. Zamislite da ste na otoku. Na koliko načina možete napustiti ovaj otok? Možete brodom ili plivajući. Imate samo ove dvije mogućnosti napuštanja ovoga malog otoka. Možete otploviti brodom na kopno. Sada, kada ste došli na kopno, na koliko se načina može kretati po kopnu? Možete opet ploviti brodom, ali se sada možete voziti prema drugim odredištima. O tome gdje ste (vaše stanje) određuje na koliko načina možete putovati (vaš izlaz).

Neka stanja kodera dopuštaju izlaze 11 i 00, a neka dopuštaju samo stanja 01 i 10 stanja. Nema stanja što dopušta sve četiri mogućnosti.[6]

Kako možemo kodirati niz 1011 koristeći dijagram stanja (vidi i prati sliku 7)?

a. Počnimo stanjem 000 (a). Dolazak bita 1 daje izlaz 11 i vodi nas u stanje 100 (b).

b. Dolazak idućega bita 0, šalje na izlaz bitove 11 i vodi nas u stanju 010 (c).

c. Dolazak idućega bita 1, šalje na izlaz bitove 01 i vodi nas u stanju 101 (d).

d. Dolazak posljednjega bita 1 šalje na izlaz bitove 11 i vodi nas u stanje 110 (e). Tako sada imamo na izlazu prikupljen niz bitova 11 11 01 11. No, ovo nije kraj. Moramo dovesti koder natrag na stanje „sve nule“.

e. Idući dolazni bit je 0 pa se iz stanja 110, ide u stanje 011, a na izlazu se pojavljuju bitovi 01 i pridružuju nizu 11 11 01 11 s desne strane.

f. Iz stanja 011 (f) ide se u stanja 001 (g), a na izlazu se pojavljuju bitovi 11 i pridružuju se gornjemu nizu s desna, a

g. zatim se 0 na ulazu ide prema stanju 000 (h) i konačnome izlaznom nizu: 11 11 01 11 01 01 11.

Ovo je isti odgovor kao što smo ga dobili zbrajanjem pojedinačnih *impulsnih odziva* za bitove 1011000.

1.13. 1.13 Dijagram stabla

Slika 9 prikazuje dijagram stabla za kôd (2, 1, 4). Dijagram stabla pokušava pokazati prolaz vremena, kao idemo dublje u grane drveća. On je nešto bolji od dijagrama stanja, ali još uvijek ne preferira kao pristup za predstavljanje konvolucijskih kodova.

Ovdje umjesto skakanja iz jednoga stanja u drugo, spuštamo se niz grane stabla, ovisno o tome je li se primila 1 ili 0.

Na slici 9, pune crte pokazuju dolazak bita 0, a crtkane linije dolazak bita 1. Prva dva bita pokazuju izlazne bitove, a broj unutar zagrada je stanje izlaza.

Prva grana na slici 9 označava dolazak bita 0 ili 1. Za početno stanje pretpostavlja se da je 000. Ako se primi 0, penjemo se, a ako se primi 1, onda se spuštamo.

Neka je kodni niz 1011 kao i prije. Na grani 1, spuštamo se. Izlaz je 11, a mi smo sada u stanju 111. Sada se primila 0, pa se penjemo. Izlazni bitovi su 11, a stanje je sada 011.

Sljedeći dolazni bit je 1. Spuštamo se i dobivamo izlaz 01 pa je sada stanje izlaz 101.

Sljedeći dolazni bit je 1 pa se spuštamo i opet su izlazni bitovi 11. Od ove točke, u odzivu na ulazni bit 0 pa je izlaz 01, a stanje izlaza je 011.

Što ako je niz duži, što onda? Ponestane grana na drvetu. Dijagram stabla sada se ponavlja. U stvari moramo „isprati“ koder, tj. njegova stanja popuniti nulama, tako je naš niz zapravo 1011000, a zadnja 3 bita (000) zovu se „bitovi za ispiranje“ (*flush bits*) ili „punjenje repa bitovima“ (*tail-biting*).

Sada skačemo u točku 2 na stablu i penjemo se za tri grane. Sada na izlazu imamo potpun niz, a to je 11110111101011. Možda niste iznenađeni da je to također isti odgovor kao i onaj koji smo dobili iz dijagrama stanja.

Crvene okomite crte znače dolazak bita 0.	

[6] Objasniti ovu rečenicu!?!

Plave okomite crte označavaju dolazak bita 1.	
Na vodoravnim crtama, prva 2 bita su izlazni.	
Broj unutar zagrada je stanje registra	
Prva grana na slici označava dolazak 0 ili 1	
Za početno stanje pretpostavlja se da je 000.	
Ako se primi 0, penjemo se.	
Ako se primi 1, spuštamo se.	
Neka je kodni niz 1011 kao i prije.	
Na grani 1, spuštamo se (crtkano).	
Izlaz je 11 (u plavome krugu), a stanje je 111.	
Sada se primila 0, pa se penjemo (crtkano).	
Izlazni bitovi su 11, a stanje je sada 011.	
Sljedeći dolazni bit je 1.	
Spuštamo se, jer smo primili izlaz 01	
Sada je stanje izlaza 101.	
Sljedeći dolazni bit je 1.	
Spuštamo se i opet su izlazni bitovi 11.	
Od ove točke, u odzivu na ulazni bit 0	
izlaz je 01, a stanje izlaza je 011.	
Što ako je niz duži,	
što ako ponestane grana na drvetu?	
Registri kodera se „ispiru“	
tj, njegova stanja popunjavaju se nulama	
tako je naš ulazni niz zapravo 1011000.	
Zadnja 3 bita (000) zovu se	
„bitovi za ispiranje“ (*flush bits*) ili	
„punjenje repa bitovima“ (*tail-biting*).	
Sada skačemo u točku 2 na stablu (strelica)	
i penjemo se za tri grane	
pa na izlazu imamo potpun niz, a to je:	
11110111101011.	

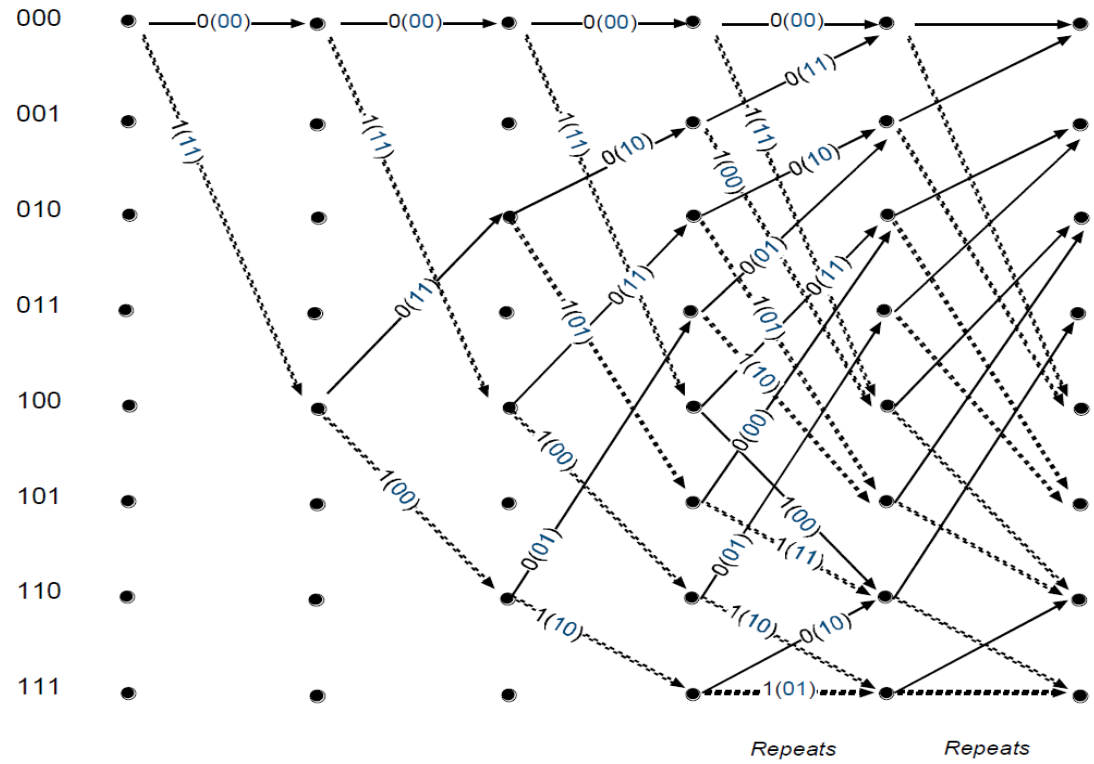
Slika 9 - Dijagram stabla (2, 1, 4) koda

1.14. 1.14. Dijagram rešetke

Dijagrami rešetke su neuredni, ali općenito imaju prednost pred dijagramima stabla i stanja, jer predstavljaju linearan vremenski redoslijed događaja. X-os prikazuje diskretno vrijeme, a sva moguća stanja prikazana su na y-osi. Krećemo se horizontalno kroz rešetku u vremenu. Svaki prijelaz znači dolazak novih bitova.

Dijagram rešetke ostvaruje se crtanjem svih mogućih stanja (2^L) na vertikalnoj osi. Onda spajamo svako stanje sa sljedećim stanjem dopuštene kodne riječi za to stanje. Postoje samo dva moguća izbora u svakome stanju. To je određeno dolaskom bita 0 ili 1. Strelice pokazuju ulazni bit, a izlazni bitovi su prikazani u zagradama. Strelice što idu prema gore predstavljaju bit 0, a one što idu prema dolje, predstavljaju bit 1. Dijagram rešetke jedinstven je za svaki kod, isto kao što su dijagrami stanja i stabla. Možemo nacrtati rešetku za onoliki broj razdoblja koliki želimo. Svako razdoblje ponavlja moguće prijelaze.

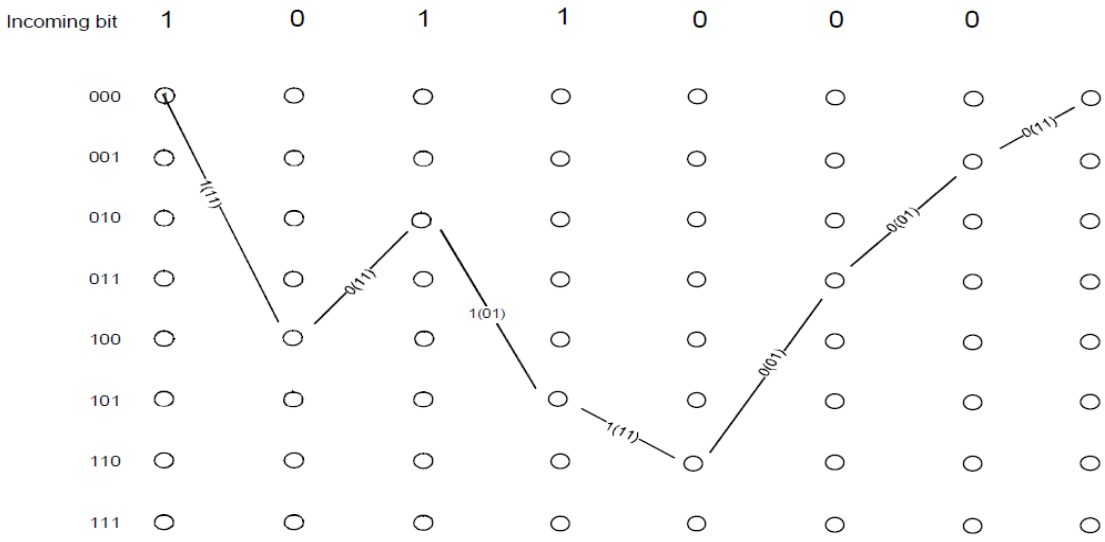
Uvijek počinjemo sa stanjem 000. Počevši od ovoga, rešetka se širi na L bitova i postaje potpuno popunjena, kao da su svi prijelazi mogući. Prijelaze ponovite iz ove točke.



Slika 10 - Dijagram rešetke za (2, 1, 4) kod[7]

1.15. 1.12. Kako kodiranje koristi dijagram rešetke

Na slici 10a (dolje), dolazni bitovi prikazani su na vrhu. Možemo započeti samo u točki 1. Kodiranje je jednostavno. Jednostavno idemo gore ako je bit 0, a dolje ako je bit 1. Trag bitova u našem primjeru je niz (1011000) pokazuju crte. Vidimo da dijagram rešetke daje isti izlazni niz kao i ostale tri metode, naime *impulsni odziv*, *dijagram stanja* i *dijagram stabla.* Svi ovi dijagrami izgledaju slično, ali, moramo priznati da su jedinstveni za svaki kod.



Slika 10a – Kodiran niz, ulazni bitovi 1011000, izlazni bitovi 11 11 01 11 01 01 11

1.16. 1.16. Dekodiranje

Postoji nekoliko različitih pristupa za dekodiranje konvolucijskih kodova. Oni su združeni u dvije osnovne kategorije.

[7] ZADATAK: Nacrtati dijagram rešetke za zadan primljen niz kodiranih binarnih sumbola!

1. Serijsko dekodiranje
   - Fano algoritam
2. Dekodiranje najvećom vjerojatnosti
   - Viterbijevo dekodiranje

Obje ove metode predstavljaju dva različita pristupa istoj osnovnoj ideji dekodiranja.

1.17. 1.17. Osnovna ideja dekodiranja

Pretpostavimo da su poslana 3 bita brzinom koda od 1/2. Primamo 6 bitova. (Za sada zanemarimo bitove za ispiranje.) Ovih šest bitova mogu, ali i ne moraju imati pogrešaka. Znamo iz procesa kodiranja da se ti bitovi jedinstveno preslikavaju. Dakle, niz od 3 bita imat će jedinstven izlaz od 6 bitova. No, zbog pogrešaka, možemo dobiti bilo koju ili sve moguće kombinacije od 6 bitova.

Permutacije 3 ulazna bita rezultira u osam mogućih ulaznih nizova. Svaki od njih pomoću koda, jedinstveno se preslikava u niz od šest izlaznih bitova. Ovo čini skup dopuštenih nizova pa je zadatak dekodera odrediti koji je niz poslan.

Tablica 4 - Sporazum o bitovima koristi se kao mjera odluke između primljenoga niz i osam mogućih ispravnih vrijednosti kodiranoga niza.

Ulaz	Ispravan kodiran niz	Primljen niz	Broj podudarnih bitova
000	000000	111100	2
001	000011	111100	0
010	001111	111100	2
011	001100	111100	4
100	**111110**	111100	**5**
101	**111101**	111100	**5**
110	110001	111100	3
111	110010	111100	3

Recimo da smo primili 111100. To nije ni jedan od 8 mogućih nizova. Kako ga dekodirati? Možemo učiniti dvije stvari:

1. Možemo usporediti primljen niz svim dopustivim nizovima i odabrati onaj s najmanjom Hammingovom udaljenošću (ili najmanjim neslaganjem među bitovima)
2. Možemo napraviti korelaciju i pokupiti nizove s najboljom korelacijom.

Prvi postupak je u osnovi ono što se krije pod nazivom dekodiranje *tvrdom odlukom* (*hard decision decoding*), a drugi postupak je dekodiranje *mekom odlukom* (*soft decision decoding*). Podudarnost bitova, kao i umnožak između primljenoga niza kodne riječi, pokazuje da smo još uvijek dobili dvosmislen odgovor pa još uvijek ne znamo što se poslalo.

Kako se broj bitova povećava, povećava se broj izračuna potrebnih za dekodiranje na brutalan način, pa dekodiranje ovaj način više nije praktično. Moramo pronaći učinkovitiji način da se ne ispituju sve opcije, a da imamo načina rješavanja nejasnoća kao što su ovdje, gdje imamo dva moguća odgovora. (U tablici 4 prikazana podebljano i osjenčano).

Ako se dobila poruka o dužini s bitova, onda je moguć broj kodnih riječi 2^s. Kako možemo dekodirati niz bez provjere svake od tih 2^s kodnih riječi? To je osnova ideje postupka dekodiranja.

1.18. 1.14. Serijsko dekodiranje

Serijsko dekodiranje bilo je jedna od prvih metoda predloženih za dekodiranje konvolucijski kodiranoga toka bitova. Nju je prvi predložio Wozencraft, a poslije je bolju verziju predložio Fano. Serijsko dekodiranje najbolje se opisuju analogijom. Daju vam se neke smjernice. Upute se sastoje od

[9] Pretraživanje unatrag je opći algoritam za pronalaženje svih (ili nekih) rješenja za neke računalne probleme, na način da se postupno grade kandidati za rješenja, a napušta se pojedini djelomičan kandidat c ("*backtracks*"), čim se utvrdi da c ne može dovesti do pravoga rješenja.

nekih poznatih smjernica. No, osoba koja Vam je dala upute nije učinio vrlo dobar posao, a ponekad ne prepoznaje orijentir pa ćete završiti na krivome putu. Ali zbog toga što ne vidite niti jednu oznaku, imate osjećate da ste na krivome putu. Vi se vraćate (*backtrack*)[9] do točke gdje možete prepoznati orijentir, a onda odabirete alternativan put sve dok ne naiđete na sljedeći orijentir i na kraju na taj način možete doći do svojega odredišta. Vi se možete vratiti nekoliko puta u procesu, ovisno o tome koliko su dobre bile smjernice o putovima.

Slično, u serijskome dekodiranje Vi se bave samo jednim putom istovremeno. Može te se odreći puta u bilo koje vrijeme i vratiti se natrag te slijediti drugi put, ali važna stvar je da u bilo kojemu trenutku slijedite samo jedan put.

Serijsko dekodiranje omogućuje Vam kretanja kroz rešetku naprijed i natrag. Dekoder prati njegove odluke i svaki put donosi dvosmislenu odluku, što se podudara[12]. Ako podudarnost[13] raste brže od vrijednosti nekoga praga, dekoder odustaje od toga puta i vraća se natrag na zadnji put gdje je podudarnost bila ispod toga praga.

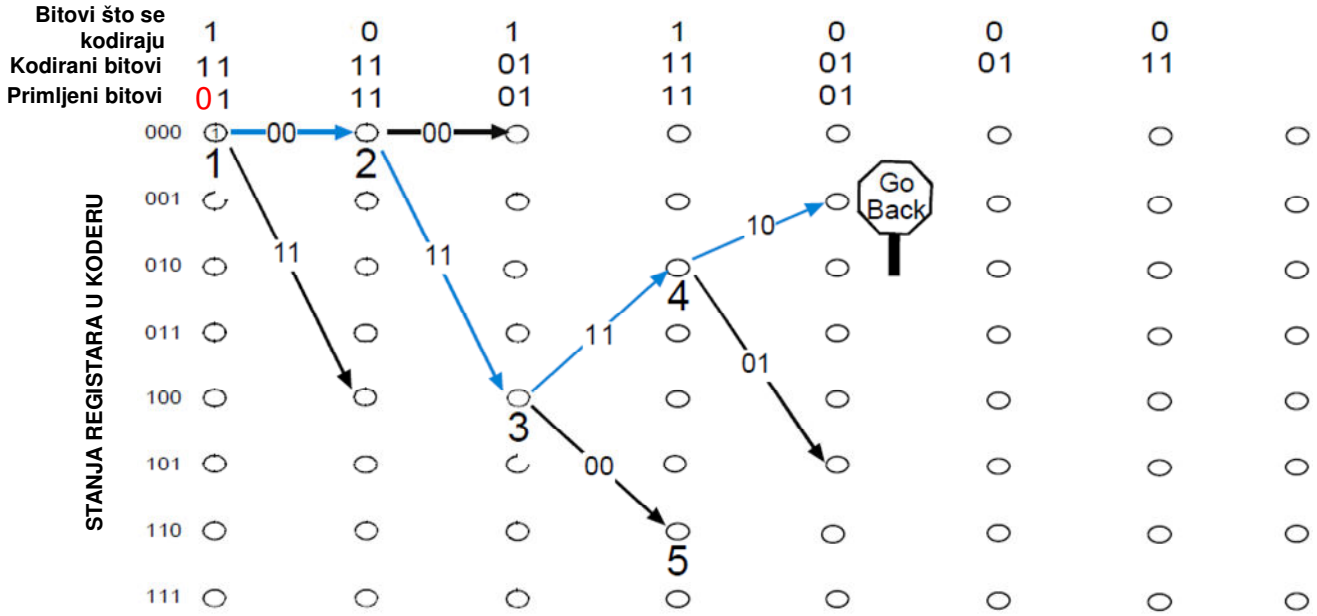
7. Dajmo jedan primjer.

Kod je: (2, 1, 4) za koje smo načinili dijagrame kodera u prethodnome poglavlju. Pretpostavimo da je poslan niz bitova 1011000. (Sjetite se da su posljednja tri bita potrebna za ispiranja registara. Njihov je naziv *bitovi na repu* (*tail bits*).) Ako se nije pojavila pogreška, dobit ćemo: 11 11 01 11 01 01 11.

Ali recimo da smo umjesto toga niza dobili: 01 11 01 11 01 01 11. Pojavila se jedna pogreška. Prvi primljen bit je 0 umjesto 1.

1.19. 1.16. Dekodiranje pomoću algoritma dekodiranja nizova

1. **Točka odluke 1** Dekoder promatra prva dva bita, 01. Odmah se uočava da je došlo do pogreške zbog toga što prva dva bita mogu biti samo 00 ili 11. No, koji je od dva bita primljen kao pogrešan, prvi ili drugi? Dekoder slučajno odabire 00 kao polazni izbor. Da bi bitovi odgovarali kombinaciji 00, dekodira se ulazni bit kao 0. U svoj brojač pogrešaka dekoder stavlja 1. Sada se dekoder nalazi u točki 2.



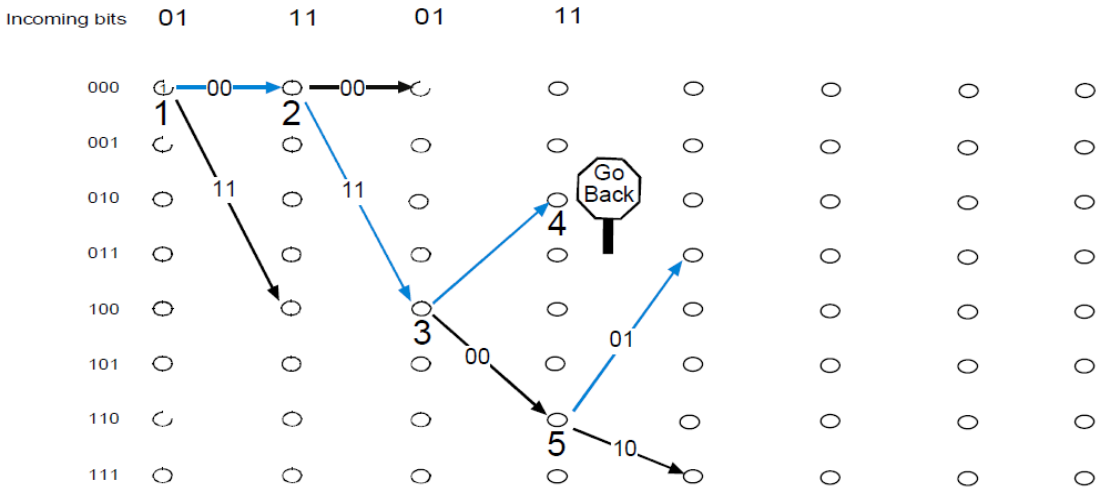
Slika 11a - Serijsko dekodiranje pretragom puta

[12] tallies … podudara
[13] tally … podudarati se

2. **Točka odluke 2** Dekoder promatra sljedeći skup od 2 primljena bita, a oni su 11. Odavde, donosi odluku da je poslana 1 što odgovara upravo jednoj od kodnih riječi. Ova odluka dovodi dekoder do točke 3.

3. U **točki odluke 3.** Primljeni bitovi su 01, ali pod-izbori za kodne riječi su 11 i 00. Uočava se kao pogreška i broj pogrešaka se povećan na 2. Dok god je broj pogrešaka manji od praga vrijednost 3 (koje smo postavili na temelju statistike kanala) dekoder nastavlja naprijed. On proizvoljno odabire gornji put (11) (prema točki 4) i nastavlja do donošenja odluke u točki broj 4 pošto se poslala 0.

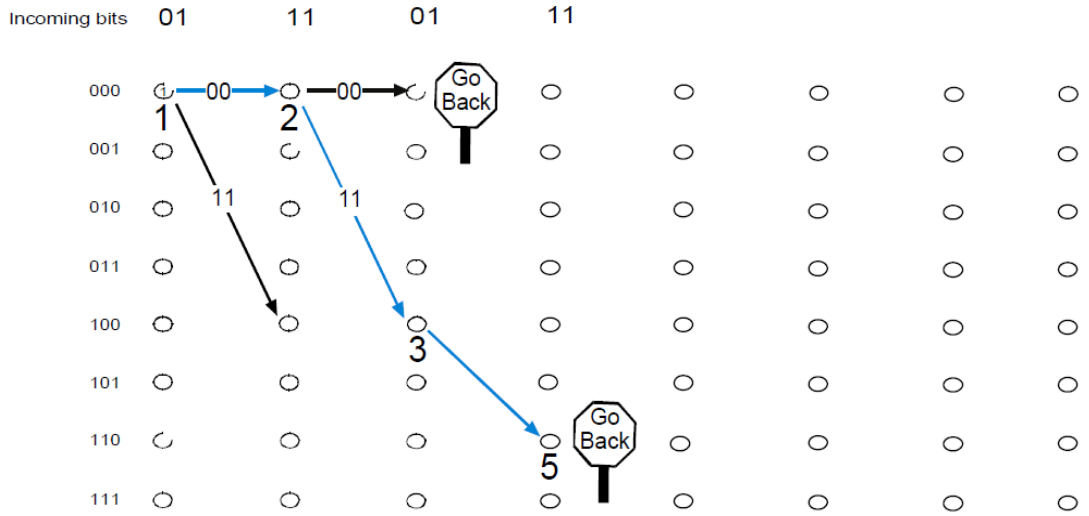
4. U **točki odluke 4**. Dekoder prepoznaje još jednu pogrešku, jer su primljeni bitovi 11, ali su mogući izbori za kodne riječi 10 i 01. Broj pogreška povećava ukupan zbroj i izjednačava s 3 pa to upućuje dekoder da se vrati natrag jedan korak.



Slika 11b - Traženje puta serijskim dekodiranjem

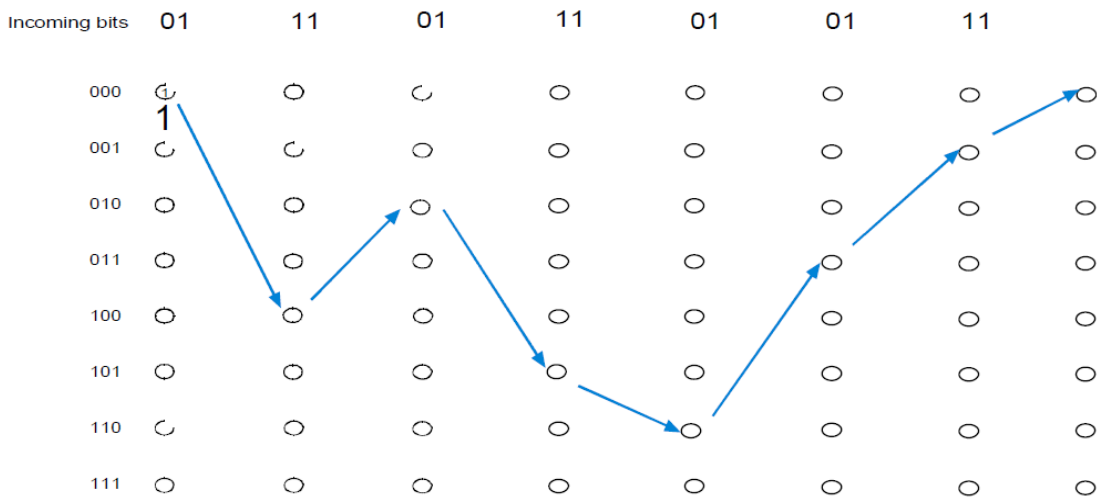
5. Dekoder se vraća do točke 3, gdje je zbroj pogrešaka manji od 3 i izabire drugi put iz prethodne točke 3 u točku 5. Dekoder opet nailazi na pogrešku. Primljeni bitovi su 11, ali moguće kodne riječi su 01 i 10. Brojač pogrešaka opet raste na 3 pa se dekoder vraća natrag.

6. Oba moguća puta iz točke 3 su iscrpljena. Dekoder mora ići natrag do točke 2 (prije 3. točke). On se i vraća do točke 2, ali ovdje, ako prati točku 2, brojač pogrešaka odmah se povećava na ukupan zbroj 3. Dakle, dekoder se mora vratiti iz točke 2 natrag do točke 1.



Slika 11c - Traženje puta serijskim dekodiranjem

Od točke 1, svi izbori na koje se naiđe, savršeno se podudaraju s izborom kodne riječi i dekoder uspješno dekodira poruku kao 1011000



Slika 11e - Traženje puta serijskim dekodiranjem

Za serijsko dekodiranje nizova upotrebljava se memorija pa se tako ova metoda koristi zajedno s kodovima duge duljine ograničenja gdje je *S/N* također malen. Neke veze NASAinih planetarnih misija koristile su prednost serijskoga dekodiranja.

1.20. 1.20. Maksimalna vjerojatnost i Viterbijevo dekodiranje

Viterbijevo dekodiranje je najpoznatija provedba dekodiranja najvećom vjerojatnosti. Ovdje smo sustavno suziti mogućnosti u svakome vremenskom intervalu. Glavna korist smanjenja izbora je:

1. Pogreške se javljaju rijetko. Vjerojatnost pogreške je mala.
2. Vjerojatnost dvije pogreške u nizu je mnogo manja od vjerojatnosti pojave jedne pogreške tj. one su slučajno raspodijeljene.

Viterbijev dekoder ispituje čitav primljen niz zadane duljine. Dekoder izračunava mjeru za svaki put i donosi odluku na temelju toga pokazatelja. Istražuju se svi putovi sve dok se dva puta ne spoji u jednome čvoru. Zatim se odabire put s višom mjerom, a onaj s nižom mjerom se odbacuje. Staze što se odaberu zovu se *preživjeli* (*survivors*).

Za niz od N bitova, ukupan broj mogućih primljenih nizova je 2^N. Od njih, samo 2^{kL} su valjani. Viterbijev algoritam primjenjuje *načela najveće vjerojatnosti* (*maximum-likelihood principles*) za ograničiti usporedbu na 2^{kL} preživjelih staza umjesto da provjerava sve staze.

Najčešća korištena mjera udaljenosti binarnih simbola je Hammingove mjera. Ona je samo običan umnožak[14] između *primljene* i *dopuštene* kodne riječi. Ostale mjere također se koriste i o njima će se govoriti poslije.

Tablica 5 - Svaka grana ima Hammingovu mjeru ovisno o tome što se primilo i o ispravnoj kodnoj riječi u tome stanju

primljeni bitovi	ispravna kodna riječ 1	ispravna kodna riječ 2	Hammingova mjera 1	Hammingova mjera 2
00	00	11	2	0
01	10	01	0	2
10	00	11	1	1

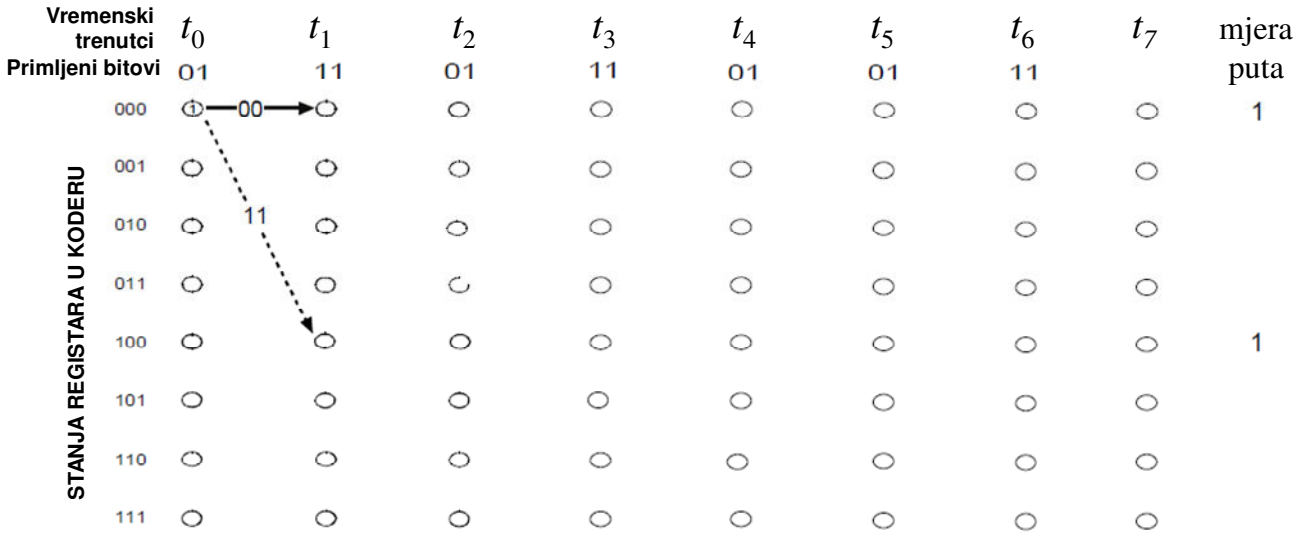
Ove mjere su kumulativne, tako da je put s najvećom ukupnom mjerom konačan pobjednik. No, sve to ne bi imalo smisla dok ne vidite algoritam u radu.

Neka se dekodira primljen niz **0**1 11 01 11 01 01 11 pomoću Viterbijeva dekodiranja.

1. U trenutku t = 0, primili smo bitove 01. Dekoder uvijek počinje iz stanja 000. Od ove točke postoje dvije raspoložive staze, ali one se ne podudaraju s dolaznim bitovima. Dekoder izračunava mjeru grane za obje staze i nastavlja istodobno duž obje ove grane što je u suprotnosti

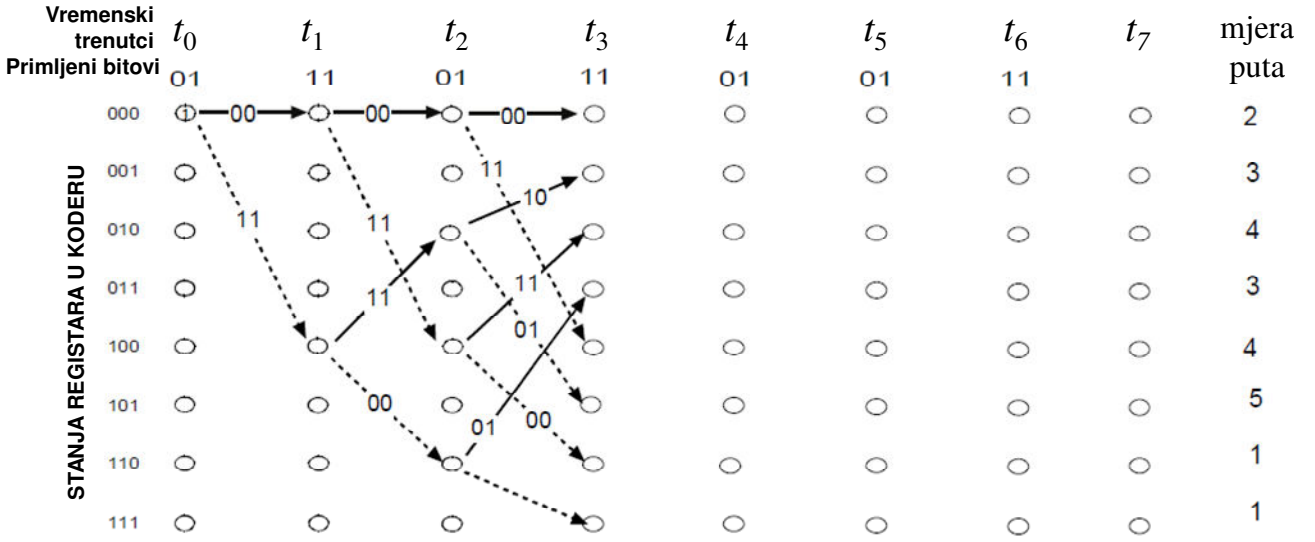
[14] VIP!

sa serijskim dekodiranjem gdje se izbor pravi u svakoj točki odluke. Mjera za oboje grane je jednaka 1, što znači da jedan od dva bita nije u skladu s dolaznim bitovima.



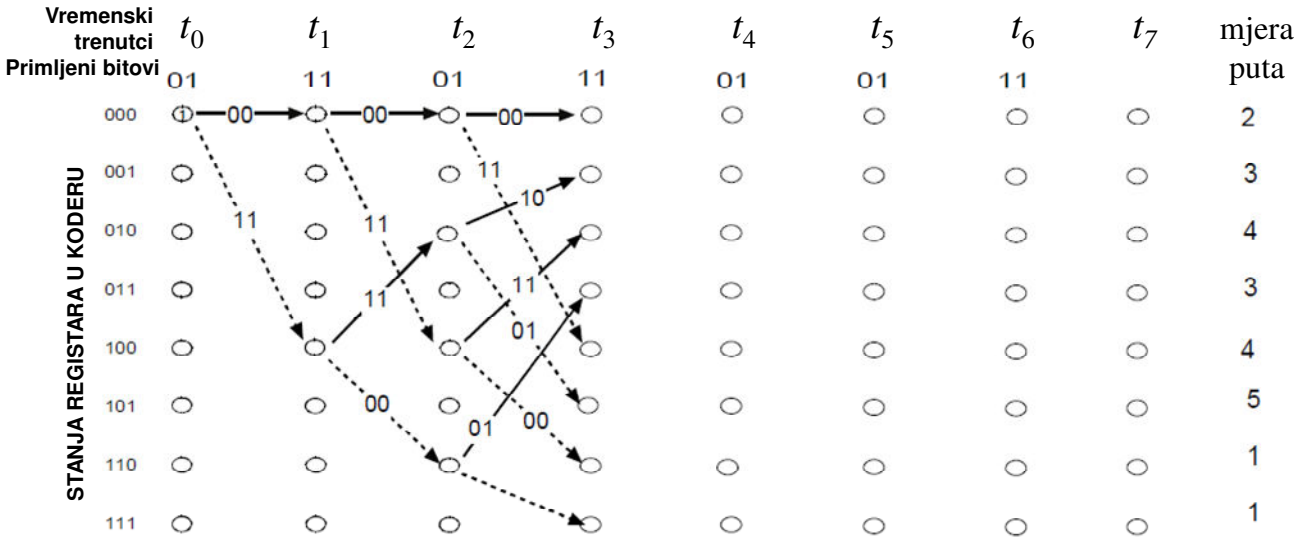
Slika 12a – Viterbijevo dekodiranja, korak 1

2. U trenutku t = 1, dekoder izlazi iz ovih dvaju mogućih stanja prema četiri stanja. Računaju se mjere za ove grane promatranjem podudarnosti s kodnom riječi i dolaznim bitovima koji su jednaki 11. Nova mjera je prikazana na desnoj strani rešetke.



Slika 12b - Viterbijevo dekodiranje, korak 2

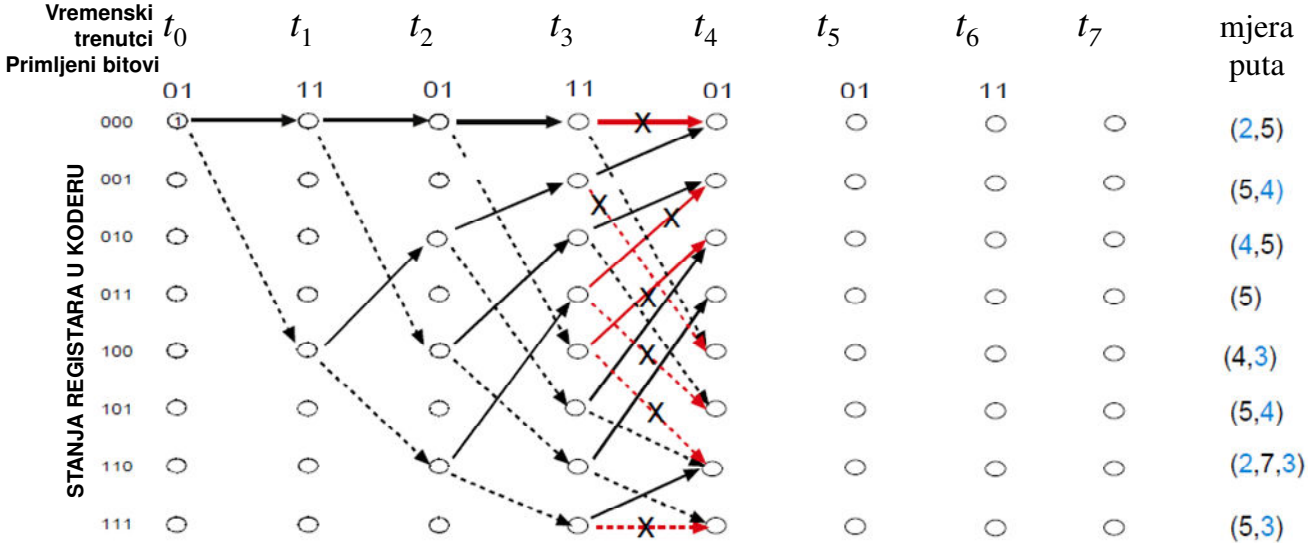
3. U trenutku t = 2, izašlo se iz četiri stanja u osam za pokazati sve moguće staze. Računaju se mjere putova za bitove 01 i dodaju se prijašnjim mjerama u trenutku t = 1.



Slika 12c - Viterbijevo dekodiranja, korak 3

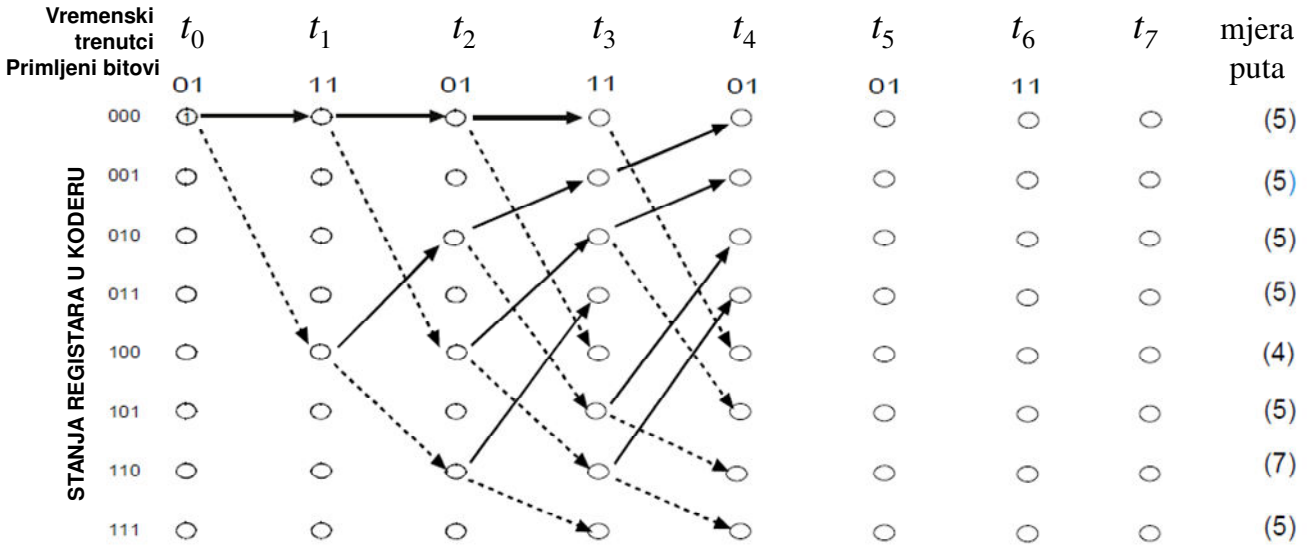
4. U trenutku $t = 4$, rešetka je u potpunosti ispunjena. Svaki čvor ima barem jedan put što ulazi u njega. Mjere su prikazane na slici.

5. U trenutku $t = 5$, napreduje se stazama prema naprijed i sada se počinju približavati čvorovima. Dvije mjere su dane za svaku od staza što ulaze u čvor. Prema načelu najveće vjerojatnosti, u svakome čvoru odbacujemo put s nižom mjerom, jer je on najmanje vjerojatan. Ovo odbacivanje puta, u svakome čvoru pomaže da se smanji broj staza koje treba ispitati i osnažuje Viterbijevu metodu.



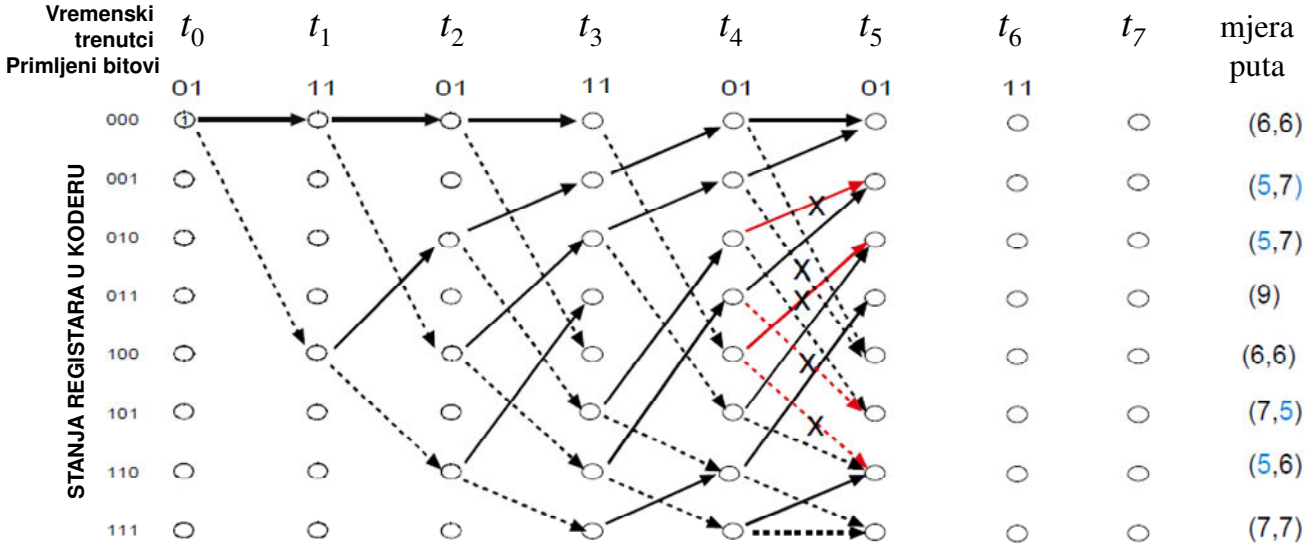
Slika 12d - Viterbijevo dekodiranja, korak 4

6. Sada, u svakome čvoru, imamo jedan ili više putova što se razdvajaju. Mjere za sve staze nalaze se na desnoj strani. U svakome čvoru, „držimo“ se samo puta s najvišom mjerom, a odbacujemo sve druge putove, što prikazuje crveno obojana crta. Nakon odbacivanja staze s manjom mjerom, imamo sljedeće preostale staze. Prikazana mjera je „pobjednik“ puta.



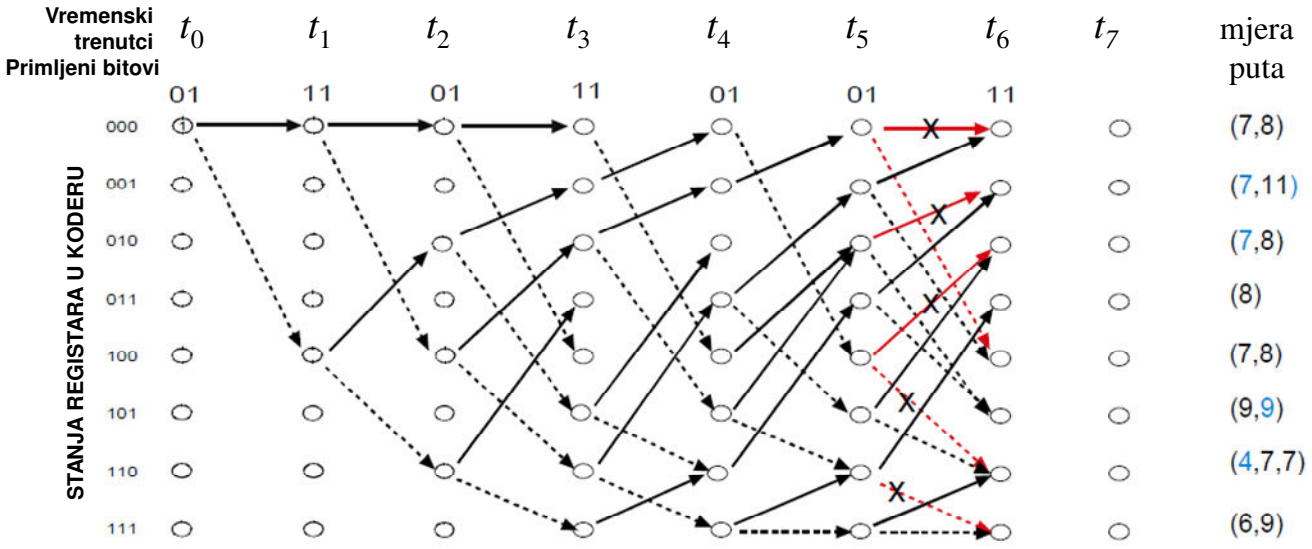
Slika 12e - Viterbijevo dekodiranja, 4. korak, nakon odbacivanja

7. U trenutku t = 5, nakon odbacivanja staza, kako je prikazano, ponovno idemo naprijed i računamo nove mjere. U sljedećem čvoru, ponovno se staze približavaju i opet odbacujemo one s manjim mjerama.



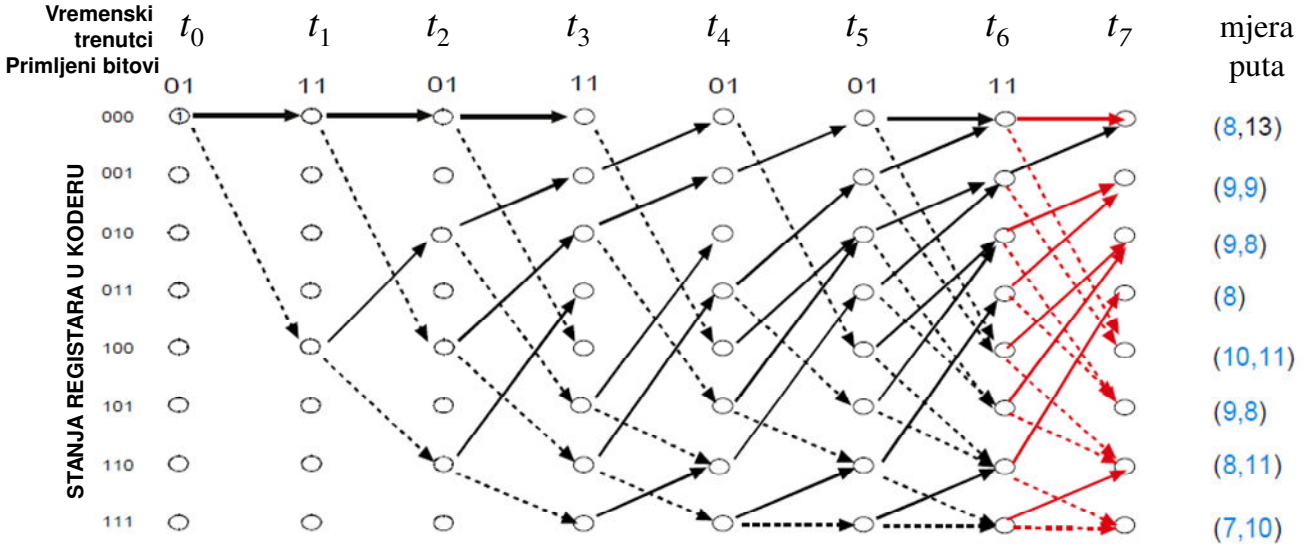
Slika 12f - Viterbijevo dekodiranje, korak 5

8. U trenutku t = 6, primamo bitove 11. Opet se računaju mjere za sve staze. Odbacujemo sve manje mjere ali zadržavamo obje mjere ako su jednake.



Slika 12g - Viterbijevo dekodiranje, korak 6

9. U sedmome koraku rešetka završava. Sada tražim put s najvećom mjerom. Imamo pobjednika. Put što ga pratimo prema stanjima 000, 100, 010, 101, 110, 011, 001, 000 i pošto on odgovara bitovima 1011000, predstavlja dekodiran niz.



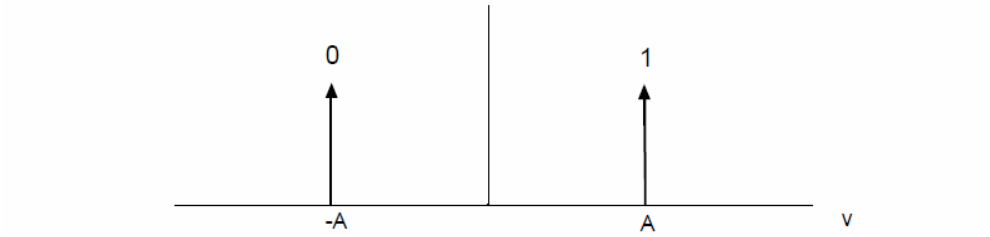
Slika 12h - Viterbijevo dekodiranje, korak 7

Duljina ove rešetke bila je 4 bita + *m* bitova. Idealno, ovo bi trebalo biti jednako duljini poruke, ali zbog postupka skraćivanja i zahtjeva skladištenje, može se smanjiti pa dekodiranje ne treba kasniti do kraja prijenosa niza. Tipična duljina sakaćenja konvolucijskoga kodera je 128 bita ili 5-7 puta veća od duljine ograničenja.

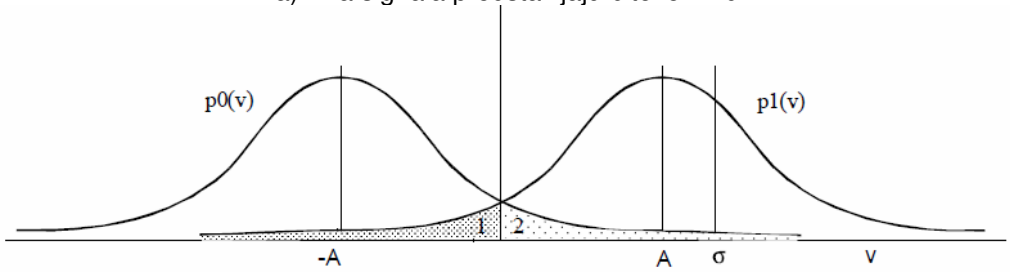
Viterbijevo dekodiranje je vrlo važno, jer ono također vrijedi i za dekodiranje blok-kodova. Ovaj oblik dekodiranja rešetkom također se koristi za *modulaciju kodiranom rešetkom* TCM (*Trellis-Coded Modulation*).

1.21. 1.18. Dekodiranje mekom odlukom

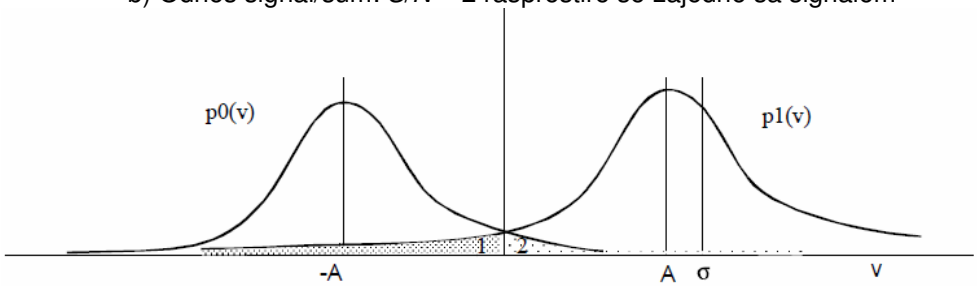
Spektar dva signala koji se koriste za predstavljanje bitova 0 i 1 kada se podvrgne šumu, jako otežava proces odlučivanja. Signal se prostire i energija s jednoga signala prelijeva se u drugi. Slika 13.b i 13.c, prikazuje dva različita slučaja za različite odnose *S/N*. Ako je dodan šum malen, tj. njegova varijance je mala (jer je snaga šuma = varijanca), onda će raspršenjem biti još manja, kao što možemo vidjeti na slici 13.c u odnosu na sliku 13.b. Intuitivno možemo vidjeti iz ovih slika kako bi se napravila manje vjerojatna pogreška pri dekodiranju, ako bi odnos *S/N* bio velik ili ako bi varijanca šuma bila mala.



a) Dva signala predstavljaju bitove 1 i 0



b) Odnos signal/šum: *S*/*N* = 2 rasprostire se zajedno sa signalom



c) Odnos signal/šum: *S*/*N* = 4 rasprostire se i prelijeva energiju iz jednoga područja u drugo

Slika 13: Prelijevanje energije iz jednoga područja u drugo

Dekodiranje tvrdom odlukom (*hard decision decoding*) znači da se između dva signala obično odabire jednostavan prag odluke, tako da, ako je primljen napon pozitivan onda se signal dekodira kao 1 inače se dekodira kao 0. U vrlo jednostavnim uvjetima to znači maksimalnu vjerojatnost dekodiranja.

Ovom metodom odlučivanja možemo brojčano izraziti (kvantificirati) napravljenu pogrešku. Vjerojatnost da će se dekodirati 0, s obzirom da je poslana 1, predstavljena je funkcijom dvaju osjenčanih područja kao što prikazuje slika 13.

Područje 1 predstavlja tu energiju što pripadaju signalu za 1, što se „prelila“ u suprotno područje odluke pa se time bit pogrešno dekodirao kao 0 i u područja 2, što je energija koja pripada bitu 0, što se prelilo u područje. To se oduzima od primljenoga napona pa stoga potencijalno uzrokuje pogrešku pri odluci.

S obzirom da je poslana 1, vjerojatnost da će se 1 dekodirati kao 0 je:

$$Pe1 = \frac{1}{2}\,erfc\left(\frac{A - v_t}{\sqrt{2}\sigma}\right)$$

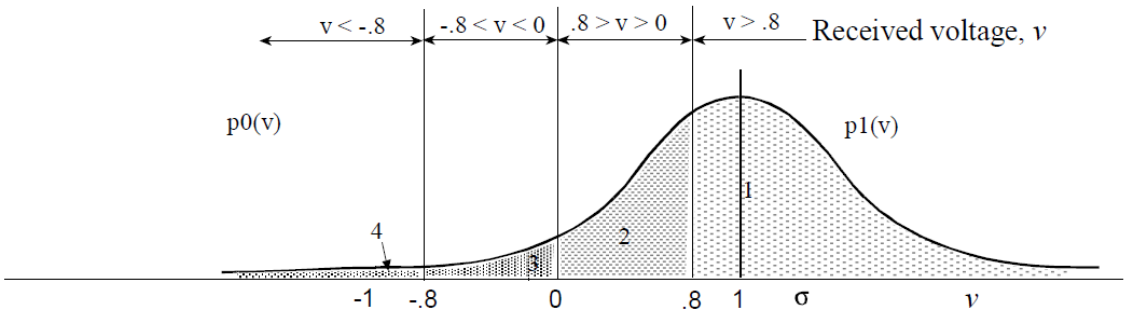
v_t … napon praga odluke, što je u našem slučaju 0.

σ … varijanca šuma ili njegova snaga.

Možemo preraditi ovu jednadžbu kao funkciju *S*/*N*:

$$Pe1 = \frac{1}{2}\,erfc\left(\sqrt{\frac{S}{N}}\right)$$

Ovo je poznata jednadžba odnosa pogrešnih bitova. Vidimo da se pretpostavlja dekodiranje tvrdom odlukom. Ali što ako učinimo sljedeće, umjesto da imamo samo dva područja odluke, podijelimo raspoloživo područja u četiri područja kao što se prikazuje u nastavku.



Slika 14 - Stvaranje četiri područja za odluku

Vjerojatnost da je odluka ispravna može se izračunati iz područja ispod Gaussove krivulje.

Četiri područja su organizirana na sljedeći način:

Područje 1 = ako je primljen napon veći od 0,8 V

Područje 2 = ako je primljen napon veći od 0, ali manji od 0,8 V

Područje 3 = ako je primljen napon veći od –0,8 V, ali manji od 0 V

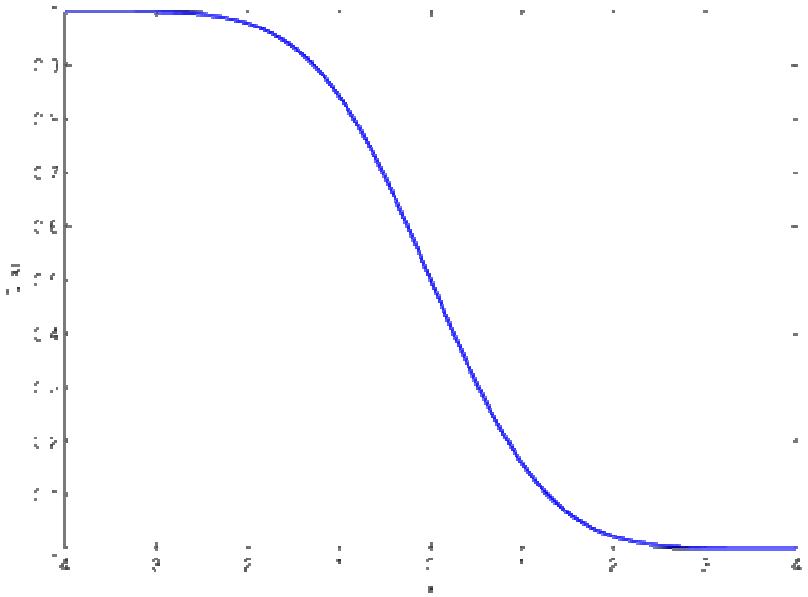
Područje 4 = ako je primljen napon manji od –0,8 V

Sada se pitamo? Ako primljeni napon padne u područje 3, kolika je onda vjerojatnost pogreške da se poslala 1?

Za dekodiranje tvrdom odlukom, odgovor je jednostavan. On se može izračunati iz jednadžbe. A kako bismo izračunati slične vjerojatnosti za prostor s višestrukim područjima?

Koristimo *Q* funkciju koja je dana u tablicama mnogih knjiga.

1.22. 1.22 Q-funkcija



Formalno, *Q*-funkcija definirana je kao

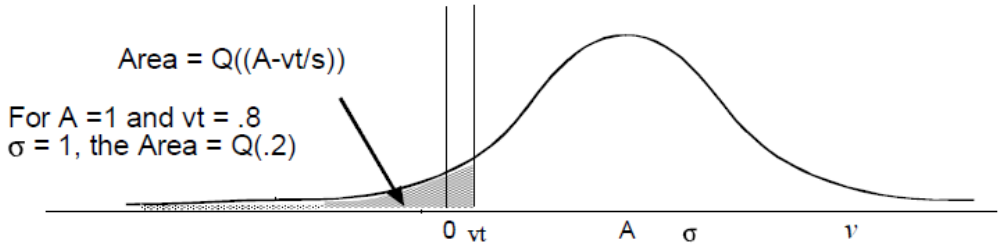
$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp\left(-\frac{u^2}{2}\right) du.$$

Dakle,

$$Q(x) = 1 - Q(-x) = 1 - \Phi(x),$$

Gdje je $\Phi(x)$ kumulativna funkcija razdiobe normalne Gaussove razdiobe.

Q funkcija definira nam prostor ispod repa krivulje što se definira udaljenošću od srednje vrijednosti prema bilo kojoj drugoj vrijednosti. Dakle *Q*(2) definira signal s aritmetičkom sredinom 2 i daje nam vrijednost vjerojatnosti koja je jednaka ili veća od 4.



Slika 15 - Q funkcija je jednostavan način da se utvrdi vjerojatnost normalno distribuirane varijable

Pretpostavlja se da je $vt = 0{,}8$ (što je standardna vrijednost broja za 4 razine), ali to može biti bilo koji broj. Jednadžbe se mogu napisati skoro intuitivno, one su tako očite.

P_{e1} (vjerojatnost da je poslana 1 ako je primljeni napon u području 1) = $1 - Q(-vt/\sigma)$

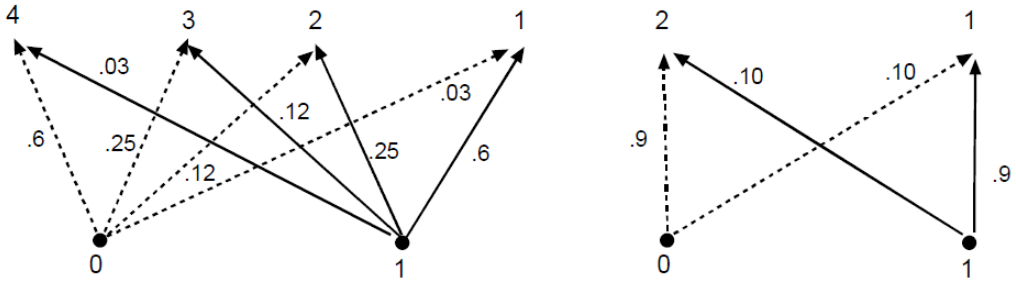
P_{e4} (vjerojatnost da je poslana 1 ako je primljeni napon u području 4) = $Q(2(A + vt/\sigma)$

P_{e2} (vjerojatnost da je poslana 1 ako je primljeni napon u području 2) = $1 - P_{e1} - Q(A/\sigma)$

P_{e3} (vjerojatnost da je poslana 1 ako je primljeni napon u području 3) = $1 - P_{e1} - P_{e2} - P_{e4}$

Izračunali smo da je ovaj $S/N = 1$ ako je $vt = 0{,}8$, a $A = 1{,}0$. Također smo pretpostavili da su bitovi 0 i 1 jednako vjerojatni. Ovo se također nazivaju *apriorne* vjerojatnosti za bitove 0 i 1 i gotovo se uvijek pretpostavlja da su jednake u komunikaciji osim za radarske sustave, gdje apriorne vjerojatnosti obično nisu poznate.

Dakle, područje 4 ili P_{e4} jednako je $Q(1{,}8)$ što se može iščitati iz tablica. Područje 1 tada je jednako $1–Q(0{,}2)$. Ostala područja mogu se brzo izračunati na ovaj način. Možemo pokazati uvjetne vjerojatnosti izračunate na grafički način.



Slika 16 - Uvjetne vjerojatnosti dekodiranja 0 ili 1 ovisno o veličini primljenoga napona, kada se napon kvantizira u 4 razine u odnosu na dvije razine.

Ovaj proces diobe prostora odluke u područja veća od dva, kao što je ovaj primjer s 4 razine, naziva se *dekodiranje mekom odlukom* (*soft decision decoding*). Ove vjerojatnosti također se nazivaju *prijelazne vjerojatnosti*.

Postoje 4 različite vrijednosti napona za svaki primljeni signal kojima donosimo odluku. Među signalima, kao i u stvarnome životu, više informacija znači bolje odluke. Meka odluka unaprjeđuje osjetljivost mjere dekodiranja i poboljšava svojstva za čak 3 dB u slučaju meke odluke s 8 razina.

Sada, za izračunati mjeru meke odluke napraviti smo tablicu od gore navedenih vjerojatnosti.

poslano	v4	v3	v2	v1
1	.03	.12	.25	.60
0	.6	.25	.12	.03

Sada se računa prirodan logaritam svake od ovih vrijednosti i normalizira tako da je jedna od vrijednosti 0. Nakon maloga rukovanja brojevima, dobivamo

poslano	v4	v3	v2	v1
1	.0	–1	–4	–10
0	–10	–4	–1	0

U dijelu o dekodiranju, izračunali smo Hammingovu mjeru množenjem dobivenih bitova kodnim riječima. Sada radimo istu stvar, osim što umjesto primanja 0 i 1, primamo jedan od ovih napona. Dekoder gleda mjeru za taj napon iz tablice kao što je ona iznad što je drži u svojoj memoriji i čini sljedeći izračun.

Pretpostavimo da smo primili naponski par (v_3, v_2). Dopuštene kodne riječi su 01 i 10.

Mjera za 01 = $p(0/v_3) + p(1/v_2) = -4 + -4 = -8$

Mjera za 10 = $p(1/v_3) + p(0/v_2) = -1 + -1 = -2$

Promatrajući samo ove dvije mjere možemo kazati da je pojava 01 u odnosu na 10 puno vjerojatnija. Kada se ove mjere dodaju, one preuveličavaju razlike i pomažu otkrivanje rezultata dekodiranja.

1.23. 1.19. Logaritamski odnos

Postoje i drugi načini za poboljšanje svojstava dekodiranja poigravanjem s mjerom. Važna mjera koju treba poznavati zove se mjera logaritamske vjerojatnosti (*log likelihood metric*). Ovaj mjera uzima u obzir vjerojatnost pogreške kanala, a definirana je kao

$$\text{mjera podudaranja: } = \frac{\log_{10} 2(1-p)}{\log_{10} 2}$$

$$\text{mjera neslaganja =: } \frac{\log_{10} 2(p)}{\log_{10} 2}$$

Za $p = 0{,}1$

Mjera podudaranja = –20

Mjera neslaganja = –1

Dakle, ako smo primili bitove 01, a kodna riječ je 00, ukupna mjera bit će –20 + –1 = –21, a mjerilo za potpuno podudaranje bit će –40.

Fano algoritam što se koristi za serijsko dekodiranje upotrebljava malo drugačiju mjeru, a svrha svega toga je poboljšanje osjetljivosti.

1.24. MATLAB implementation

MATLAB supports convolutional codes. For example the encoder shown on Img. 1 can be implemented as follows:

```
G1 = 7;% octal 7 corresponds to binary 111 n1 = m1 + m0 + m-1
G2 = 3;% octal 3 corresponds to binary 011 n1 = m0  + m-1
G3 = 5;% octal 5 corresponds to binary 101 n1 = m1  + m-1
constLen = 3;   % Constraint length

% Create the trellis that represents the convolutional code
convCodeTrellis = poly2trellis(constLen, [ G1 G2 G3 ]);
uncodedWord = [1 ];
codedWord1 = convenc(uncodedWord, convCodeTrellis)
uncodedWord = [1 0 0 0];
codedWord2 = convenc(uncodedWord, convCodeTrellis)
```

The output is the following:

codedWord1 =

```
1     0     1
```

codedWord2 =

```
1     0     1     1     1     0     1     1     1     0     0     0
```

The bits of the first output stream are at positions 1,4,7,...,3k+1,... in output vector *codedWord*, respectively second stream at positions 2,5,...,3k+2,... and the third 3,6,...,3k,...

Initial state is by default initialized by all zeros.

S. Lin and D. J. Costello, Error Control Coding. Englewood Cliffs, NJ: Prentice Hall, 1982.

A. M. Michelson and A. H. Levesque, Error Control Techniques for Digital Communication. New York: John Wiley & Sons, 1985.

W. W. Peterson and E. J. Weldon, Jr., Error Correcting Codes, 2nd ed. Cambridge, MA: The MIT Press, 1972.

V. Pless, Introduction to the Theory of Error-Correcting Codes, 3rd ed. New York: John Wiley & Sons, 1998.

C. Schlegel and L. Perez, Trellis Coding. Piscataway, NJ: IEEE Press, 1997

S. B. Wicker, Error Control Systems for Digital Communication and Storage. Englewood Cliffs, NJ: Prentice Hall, 1995.

Pitanja:

1. Što je to *impulsni* konvolucijskoga *odziv* kodera?
2. Što je to *brzina* koda?
3. Objasnite konvoluciju?
4. Koji su parametri konvolucijskoga koda?
5. Kakva je struktura konvolucijskoga koda?
6. Kako se odabiru polinomi konvolucijskoga koda?
7. Što su to stanja konvolucijskoga koda?
8. Objasnite pojam probušenoga konvolucijskog koda?
9. Objasnite strukturu koda za $k > 1$!
10. U čemu se razlikuju sustavan u odnosu na nesustavan konvolucijski kod?
11. Kako se kodira dolazni niz bitova konvolucijskog koda?
12. Kako se oblikuje konvolucijski koder?
13. Objasnite rad dijagrama stanja!
14. Objasnite rad dijagrama stabla!
15. Objasnite rad dijagrama rešetke!
16. Kako dijagram rešetke koristi kodiranje?
17. Objasnite dekodiranje!
18. Što je osnovna ideja dekodiranja?
19. Kakvo je to serijsko dekodiranje?
20. Objasnite dekodiranje algoritmom dekodiranja nizova!
21. Objasnite algoritam dekodiranja maksimalnom vjerojatnosti!
22. Objasnite algoritam Viterbijeva dekodiranja!
23. Što je to dekodiranje mekom odlukom?
24. Što je to dekodiranje tvrdom odlukom?
25. Čemu služi CLK ulaz u shift registar?
26. Tablice istine
27. Ekskluzivno ILI
28. Serial-in to Parallel-out (SIPO)
29. Objasniti Basic Movement of Data through a Shift Register dijagramom!

~~Logaritamski odnos~~

Laboratorijske vježbe:

1. Koristiti program Logisim Version 2.7.1, 2011.
2. Pojam i rad bistabila (flip-flop sklop)
3. Objasniti rad svih relevantnih komponenti
4. D bistabil
5. Vrste registara
6. Serijski ulaz - paralelan izlaz
7. 4-bit Universal Shift Register 74LS194
8.
9.