VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

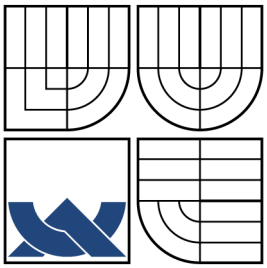
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY DEKÓDOVÁNÍ STROMOVÝCH KÓDŮ

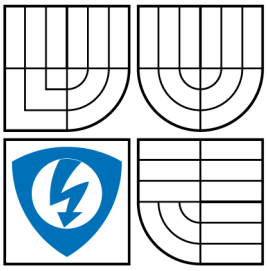
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ZDENĚK ZAMAZAL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY DEKÓDOVÁNÍ STROMOVÝCH KÓDŮ
TREE CODES DECODING METHODS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK ZAMAZAL
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. KAREL NĚMEC, CSC.
SUPERVISOR

BRNO 2008

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Práce se zabývá využitím poznatků z oblasti kanálového kódování dat a zaměřuje se na možnosti dekódování stromových kódů. Popisuje několik způsobů dekódování a zjišťuje jejich výhody či nevýhody. V problematice stromových kódů je nutné znát základy teorie informace, způsoby vytváření kódů a definice kodérů. V práci jsou popsány kritéria výběru dekódovací metody a je uveden nástin jejich hodnocení vzhledem k požadavkům kladeným na protichybový kódový systém. Je vypracován návrh dekodéru pomocí vybrané metody na základě uvedených kritérií. Součástí práce je implementace dekodéru v simulačním prostředí Matlab Simulink. Funkce dekodéru je ověřena odsimulováním modelu protichybového kódového systému.

KLÍČOVÁ SLOVA

dekódování, stromový, kód, konvoluční, Viterbi, Fano, pravděpodobnostní, sekvenční, tabulková, dekodér, mříž, kódový, syndrom

ABSTRACT

This thesis deals with applying knowledge of data channel coding and concerns about possibilities of tree codes decoding. It describes several decoding methods and inquires their ins and outs. Basic knowledge of theory of information is necessary, it is presumed that reader is familiar with convolutional codes and coder definitions. Criterions for choosing decoding method are introduced and lightely described. A decoder is designed using method choosed by denoted criterions. Implementation of the decoder, using Matlab Simulink enviroment, has taken part in the thesis. Function of the decoder is verified in simulation.

KEYWORDS

decoding, tree, code, convolutional, Viterbi, Fano, probability, sequential, list, decoder, trellis, syndrome

ZAMAZAL Z. *Metody dekódování stromových kódů*. Místo: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, 2008. Počet stran 50s., Počet stran příloh 2s. Bakalářská práce. Vedoucí práce byl doc. Ing. Karel Němec, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Metody dekódování stromových kódů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

Děkuji svému vedoucímu bakalářské práce doc. Ing. Němcovi, CSc. za vedení a pomoc při tvorbě této práce a Ing. Křivánkovi za poskytnutí inspirujících materiálů, které mi velmi pomohly dokončit dílo.

OBSAH

Úvod **12**

1 Struktura stromových kódů **13**
1.1 Matematický model stromových kódů – vzorce 13

2 Polynomické vyjádření **14**

3 Pravděpodobnostní dekódování **16**
3.1 Postupné pravděpodobnostní dekódování 16
3.2 Viterbiho dekódování 19

4 Sekvenční dekódování **22**
4.1 Úvod 22
4.2 Fanova vzdálenost 22
4.3 Algoritmus střadače 23
4.4 Fanoův algoritmus 24

5 Tabulkové dekódování **26**
5.1 Úvod 26
5.2 Tabulková metoda 26

6 Většinové dekódování **29**
6.1 Většinové dekódování s tvrdou volbou 29
6.1.1 Jednoznačná metoda Většinového dekódování 35

7 Kritéria výběru dekódovací metody **40**
7.1 Základní kritéria: 40
7.1.1 Celkové hodnocení 44
7.2 Ostatní kritéria 44

8 Návrh dekodéru **46**
8.1 Rozbor zadaného kodéru 46
8.1.1 Volba dekódovací metody 47
8.2 Navržení dekodéru a simulace 47

9 Závěr **50**

Literatura **51**

A První příloha **52**

SEZNAM OBRÁZKŮ

3.1 stromový graf konvolučního kódu 17
3.2 stromový graf (2,1) konvolučního kodéru $G(D) = [1 + D^2\ 1 + D + D^2]$ 18
3.3 a) zapojení kodéru, b) částečně prozkoumaný stromový graf (2,1)3 kódu 20
3.4 Mřížový graf konvolučního kódu.
Plná čára znamená 1 na vstupu. Čárkovaná čára znamená 0 na vstupu.
Čísla u větví grafu znamenají stavy na výstupu. 21
3.5 Dekódovací kroky Viterbiho algoritmu
Plná čára představuje přežívající cestu mřížovým grafem.
Čárkovaná čára představuje cestu, která nevyhověla. 21
4.1 Diagram algoritmu střadače 24
4.2 Diagram Fanova algoritmu 25
5.1 Zapojení kodéru 27
5.2 částečně prozkoumaný stromový graf (2,1)3 kodéru 28
5.3 Seznam cest a jejich akumulované vzdálenosti 28
6.1 jednoznačná metoda většinového dekódování konvolučních kódů s tvrdou volbou 35
6.2 Jednoznačná metoda většinového dekódování pro (2,1)2 systematický stromový kód 38
6.3 Jednoznačné metoda většinového dekódování pro (3,1)3 systematický stromový kód 39
8.1 Zapojení kodéru stromového kódu 46
8.2 Struktura mříže v proměnné Matlabu 46
8.3 Dekódování jednoznačnou metodou 48
8.4 Model PKS s dekodérem 48
8.5 Zapojení dekodéru 49
8.6 Výsledné průběhy procesu dekódování 49
A.1 Graf přechodu stavů v mřížovém grafu 52
A.2 Graf výstupů stromového kódu 52
A.3 Srovnání účinnosti typů kódování 53

SEZNAM TABULEK

3.1 Bitové vzdálenosti pro $p = 0, 1$ 17
4.1 Bitové vzdálenosti 23
7.1 Rozdíl mezi tvrdou a měkkou volbou 40
7.2 Hodnocení při rozumném uvážení parametrů 41
7.3 Hodnocení propustnosti při rozumném uvážení parametrů 42
7.4 Hodnocení náročnosti na paměť při rozumném uvážení parametrů . . 43
7.5 Hodnocení komplexnosti při rozumném uvážení parametrů 44
7.6 Celkové průměrné hodnocení dekódovacích metod 44

ÚVOD

Během uplynulých desítek let výrazně narostla potřeba přenášet digitální data a zároveň došlo k prudkému vývoji elektroniky, která umožňuje vytvořit a implementovat sofistikované algoritmy opravující případné chyby. Výzkum v oblasti datových přenosů prokázal, že vložení umělé nadbytečnosti do přenesené informace dokáže přispět k detekci a opravě chyb. Protichybové kódování by mělo chránit data v digitální podobě před chybami, které vznikají při přenosu přes zarušený kanál.

Výběr protichybového zabezpečovacího kódu záleží na statistickém výskytu chyb na přenosovém kanále. Na jeho základě se vytvoří protichybový kodér. K němu patří odpovídající dekodér. Dekodér je realizován buď vhodnými integrovanými obvody nebo pomocí naprogramovaného mikropočítače. Dekódovací metody stromových zabezpečovacích kódů rozdělujeme podle způsobu, které k jejich řešení používáme. Poznáváme syndromové a postupné dekódování. Syndromové dekódování předpokládá existenci syndromu, jako výsledku kontroly správnosti přenesené posloupnosti bitů. Postupné dekódování využívá odchylek mezi přijatou posloupností bitů a možnými posloupnostmi bitů, kterým odpovídají cesty grafem stromového kódu. Pro určitý stromový kód existuje konečný počet možných cest. Postupné dekódování používá výběr nejpravděpodobnější možné cesty vzhledem k přijaté cestě.

Stromové zabezpečovací kódy se nyní používají převážně v bezdrátové komunikaci, kde je přenosový kanál velmi náchylný ke vzniku chyb vlivem šumu a nechtěných odrazů elektromagnetických vln. V dnešní době stromové kódy naleznou uplatnění ve vesmírných přenosech, satelitních a mobilních komunikacích a běžně používané je například u digitálního televizního vysílání.

Cílem této bakalářské práce je prostudovat některé z možných metod dekódování stromových kódů, dále vytvořit soubor kritérií pro výběr optimální varianty a následně navrhnout odpovídající dekodér vybranou metodou.

1 STRUKTURA STROMOVÝCH KÓDŮ

1.1 Matematický model stromových kódů – vzorce

Informační rychlost (kódový poměr)

$$R = \frac{k}{n}$$

$k \ldots$ vstupní bitový úsek , $n \ldots$ výstupní bitový úsek

vstupní sekvence

$$\mathbf{X} = [\mathbf{X}_1 \mathbf{X}_2 \ldots \mathbf{X}_l \ldots] \tag{1.1}$$

vstupní vektor s k vstupy

$$\mathbf{X}_l = \left[x_l^{(1)} x_l^{(2)} \ldots x_l^{(k)}\right] \tag{1.2}$$

výstupní sekvence

$$\mathbf{Y} = [\mathbf{Y}_1 \mathbf{Y}_2 \ldots \mathbf{Y}_l \ldots] \tag{1.3}$$

výstupní vektor s n výstupy

$$\mathbf{Y}_l = \left[y_l^{(1)} y_l^{(2)} \ldots y_l^{(n)}\right] \tag{1.4}$$

Mezi vstupní a výstupní sekvencí platí vztah

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{G} \tag{1.5}$$

vytvářecí matice

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_0 & \mathbf{G}_1 & \mathbf{G}_2 & \ldots & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{0} & \mathbf{G}_0 & \mathbf{G}_1 & \ldots & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \mathbf{0} & \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_0 & \ldots & \mathbf{G}_{m-2} & \mathbf{G}_{m-1} & \mathbf{G}_m & \mathbf{0} & \\ \vdots & & & & & & & & \ddots \end{bmatrix} \quad 0 \leq l' \leq m \tag{1.6}$$

$\mathbf{0}$ = matice nul $\frac{n}{k}$

dílčí vytvářecí matice

$$\mathbf{G}_{l'} = \begin{bmatrix} g_{1,l'}^1 & g_{1,l'}^2 & \ldots & g_{1,l'}^n \\ g_{2,l'}^1 & g_{2,l'}^2 & \ldots & g_{2,l'}^n \\ \vdots & \vdots & & \vdots \\ g_{k,l'}^1 & g_{k,l'}^2 & \ldots & g_{k,l'}^n \end{bmatrix} \tag{1.7}$$

2 POLYNOMICKÉ VYJÁDŘENÍ

Definujeme:
$D \ldots$ operátor zpoždění

vstupní sekvence

$$\mathbf{X}(D) = \left[x^{(1)}(D)x^{(2)}(D)\ldots x^{(k)}(D)\right] \tag{2.1}$$

vstupní dílčí tok

$$x^{(i)}(D) = x_0^{(i)} + x_0^{(i)}(D) + \ldots + x_l^{(i)}(D)^l + \ldots \tag{2.2}$$

výstupní sekvence

$$\mathbf{Y}(D) = \left[y^{(1)}(D)y^{(2)}(D)\ldots y^{(k)}(D)\right] \tag{2.3}$$

výstupní dílčí tok

$$y^{(j)}(D) = y_0^{(j)} + y_0^{(j)}(D) + \ldots + y_l^{(j)}(D)^l + \ldots \tag{2.4}$$

Mezi vstupní a výstupní sekvencí platí vztah

$$\mathbf{Y}(D) = \mathbf{X}(D) \cdot \mathbf{G}(D) \tag{2.5}$$

vytvářecí matice

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{G}_1^1(D) & \mathbf{G}_1^2(D) & \cdots & \mathbf{G}_1^n(D) \\ \mathbf{G}_2^1(D) & \mathbf{G}_2^2(D) & & \mathbf{G}_2^n(D) \\ \vdots & \vdots & & \vdots \\ \mathbf{G}_k^1(D) & \mathbf{G}_k^2(D) & \cdots & \mathbf{G}_k^n(D) \end{bmatrix} \tag{2.6}$$

dílčí vytvářecí polynom stromového kódu

$$\mathbf{G}_i^{(j)}(D) = g_{i,0}^{(j)} + g_{i,1}^{(j)}(D) + g_{i,2}^{(j)}(D^2) + \ldots + g_{i,m}^{(j)}(D^m) \tag{2.7}$$

délka posuvného registru

$$\mathrm{L}_i = \max_{1 \le j \le n} \left[\mathrm{deg} G_i^{(j)}(D)\right]$$

řád paměti kodéru

$$\mathrm{m} = \max_{1 \le i \le k} L_i$$

celková paměť kodéru

$$\mathrm{K} = \sum_{i=1}^{k} L_i$$

délka kódového ohraničení (omezující délka)

$$\nu = m + 1$$

K označení stromových kódů se používá tento zápis

$$(n; k)\nu$$

kontrolní matice

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{H}_1^1(D) & \mathbf{H}_1^2(D) & \cdots & \mathbf{H}_1^n(D) \\ \mathbf{H}_2^1(D) & \mathbf{H}_2^2(D) & & \mathbf{H}_2^n(D) \\ \vdots & \vdots & & \vdots \\ \mathbf{H}_{n-k}^1(D) & \mathbf{H}_{n-k}^2(D) & \cdots & \mathbf{H}_{n-k}^n(D) \end{bmatrix} \tag{2.8}$$

Mezi vytvářecí a transponovanou kontrolní maticí platí tento vztah

$$\mathbf{G}(D) \cdot \mathbf{H}^T(D) = 0 \tag{2.9}$$

Mezi výstupní sekvencí a transponovanou kontrolní maticí platí tento vztah

$$\mathbf{Y}(D) \cdot \mathbf{H}^T(D) = 0 \tag{2.10}$$

Systematické kódy

vytvářecí matice

$$\mathbf{G}(D) = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{G}_1^{k+1}(D) & \mathbf{G}_1^{k+2}(D) & \cdots & \mathbf{G}_1^n(D) \\ \mathbf{0} & \mathbf{1} & & & \mathbf{G}_2^{k+1}(D) & \mathbf{G}_2^{k+2}(D) & & \mathbf{G}_2^n(D) \\ \vdots & & \ddots & & \vdots & \vdots & & \vdots \\ \mathbf{0} & & & \mathbf{1} & \mathbf{G}_k^{k+1}(D) & \mathbf{G}_k^{k+2}(D) & \cdots & \mathbf{G}_k^n(D) \end{bmatrix} \tag{2.11}$$

kontrolní matice

$$\mathbf{H}(D) = \begin{bmatrix} \mathbf{G}_1^{(k+1)}(D) & \mathbf{G}_2^{(k+1)}(D) & \cdots & \mathbf{G}_k^{(k+1)}(D) & \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{G}_1^{(k+2)}(D) & \mathbf{G}_2^{(k+2)}(D) & & \mathbf{G}_k^{(k+2)}(D) & \mathbf{0} & \mathbf{1} & & \\ \vdots & \vdots & & \vdots & \vdots & & \ddots & \\ \mathbf{G}_1^n(D) & \mathbf{G}_2^n(D) & \cdots & \mathbf{G}_k^n(D) & \mathbf{0} & & & \mathbf{1} \end{bmatrix} \tag{2.12}$$

3 PRAVDĚPODOBNOSTNÍ DEKÓDOVÁNÍ

3.1 Postupné pravděpodobnostní dekódování

Postupné pravděpodobnostní dekódování (viz. [4]) lze nejlépe vysvětlit na stromovém grafu $(n;k)\nu$ konvolučního kódu. Pomůckou zde bude obrázek stromového grafu konvolučního kodéru (viz obr. 3.1). Předpokládejme, že vycházíme z kořenového uzlu 0', z nějž vede $2^{\nu k}$ cest pro ν větví. Nechť $Y_0, Y_1, \ldots Y_{\nu-1}$ značí možnou sekvenci spojenou s ν různými cestami stromovým grafem z daného uzlu. Pro jednoduchost zvolme *délku kódového ohraničení* ν rovno 2. Z přijaté sekvence $R_0, R_1, \ldots R_{\nu-1}$ dekodér vypočítá všech $2^{\nu k}$ pravděpodobnostních funkcí

$$\prod_{l=0}^{\nu-1} P(R_l/Y_l)$$

pro $2^{\nu k}$ cest v soustavách $A_0, A_1, \ldots A_{2^k-1}$. Z nich je vybrána nejpravděpodobnější cesta a dekodér vynese na výstup X_0', (odhad X_0). Dekodér nyní vybere soustavu A_0 a přesune se k uzlu 1'. Zde stejným způsobem prověří soustavy $A_0', A_1', \ldots A_{2^k-1}'$, aby nalezl X_1', odhad X_1. V závislosti na přijatém vektoru R_ν dekodér pokračuje v dekódování podle stromového grafu. Dekodér tedy vykonává hledání největší pravděpodobnosti přes funkci

$$\prod_{l=1}^{\nu} P(R_l/Y_l).$$

Protože $\log P(R_l/Y_l)$ je monotónně rostoucí funkce, vyplývá, že hledání maximální pravděpodobnosti je ekvivalentní hledání maxima logaritmické funkce

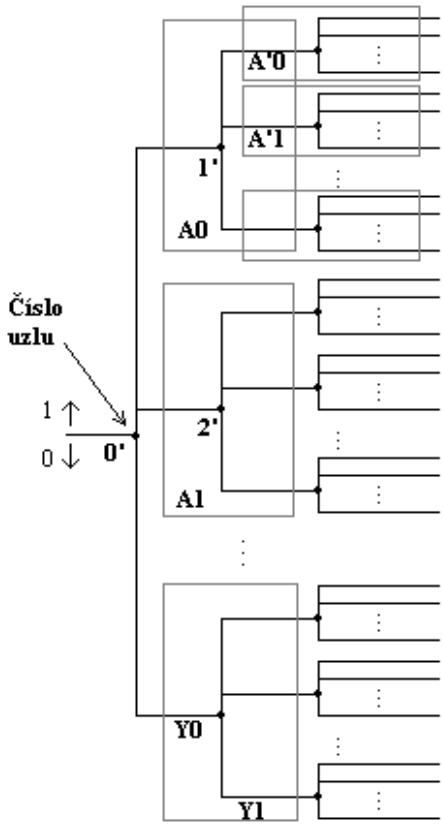
$$\prod_{l=1}^{\nu} \log P(R_l/Y_l).$$

V procesu dekódování je nejpravděpodobnější cesta shodná s tou, která má největší akumulovanou vzdálenost. Akumulovanou vzdáleností se myslí hodnota, která bude definována posléze.

V každém kroku dekodér porovnává vzdálenosti všech $2^{\nu k}$ cest v soustavě a dekóduje cestu s největší vzdáleností. Proces je opakován při každém přijetí segmentu zprávy. Schopnost opravy chyby postupným pravděpodobnostním dekódováním je závislá na rozsahu dekódovacího okna, což je interval, ve kterém dekódování probíhá. Výkon je nejlepší, pokud se velikost okna blíží nekonečnu.

Při přenosu lineárním kanálem jsou symboly sekvence R_l kvantovány do stavů 0 a 1. Uvažujeme totožnou pravděpodobnost jejich výskytu a pravděpodobnost p chybného přechodu z 0 do 1 nebo z 1 do 0. V tomto případě definujeme bitovou vzdálenost v čase l.

$$M(r_l^{(j)}/y_l^{(j)}) = \log_{10} p \tag{3.1}$$



Obr. 3.1: stromový graf konvolučního kódu

pro $r_l^{(j)} \neq y_l^{(j)}$

$$M(r_l^{(j)}/y_l^{(j)}) = \log_{10}(1-p) \tag{3.2}$$

pro $r_l^{(j)} = y_l^{(j)}$.

Např.: Pro pravděpodobnost přechodu $p = 0,1$ dostaneme
$M(r_l^{(j)}/y_l^{(j)}) = -1$ pro $r_l^{(j)} \neq y_l^{(j)}$; $M(r_l^{(j)}/y_l^{(j)}) = -0.05$ pro $r_l^{(j)} = y_l^{(j)}$.
Pro další použití je vhodné vynásobit čísla tak, abychom získali celá čísla: -20; -1
Platí, že akumulovaná vzdálenost cesty je součtem bitových vzdáleností.

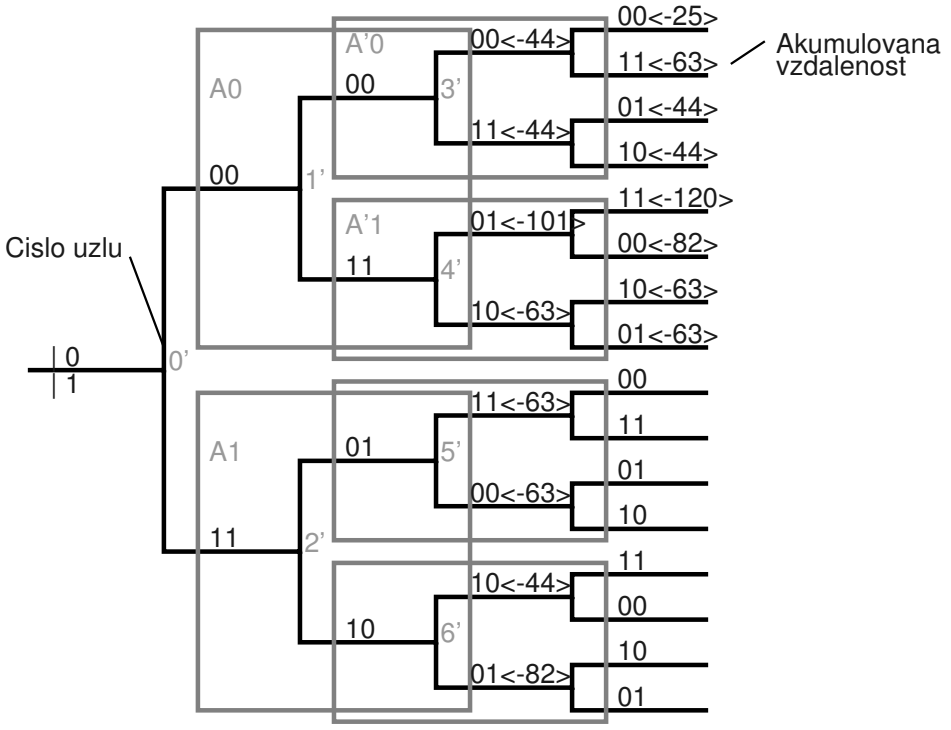
Příklad 3.1

Předpokládejme výstupní nulovou sekvenci $\mathbf{Y} = [00\,00\,00\,\ldots]$, přijatou sekvenci $\mathbf{R} = [R_0,\, R_1,\, R_2,\, R_3] = [10\,00\,10\,00]$ a velikost dekódovacího okna 3.

$r_l^{(j)} \backslash y_l^{(j)}$	0	1
0	-1	-20
1	-20	-1

Tab. 3.1: Bitové vzdálenosti pro $p = 0,1$

Práci dekodéru vysvětlíme na stromovém kódu s vytvářecí maticí danou mnohočleny $\mathbf{G}(D) = [1 + D^2\ 1 + D + D^2]$. Na počátku dekodér vychází z kořenového uzlu 0'. Přijaté vektory R_0, R_1, R_2 jsou porovnány s 8 možnými vektory ze soustav $A_0 = \{00\,00\,00,\ 00\,00\,11,\ 00\,11\,01,\ 00\,11\,10\}$ a $A_1 = \{11\,01\,11,\ 11\,01\,00,\ 11\,10\,10,\ 11\,10\,01\}$. Každé soustavě je přiřazena akumulovaná vzdálenost – je uvedena v <> uvozovkách. Na příklad akumulovaná vzdálenost cesty samých nul v soustavě A_0 je: $(-20)+(-1)+(-1)+(1)+(-20)+(-1) = -44$. Mezi soustavami A_0 a A_1 existuje shoda.



Obr. 3.2: stromový graf (2,1) konvolučního kodéru $G(D) = [1 + D^2\ 1 + D + D^2]$

Protože více cest s větší vzdáleností existuje v soustavě A_0, dekodér zvolí soustavu A_0. Náhodně vybere cestu 00 00 00 a na výstup vynese $X'_0 = 0$. Dekodér nyní přejde k uzlu 1' a hledá vzdálenosti cest v soustavách A'_0, A'_1. Musí předem přijmout vektor R_3. Nyní porovnává R_1, R_2, R_3 s 8 vektory ze soustav $A'_0 = \{00\,00\,00, 00\,00\,11, 00\,11\,01, 00\,11\,10\}$ a $A'_1 = \{11\,01\,11, 11\,01\,00, 11\,10\,10, 11\,10\,01\}$. Dekodér vybere vektor 00 00 00 ze soustavy (A'_0 za cestu s největší vzdáleností a na výstup vynese $X'_1 = 0$. V dalším kroku vycházíme z uzlu 3' a potřebujeme vektor R_4, abychom získali odhad X'_2. Proces dekódování pokračuje, dokud není celá zpráva přijata a dekódována. V příkladu je přijatá sekvence $R_0, R_1, R_2, R_3 =$ 10, 00, 10, 00 dekódována na $Y_0, Y_1, \ldots =$ 00, 00, ... , která odpovídá očekávané sekvenci $X'_0, X'_1 = 0, 0, \ldots$

3.2 Viterbiho dekódování

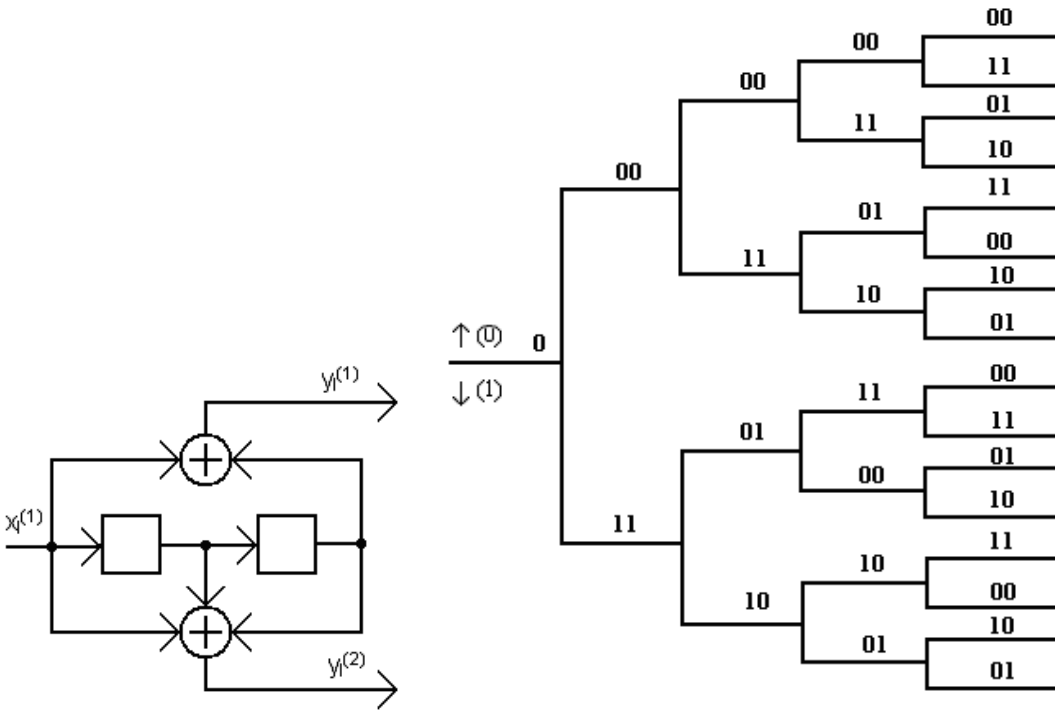
Viterbiho dekódovací algoritmus (viz. [4])představuje možnost dekódovat zprávu po úsecích. Dokáže dosáhnout enormní dekódovací schopnosti, která minimalizuje možnost vzniku chyb v dekódování celistvé sekvence. Spočívá v nalezení nejsprávnější cesty přes váhovaný mřížový graf. Mřížový graf má 2^k stavů, kterých může nabývat vektor $\mathbf{Y}$. Tyto stavy odpovídají uzlům mřížového grafu pro odpovídající hloubku ν . Z každého uzlu vychází 2^k větví do uzlu v hloubce $\nu + 1$. Jestliže vycházíme z kořenového uzlu ve stavu 0, existuje $2^{(\nu-1)k}$ cest pro úsek délky $\nu - 1$. V praktické realizaci se volí *ochranná dekódovací hloubka* æ tak, aby byla 4 nebo 5ν.

Při Viterbiho dekódování nesrovnáváme přijatou posloupnost se všemi možnými, ale pouze úsek po úseku. Úseky na sebe musí plynule navazovat, protože společně spolu tvoří cestu v grafu. V důsledku chyb v přenosu můžeme nalézt v grafu několik možných cest uvnitř intervalu æ. Proto podle korekčních schopností kódu stanovíme práh rozhodování T k vyřazení málo pravděpodobných cest. Ty cesty, které prahu vyhovují, se účastní dalšího dekódování a nazýváme je přežívající cesty. Cesty odhalujeme postupně, může se stát, že na konci dekódovaného úseku o délce æ, bude několik přežívajících cest. Správnou cestu vybereme pomocí nejmenší akumulované vzdálenosti d_A. Akumulovaná vzdálenost d_A je součet Hammingových vzdáleností jednotlivých úseků ve stromovém grafu od přijmutých úseků. Tuto cestu zvolíme ke korekčnímu dekódování, tzn. dekódujeme ji jako celek. V případě, že na konci intervalu æ zůstává několik přežívajících cest se stejnou hodnotou akumulované vzdálenosti, postupujeme podle jedné z následujících možností: náhodný výběr ze skupiny cest – s ohledem na požadovanou chybovost se však nehodí na všechny situace nebo prodloužení ochranné dekódovací hloubky æ a následný výběr cesty jdoucí větvemi s nejmenší Hammingovou vzdáleností.

Příklad 3.2:

Kodér (2,1)3 s vytvářecí maticí $G(D) = [1 + D^2 \ 1 + D + D^2]$. Byla vyslána nulová posloupnost $\mathbf{Y} = [00\,00\,00\,00\,\ldots]$. Přijata byla sekvence $\mathbf{R} = [10\,00\,10\,00\,\ldots]$ s 2 náhodnými chybami. Dekódovací hloubku volíme 2ν, přičemž $\nu = 3$.

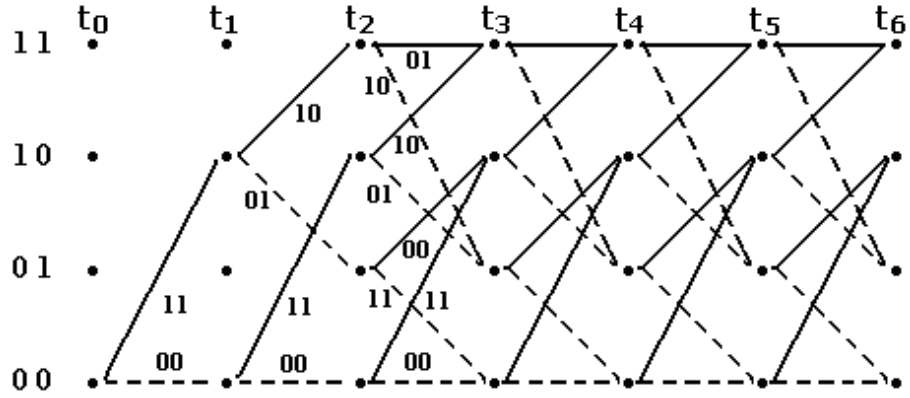
Musíme zvolit práh dekódování T, abychom mohli vyřadit cesty, které nepřežívají. Zvolme tedy práh dekódování T=3, vyřadíme tak cesty, které mají akumulovanou vzdálenost d_A větší než 3. Při dekódování postupujeme následovně: Vycházíme ze struktury mřížového grafu. Z každého uzlu v čase $t > 1$ vychází 2^k cest (zde $k = 1$). Rozvážíme podobnost mezi vektorem $\mathbf{R}$ a vektory $\mathbf{Y}_1, \mathbf{Y}_2, \ldots \mathbf{Y}_k$ a zapíšeme Hammingovu vzdálenost. Vzdálenosti spolu sčítáme tak, jak procházíme uzly podle času t, tím dostáváme hodnotu akumulované vzdálenosti d_A. Pokud je větší, než práh rozhodování, cestu neuvažujeme. Může se stát, že do uzlu míří více cest, zvolíme



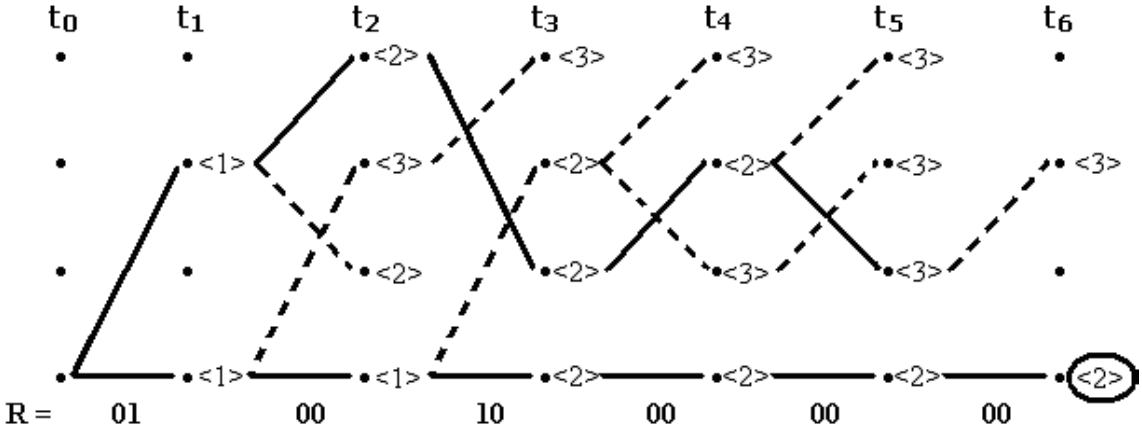
a) b)

Obr. 3.3: a) zapojení kodéru, b) částečně prozkoumaný stromový graf (2,1)3 kódu

tu, která má menší vzdálenost d_A. Jestliže mají cesty vzdálenost stejnou, vybereme jednu náhodně. V dekódování pokračujeme přes celý interval æ. Nalevo v mřížovém grafu zůstává cesta dekódované posloupnosti. V příkladu je kódovaná sekvence opravena na $\mathbf{Y} = [00\,00\,00\,00\,\ldots]$, což odpovídá vstupní posloupnosti $\mathbf{X} = [0\,0\,0\,0\,\ldots]$ kodéru. Spatříme, že dekodér dokáže v tomto příkladě opravit 1 chybu v kódovém ohraničení.



Obr. 3.4: Mřížový graf konvolučního kódu.
Plná čára znamená 1 na vstupu. Čárkovaná čára znamená 0 na vstupu.
Čísla u větví grafu znamenají stavy na výstupu.



Obr. 3.5: Dekódovací kroky Viterbiho algoritmu
Plná čára představuje přežívající cestu mřížovým grafem.
Čárkovaná čára představuje cestu, která nevyhověla.

4 SEKVENČNÍ DEKÓDOVÁNÍ

4.1 Úvod

V předchozí kapitole jsme probrali Viterbiho dekódovací metodu. Složitost této metody roste exponenciálně s délkou kódového ohraničení daného kódu. Jestliže v určité aplikaci požadujeme opravdu velmi nízkou chybovost, musíme využít kodér s velkou délkou kódového ohraničení, řekněme $\nu > 20$. V tomto případě se Viterbiho dekodér stane beznadějně složitým. Abychom dosáhli libovolně malé chybovosti, je žádoucí nalézt takový dekódovací proces, který je nezávislý na celkové paměti kodéru, a proto můžeme použít kód využívající větší délku kódového ohraničení.

Algoritmy sekvenčního dekódování zcela nezávisí na paměti kodéru. Jsou založeny na principu, že v každém kroku zkouší pouze jeden stav kodéru. S využitím zpětného trasování, sekvenční dekódování dosáhne správného odhadu přenesené sekvence s nejvyšší pravděpodobností. Tato metoda (tzv. asymptoticky) plně vyčerpává korekční schopnosti kódu.

Uvažujeme-li stromový kód, který využívá délku kódového ohraničení tak velkou, že použití Viterbiho dekodéru je nepraktické, potom kódovací proces vnímáme jako cestu stromovým grafem namísto mřížového. Základní předpoklad sekvenčního dekódování je založen na myšlence, že bychom měli hledat pouze nejslibnější cesty. Jestliže cesta k danému uzlu není pravděpodobná, můžeme všechny cesty z tohoto uzlu jednoduše vyloučit.

4.2 Fanova vzdálenost

K následnému způsobu dekódování je nutné zavést nový nástroj k určení bitové vzdálenosti. Protože při sekvenčním dekódování se porovnávají cesty různých délek, nelze využít stejnou techniku jako při Viterbiho dekódování. Zavádíme tedy Fanovu vzdálenost: (viz. [4]) Předpokládejme, že dekodér prohledal $l'+1$ větví, odpovídající akumulovaná vzdálenost je:

$$\begin{aligned} M_F = M_{l'}(R_0/Y_0, R_1/Y_1, \ldots R_{l'}/Y_{l'}) &= \sum_{l=0}^{l'} \sum_{j=1}^{n} \log_2 P(r_l^{(j)}/y_l^{(j)}) \\ &+ \sum_{l=0}^{l'} \sum_{j=1}^{n} \left[\log_2 \frac{1}{P(r_l^{(j)})} - R \right] \end{aligned} \tag{4.1}$$

kde $P(r_l^{(j)}/y_l^{(j)})$ je pravděpodobnost chyby v kanále, $P(r_l^{(j)})$ je pravděpodobnost výskytu symbolu a R je informační rychlost kódu.

První část vzorce je shodná s bitovou vzdáleností používanou Viterbiho algoritmem, druhá část je vždy kladná a roste lineárně s délkou cesty. Cesta s největší Fanovou vzdáleností je považována za správnou cestu.

Příklad 4.1:

Pro symetrický binární kanál s pravděpodobností chyby $p = 0,1$ a informační rychlostí $R = 1/2$ je výpočet bitové vzdálenosti následující:

$$M(r_l^{(j)}/y_l^{(j)}) = \log_2 2p - R \tag{4.2}$$

pro $r_l^{(j)} \neq y_l^{(j)}$

$$M(r_l^{(j)}/y_l^{(j)}) = \log_2 2(1-p) - R \tag{4.3}$$

pro $r_l^{(j)} = y_l^{(j)}$.
Zde: $M(r_l^{(j)}/y_l^{(j)}) = -1$ pro $r_l^{(j)} \neq y_l^{(j)}$; $M(r_l^{(j)}/y_l^{(j)}) = -0.05$ pro $r_l^{(j)} = y_l^{(j)}$. Pro praxi se používá převod do celých čísel, např.: $-8; 1$.

$r_l^{(j)} \backslash y_l^{(j)}$	0	1
0	1	-8
1	-8	1

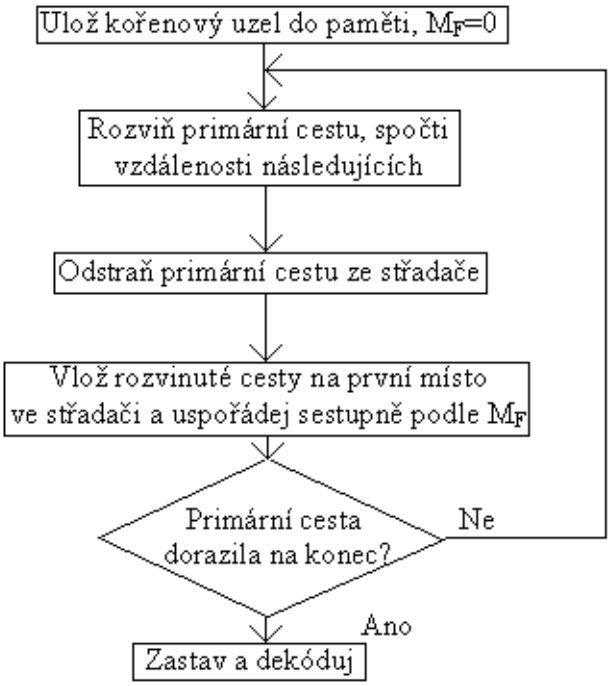
Tab. 4.1: Bitové vzdálenosti

4.3 Algoritmus střadače

Tento algoritmus spočívá v uchovávání všech možných cest v paměti spolu s jejich Fanovou vzdáleností. Cesta s nejvyšší hodnotou je rozšířena dál. Cesty stromovým grafem jsou prozkoumávány po částech podle pořadí ve střadači. nejlepší cesta je zařazena na první místo, pak následuje druhá nejlepší a tak dál.

Algoritmus střadače

1. Do střadače uložíme kořenový uzel, jeho vzdálenost je 0.
2. Rozšíříme cestu na prvním místě a spočítáme vzdálenosti 2^k nových cest.
3. Nahradíme primární cestu 2^k novými cestami a seřadíme je sestupně podle hodnoty Fanovy vzdálenosti.
4. Jestliže dorazíme na konec stromového grafu, zastavíme a cestu dekódujeme. Jinak pokračujeme bodem 2.

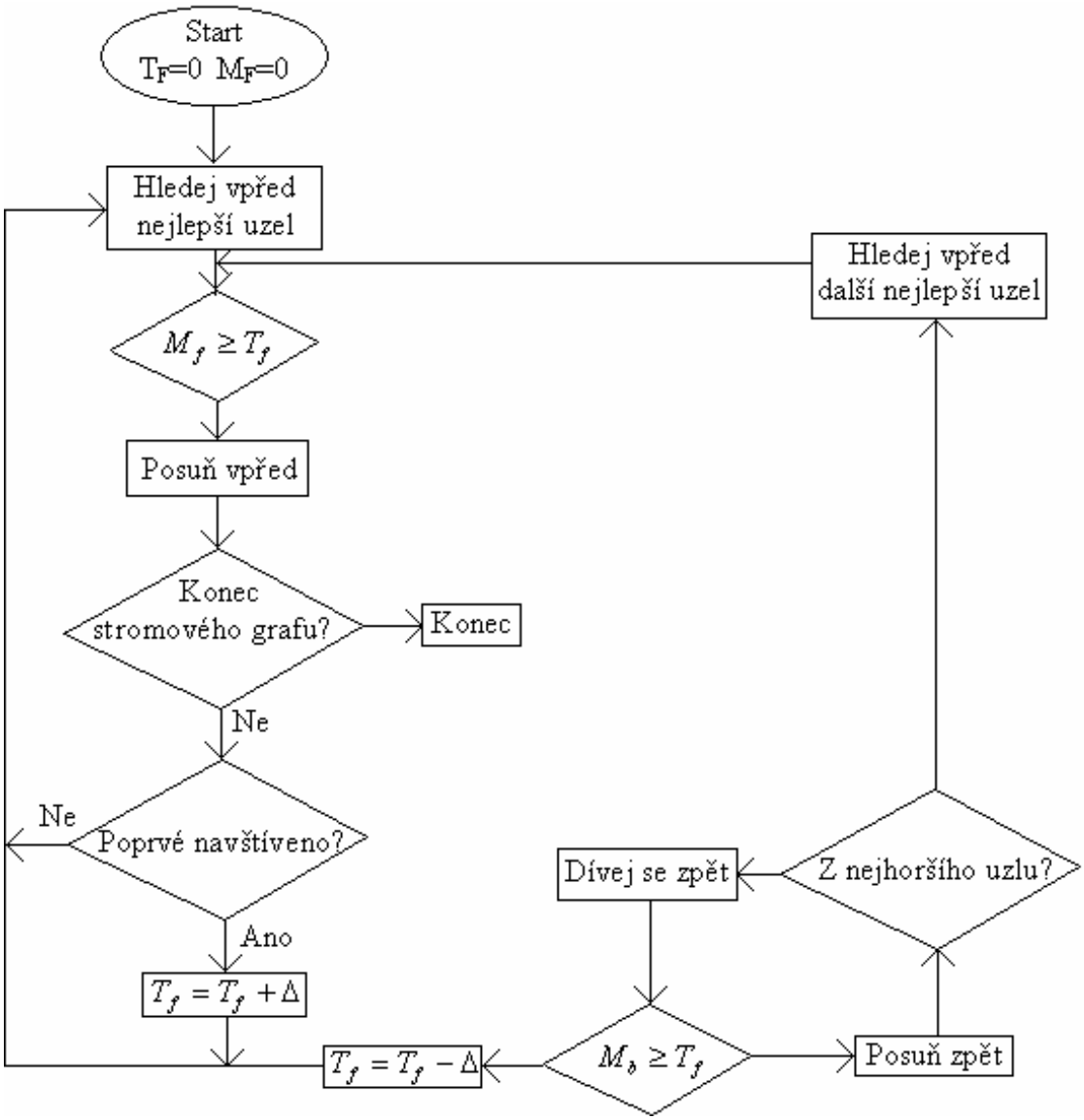


Obr. 4.1: Diagram algoritmu střadače

4.4 Fanoův algoritmus

Dekodér tohoto algoritmu pracuje v každém čase pouze na jediné cestě stromového grafu. Posouvá se o uzel vpřed nebo vzad po cestě a porovnává hodnotu akumulované vzdálenosti s prahem dekódování. Protože dekodér hledá v omezeném počtu cest, dekódovaná cesta se může lišit od nejpravděpodobnější cesty zvolené Viterbiho dekodérem.

Dekodér začíná v kořenovém uzlu stromového grafu a končí v jednom z uzlů na konci stromu. V každém dekódovacím kroku se dekodér nachází v nějakém uzlu N. Z tohoto uzlu se dekodér dívá do následujících 2^k uzlů a počítá *dopřednou vzdálenost* M_f pro každý uzel. Tyto uzly jsou seřazeny sestupně podle jejich vzdálenosti. Nejlepší nebo nejslibnější uzel má nejvyšší hodnotu, nejhorší má hodnotu nejnižší. Dekodér se posune k nejlepšímu uzlu a porovnává, zda je vzdálenost cesty M_f větší nebo rovna *prahovací úrovni* T_F. Pokud dekodér dorazí na konec, je proces ukončen a cesta je dekódována. Pokud dekodér ještě nedorazil na konec a následující uzel nebyl zkoušen, prahovací úroveň T_F je navýšena o kladnou konstantu Δ a dekodér se dívá znovu dopředu. Jestliže se dekodér nemůže pohnout dopředu z uzlu N, dívá se zpět do uzlu M, z kterého vychází cesta do uzlu N. Zpětná vzdálenost cesty M_b se vypočítá jako rozdíl akumulované vzdálenosti v uzlu N a akumulované vzdálenosti cesty do uzlu N. Jestliže je zpětná vzdálenost menší než prahovací úroveň T_F, dekodér se nemůže vrátit zpět. Prahovací úroveň se sníží o Δ a dekodér se znovu



Obr. 4.2: Diagram Fanova algoritmu

dívá dopředu. Jestliže je zpětná vzdálenost větší nebo rovna T_F, dekodér rozezná nepravděpodobnou cestu a přesune se zpět k uzlu M. Pokud uzel N byl nejhorším uzlem, když se dekodér posunul z uzlu M do N v některém z předchozích kroků, dekodér pokračuje v dívání se zpět. Jinak dekodér hledá novou dopřednou cestu k nejlepšímu uzlu.

5 TABULKOVÉ DEKÓDOVÁNÍ

5.1 Úvod

Tabulkové dekódování je účinná dekódovací metoda, která plně nevyčerpává korekční schopnosti kódu. Dekodér využívající tabulkové dekódování je účinnější než podobně složitý Viterbiho dekodér. Tabulkové dekódování je zajímavá metoda založená na myšlence vytvářet ve stromovém grafu seznam nejvíce slibných cest o velikosti L. Tabulkové dekódování systematických kódů je, při stejné složitosti dekodéru, v podstatě nadřazeno Viterbiho dekódování stromových kódů vytvořených nesystematickými maticemi.

5.2 Tabulková metoda

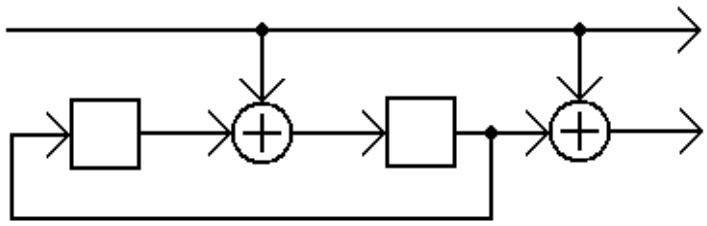
Tabulková metoda (viz. [1])je metoda, která používá dopředné hledání cesty stromovým grafem. V určité dekódovací hloubce je rozvinuto pouze L nejvíce slibných cest, ne všechny, jak je tomu u Viterbiho metody. Tyto cesty tvoří seznam o velikosti L. Začínáme u kořenového uzlu, procházíme všechny cesty, dokud neobdržíme L nebo více cest. Všechny rozvinuté cesty mají stejnou délku, z nich vybereme L nejslibnějších cest na základě hodnoty akumulované vzdálenosti d_A dané cesty. Akumulovaná vzdálenost je součtem Hammingových vzdáleností všech uzlů, kterým daná cesta prohází. Cesty s největší d_A tvoří seznam velikosti L, pouze tyto cesty dále uvažujeme a vycházíme z nich. Nebezpečím je ztráta správné cesty, tato chyba je závažným nedostatkem typickým pro tabulkovou metodu.

Algoritmus tabulkové metody:

1. Načti seznam cest vycházejících z kořenového uzlu.
2. Rozviň všechny uložené cesty do délky $\nu + 1$, vlož L nejlepších (podle akumulované vzdálenosti d_A) na seznam.
3. Jestliže jsi již dosáhl konce, zastav a zvol nejlepší cestu k dekódování. Jinak zvětši ν o 1 a vrať se k bodu 2.

Příklad 5.1

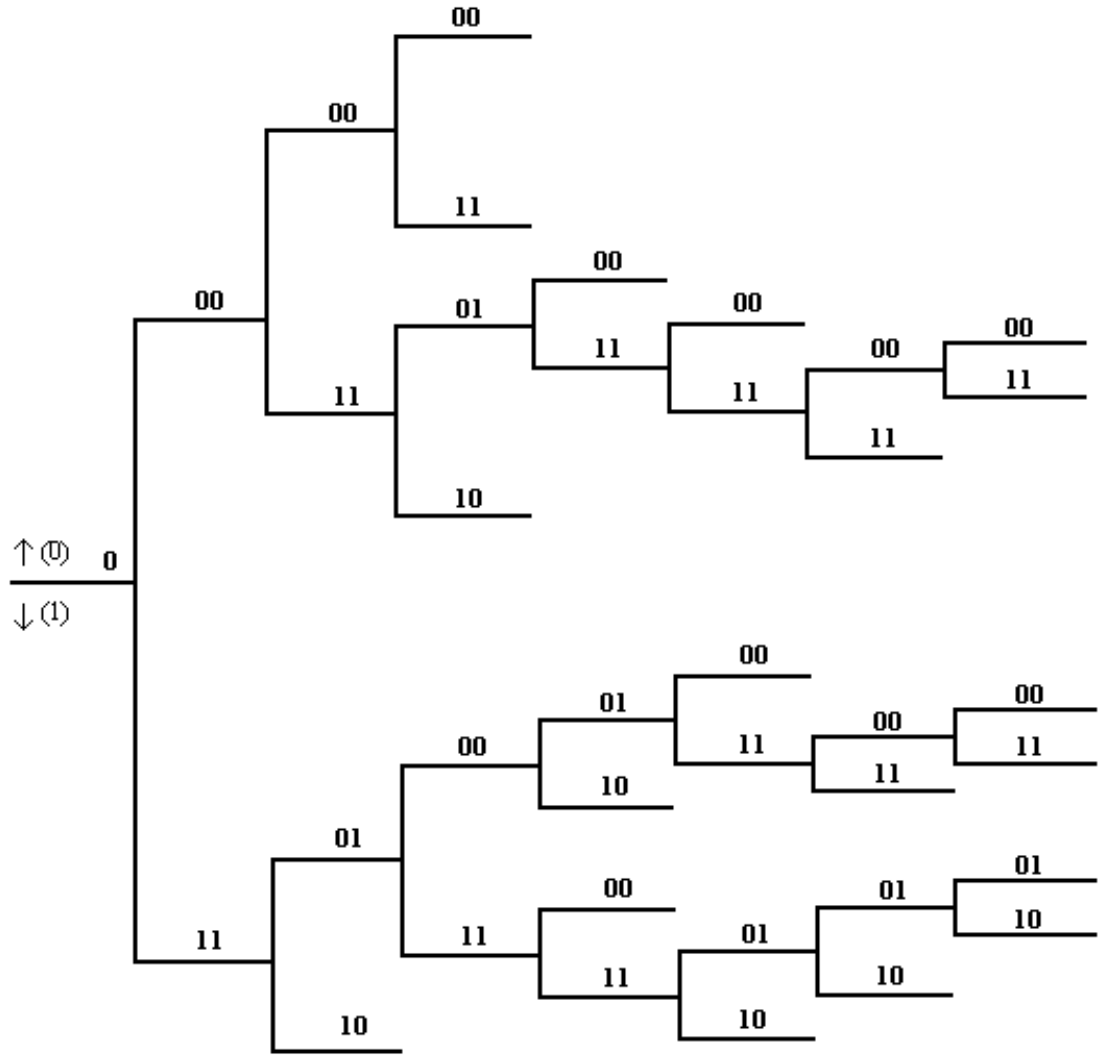
Kodér (2,1)3 stromového kódu je zadaný pomocí obrázku. Byla vyslána sekvence přes zarušený kanál $\mathbf{Y} = [11\,01\,11\,11\,01\,00\,01]$, sekvence $\mathbf{R} = [10\,01\,01\,11\,11\,00\,01]$ byla přijata se 3 náhodnými chybami.



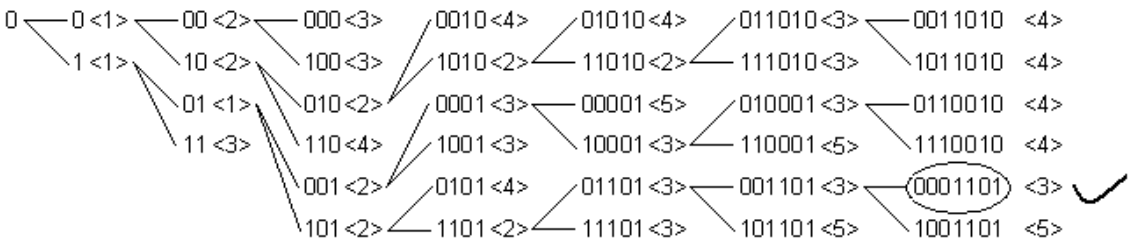
Obr. 5.1: Zapojení kodéru

Zvolili jsme tabulkovou metodu se seznamem o velikosti $L = 3$. Vycházíme z kořenového uzlu, z něj vychází 2 větve do hloubky $\nu = 1$, jejich Hammingova vzdálenost je v obou případech 1. Uložíme je do seznamu a rozvineme do hloubky $\nu = 2$, dostaneme nyní 4 větve. V uzlech na koncích cest spočteme jejich akumulovanou vzdálenost d_A jako součet Hammingových vzdáleností všech větví, kterými cesta prochází. Hammingovu vzdálenost odvodíme porovnáním stavů v odpovídajících si uzlech. Například, pro cestu 1 0 platí $(1) + (1) = (2)$. Vybereme L cest, které mají nejmenší akumulovanou vzdálenost a uložíme je do seznamu. Pokud má více cest stejnou d_A, zbývá nám vybrat cestu náhodně. Nyní rozvineme cesty v seznamu do hloubky $\nu + 1$ a opět v uzlech spočteme akumulovanou vzdálenost d_A. Z cest znovu vybereme L nejlepších a v tomto smyslu pokračujeme dál. Až dorazíme na konec, zastavíme a dekódujeme. Zvolíme k dekódování tu „nejlepší“ cestu, která má nejnižší hodnotu akumulované vzdálenosti d_A. V tomto případě jsme zvolili cestu 1011000. Výpočet vzdálenosti byl $(1) + (0) + (1) + (0) + (1) + (0) + (0) = (3)$.

Určili jsme cestu grafem jako posloupnost [0 0 0 1 1 0 1], ale máme zde otočeno MSB a LSB, protože bity jsme při připisovali zprava doleva. Dekódovaná sekvence je tedy $\mathbf{X}' = [1\,0\,1\,1\,0\,0\,0]$. Dekódovali jsme správně a dekodér dokázal opravit 3 bitové chyby v přijaté sekvenci.



Obr. 5.2: částečně prozkoumaný stromový graf (2,1)3 kodéru



Obr. 5.3: Seznam cest a jejich akumulované vzdálenosti

6 VĚTŠINOVÉ DEKÓDOVÁNÍ

Většinové dekódování (viz. [4], Majority-logic decoding) je specifickou alternativou pro dekódování konvolučních kódů. Je relativně snažší implementovat tento způsob, než dekódování Viterbiho nebo sekvenčními metodami. Většinové dekódování je založeno na ortogonálních paritních součtech a dekodér vykonává činnost pouze uvnitř jedné délky kódového ohraničení z přijaté posloupnosti. Z toho vyplývá, že se jedná o podoptimální systém, ale implementace je mnohem jednodušší a rychlost dekódování větší než například u Viterbiho nebo sekvenčních dekodérů. Většinové dekódování je tudíž použitelné v přístrojích nižší kvality, kde můžeme očekávat výkon úměrný kvalitě.

Zde se budeme zabývat pouze systematickými stromovými kódy, jelikož jakýkoli nesystematický stromový kód lze vhodnými řádkovými transformacemi vytvářecí matice převést na systematický. Hammingova vzdálenost uvnitř jedné délky kódového ohraničení přitom zůstane nezměněna. Použitím nesystematických kódů spolu s většinovým dekódováním bychom nedosáhli nijak lepších výsledků při oprave chyb.

6.1 Většinové dekódování s tvrdou volbou

Systematický konvoluční kód $(n; k)\nu$ je daný polonekonečnou vytvářecí maticí

$$\mathbf{G} = \begin{bmatrix} \mathbf{IP}_0 & \mathbf{0P}_1 & \mathbf{0P}_2 & \dots & \mathbf{0P}_m & & & \dots \\ & \mathbf{IP}_0 & \mathbf{0P}_1 & \dots & \mathbf{0P}_{m-1} & \mathbf{0P}_m & & \\ & & \mathbf{IP}_0 & \dots & \mathbf{0P}_{m-2} & \mathbf{0P}_{m-1} & \mathbf{0P}_m & \\ \vdots & & & & & & & \ddots \end{bmatrix} \tag{6.1}$$

kde $\mathbf{I}$ je matice identity o velikosti k na k, $\mathbf{0}$ je nulová matice o velikosti k na k, a každá $\mathbf{P}_{l'}$, $0 \leq l' \leq m$, je submatice o velikosti k na *(n-k)*:

$$\mathbf{P}_{l'} = \begin{bmatrix} g_{1,l'}^{k+1} & g_{1,l'}^{k+2} & \dots & g_{1,l'}^{n} \\ g_{2,l'}^{k+1} & g_{2,l'}^{k+2} & \dots & g_{2,l'}^{n} \\ \vdots & \vdots & & \vdots \\ g_{k,l'}^{k+1} & g_{k,l'}^{k+2} & \dots & g_{k,l'}^{n} \end{bmatrix} \tag{6.2}$$

s binárními členy. Polonekonečná matice parity systematického kódu může být vyjádřena stejnými členy jako v $\mathbf{G}$, to je,

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^T\mathbf{I} & & & & & & \dots \\ \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & & & \\ \vdots & & \ddots & & & & \\ \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & & & \\ & \mathbf{P}_m^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & \\ & & \ddots & & & \ddots & \\ & & & \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} \\ \vdots & & & & & & \ddots \end{bmatrix} \tag{6.3}$$

kde, v tomto případě, $\mathbf{I}$ je matice identity o velikosti *(n-k)* na *(n-k)*, $\mathbf{0}$ je nulová matice o velikosti *(n-k)* na *(n-k)* a $\mathbf{P}_{l'}^T$ je transponovaná submatice $\mathbf{P}_{l'}$.

Uvažujme, že $\mathbf{Y}$ je kódovaná sekvence, která je přenášena přes zarušený kanál, a $\mathbf{R}$ bude šumem poškozená sekvence, která byla přijata. $\mathbf{R}$ vznikne součtem kódové sekvence $\mathbf{Y}$ a chybové sekvence kanálu $\mathbf{E}$, tedy

$$\mathbf{R} = \mathbf{Y} + \mathbf{E} \tag{6.4}$$

kde platí, že součet vektorů nebo matic je dělený modulo-2, pokud to z kontextu neplyne jinak. Přijatá a chybová sekvence se poto vyznačí dvěma řádkovými vektory

$$\mathbf{R} = [\mathbf{R}_0\mathbf{R}_1 \dots \mathbf{R}_l \dots] \tag{6.5}$$

a příznačně

$$\mathbf{E} = [\mathbf{E}_0\mathbf{E}_1 \dots \mathbf{E}_l \dots] \tag{6.6}$$

$\mathbf{R}_l$ je zde přijatý vektor v čase l a je zde reprezentován řádkovým vektorem s n komponenty

$$\mathbf{R}_l = \left[r_l^{(1)} r_l^{(2)} \dots r_l^{(n)} \dots\right] \tag{6.7}$$

a chybový vektor $\mathbf{E}_l$ v čase l

$$\mathbf{E}_l = \left[e_l^{(1)} e_l^{(2)} \dots e_l^{(n)} \dots\right] \tag{6.8}$$

Při přijetí poškozené sekvence $\mathbf{R}$, přijímač určí syndromovou sekvenci $\mathbf{S}$. Sekvence syndromu $\mathbf{S}$ je definována jako

$$\mathbf{S} = \mathbf{R}\mathbf{H}^T \tag{6.9}$$

$$= \mathbf{Y}\mathbf{H}^T + \mathbf{E}\mathbf{H}^T \tag{6.10}$$

kde

$$\mathbf{S} = [\mathbf{S}_0 \mathbf{S}_1 \dots \mathbf{S}_l \dots] \tag{6.11}$$

a syndrom $\mathbf{S}_l$ v čase l je reprezentován řádkovým vektorem s $(n-k)$ komponenty

$$\mathbf{S}_l = \left[s_l^{(k+1)} s_l^{(k+2)} \dots s_l^{(n)} \dots \right] \tag{6.12}$$

Protože platí, že $\mathbf{Y}\mathbf{H}^T = 0$, můžeme napsat

$$\mathbf{S} = \mathbf{E}\mathbf{H}^T \tag{6.13}$$

Je patrné, že syndrom $\mathbf{S}$ závisí pouze na chybové sekvenci a ne na přenášené informaci. Znalost syndromu $\mathbf{S}$ se vyrovná znalosti sekvence $\mathbf{E}$ a dekodér lze navrhnout tak, aby pracoval s $\mathbf{S}$ namísto $\mathbf{R}$. Pokud provedeme transpozici obou stran, struktura vypadá

$$\mathbf{S}^T = \mathbf{H}\mathbf{R}^T \tag{6.14}$$

a

$$\begin{bmatrix} \mathbf{S}_0^T \\ \mathbf{S}_1^T \\ \vdots \\ \mathbf{S}_l^T \\ \mathbf{S}_{l+1}^T \\ \vdots \\ \mathbf{S}_{l+m}^T \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{P}_0^T\mathbf{I} & & & & & & & \dots \\ \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & & & & \\ \vdots & & \ddots & & & & & \\ \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & & & & \\ & \mathbf{P}_m^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & \\ & & \ddots & & & \ddots & & \\ & & & \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & \\ \vdots & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{R}_0^T \\ \mathbf{R}_1^T \\ \vdots \\ \mathbf{R}_l^T \\ \mathbf{R}_{l+1}^T \\ \vdots \\ \mathbf{R}_{l+m}^T \\ \vdots \end{bmatrix} \tag{6.15}$$

Také platí

$$\mathbf{S}^T = \mathbf{H}\mathbf{E}^T \tag{6.16}$$

a

$$\begin{bmatrix} \mathbf{S}_0^T \\ \mathbf{S}_1^T \\ \vdots \\ \mathbf{S}_l^T \\ \mathbf{S}_{l+1}^T \\ \vdots \\ \mathbf{S}_{l+m}^T \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{P}_0^T\mathbf{I} & & & & & & & \dots \\ \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & & & & \\ \vdots & & \ddots & & & & & \\ \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & & & & \\ & \mathbf{P}_m^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & \\ & & \ddots & & & \ddots & & \\ & & & \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & \\ \vdots & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{E}_0^T \\ \mathbf{E}_1^T \\ \vdots \\ \mathbf{E}_l^T \\ \mathbf{E}_{l+1}^T \\ \vdots \\ \mathbf{E}_{l+m}^T \\ \vdots \end{bmatrix} \tag{6.17}$$

Jestliže je dekodér navrhnut tak, aby dekódoval přes jednu délku kódového ohraničení $(n;k)\nu$ stromového kódu, stanovíme vyjádření pro ukončený průběh, který napíšeme jako

$$\mathbf{S}'^{T} = \mathbf{H}'\mathbf{R}'^{T} \tag{6.18}$$

a

$$\begin{bmatrix} \mathbf{S}_{l^T} \\ \mathbf{S}_{l+1}^T \\ \vdots \\ \mathbf{S}_{l+m}^T \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & & & & \\ & \mathbf{P}_m^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & \\ & & \ddots & & & \ddots & & \\ & & & \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{l-m}^T \\ \vdots \\ \mathbf{R}_{l-1}^T \\ \mathbf{R}_l^T \\ \mathbf{R}_{l+1}^T \\ \vdots \\ \mathbf{R}_{l+m}^T \end{bmatrix} \tag{6.19}$$

Podobně můžeme ukončenou verzi napsat jako

$$\mathbf{S}'^{T} = \mathbf{H}'\mathbf{E}'^{T} \tag{6.20}$$

a

$$\begin{bmatrix} \mathbf{S}_l^T \\ \mathbf{S}_{l+1}^T \\ \vdots \\ \mathbf{S}_{l+m}^T \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_0^T\mathbf{I} & & & & \\ & \mathbf{P}_m^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} & & & \\ & & \ddots & & & \ddots & & \\ & & & \mathbf{P}_m^T\mathbf{0} & \mathbf{P}_{m-1}^T\mathbf{0} & \dots & \mathbf{P}_1^T\mathbf{0} & \mathbf{P}_0^T\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{E}_{l-m}^T \\ \vdots \\ \mathbf{E}_{l-1}^T \\ \mathbf{E}_l^T \\ \mathbf{E}_{l+1}^T \\ \vdots \\ \mathbf{E}_{l+m}^T \end{bmatrix} \tag{6.21}$$

Nahrazením předešlých vzorců a přeskupením členů v maticích dostáváme

$$\begin{bmatrix} s_l^{(k+1)} \\ \vdots \\ s_{l+m}^{(k+1)} \\ s_l^{(k+2)} \\ \vdots \\ s_{l+m}^{(k+2)} \\ \vdots \\ s_l^{(n)} \\ \vdots \\ s_{l+m}^{(n)} \end{bmatrix} = \begin{bmatrix} g_{1,m}^{(k+1)} & \cdots & g_{1,0}^{(k+1)} & \cdots & 0 & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(k+1)} & \cdots & g_{1,0}^{(k+1)} & 0 & \cdots & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} \\ g_{1,m}^{(k+2)} & \cdots & g_{1,0}^{(k+2)} & \cdots & 0 & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(k+2)} & \cdots & g_{1,0}^{(k+2)} & 0 & \cdots & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} \\ \vdots & & & & & \vdots & & & & \\ g_{1,m}^{(n)} & \cdots & g_{1,0}^{(n)} & \cdots & 0 & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(n)} & \cdots & g_{1,0}^{(n)} & 0 & \cdots & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} \end{bmatrix}$$

$$\begin{bmatrix} \cdots & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(k+1)} & \cdots & g_{k,0}^{(k+1)} \\ \cdots & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(k+2)} & \cdots & g_{k,0}^{(k+2)} \\ & \vdots & & & & \\ \cdots & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(n)} & \cdots & g_{k,0}^{(n)} \end{bmatrix} \begin{bmatrix} r_{l-m}^{(1)} \\ \vdots \\ r_l^{(1)} \\ \vdots \\ r_{l+m}^{(1)} \\ r_{l-m}^{(2)} \\ \vdots \\ r_l^{(2)} \\ \vdots \\ r_{l+m}^{(2)} \\ \vdots \\ r_{l-m}^{(k)} \\ \vdots \\ r_l^{(k)} \\ \vdots \\ r_{l+m}^{(k)} \end{bmatrix} + \begin{bmatrix} r_l^{(k+1)} \\ \vdots \\ r_{l+m}^{(k+1)} \\ r_l^{(k+2)} \\ \vdots \\ r_{l+m}^{(k+2)} \\ \vdots \\ r_l^{(n)} \\ \vdots \\ r_{l+m}^{(n)} \end{bmatrix} \tag{6.22}$$

a

$$
\begin{bmatrix} s_l^{(k+1)} \\ \vdots \\ s_{l+m}^{(k+1)} \\ s_l^{(k+2)} \\ \vdots \\ s_{l+m}^{(k+2)} \\ \vdots \\ s_l^{(n)} \\ \vdots \\ s_{l+m}^{(n)} \end{bmatrix} = \begin{bmatrix} g_{1,m}^{(k+1)} & \cdots & g_{1,0}^{(k+1)} & \cdots & 0 & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(k+1)} & \cdots & g_{1,0}^{(k+1)} & 0 & \cdots & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} \\ g_{1,m}^{(k+2)} & \cdots & g_{1,0}^{(k+2)} & \cdots & 0 & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(k+2)} & \cdots & g_{1,0}^{(k+2)} & 0 & \cdots & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} \\ \vdots & & & & & \vdots & & & & \\ g_{1,m}^{(n)} & \cdots & g_{1,0}^{(n)} & \cdots & 0 & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} & \cdots & 0 \\ & \ddots & & \ddots & & & \ddots & & \ddots & \\ 0 & \cdots & g_{1,m}^{(n)} & \cdots & g_{1,0}^{(n)} & 0 & \cdots & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} \end{bmatrix}
$$

$$
\begin{bmatrix} \cdots & g_{2,m}^{(k+1)} & \cdots & g_{2,0}^{(k+1)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(k+1)} & \cdots & g_{k,0}^{(k+1)} \\ \cdots & g_{2,m}^{(k+2)} & \cdots & g_{2,0}^{(k+2)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(k+2)} & \cdots & g_{k,0}^{(k+2)} \\ & \vdots & & & & \\ \cdots & g_{2,m}^{(n)} & \cdots & g_{2,0}^{(n)} & \cdots & 0 \\ \cdots & & \ddots & & \ddots & \\ \cdots & 0 & \cdots & g_{k,m}^{(n)} & \cdots & g_{k,0}^{(n)} \end{bmatrix} \begin{bmatrix} e_{l-m}^{(1)} \\ \vdots \\ e_l^{(1)} \\ \vdots \\ e_{l+m}^{(1)} \\ e_{l-m}^{(2)} \\ \vdots \\ e_l^{(2)} \\ \vdots \\ e_{l+m}^{(2)} \\ \vdots \\ e_{l-m}^{(k)} \\ \vdots \\ e_l^{(k)} \\ \vdots \\ e_{l+m}^{(k)} \end{bmatrix} + \begin{bmatrix} e_l^{(k+1)} \\ \vdots \\ e_{l+m}^{(k+1)} \\ e_l^{(k+2)} \\ \vdots \\ e_{l+m}^{(k+2)} \\ \vdots \\ e_l^{(n)} \\ \vdots \\ e_{l+m}^{(n)} \end{bmatrix} \tag{6.23}
$$

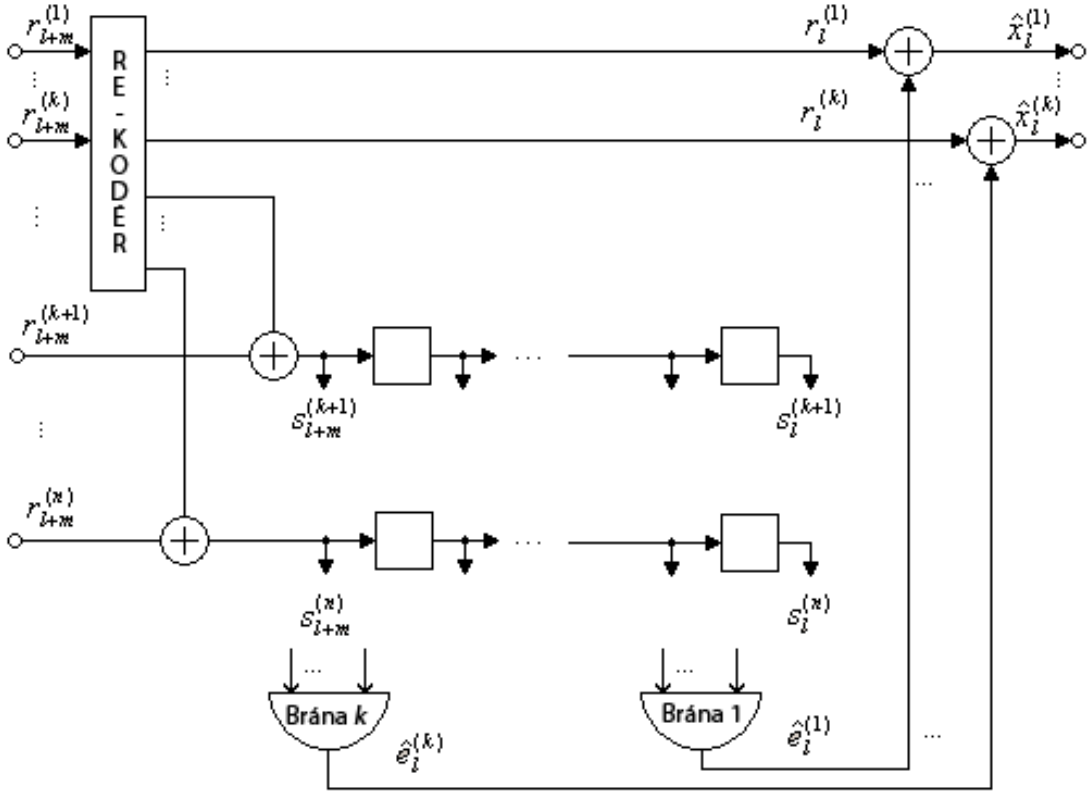
Můžeme vidět, že každý syndromový vektor může být vytvořen opětovným zakódováním části přijatého vektoru, která nese informace a následným přičtením znovu zakódovaného vektoru kontroly parity k části přijatého vektoru nesoucí kontrolu parity, což je

$$
s_l^{(j)} = \left(\sum_{i-1}^{k} \sum_{l'=0}^{m} r_{l-l'}^{(i)} g_{i,l'}^{(j)} \right) + r_l^{(j)}
$$

pro $(k+1) \leq j \leq n$. Proto vzorec obsahující přijatý vektor může být použit k výpočtu syndromových bitů, a vzorec obsahující chybový vektor lze využít při návrhu dekodérů většinovou metodou dekódování.

6.1.1 Jednoznačná metoda Většinového dekódování

Dekodér na obrázku se nazývá jednoznačný dekodér metody většinového dekódování. Do syndromových registrů se nevkládá žádná odhadovaná bitová chyba před dalším



Obr. 6.1: jednoznačná metoda většinového dekódování konvolučních kódů s tvrdou volbou

odhadem chyby. Ukončený syndromový vektor S'^T (6.23) nepotřeboval žádné úpravy

a je vyjádřen jako

$$\begin{bmatrix} s_l^{(k+1)} \\ \vdots \\ s_{l+m}^{(k+1)} \\ s_l^{(k+2)} \\ \vdots \\ s_{l+m}^{(k+2)} \\ \vdots \\ s_l^{(n)} \\ \vdots \\ s_{l+m}^{(n)} \end{bmatrix} = \begin{bmatrix} G'^{(k+1)}_1 & G'^{(k+1)}_2 & \dots & G'^{(k+1)}_k \\ G'^{(k+2)}_2 & G'^{(k+2)}_2 & \dots & G'^{(k+2)}_{2,l'} \\ \vdots & \vdots & & \vdots \\ G'^{(n)}_1 & G'^{(n)}_2 & \dots & G'^{(n)}_k \end{bmatrix} \begin{bmatrix} e_{l-m}^{(1)} \\ \vdots \\ e_l^{(1)} \\ \vdots \\ e_{l+m}^{(1)} \\ e_{l-m}^{(2)} \\ \vdots \\ e_l^{(2)} \\ \vdots \\ e_{l+m}^{(2)} \\ \vdots \\ e_{l-m}^{(k)} \\ \vdots \\ e_l^{(k)} \\ \vdots \\ e_{l+m}^{(k)} \end{bmatrix} + \begin{bmatrix} e_l^{(k+1)} \\ \vdots \\ e_{l+m}^{(k+1)} \\ e_l^{(k+2)} \\ \vdots \\ e_{l+m}^{(k+2)} \\ \vdots \\ e_l^{(n)} \\ \vdots \\ e_{l+m}^{(n)} \end{bmatrix} \tag{6.24}$$

kde

$$G'^{(j)}_i = \begin{bmatrix} g_{i,m}^{(j)} & g_{i,m-1}^{(j)} & \cdots & g_{i,0}^{(j)} & & & \\ & g_{i,m}^{(j)} & \cdots & g_{i,1}^{(j)} & g_{i,0}^{(j)} & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ & & & g_{i,m}^{(j)} & g_{i,m-1}^{(j)} & \cdots & g_{i,0}^{(j)} \end{bmatrix} \tag{6.25}$$

pro $1 \leq i \leq k$ a $(k+1) \leq j \leq$ n. Každá podmatice $G_i^{(j)}$ se nazývá paritní paralelogram daného kódu. Chybovou sekvenci přenosového kanálu lze rozdělit do informačních chybových sekvencí (chyby spojené s k informačními bity) a do paritní chybové sekvence (chyby spojené s $n-k$ kódovými bity). Každý bit syndromu nebo součet syndromových bitů reprezentuje součet chybových bitů přenosového kanálu a nazývá se kontrolní paritní součet. Celkový počet paritních paralelogramů je $(n-k) \cdot k$, paritní paralelogramy mohou být použity, abychom vybrali k soustav z J ortogonálních kontrolních součtů přes $e_l^{(1)}, e_l^{(2)}, \dots, e_l^{(k)}$. Soustava J kontrolních součtů je ortogonální na každý z těchto chybových bitů, jestliže určitý chybový bit $e_l^{(1)}, 1 \leq i \leq k$, je prověřen všemi kontrolními součty v soustavě a žádný jiný chybový bit není prověřen více než jedním kontrolním součtem v soustavě. Celkový počet různých chybových bitů prověřených k soustavami J ortgonálních kontrolních součtů se nazývá se nazývá efektivní délka kódového ohraničení kódu ν_E.

Lze ukázat, že jednoznačná metoda většinového dekódování systematického konvolučního kódu může opravit až $t_{\mathrm{ML}} = J/2$ chyb uvnitř ν_E.

Kroky většinového dekódování jsou:

1. Spočti syndromový vektor S'^T délky $(m+1)\cdot(n-k)$.

2. Soustava J ortogonálních kontrolních součtů na každém $e_l^{(i)}$, pro $1 \leq i \leq k$, je vytvořena ze syndromového vektoru S'^T.

3. Každá soustava J kontrolních součtů je zavedena zpět do brány většinové logiky. Jestliže více než polovina vstupních bitů jsou jedničky, předpokládá se, že $r_l^{(i)}$ je chybným, stanoví se $\hat{e}_l^{(i)} = 1$ pro $1 \leq i \leq k$. Jestliže jsou méně než polovina vstupních bitů jedničky, $r_l^{(i)}$ se předpokládá být bez chyb, stanoví se $\hat{e}_l^{(i)} = 0$.

4. Uskutečni opravu chyb sečtením $r_l^{(i)}$ a $\hat{e}_l^{(i)}$ a vydej očekávané informační bity $\hat{x}_l^{(i)} = 1$ pro $1 \leq i \leq k$. Syndromové registry jsou o jedno posunuty doprava. Poté, jsou n nově přijatých bitů vloženy do dekodéru. Spočítá se nová soustava $s_{l+m+1}^{(k+1)}, s_{l+m+1}^{(k+2)}, \ldots, s_{l+m+1}^{(n)}$ $n-k$ syndromových bitů, a následně je zavedena do syndromových registrů.

5. Kroky 2, 3, 4 jsou opakovány k odhadu chybových bitů, ke kterým došlo při přenosu $e_{l+1}^{(1)}, e_{l+1}^{(2)}, \ldots, e_{l+1}^{(k)}$. Dekódovací proces tímto způsobem dále pokračuje.

Pro $(n,k)\nu$ systematický stromový kód platí, že existuje $n-k$ nově tvořených syndromových bitů vztahujících se ke k informačních bitů, které mají být odhadnuty v každém kroku.

Příklad 6.1

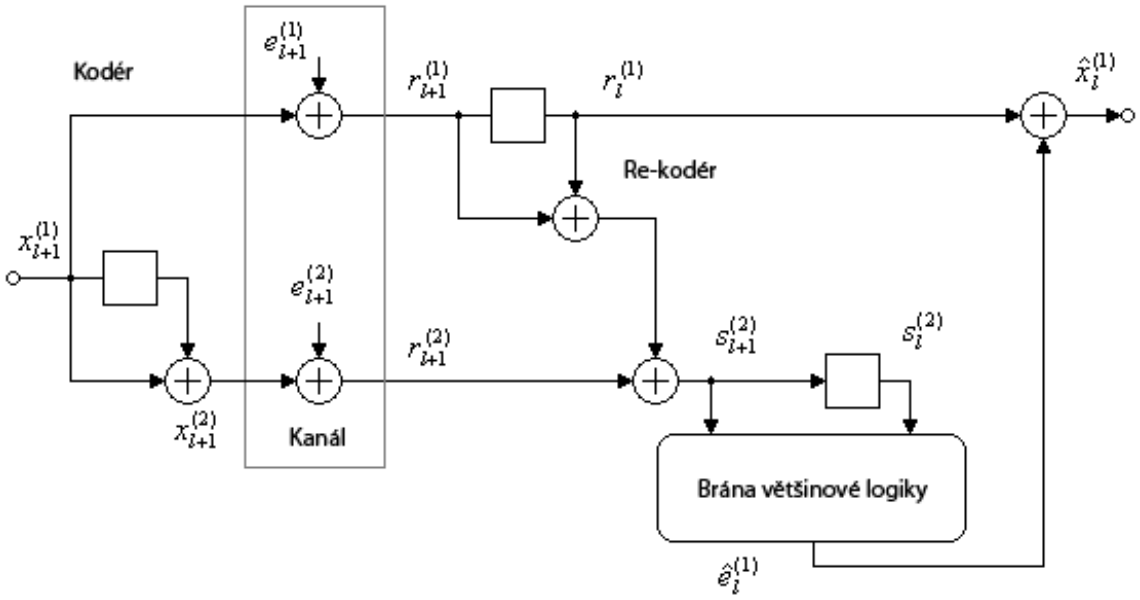
Uvažujme (2,1)2 systematický konvoluční kód s $G(D) = [1\ 1+D]$. Při přijetí $r_{l+1}^{(j)}$, $1 \leq j \leq 2$ ukončený syndromový vektor S'^T pro kód je

$$\begin{bmatrix} s_l^{(2)} \\ s_{l+1}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} e_{l-1}^{(1)} \\ e_l^{(1)} \\ e_{l+1}^{(1)} \end{bmatrix} + \begin{bmatrix} e_l^{(2)} \\ e_{l+1}^{(2)} \end{bmatrix} \tag{6.26}$$

Syndromové bity $s_l^{(2)}$ a $s_{l+1}^{(2)}$ tvoří soustavu dvou ortogonálních kontrolních součtů chybového informačního bitu $e_l^{(2)}$. Navíc,

$s_l^{(2)}$ kontroluje bit chyby informace $e_{l-1}^{(1)}$ a bit chyby parity $e_l^{(2)}$.
$s_{l+1}^{(2)}$ kontroluje bit chyby informace $e_{l+1}^{(1)}$ a bit chyby parity $e_{l+1}^{(2)}$.

Celkem 5 rozdílných bitových kanálových chyb je kontrolováno dvěma ortogonálními kontrolními součty, efektivní délka kódového ohraničení kódu je 5. Jednoznačným většinovým dekodérem mohou být opraveny ojedinělé chyby.



Obr. 6.2: Jednoznačná metoda většinového dekódování pro (2,1)2 systematický stromový kód

Ortogonální kontrolní součty pro kód jsou:

$$\begin{array}{rcccccccc} s_l^{(2)} & = & e_{l-1}^{(1)} & + & e_l^{(1)} & & & + & e_l^{(2)} \\ s_{l+1}^{(2)} & = & & & e_l^{(1)} & + & e_{l+1}^{(1)} & + & e_{l+1}^{(2)} \end{array}$$

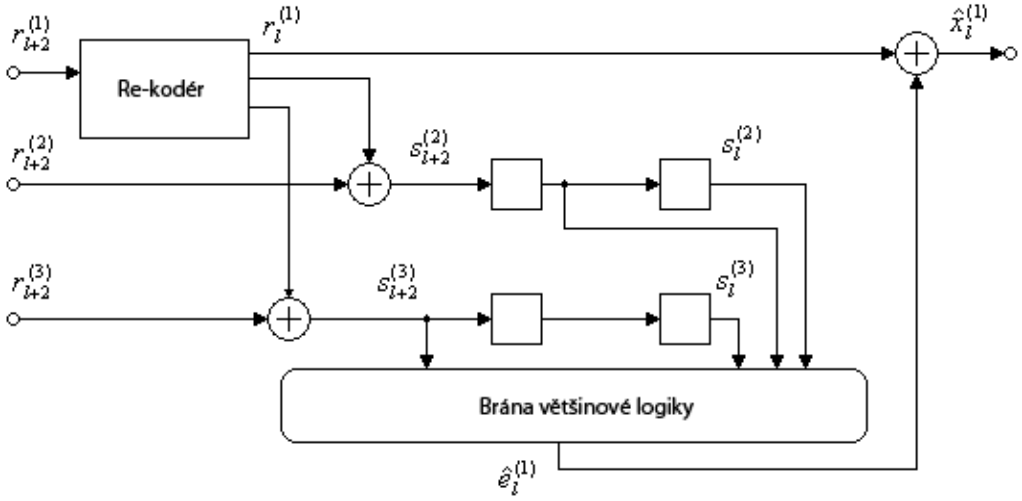
Očividně, pokud nedošlo k chybám, oba syndromové bity jsou vždy 0. Pokud se objeví chyby, dekodér nyní vybere odhad $\hat{e}_l^{(1)} = 1$ jen a pouze, jestliže oba ortogonální chybové součty, $s_l^{(2)}$ a $s_{l+1}^{(2)}$, přes $e_l^{(1)}$ mají hodnotu 1.

Příklad 6.2

Uvažujme (3,1)3 systematický konvoluční kód s $G(D) = [1\ 1+D\ 1+D^2]$. Při přijetí $r_{l+2}^{(j)}$, $1 \le j \le 2$ ukončený syndromový vektor S'^T pro kód je

$$\begin{bmatrix} s_l^{(2)} \\ s_{l+1}^{(2)} \\ s_{l+2}^{(2)} \\ \\ s_l^{(3)} \\ s_{l+1}^{(3)} \\ s_{l+2}^{(3)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ & & & & \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_{l-2}^{(1)} \\ e_{l-1}^{(1)} \\ e_l^{(1)} \\ e_{l+1}^{(1)} \\ e_{l+2}^{(1)} \end{bmatrix} + \begin{bmatrix} e_l^{(2)} \\ e_{l+1}^{(2)} \\ e_{l+2}^{(2)} \\ \\ e_l^{(3)} \\ e_{l+1}^{(3)} \\ e_{l+2}^{(3)} \end{bmatrix} \tag{6.27}$$

Syndromové bity $s_l^{(2)}, s_{l+1}^{(2)}, s_l^{(3)}$ a $s_{l+2}^{(3)}$ tvoří soustavu dvou ortogonálních kontrolních součtů chybového informačního bitu $e_l^{(2)}$. Navíc,



Obr. 6.3: Jednoznačné metoda většinového dekódování pro (3,1)3 systematický stromový kód

$s_l^{(2)}$ kontroluje bit chyby informace $e_{l-1}^{(1)}$ a bit chyby parity $e_l^{(2)}$.
$s_{l+1}^{(2)}$ kontroluje bit chyby informace $e_{l+1}^{(1)}$ a bit chyby parity $e_{l+1}^{(2)}$.
$s_l^{(3)}$ kontroluje bit chyby informace $e_{l-2}^{(1)}$ a bit chyby parity $e_1^{(3)}$.
$s_{l+2}^{(3)}$ kontroluje bit chyby informace $e_{l+2}^{(1)}$ a bit chyby parity $e_{l+2}^{(3)}$.

Celkem 9 rozdílných bitových kanálových chyb je kontrolováno čtyřmi ortogonálními kontrolními součty, efektivní délka kódového ohraničení kódu je 9. Jednoznačným většinovým dekodérem mohou být opraveny ojedinělé chyby. V případě výskytu chyb, dekodér vybere odhad $\hat{e}_l^{(1)} = 1$ jen a pouze, jestliže 3 nebo více ortogonálních kontrolních součtů přes $e_l^{(1)}$ má hodnotu 1. Jednoznačný většinový dekodér zde opraví 2 a méně chyb.

7 KRITÉRIA VÝBĚRU DEKÓDOVACÍ METODY

Vhodnou dekódovací metodu nelze vybrat náhodně, záleží na mnoha parametrech použitého kódu a také tom, jakým způsobem se danou metodu rozhodneme implementovat. Dekódování je možné realizovat alespoň dvěma způsoby: pomocí pevně zapojených elektronických obvodů (hardwarová realizace) nebo pomocí vhodně naprogramovaného mikropočítače (softwarová realizace). Výběr vhodné dekódovací metody provádíme na základě stanovení podmínek, co od dané dekódovací metody očekáváme. Platí, že čím větší má daný kód omezující délku ν, tím větší jsou korekční schopnosti kódu. Následná kritéria jsou vztažena pouze pro dekódování „tvrdou volbou“. Na druhou stranu je dobré si uvědomit, že při použití kódování je plýtváním informací dělat „tvrdou volbu“. Protože vliv každého informačního bitu dosahuje přes několik kanálových symbolů, dekodér může těžit z využití indikace hodnoty Z_i (u tvrdé volby záleží pouze na znaménku Z_i), aby odhadl míru spolehlivosti, zda byl symbol správně přijat. Je praktické výstup z demodulátoru s analogovým výstupem hodnoty Z_i kvantizovat. Například 3-bitová kvantizace rozdělí interval hodnot Z_i do 8 shodných intervalů a tím dostaneme „měkce“-kvantizovaný diskrétní kanál bez paměti. Tímto způsobem dosáhneme toho, že daný kód je mnohem účinnější už při nižším odstupu signálu od šumu.

	$-Z_i$				$+Z_i$			
tvrdá volba	0				1			
měkká volba	0	1	2	3	4	5	6	7
	$\rightarrow Z_i$							

Tab. 7.1: Rozdíl mezi tvrdou a měkkou volbou

7.1 Základní kritéria:

1. **Spolehlivost**

   Spolehlivost je schopnost plnit požadované funkce během stanovené doby. Dekódovací metodu vybíráme tak, abychom zajistili, že v dané aplikaci nedojde k jejímu selhání. Selháním můžeme rozumět neschopnost opravit v daném čase určité množství vzniklých chyb, nebo v nejhorším případě rozmnožení chyb do následujících přijatých sekvencí. U většiny systémů s kódovým zabezpečením nepovažujeme 100% přizpůsobení kódu na kanál za prakticky možné, proto v reálných systémech připouštíme v určitou zbytkovou chybovost. Našim cílem je zajistit správnou opravu chyb, tj. snížit BER (Bit Error Rate) na minimum.

A) Postupné pravděpodobnostní dekódování. Zde záleží na velikosti dekódovacího okna. Čím větší jej zvolíme, tím větší spolehlivost získáme. Velikost okna narůstá spolu s omezující délkou ν kódu. Nevýhodou je větší náročnost na paměť.

B) Viterbiho algoritmus. Záleží na délce přeživší cesty. Opět zde platí, čím větší ν, tím větší nutnost uložit do paměti delší cestu. Podmínkou je zvolení prahovací úrovně T. Je vhodný k opravě náhodných chyb.

C) Střadač algoritmus. Lze ho použít jako náhradu za Viterbiho algoritmus, kdy ν je příliš velké. Asymptoticky vyčerpává korekční schopnosti kódu.

D) Fanoův algoritmus. Platí podobné tvrzení jako u předchozího algoritmu. Spolehlivost je závislá na hodnotě prahovací úrovně T.

E) Tabulková metoda. Nevyužívá zpětný směr ve vyhledávání. Její spolehlivost je o poznání nižší, než např. u Viterbiho algoritmu (srovnání s ním zde uvádíme, protože je nejčastěji používaný). Plně nevyčerpává korekční schopnosti kódu. U této metody záleží na velikosti seznamu L.

F) Jednoznačná metoda. Zde je spolehlivost na relativně nižší úrovni. Tato metoda je vhodná do méně komplexních zařízení. Nejlépe opravuje shlukové chyby.

Metoda	A	B	C	D	E	F
Hodnocení	70%	90%	80%	80%	60%	50%

Tab. 7.2: Hodnocení při rozumném uvážení parametrů

2. **Propustnost**

   Rychlost dekódování, tuto vlastnost definujeme spíše jako propustnost dekodéru, tzn. maximální množství informace přenesené zařízením za jednotku času. Velikost je určena časovou náročností procesů, které zpracovávájí informaci při průchodu tímto zařízením. Je žádoucí omezit vznik zpoždění, dochází k němu obvykle v paměťových blocích.

   A) Postupné pravděpodobnostní dekódování. Relativně rychlá metoda, používá pouze dopředné vyhledávání. Zpoždění může vznikat díky paměťovým blokům, které je tak přímo úměrné velikosti dekódovacího okna a potažmo omezující délce ν.

   B) Viterbiho algoritmus. Taktéž nevyužívá zpětný směr. Je efektivní pro nízký kódový poměr. Ke zpoždění v dekodéru dochází tím, že na vstup je vypraven pouze nejstarší segment ideální cesty. Ideální Viterbiho dekodér by nemohl

rozhodnout o ideální cestě, dokud nepřijme celou sekvenci. Reálný dekodér musí limitovat délku přeživší cesty na přibližně 6ν, aniž by došlo ke ztrátě na kvalitě procesu dekódování.

C) Střadač algoritmus. Propustnost je omezena vyhledáváním ideální cesty i ve zpětném směru. V paměti je drženo 2^k z jedné a zbytek z ostatních cest.

D) Fanoův algoritmus. Je podobný algoritmu střadače, propustnost zde záleží na počtu kroků v dekódování. Počet kroků závisí na velikosti malé hodnoty Δ.

E) Tabulková metoda. Tato metoda opět využívá pouze dopředný směr. Na výstup se dostává nejstarší část přeživší cesty. Může vznikat zpoždění v závislosti na délce přeživší cesty.

F) Jednoznačná metoda. Rychlost dekódování touto metodou je velmi vysoká, k opravě se využívá namísto výběru správné cesty z paměti výpočty syndromových bitů. Zpoždění, vznikající v paměti, je nízké a odvíjí se od omezující délky ν kódu.

Metoda	A	B	C	D	E	F
Hodnocení	80%	70%	60%	50%	80%	100%

Tab. 7.3: Hodnocení propustnosti při rozumném uvážení parametrů

3. **Náročnost na paměť**

Náročnost dekodéru na paměť udává potřebné množství paměťových bloků, pokud se jedná o hardwarovou realizaci nebo velikost alokované paměti v případě softwarové realizace. Náročnost na paměť je důležitý parametr dekodéru, nezbytný pro správnou funkci přístroje. S větším množstvím potřebné paměti roste složitost, ale také třeba cena dekodéru. V paměti navíc dochází ke zpoždění, což je nežádoucí pro vlastní výkon procesu. Čím méně paměti je zapotřebí, tím lépe hodnotíme dekódovací metodu.

A) Postupné pravděpodobnostní dekódování. Platí, že pravděpodobnostní dekódování je charakteristické uchováváním možných cest v paměti. Velikost paměti, která je nutná pro tuto metodu se zvyšuje s omezující délkou ν a zvolenou velikostí dekódovacího okna. Pro vysoká ν je tato metoda nevhodná a příliš zatěžující.

B) Viterbiho algoritmus. V náročnosti na paměť si nezadá s předchozí metodou. Spotřeba paměti velmi stoupá pro vysoká ν. Dekodér v paměti uchovává 2^k cest pro každý předcházející uzel.

C) Střadač algoritmus. Dekódovací proces je nezávislý na celkové paměti kodéru, takže jej lze použít i pro konvoluční kódy s velmi dlouhou omezující délkou ν. Střadač uchovává cesty i s jejich vzdáleností, je potřeba tolik paměti, jak dimenzovaný je střadač.

D) Fanoův algoritmus. Dekodér fanova algoritmu se vždy pohybuje na jednom uzlu v určitém kroku. V paměti uchovává vždy několik uzlů ve zpětném směru a potřebuje vědět o vlastnostech dalších uzlů stromového grafu v dopředném směru. Náročnost dekodéru na paměť je mírná.

E) Tabulková metoda. Vyznačuje se tím, že uchovává počet cest o velikosti až $2L$, kde L je velikost seznamu cest. Celkově tato metoda neklade na paměť nijak závratné nároky.

F) Jednoznačná metoda. Zde je spotřeba paměti minimální, dekodér potřebuje pouze přibližně dvakrát tolik paměťových bloků, co kodér. Je to nenáročná metoda.

Metoda	A	B	C	D	E	F
Hodnocení	70%	60%	90%	90%	80%	100%

Tab. 7.4: Hodnocení náročnosti na paměť při rozumném uvážení parametrů

4. **Komplexnost a složitost realizace**

Komplexnost metody lze chápat jako složitost procesu, náročnost přístroje na logiku a počet potřebných obvodů. Vysoká komplexnost znesnadňuje návrh dekodéru a klade větší nároky na zdroje. Komplexnost přístrje nelze srovnávat se spolehlivostí, ikdyž bývá pravidlem, že komplexnější dekodér je často spolehlivější. Snažíme se o co nejnižší komplexnost, žádoucí je metoda co nejjednodušší.

A) Postupné pravděpodobnostní dekódování.Implementace této metody začne být velmi složitá pro vysoká ν.

B) Viterbiho algoritmus. Komplexnost značně roste s vysokým kódovým poměrem a omezující délkou ν.

C) Střadač algoritmus. Implementace této metody není zrovna z nejjednodušších. Výhodou je zde nezávislost na omezující délce kódu.

D) Fanoův algoritmus. Dá se říci, že složitost Fanova algoritmu je poněkud větší a tudíž realizace obtížnější než Střadač.

E) Tabulková metoda. Složitost není vysoká a drží se pod přijatelnou hranicí.

F) Jednoznačná metoda. Komplexnost této metody je přímo úměrná kódovému poměru a omezující délce ν. Pro nízké hodnoty je docela jednoduchá.

Metoda	A	B	C	D	E	F
Hodnocení	80%	70%	70%	60%	90%	90%

Tab. 7.5: Hodnocení komplexnosti při rozumném uvážení parametrů

7.1.1 Celkové hodnocení

Vzhledem ke všem výše uvedeným kritériím nyní provedeme závěrečné porovnání metod dekódování. Hodnocení budiž pouze orientační a berme v úvahu, že při výběru metody se orientujeme podle aktuálních potřeb. Průměrem hodnocení předešlých 4 kritérií vznikne výsledná tabulka:

Metoda	A	B	C	D	E	F
Hodnocení	75%	72,5%	75%	70%	77,5%	85%

Tab. 7.6: Celkové průměrné hodnocení dekódovacích metod

7.2 Ostatní kritéria

Při návrhu dekodéru bychom měli vycházet také z toho, jaké vlastnosti má přenosový kanál. Na charakteru přenosového kanálu se podepisuje ponejvíce, zda na něm dochází spíše k náhodným nebo shlukovým chybám. Pro náhodné chyby lze doporučit Viterbiho dekódování, zatímco shlukové chyby jsou opravitelné využitím syndromů a tedy pomocí Většinového dekódování.

Následující přehled parametrů stromových kódů značně ovlivňuje výběr dekódovací metody.

- Omezující délka ν

  Pro dlouhé ν rozhodně budeme volit sekvenční dekódování, tedy jeden z algoritmů Střadač nebo Fanoův algoritmus. Nejméně je vhodné pravděpodobnostní dekódování, což představují metody Postupné pravděpodobnostní dekódování a Viterbiho algoritmus.

- kódový poměr $R = \frac{k}{n}$

  Je na místě pro vysoký kódový poměr nejlépe využít jednu z metod sekvenčního dekódování. Ostatní metody nebudou výkonově stačit a raději je volíme pro nižší kódové poměry.

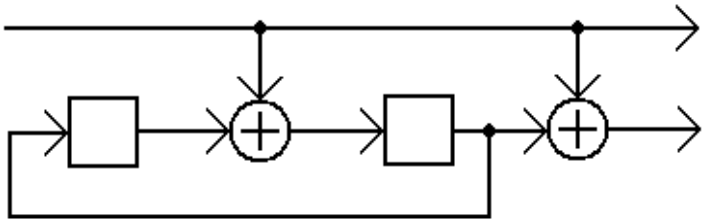
- Systematický a nesystematický kód

  S jistotou lze tvrdit, že všechy uvedené metody dokáží dekódovat systematické kódy. Není tomu tak u nesystematických kódů. Metoda Většinového dekódování potřebuje ke své činnosti kód systematický a žádný jiný. Naštěstí je možné každý nesystematický konvoluční kód s vytvářecí maticí přetvořit na kód systematický vhodnými řádkovými transformacemi vytvářecí matice. Systematické kódy navíc nejsou nikdy katastrofické, což znamená, že u nich nemůže dojít k nekonečnému rozmnožení chyb, zatímco u nesystematických kódů může.

8 NÁVRH DEKODÉRU

8.1 Rozbor zadaného kodéru

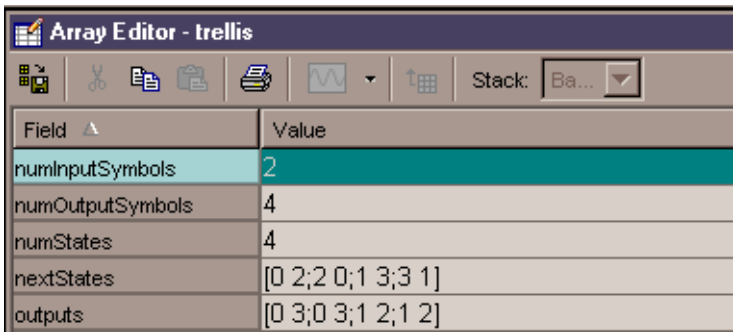
Jedná se o kodér systematického kódu s nízkým kódovým poměrem $R = 1/2$ a krátkou omezující délkou $\nu = 3$. Obsahuje zpětnou vazbu, která zvyšuje korekční schopnosti kódu.



Obr. 8.1: Zapojení kodéru stromového kódu

Těmito příkazy lze zadat mřížovou strukturu kodéru v prostředí Matlab, obyčejně by bylo možné ji vytvořit například funkcí `poly2trellis`, ale zadaný kodér má neobvyklou zpětnou vazbu.

```
% Definice mřížové struktury
trellis.numInputSymbols = 2;
trellis.numOutputSymbols = 4;
trellis.numStates = 4;
trellis.nextStates = [0 2;2 0;1 3;3 1];
trellis.outputs = [0 3;0 3;1 2;1 2];
```



Obr. 8.2: Struktura mříže v proměnné Matlabu

8.1.1 Volba dekódovací metody

Využitím předem vypracovaných kritérií bychom se měli dopracovat k metodě dekódování, která bude vhodná a realizovatelná dostupnými prostředky. Zájmem zde bude zaměřit se na co nejmenší komplexnost metody a tedy schopnost úspěšné realizace dekodéru. Vzhledem k tomu, že kodér je systematický, nejsme nijak omezeni při výběru metody, naopak spíše směrováni k využití Většinového dekódování. Nízký kódový poměr také nijak neomezuje monožinu dekódovacích metod, nevelká omezující délka je výhodou pro jednodušší návrh. Prostředí Matlab Simulink napomáhá při tvorbě blokových schémat, je zde dobrá možnost tvorby syndromů z bitových toků. Výpočet syndromů využívá Většinové dekódování, zvolíme tuto metodu.

8.2 Navržení dekodéru a simulace

Uvažujme systematický stromový kód zadaný podle obrázku. Při přijetí $r_{l+1}^{(j)}$, $1 \leq j \leq 2$ ukončený syndromový vektor S'^T pro kód je

$$\begin{bmatrix} s_l^{(2)} \\ s_{l+1}^{(2)} \\ s_{l+2}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} e_{l-2}^{(1)} \\ e_{l-1}^{(1)} \\ e_l^{(1)} \\ e_{l+1}^{(1)} \\ e_{l+2}^{(1)} \end{bmatrix} + \begin{bmatrix} e_l^{(2)} \\ e_{l+1}^{(2)} \\ e_{l+2}^{(2)} \end{bmatrix}$$

Ortogonální kontrolní součty pro kód jsou:

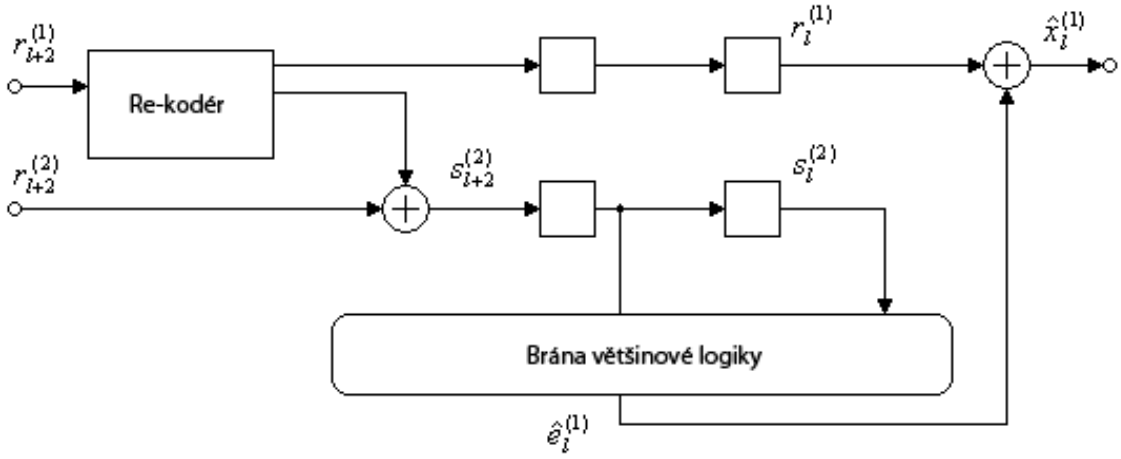
$$\begin{array}{lcccccccc} s_l^{(2)} & = & e_{l-1}^{(1)} & + & e_l^{(1)} & & & + & e_l^{(2)} \\ s_{l+1}^{(2)} & = & & & e_l^{(1)} & + & e_{l+1}^{(1)} & + & e_{l+1}^{(2)} \\ s_{l+2}^{(2)} & = & & & & & e_{l+1}^{(1)} + e_{l+2}^{(1)} & + & e_{l+2}^{(2)} \end{array}$$

Vstupní data dekodéru se dělí na informační a zabezpečující část. Informační část představuje vlastní přenášená data, která mohou být poškozena. K ověření korektnosti přijatých dat se proto opět počítají zabezpečující data.

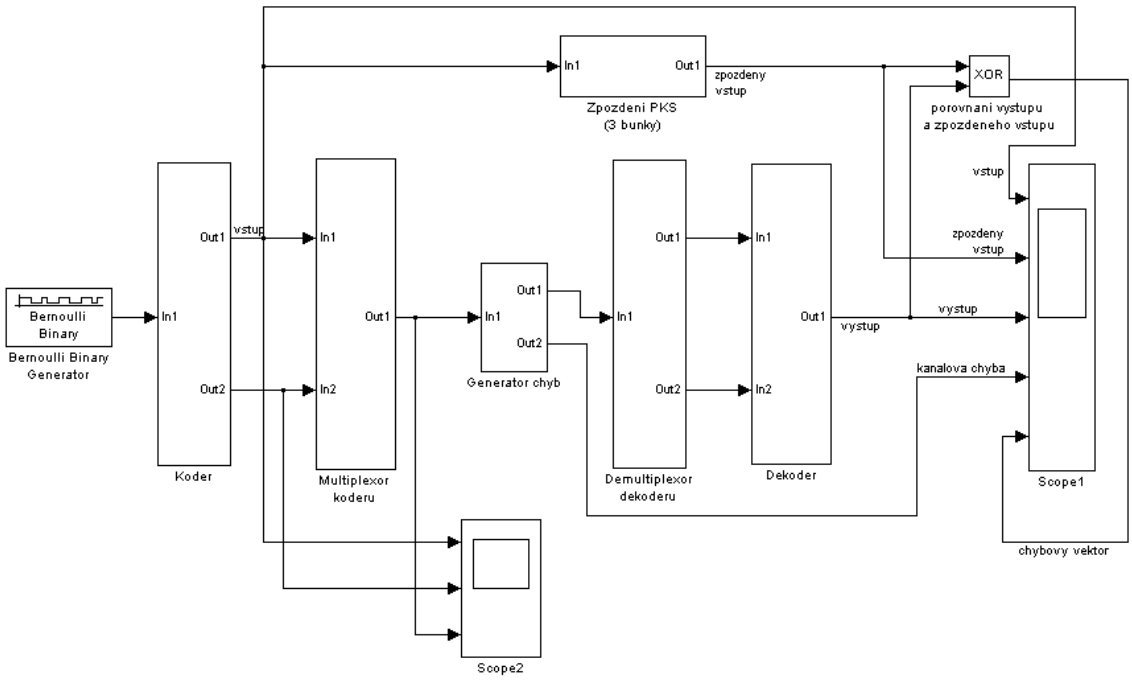
Při návrhu jsem vycházel z literatury [2]. Zapojení dekodéru v Simulinku je velice podobné obrázku.

Na obrázku jsou vidět průběhy vstupu a výstupu, chybu vloženou přenosovým kanálem a rozdílový vektor. Kanálová chyba má průběh s dvojnásobnou frekvencí, protože sleduje multiplexovaný datový tok.

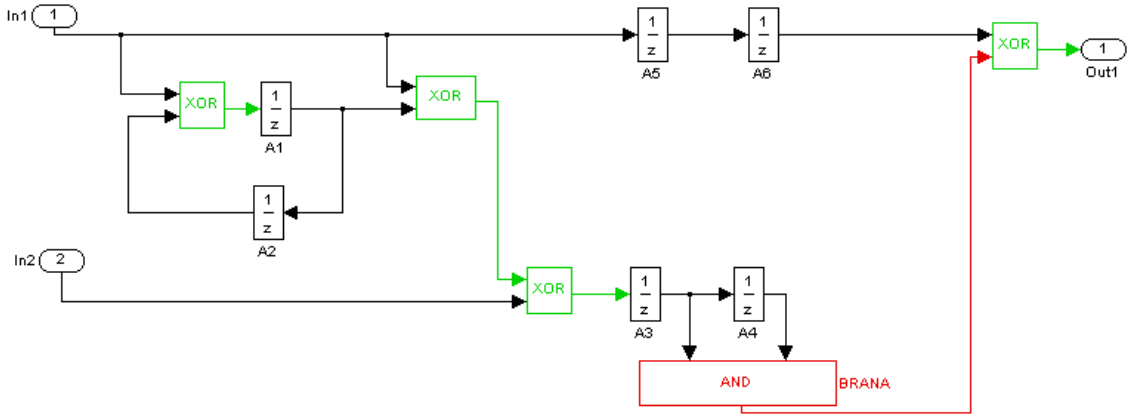
Dekodér v dekódovacím procesu dokázal zredukovat 3 kanálové bitové chyby na 1 bit chyby datového slova. Výkon tohoto dekodéru je malý, zůstává velká zbytková chybovost. Upřednostnili jsme zde nízkou komplexnost a rychlost dekódování nad spolehlivostí.



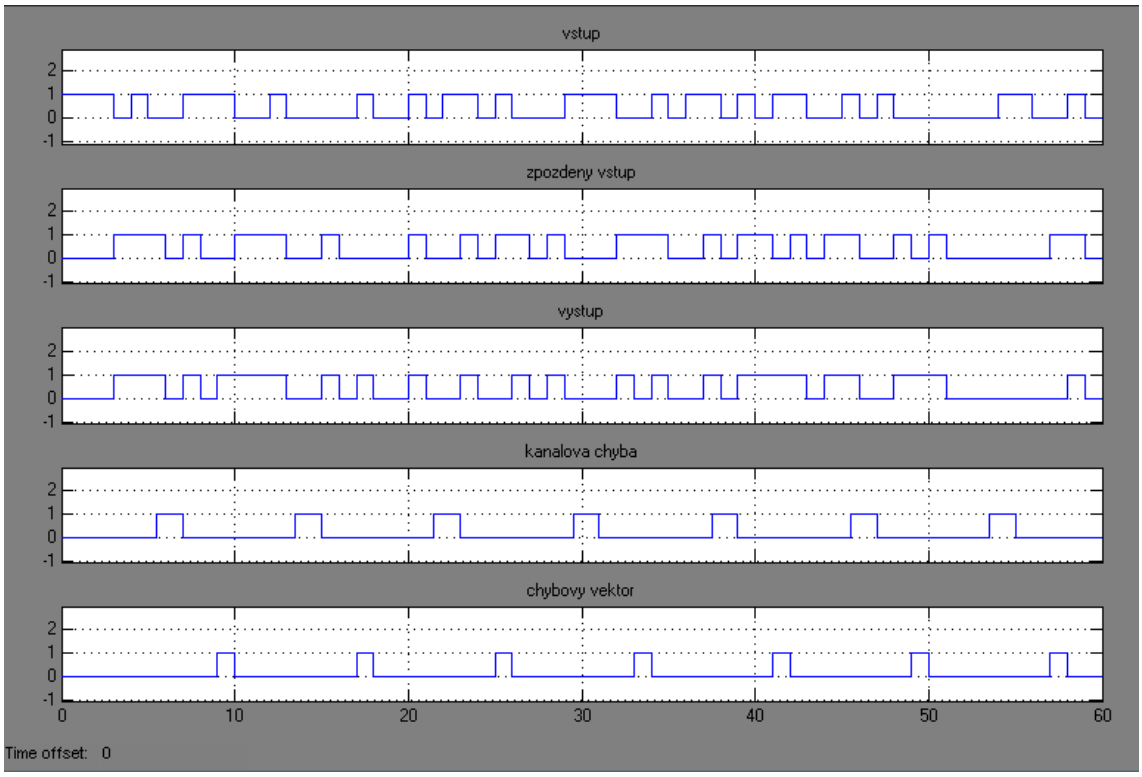
Obr. 8.3: Dekódování jednoznačnou metodou



Obr. 8.4: Model PKS s dekodérem



Obr. 8.5: Zapojení dekodéru



Obr. 8.6: Výsledné průběhy procesu dekódování

9 ZÁVĚR

Tato bakalářská práce se zabývala analýzou známých metod dekódování stromových kódů. Značná část těchto metod je již dlouho objevena. Záměrně jsem se věnoval pouze dekódování s „tvrdou“ volbou. Lze na nich lépe vysvětlit všechny metody zmíněné v práci. Rozdíl mezi tvrdou a měkkou volbou naleznete v kapitole 7. V kapitole 3 jsem rozebral metody pravděpodobnostního dekódování, kam patří hlavně Viterbiho algoritmus, pak jsem v kapitole 4 popsal sekvenční dekódování a nastínil jsem jednoduchou tabulkovou metodu v kapitole 5 i jednu metodu většinového dekódování v kapitole 6.

Důležitou částí této práce je v podstatě návod k výběru vhodného algoritmu, obsažen je v kapitole kritéria výběru dekódovací metody. Text se zabývá parametry kódu v návaznosti na výsledné možnosti kódu využitím různých dekódovacích metod. Nikde v žádné literatuře jsem nenašel ucelený přehled kritérií, proto jsem vytvořil svá kritéria na základě obecných vlastností dekódovacích metod, které jsem čerpal z literatury [1] a [4]. Mezi základní kritéria patří spolehlivost, propustnost, náročnost na paměť a komplexnost dekodéru. Pro každé kritérium platí, že různé metody získají jiné ohodnocení. Metody jsou hodnoceny procentuálně. Víc procent znamená, že danému kritériu vyhovuje lépe. Méně procent znamená, že je podle toho kritéria vhodná méně.

Podle kritérií jsem rozhodl, že k návrhu použiju metodu většinového dekódování. Navrhl jsem dekodér v prostředí Simulink. Návrh dekodéru je popsán v kapitole 8. Ověřil jsem jeho funkci simulací vložením umělé shlukové chyby do přenosové cesty. Dekodér podle předpokladů nedokáže opravit ojedinělé chyby. Shluková kanálová chyba o délce 3 bity byla zredukována na 1 chybný informační bit. Přestože dekodér nedokázal uskutečnit ideální korekci přenášených dat, znamená to funkčnost realizovaného modelu. Proč jsem zvolil vybranou metodu, jsem výstižně popsal v sekci volba dekódovací metody.

LITERATURA

[1] JOHANNESON, R., ZIGANGIROV, K. S.,
Fundamentals of convolutional coding. 1999. IEEE Press, ISBN 0-7803-3483-3.

[2] KŘIVÁNEK, V., *Návrh kodérů a dekodérů umožňující opravu shlukových chyb a jejich simulace*. Publikace v internetovém magazínu Elektrorevue. Dostupné z URL: <http://www.elektrorevue.cz/clanky/06008/>. Brno 2006

[3] KŘIVÁNEK, V., ČÍKA, P., *Simulace shlukových chyb v Matlabu*. Elektrorevue, ISSN 1213-1539. Brno, roč. 2006, č. 35.

[4] LEE, L. H. Charles, *Convolutional Coding - Fundamentals and Applications*. 1997. Artech House, Boston, London, ISBN 0-89006-914-X.

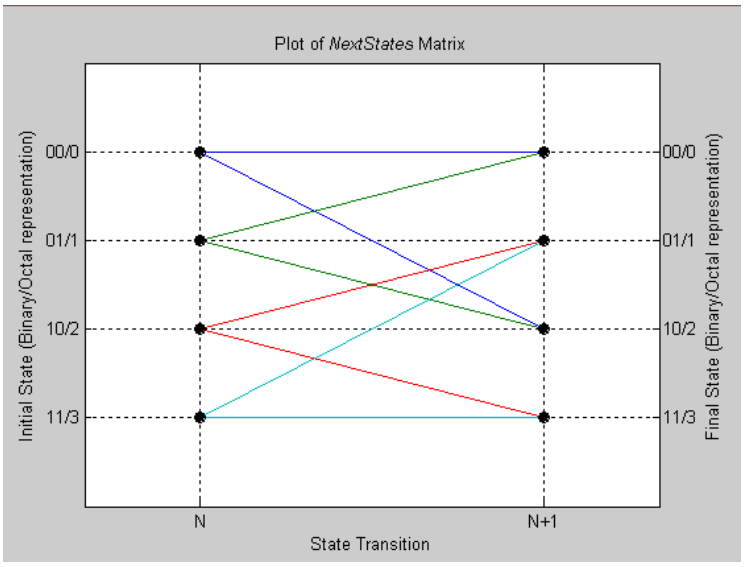
[5] NĚMEC, K., *Protichybové kódové systémy*. Elektrorevue 2001/29. Dostupné z URL: <http://www.elektrorevue.cz/clanky/01029>.

[6] NĚMEC, K., *Řešení požadavků kladených na protichybový kódový systém*. Elektrorevue 2006/17. Dostupné z URL: <http://www.elektrorevue.cz/clanky/06017>.

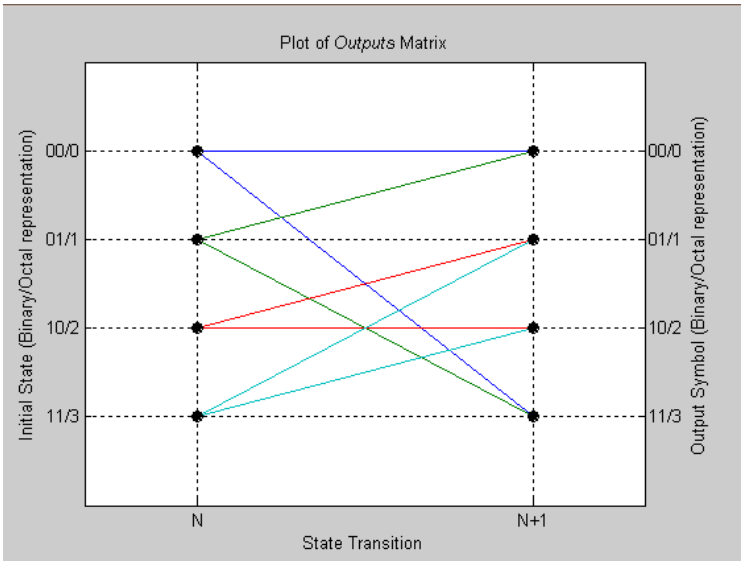
[7] NĚMEC, K. *Šestá přednáška BDAK 2006/2007*.

[8] PROAKIS, John G., MASOUD, Salehi,
Contemporary communication systems using MATLAB. 1998. PWS Publishing Company, ISBN 0-534-93084-3.

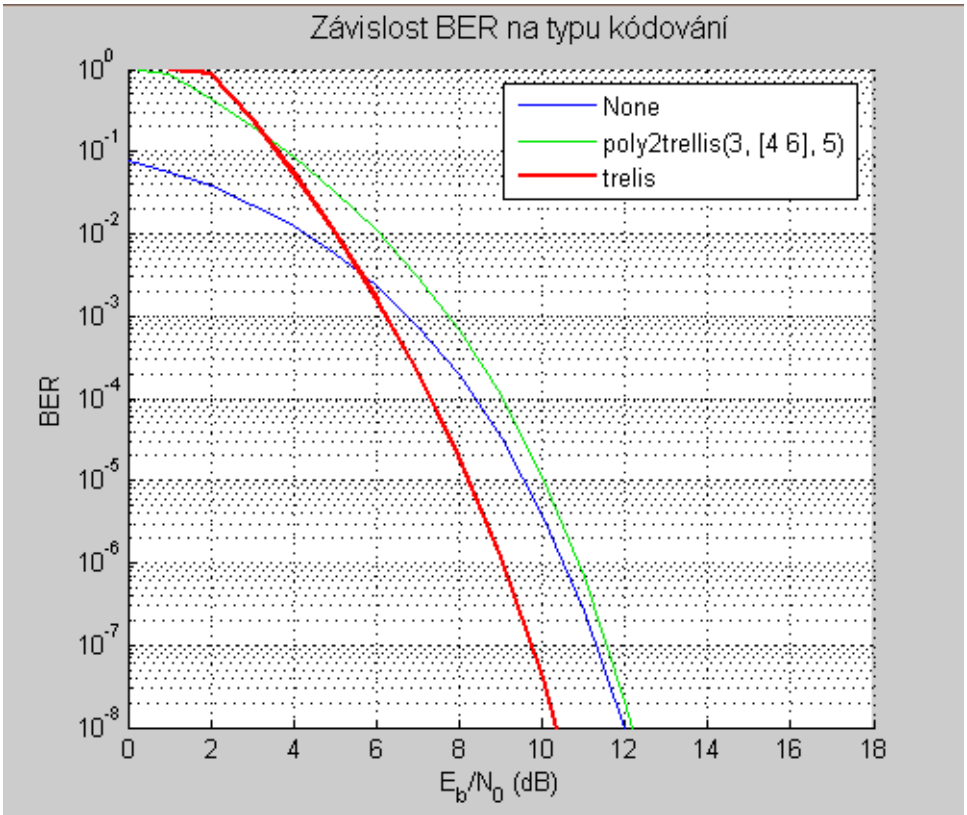
A PRVNÍ PŘÍLOHA



Obr. A.1: Graf přechodu stavů v mřížovém grafu



Obr. A.2: Graf výstupů stromového kódu



Obr. A.3: Srovnání účinnosti typů kódování