INFORMATICA 2/1979

KONVOLUCIJSKI KODIRNI IN DEKODIRNI POSTOPEK PRI MIKRORAČUNALNIKU ISKRA DATA 1680

M. KAPUS
G. DEVIDE
B. HORVAT

UDK: 681.325.3

VISOKA TEHNIŠKA ŠOLA, MARIBOR

POVZETEK - Brez dvoma je eden najmodernejših in atraktivnih dekodirnih postopkov v prenašanju in zavarovanju digitalne informacije Viterbijev algoritem. V naslednjem desetletju pričakujemo v komunikaciji podatkov ta algoritem realiziran v integrirani obliki s pomočjo materialne opreme. Zaradi visoke redundance se tak postopek koristi predvsem pri prenosu podatkov FEC brez povratnega komunikacijskega kanala. Pri komuniciranju med procesorji in v želji po čim večji zanesljivosti smo koristili ta kodni in dekodni postopek in ga realizirali s pomočjo programske opreme.
Kodirnik je preprost pomikalni register povratno povezan s seštevalniki po m = 2. Dekodiranje je bolj zapleteno in je bistvo članka. Simulirali smo celoten prenosni sistem za ocenitev prenosa. Komunikacijski kanal smo ponazorili kot normalno porazdelitev z generatorjem naključnih števil. Rezultati so obetavni in kažejo na veliko zanesljivost prenosa kljub množici motenj v kanalu.

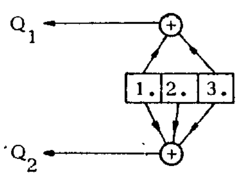
CONVOLUTIONAL CODING AND DECODING PROCEDURE BY MICROCOMPUTER ISKRA DATA 1680 - Without doubt is in communication the Viterbi algorithm one of the up to date decoding procedure. In next decade we expect that the procedure be realized in integrated form on the chip. The procedure take more use in FEC transmission and need for deplexing the same number information as control bits. By communication between processors to achieve more transmission reliability we use this coding and decoding procedures with software suport.
The coder is simple shift register feedback conected with exlcusive or elements. The decoding procedure is more complex and the mathematical structure which we take is explained in this article. We simulate the communication channel with a random number generator, to value the transmission system. The decoding procedure has shown succesfull transmission with a great degree of reliability on channel errors.

1. Uvod

Če želimo dvigniti zanesljivost prenašanih sporočil v digitalnih sistemih, pri katerih ni povratnega kanala, se odločamo najpogosteje za popravljanje napak na sprejemni strani. Tak način prenosa zahteva dokajšno redundanco prenešenega sporočila v obliki kontrolnih bitov, ki jih privesimo informacijskim bitom. Ravno tako moramo računati s povečanim procesorskim časom in povečano pomnilniško kapaciteto. Želeli smo ustvariti poseben minimalni procesorski modul, ki bi bil kot komunikacijski procesor priključen na večji računalniški sistem in bi tako zagotavljal povečano zanesljivost vsem onim vhodno/izhodnim vodilom, kjer ni možnosti za povraten kanal, torej za prenos ARQ sistema. Takšni komunikacijski kanali so pogosto vsi prenosi podatkov na pomnilniške medije in vse prenosne poti, kjer pogosto ponavljanje informacijskih paketov ne bi zavrlo prepustnost informacijskega pretoka. Razumljivo je, da komunikacijska pot mora imeti določeno kvaliteto in nenavadne neprevelike aditivne napake kanala nam dekodirni proces izloči. Sam kodirni postopek je zelo preprost in ne zahteva procesorskega časa in ga najpreprosteje rešimo v materialni opremi. Dekodirni proces je mnogo bolj zamotan in podana je njegova programska rešitev. Izbrali smo minimalno konfiguracijo registrov in si s tem zagotovili manjši čas procesiranja in manjšo pomnilniško kapaciteto.

2. Opis kodirnega in dekodirnega algoritma

Pri sinhronem prenosu sporočil nimamo povratne kontrole, ali je bil prenos pravilen. Zato moramo uporabiti načine kodiranja, kjer bo šifra čim manj občutljiva na motnje. Eden takih načinov je konvolucijsko kodiranje. Odločili smo se za kodirnik z delovno besedo dolžine treh bitov.



informacijski biti

Informacijske bite vodimo skozi pomikalni register in vsakokrat izračunamo Q_1 in Q_2. Namesto sekvence informacijskih bitov oddajamo sekvenco urejenih parov (Q_1, Q_2). Ta je dvakrat daljša od prvotne in implicitno vsebuje razen osnovne informacije še odnose med zaporednimi tremi biti. Zaradi tega je prenos zanesljivejši.

Ker ne vemo, koliko se je informacija pri prenosu pokvarila, jo lahko dekodiramo le z verjetnostnim računom. Seveda ni mogoče zajeti vseh motenj, ki se lahko pojavijo, saj bi potrebovali ogromen spomin. Zato zajamemo le najverjetnejše, to je minimalne motnje. Rezultat teh motenj je zaporedje, ki se kar najmanj razlikuje od oddane sekvence.

Na tem principu sloni Viterbijev algoritem.
Izmed informacijskih bitov, ki so trenutno v postopku kodiranja, bosta 2. in 3. bit skupaj z naslednjim informacijskim bitom I določala vrednost Q_1, Q_2 v naslednjem koraku. Označimo urejeni par 2. in 3. bita z

x_i ; x_i $a = 00$, $b = 01$, $c = 10$, $d = 11$

S sprejemom novega inf. bita dobimo par x_{i+1}. Pri tem so možni prehodi, kot jih kaže tabela 1.

$x_i \quad I_{i+1} \quad x_{i+1}$

Urejena trojica (x_i, I_{i+1}) določa par $(Q_1, Q_2)_{i+1}, x_i$. Ta se pri prenosu moti in dejansko sprejmemo par $(Q_1, Q_2)_{i+1}$, ki se od para $(Q_1, Q_2)_{i+1}, x_i$ razlikuje na m_{i+1,x_i} mestih. $m_{i+1}, x_i \quad 0,1,2$

Definirajmo funkcijo poti $P_{i+1,x_{i+1}}$.

$$P_{i+1,\, x_{i+1}} \begin{array}{l} = 0 \;==\; x_i \quad a,b \\ \;\;\, 1 \;==\; x_i \quad c,d \end{array}$$

x_{i+1}	$I_{x_{i+1}}$	x_i	$(Q_1,Q_2)_{i+1,x_i}$	$P_{i+1,x_{i+1}}$	m_{i+1,x_i} $(Q_1,Q_2)_{i+1}$ 00	01	10	11
a	0	a	00	0	0	1	1	2
		c	11	1	2	1	1	0
b	1	a	11	0	2	1	1	0
		c	00	1	0	1	1	2
c	0	b	10	0	1	2	0	1
		d	01	1	1	0	2	1
d	1	b	01	0	1	0	2	1
		d	10	1	1	2	0	1

Tabela 1

Sprejeli smo sekvenco Q_i parov $(Q_1, Q_2)_j$; $j = 1,\dots,i$. Iz nje lahko sklepamo, kakšna je bila sekvenca inf. bitov S_i. Sklepamo ločeno za primere, ko se S_i končuje z a, b, c oziroma d. Tako dobimo štiri sekvence S_{i,x_i};

$x_i \quad a,b,c,d$. Vsaka sekvenca S_{i,x_i} nam da sekvenco Q_{i,x_i}, ki se od dejansko sprejete sekvence Q_i razlikuje na M_{i,x_i} mestih.

$$S_{i,x_i} \;==\; Q_{i,x_i}$$

$$Q_i - Q_{i,x_i} = M_{i,x_i}$$

V naslednjem koraku sprejmemo par $(Q_1, Q_2)_{i+1}$ in za vsako končnico x_{i+1} spet poiščemo najverjetnejšo vhodno sekvenco $S_{i+1,x_{i+1}}$. Nove sekvence so seveda nastale z dodajanjem naslednjega informacijskega bita starim vhodnim sekvencam.

$$S_{i+1,x_{i+1}} = S_{i,x_i},\, I_{x_{i+1}}$$

Odnos med x_{i+1}, x_i, $I_{x_{i+1}}$ že poznamo iz tabele 1.

$$S_{i+1,x_{i+1}} == Q_{i+1,x_{i+1}} \;;\; Q_{i+1,x_{i+1}} = Q_{i,x_i},$$

$$(Q_1,Q_2)_{i+1,x_i}$$

$$Q_{i+1} - Q_{i+1,x_{i+1}} = Q_i - Q_{i,x_i} + (Q_1,Q_2)_{i+1} - (Q_1,Q_2)_{i+1,x_i} =$$

$$= M_{i,x_i} + m_{i+1,x_i} = M_{i+1,x_{i+1}}$$

Ker sta za vsak x_{i+1} možna dva x_i, izberemo verjetnejšega, to je tistega, kjer je $M_{i+1,x_{i+1}}$ manjša.

$$x_i : \; M_{i+1,x_{i+1}}(x_i) = \min.$$

Izbrani x_i določa funkcijo poti:

$$P_{i+1,x_{i+1}} = P_{i+1,x_{i+1}}(x_i)$$

Izmed štirih sekvenc $S_{i+1,x_{i+1}}$ vzamemo kot pravilno tisto, kjer je

$$x_{i+1} : \; M_{i+1,x_{i+1}}(x_{i+1}) = \min.$$

Ker vse sekvence S_{i,x_i} težijo k isti najverjetnejši sekvenci S, se začenjajo deloma ujemati v bitih, ki so nastali prej in so šli že večkrat skozi postopek preverjanja verjetnosti. S poskusi so ugotovili, da je to dovolj pogosto že na mestu (i - D.4), strožje na mesti (i - 5.D), kjer je dolžina delovne besede kodirnika, v našem primeru je D = 3. Delov sekvenc S_{i,x_i}, ki so skupni, ni več mogoče izboljšati, zato predstavljajo rezultat dekodiranja. Torej lahko predstavimo sekvence v registrih z dolžino 4.D, bit, ki vsakokrat izpade iz registra, pa je rezultat. Program smo realizirali na MC 6800, ki ima 8-bitne celice, zato smo uporabili 16-bitne registre.

Za S_{0,x_0} lahko izberemo poljubno zaporedje, prav tako za M_{0,x_0} poljubne vrednosti, saj bodo te zanemarljive napram vsoti m_{i,x_i} višjih redov. Mi smo izbrali $S_{0,x_0} = 0^{16}$, $M_{0,x_0} = 0$.

Za naslednji korak so pomembne le relativne razlike med M_{i,x_i}, zato vsakokrat odštejemo minimalno.

$$M_{i,x_i} \qquad M_{i,x_i} - M_{i,x_i}(x_i) \min.$$

Pri tem se pokaže, da so vse možne kombinacije naslednje:

M_a	0	1	0	0	0	1	0	0	1	1	1	0	0	1	2	0	2	1	1	0	1	2	2
M_b	0	0	1	0	0	1	0	1	0	1	1	0	0	2	1	2	0	1	1	1	0	2	2
M_c	0	0	0	1	0	0	1	1	1	0	1	1	2	0	0	1	1	0	2	2	2	0	1
M_d	0	0	0	0	1	0	1	1	1	1	0	2	1	0	0	1	1	2	0	2	2	1	0
M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Tabela 2

M_a	0	2	2	2	0	2	3	3
M_b	2	0	2	2	2	0	3	3
M_c	2	2	0	2	3	3	0	2
M_d	2	2	2	0	3	3	2	0
M	23	24	25	26	27	28	29	39

Vsak niz M_a, M_b, M_c, M_d nadomestimo z njegovo zaporedno številko M;
M = 0, 1, ..., 30

Zahteve pri programiranju:

1. Čim več rezultatov izračunamo vnaprej in jih po potrebi le pokličemo iz spomina.
2. Uporabimo čim več spomina tipa EPROM in čim manj RAM-a.

Za vsak M izračunamo tabelo prehodov.

M_i	Q_1	Q_2	M_{i+1}	M	P_{i+1}	J_{i+1}	S	S
a	0	0	a	d	a	d		
	0	1				J	J_x	
	1	0						
d	1	1						

Primer za M = 19.

19									
0	0	0	0222	23	0x00	1	1	0	0
1	0	1	0010	3	0010	0	0	1	0
2	1	0	0001	4	0001	0	1	1	0
2									

Pri starem stanju M_i sprejmemo bita Q_1, Q_2. Dobimo minimalne razdalje - nabor M_{i+1} z imenom M, s tem da smo uporabili optimalne poti iz nabora P_{i+1}.

P_{i+1} = 0, če smo šli po zgornji poti, n.pr. a v a .
= 1, če smo šli po spodnji poti, n.pr. c v a .
= X, če sta poti enakovredni.

Mestom M_{i+1,x_i} priredimo naslednje dvojiške vrednosti:

$M_{i+1,a}$: 11 ; $M_{i+1,b}$: 10 ; $M_{i+1,c}$: 01 ; $M_{i+1,d}$:00.

J_{i+1} je oznaka minimalne $M_{i+1,x_{i+1}}$, to je registra, iz katerega bomo prebrali rezultat.

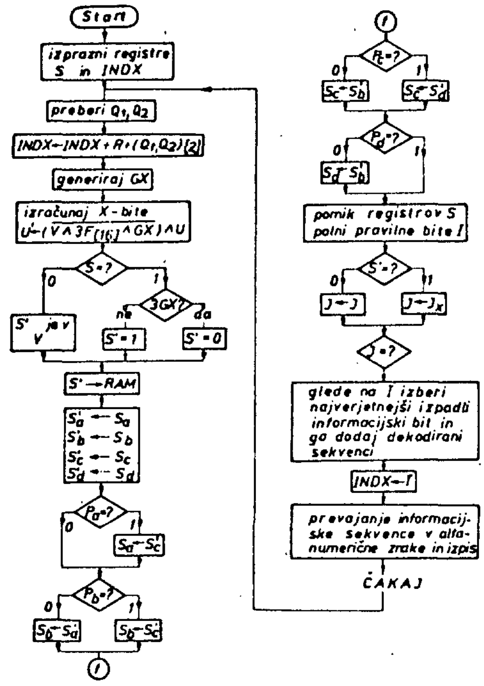
1. Če je M_{i+1} min. ena sama, pišemo njeno pozicijo v stolpec J. J_x= 00.
2. Če sta M_{i+1} min. dve, se poziciji vedno razlikujeta le po enem bitu. V stolpec J_x vpišemo skupni bit z njegovo pravo vrednostjo, bit se razlikuje, pa označimo z X. J = 00.
3. Če so M_{i+1} min. tri, dve sosednji vpišemo kot pod točko 2, tretjo pa kot pod točko 1.
4. Če so M_{i+1} min. štiri, je J_x = XX in J = 00.
5. S = 1 natanko v primeru 3, ko se moramo odločati med J in J_x z verjetnostjo p(j) : p(J_x) = 1 : 2.
6. S = 0 pomeni, da smo se odločili za J. To je v primeru 1, po naši prosti izbiri pa tudi v primeru 3, ko se S posebej izračuna v RAM-u.
7. S = 1, če se odločimo za J_x, to je v primerih 2 in 4.

Vsaki kombinaciji M, Q_1, Q_2 priredimo tri spominske celice v EPROM-u. Naj bo R najnižji uporabljeni naslov.

Potem najdemo na naslovu
$R + 4.M_i+(Q_1Q_2)_2$ celico T,
$R + 4.M_i+(Q_1Q_2)_2 + 4.31$ celico U,
$R + 4.M_i+(Q_1Q_2)_2 +8.31$ celico V.

0	M_{i+1}	0	0	T
J_{i+1}	J_x	P_{i+1}		U
S	S	X(J_x)	X(P_{i+1})	V

Biti v X(J_x) in X(P_{i+1}) imajo vrednost 1 na mestih, kjer v J_x in P_{i+1} nastopa X, sicer so 0. Prva celica je del naslova za naslednji cikel. R še ni upoštevan, zato je možno vse tri bloke spomina prestavljati na poljubne naslove brez sprememb. V začetku vsakega cikla se izračuna naslov celice T in se shrani v INDEX registru. Celice U in V so dosegljive z indeksiranim naslavljanjem.
Za polnjenje X bitov potrebujemo psevdogenerator naključnih števil GX. Glede na GX (mod 3) tudi določamo S , kadar je S = 1.
Potrebne celice RAM-a:
- GX,
- štirje dvocelični pomikalni registri za S_a, S_b, S_c, S_d,
- štirje dvocelični spomini za spravljanje registrov S, ko jih je treba premešati,
- celica za informacijske bite, ki izpadejo iz registrov,
- celica za S ,
- celica za U, ko se izračunajo X-biti,
- celica za dekodirano sekvenco informacijskih bitov.



3. Sklep

Opisani algoritem smo preizkusili na simuliranem prenosnem sistemu. Prenos sporočil je zanesljiv do določene kvalitete kanala. Pri izbiri daljših registrov, bi po pričakovanju prenosni sistem iskazal še boljšo zanesljivost. Koristili smo majhen pomnilniški prostor in razmeroma malo procesorskega časa. V vsakem primeru bomo ob uporabi integriranih vezij s tem dekodirnim algoritmom dvignili zanesljivost tudi nad mediji, kjer je zanesljivost zelo slaba in hkrati povečali prepustnost komunikacijskega kanala ter bo odpadla potreba po povratni liniji.

LITERATURA

1) A.J. Viterbi: Error Bounds for Convolutional Codes and a Asymptotically Optimum Decoding Algorithm; IEEE Trans. on Information Theory Nov.1970 Vol. IIT-16 No 6.

2) J. Justesen: Convolutional Code Construction; IEEE Trans. on Information Theory Vol. I.T- 1973.

3) B.Horvat in sodelavci: Prenos gospodarnih in vernih informacij II. Naloga SBK 785/762502, januar 1978.

4) M 6800 Microprocessor Programming Manual Motorola Inc 1975.