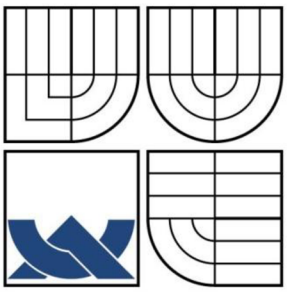
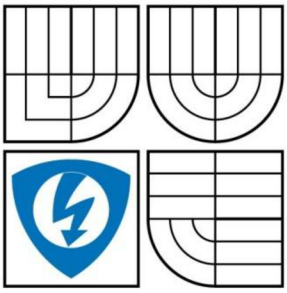


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

METODY PROKLÁDÁNÍ ZPRÁVY

METHODS OF INTERLEAVING DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

MICHAL SOUDEK
AUTHOR

BC. MICHAL SOUDEK

VÍTĚZSLAV KŘIVÁNEK
SUPERVISOR

ING. VÍTĚZSLAV KŘIVÁNEK

BRNO 2008

Obsah

1 ÚVOD 12

1.1 PŘENOSOVÉ SYSTÉMY 13
1.2.1 PKS se zpětnou vazbou 14
1.2.2 PKS bez zpětné vazby 14
1.2.2 PKS bez zpětné vazby 14

2 ROZDĚLENÍ ZABEZPEČOVACÍCH KÓDU 15

2.1 BLOKOVÉ KÓDY 15
2.2 STROMOVÉ KÓDY 15
2.3 SYSTEMATICKÉ KÓDY 15
2.4 NESYSTEMATICKÉ KÓDY 15

3 VZNIK CHYBY 17

4 KONVOLUČNI KÓDY 20

4.1 POPIS KÓDOVACÍ TECHNIKY 20
4.2 ZADÁNÍ KONVOLUČNÍHO KÓDU 23
4.2.1 Zadání kódu pomocí vytvářecí matice 23
4.2.2 Zadání kódu pomocí mnohočlenu matice 24

5 PROKLÁDÁNÍ ZPRÁVY 14

5.1 SHLUK CHYB 25
5.2 POPIS PROKLÁDÁNÍ 26
5.2.1 Blokové prokládání 27
5.2.2 Konvoluční prokládání 32
5.2.3 Diagonální prokládání 34
5.2.4 Inter – blokové prokládání 35

6 VÝBĚR TECHNIKY 37

6.1 KONVOLUČNÍ KÓDY PROTI SHLUKU CHYB 35
6.2.1 Hagelbargerův kód 37
6.2.2 Iwadari – Maseryuv kód 40
6.2.3 Berlekamp -Preparatův 43
6.2 METODA PROKLÁDÁNÍ ZPRÁVY 16

7 REALIZACE PROVEDENÍ 46

7.1 UVEDENI DO PROBLEMATIKY 46
7.2 HAGELBAGERUV KOD 52
7.3 IWADARI - MASERYUV KOD 55
7.4 BERLEKAMP – PREPARATUV KOD 58
7.5 GENERÁTOR CHYB 59

7.5 BLOKOVÉ PROKLADANÍ 62
7.6 KONVOLUČNI PROKLADANÍ 63
7.7 POROVNANI METOD 64

8 ZÁVĚR 68

9 POUŽITÁ LITERATURA 70

Seznam obrázků

Obr. 1.1 Přenosový systém 13
Obr. 1.2 Přenosový protichybový kódovací systém 13
Obr. 1.3 Přenosový systém se zpětnou vazbou 14
Obr. 2.4 Rozdělení zabezpečovacích kódů 16
Obr. 3.5 Vliv chybneho vektoru na posloupnost bitů 17
Obr. 3.6 Modul rušení kanálu 17
Obr. 3.7 Porušení posloupnosti 18
Obr. 3.8 Blokové schéma kanálu sečítačky mod2 19
Obr. 4.9 Princip konvolučního kódování 20
Obr. 4.10 Obecné blokové schéma konvolučného kodéru 20
Obr. 4.11 Blokové schéma konvolučního kodéru 21
Obr. 4.12 Obecné blokové schéma vlastního dekodéru 21
Obr. 4.13 Princip konvolučního dekodéru 22
Obr. 4.14 Blokové schéma konvolučního dekodéru 22
Obr. 5.15 Shluky chyb 25
Obr. 5.16 Umístění prokládání v PKS 26
Obr. 5.17 Schéma PKS s prokládáním bitů 26
Obr. 5.18 Správné nastavené přenosového systému 29
Obr. 5.19 Špatné nastavené přenosového systému 31
Obr. 5.20 Blokové prokládání 32
Obr. 5.21 Konvoluční kódovací systém s prokládáním 33
Obr. 5.22 Konvoluční prokládání v paměti 34
Obr. 5.23 Matice diagonálního prokládáné 35
Obr. 5.24 Diagonální prokládání 35
Obr. 5.25 Inter-blokové prokládání 36
Obr. 7.26 Model přenosového sytému 46
Obr. 7.27 Kodér s prokládáním 47
Obr. 7.28 Dekodér s prokládáním 48
Obr. 7.29 Sério – paralelní převodník 49
Obr. 7.30 Průběh na sério – paralelním převodníku 49
Obr. 7.31 Paralelně – sériový převodník 50

Obr. 7.32 Průběh na paralelně – sériovém převodníku 51
Obr. 7.33 Generator Hegelbargerových shluku chyb 58
Obr. 7.34 Průběhy Hegelbargerova generatoru chyb 59
Obr. 7.35 Generator chyb Iwadari – Maseryuv kód 59
Obr. 7.36 Generator chyb Berlekampuv – Preparatuv kód 60
Obr. 7.37 Simulace Hagelbargova kódu 60
Obr. 7.38 Simulace Hagelbagova kódu detail 61
Obr. 7.39 Simulace Iwadariho kódu 65
Obr. 7.40 Simulace Iwadariho kódu detail 65
Obr. 7.41Simulace Berlekapuv – Preparatuv kód 66
Obr. 7.42 Simulace Berlekapuv – Preparatuv kód detail 66

Seznam tabulek

Tab. 1 Sečítačka mod2 20
Tab. 2 Porovnání konvolučních kódů..... 64
Tab. 3 Porovnání prokládání konvolučních kódů..... 64

ANOTACE

Úvodní část práce je věnována celkovému obeznámení s přenosovým systémem a jeho rozdělení.

Dále je práce zaměřena na rozdělení zabezpečovacích kódu, které se používají na přenosových systémech.

V další kapitole je rozebraná problematika vzniku chyby, matematický zápis chyb, třídění chyb, které mohou při přenosu vniknout.

Následuje kapitola s popisem konvolučních kódů s popisem zabezpečení proti chybám, princip převodu sériové posloupnosti na posloupnost paralaxní a naopak. Je zde nastíněna problematika zadání konvolučních kódů. A v neposlední řadě jsou tady zmíněné tři konvoluční kódy, které se používají k ochraně před shluky chyb.

V další kapitole je zmíněna problematika prokládání zpráv – interleaving a používané metody s popisem, jak vznikají, a podrobný popis, jak se za pomocí prokládání dlouhé úseky chyb odstraňují.

V následující kapitole jsou popsány techniky, které se v praxi používají proti potlačení shluku chyb.

V poslední kapitole je praktická část diplomové práce. Jsou podrobně popsány a nasimulovány techniky, jak vzniká zabezpečení nezabezpečeného úseku, přenos po vedení, naindukování shluku chyb na přenášená zabezpečená data a následná oprava či rozdělení shluku chyb v přijímači. Pro simulaci byly použity tři konvoluční kódy pro opravu shluku chyb a dvě prokládací techniky.

Klíčová slova: přenos, shluk chyb, prokládání, kodér a simulace

ABSTRACT

The initial part of my work is dedicated to overall introduction into the transmission systems and its categories.

Further the work is focused on dividing of security codes which are used for transmission systems.

In the next chapter is analyzed the problems of error origin, mathematical transcription of errors, categorizing of errors which can generate during transmission.

The following chapter deals with description of convolutional codes describing security against errors, principle of serial consecution interpretation on parallastic consecution and visa versa. Here is outlined problems of convolutional codes input. In the next part are mentioned three convolution codes which are used for security against burst errors.

In the next chapter is mentioned problems of message interleaving and description of used methods, how they originate and detailed description, how the errors are eliminated with the help of long segment interleaving.

In the next chapter are described techniques which are used for clustered errors suppression.

The last chapter is dedicated to the practical part of my thesis. There are detail descriptions and simulation techniques how the security of non-secure section is developed, transmission on line, burst errors induction on transmitted secure data and consecutive amendment or clustered errors division in transceiver. For the simulation there were utilized three convolutional codes for clustered errors amendment and two interleaving techniques.

Keywords: transfer, burst- error, interleaving, coder and simulation

Zkratky

A ……..ochranný interval

R ……..informační rychlost

b……shluk chyby

k_0……..nezabezpečený bitový tok

n_0……..zabezpečený bitový tok

B_D…….dekadická bloková matice

B_0……..binární bloková matice

L(n)……udává počet pro zapsaní dekadického čísla pomocí dvojkové soustavy

Z_K…….zpoždění kodéru

Z_D…….zpoždění dekodéru

S_{PK}……počet paměťových míst v kodéru

S_{PD}…...počet paměťových míst v dekodéru

S_L……..počet logických operátorů

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:

Bytem:

Narozen/a (datum a místo):

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta ……

se sídlem ……

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, CSc.

(dále jen „nabyvatel“)

11

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

□ disertační práce

□diplomová práce

□ bakalářská práce

□ jiná práce, jejíž druh je specifikován

jako.....

(dále jen VŠKP nebo dílo)

Název VŠKP:

Vedoucí/ školitel VŠKP:

Ústav: telekomunikací

VŠKP odevzdal autor nabyvateli v*:

□tištěné formě – počet exemplářů 2

□elektronické formě – počet exemplářů 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané
a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

□ ihned po uzavření této smlouvy
□ 1 rok po uzavření této smlouvy
□ 3 roky po uzavření této smlouvy
□ 5 let po uzavření této smlouvy
□ 10 let po uzavření této smlouvy
(z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

* hodící se zaškrtněte

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem

o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně
nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: ……

…… ……

Nabyvatel Autor

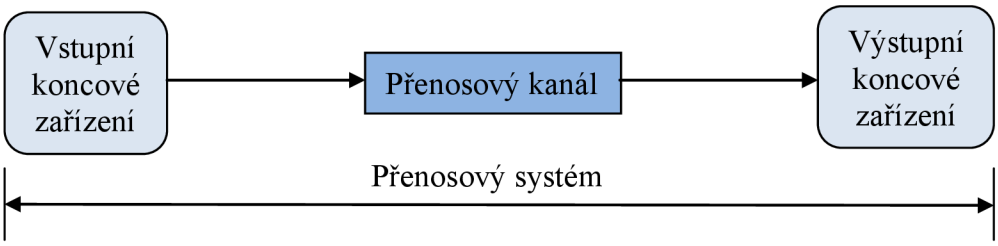
1 Úvod

V dnešní době, kdy se stala informační technologie zcela běžnou součástí každého obyvatele je potřeba přenášet obrovské objemy dat, které se přenáší od vysílače informací k přijímači, kde musíme být schopni zaručit, že všechny odeslané dojdou ke spotřebiči dat zcela neporučené a identické, jak byly odeslány. Nesmí se v přijímači nijak změnit význam jednotlivých informačních bitů a tím vyvolaná odezva na případné chybné reakce. Proto musíme jednotlivé vysílané bloky zpráv na straně vysílače zabezpečit tak, abychom byli schopni v přijímači odhalit případné chyby vzniklé během přenosu, a následně jsme mohli na danou chybu správně zareagovat. Proto se v daném oboru rozvijí zabezpečovací mechanizmy, které vloží do nezabezpečené informace zabezpečovací prvky, pomocí kterých jsme schopni při dekódovaní odhalit vzniknutou chybu a tu opravit na správnou hodnotu. Kódové zabezpečení provádíme v protichybových kódových systémech, které vychází ze znalosti zabezpečovacích kódu. Proto největší problém převážně v protichybovém kódovacím systému nastává při výběru nejvhodnějšího kódu pro zabezpečení proti chybám.

Proto je zapotřebí testovat přenosové kanál během přenosu a podle toho vhodně upravovat zabezpečovací schopnosti kanálu.

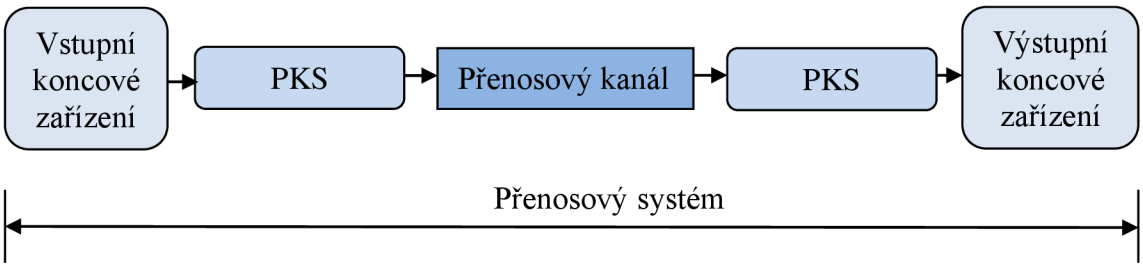
1.1 PŘENOSOVÉ SYSTÉMY

Pro přenos jakékoliv informace od zdroje (vstupního koncového zařízení) k cíli (výstupnímu koncovému zařízení) přes přenosový systém se informace přenáší pomocí přenosového kanálu. Bohužel tohle je jen zidealizovaný model, protože vlivem rušení přenosového kanálu dochází ke změně části informace.



Obr. 1.1 Přenosový systém

Abychom tomu zabránili, je zapotřebí vložit mezi koncová zařízení a přenosový kanál blok protichybový kódovací systém (dále jen PKS), který obsahuje vhodné mechanismy pro úpravu signálu Obr. 1.1, aby se ve výstupním zařízení neobjevovali chybně přenesené informace. PKS je schopen reagovat na aktuální situaci během přenosu (chybovost v přenosovém kanálu), a měnit podle situace prvky zabezpečení tzn. systém musí být flexibilní a musí se v danou situaci podřídit parametrům přenosu.



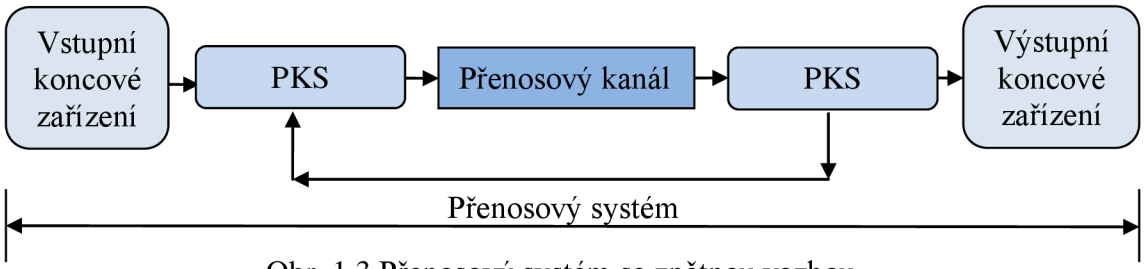
Obr. 1.2 Přenosový protichybový kódovací systém

Přenosový protichybový kódovací systém rozlišujeme na dva druhy, a to na: můžeme rozdělit na

- PKS se zpětnou vazbou
- PKS bez zpětné vazby

1.2.1 PKS se zpětnou vazbou

PKS se zpětnou vazbou neboli se zpětným kanálem poskytuje PKS informaci o stavu doručení přenášené informace. Pak dále na PKS záleží, jak je konstruovaný na dimenzovaný pro opravu chybně přenosového úseku. Může například zopakovat přenos stejného úseku, ve kterém se vyskytla chyba nebo zvýšit zabezpečovací vůči průniku chyb do informační části kódu.



Obr. 1.3 Přenosový systém se zpětnou vazbou

1.2.2 PKS bez zpětné vazby

Tento systém neposkytuje zpětný kanál, a tak se využívá mechanismu zabezpečovacích kódů. V bloku PKS bez zpětné vazby se příchozí informace rozdělí na části a k nim se přidávají zabezpečovací informace, tím zvyšujeme redundanci a snižujeme efektivnost přenášené informace. Zabezpečovací data se v přijímači PKS používají na odhalení či opravy vzniklé chyby. Rozlišujeme, následují protichybový kódové systémy:

- **PKS s detekčním kódem**

  V přijímači PKS se vyhodnotí přenesená data a případné chybně přenesené úseky jsou určeny a jsou vyjmuty z celku přenášené informace

- **PKS s detekčním a korelačním kódem**

  Protichybový kódovací systém umí případné chyb jak odhalit tak i některé je schopen pomocí korekčního kódu opravit.

- **PKS se samoopravným kódem**

  Systém je schopen opravit chyby vzniklé během přenosu.

2 ROZDĚLENÍ ZABEZPEČOVACÍCH KÓDU

Zabezpečovací kódy můžeme rozdělit podle několika skupin. Podle způsobu zabezpečení rozeznáváme:

2.1 BLOKOVÉ KÓDY

Blokové kódy se vyznačují tím, že bitový tok rozdělí na stejně dlouhé úseky o délce *k*, která dále zabezpečí a tím se zvětší délka *n* kódovaného slova.

2.2 STROMOVÉ KÓDY

Stromové kódy jsou jinou kategorií zabezpečovacích kódu, kterou lze určit z definice tohoto kódu. Název kódu udává způsob kódovaní bitového toku, kdy se nejčastěji používá graf typu strom neboli taky označovaný T strom. Mezi stromové kódy bych zařadil konvoluční a mřížkové kódy

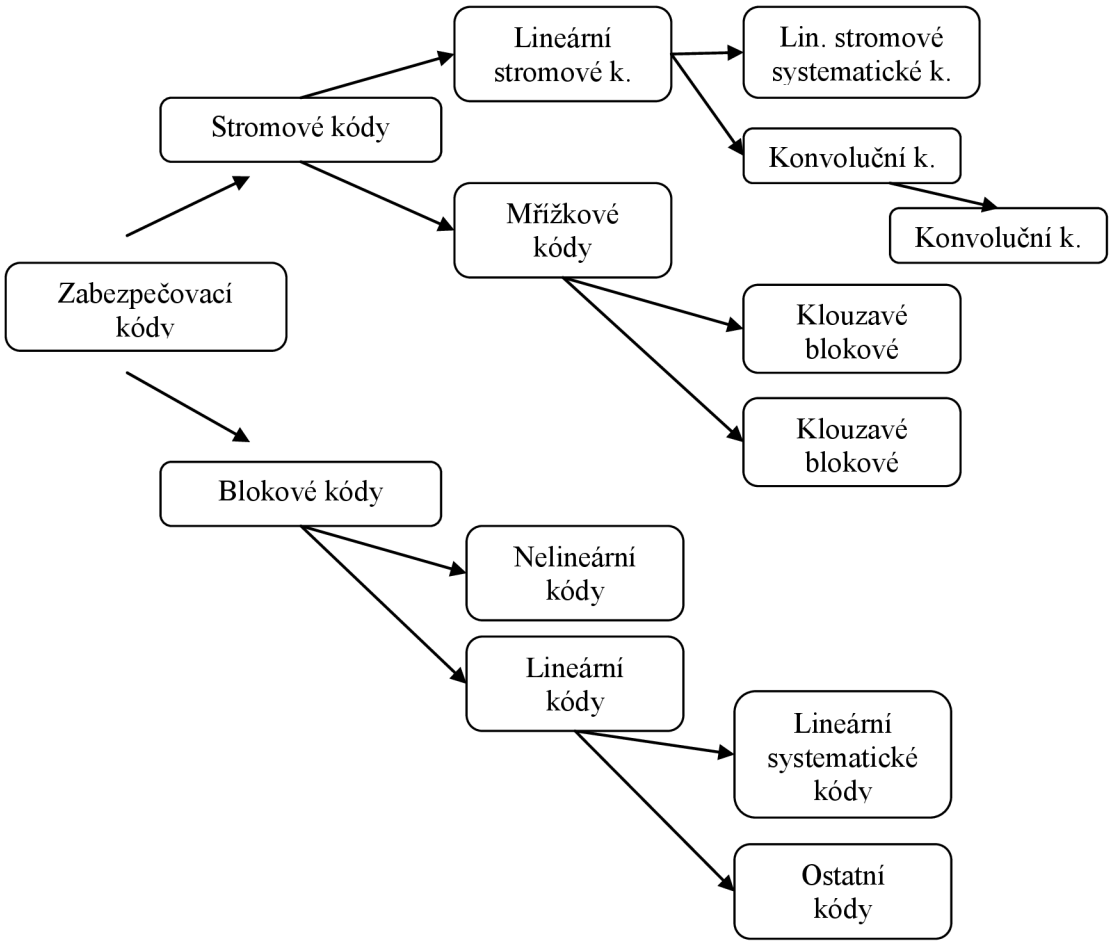
2.3 SYSTEMATICKÉ KÓDY

Podle způsobu zvětšováním nadbytečnosti v kódovaném slově rozlišujeme *systematické* a *nesystematické kódy*.

Systematické kódy rozdělí nezabezpečený bitový tok do bloků o délce k, a následně za něj připojí zabezpečovací kód o délce n, který navazuje na informační bity. V tomto kódu jsme schopni určit, jaké data představují informace a jaké zabezpečení. Proto je často označujeme jako kód (n; k).

2.4 NESYSTEMATICKÉ KÓDY

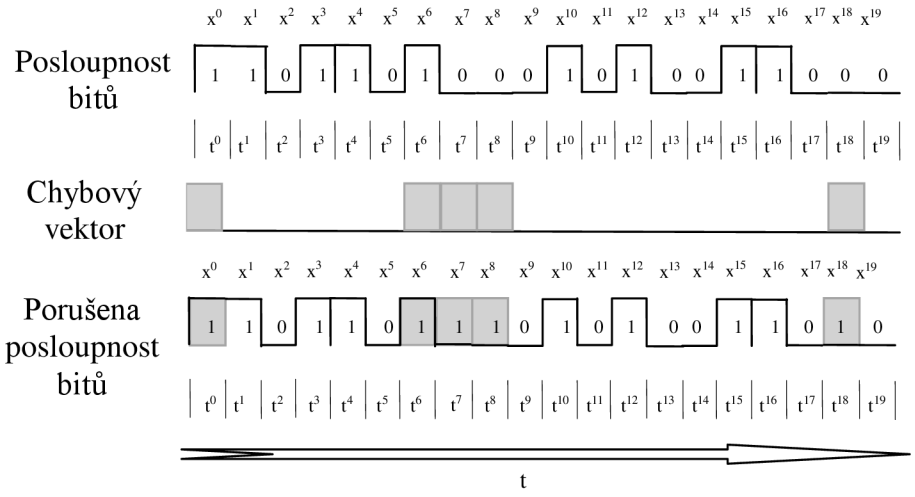
Nesystematické kódy se stávají z nezabezpečeného toku dat, ke kterým jsou přidávány nové úseky zabezpečovacího kódu, které jako všechny kódy zvyšují záměrně nadbytečnost. Přiřazovaní zabezpečení je určeno definicí daného kódu. Po zabezpečení tím kódem nejsme schopni zjistit, které bity jsou informační a které zabezpečovací



Obr. 2.4 Rozdělení zabezpečovacích kódů

3 VZNIK CHYBY

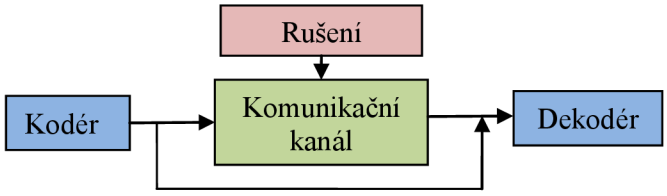
V průběhu přenosu po přenosovém médiu se nám na kanál naindukovalo rušení, které máme reprezentováno jako chybový vektor. Ten nám původní posloupnost změní ve stejných časových oblastech jako jsou signálové prvky bitového i chybového toku na výslednou porušenou bitovou posloupnost.



Obr. 3.5 Vliv chybného vektoru na posloupnost bitů

Jak je vidět na obrázku Obr. 3.5, tak v první části máme posloupnost bitového toku, který odesíláme pomocí komunikačního kanálu směrem k přijímači.

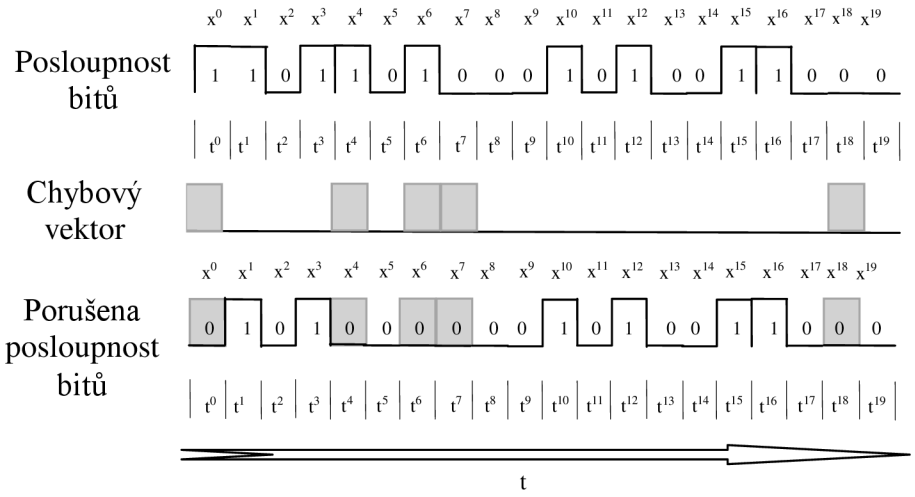
A pokud srovnáme první a třetí část tohoto obrázku, tak zjistíme, že přenesena data nejsou stejná a že se nám projevila chyba na komunikačním kanále. Pokud bychom měli ze zdroje informací ještě jeden kanál Obr. 3.6, který by nebyl ovlivněn rušením v komunikačním kanále, tak bychom byli schopni analyzovat rušení a přesně zjistit, kde proběhla změna hodnoty na opačnou a jaký byl mnohočlen rušení.



Obr. 3.6 Model rušení kanálů

Chybový mnohočlen můžeme zapsat jako posloupnost bitů s označením E(x), který obsahuje pouze signálové prvky, v kterým a=1 neboli kdy je změna z hodnoty 0 →1. V našem případě můžeme zapsat:

$$E(x) = 1 + x^7 + x^8 + x^{18} \tag{1}$$



Obr. 3.7 Porušení posloupnosti

Na tomto Obr. 3.7 vidíme porušení posloupnosti bitů kvůli změně hodnoty z 1 na hodnotu 0. Pro tento případ můžeme napsat chybový vektor, který je:

$$E(x) = 1 + x^4 + x^6 \tag{2}$$

Z předchozího textu jsme napsat, že změny v posloupnosti bitů nastávají jen tehdy, pokud se v určitý časový úsek t_n objeví chybový vektor, který v časovém intervalu t_n změní svoji hodnotu z 0 →1.

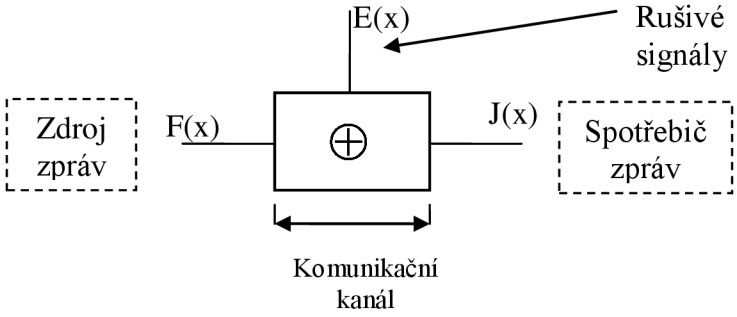
Chybový mnohočlen E(x) nastává tehdy, pokud ve stejný časový okamžik dojde ke změně hodnoty u před tím bezchybného bitového přenosu tzn. že chybový vektor se změní z 0 →1.

Ke změně bezchybné posloupnosti dochází v důsledku rušivých jevů, pomocí kterých dojde ke sečtení do té doby bezchybného přenosu F(x) a chybového přenosu E(x) stejných časových úseku a tím vzniká chybová posloupnost J(x). Pro sečítání binárních kódu platí následující pravděpodobnostní tabulka, na které je zřejmě, jak se mění výstup

dvou vstupních hodnot, pro nás tedy vstupy máme jako bezchybný přenos F(x) a E(x) (nebo taky zapsané F(x) ⊕ E(x)) a výstup je charakterizován jako posloupnost s chybami J(x). Sečítání vstupních hodnot označujeme symbolem ⊕, často taky označované součet modulo 2 nebo taky ve zkratce mod2.

Tab. 1 Sečítačka mod2

Posloupnost bitů F(x)	Chybové bity E(x)	Chybová posloupnost J(x)
0	0	0
1	0	1
0	1	1
1	1	0



Obr. 3.8 Blokové schéma kanálu sečítačky mod2

Jelikož už jsme si vysvětlili, jak vzniká chyba a jak se na kanál naindukuje, tak si nyní řekneme něco o chybách, jaké rozlišujeme. Jsou to *nezávislé chyby* a *shluky chyb*. Je třeba tyto chyby rozlišovat a podle toho používat vhodné zabezpečení.

Nezávislé chyby označujeme je písmenem t a jsou to chyby, které jsou způsobeny rušivými jevy na kanál. Rozeznáváme chyby jednoduché a vícenásobné. Jednoduché chyby způsobují v posloupnosti délky n bitů pouze jeden bit chybný. Naopak Vícenásobné chyby v posloupnosti n bitů způsobuji několik nezávislých chyb.

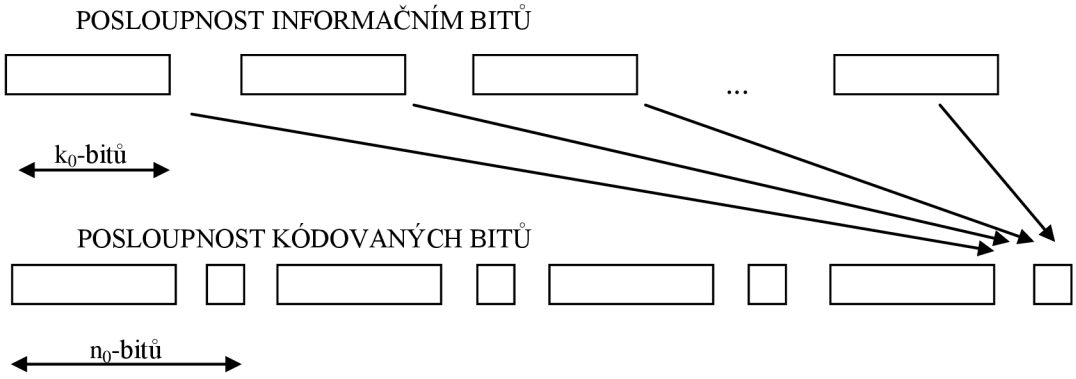
Shluky chyb označujeme písmenem *b*. Svoji četností chybně přenesených bitů výrazně převyšují četnost ve zbytku bitové zprávy.

4 KONVOLUČNÍ KÓDY

4.1 POPIS KÓDOVACÍ TECHNIKY

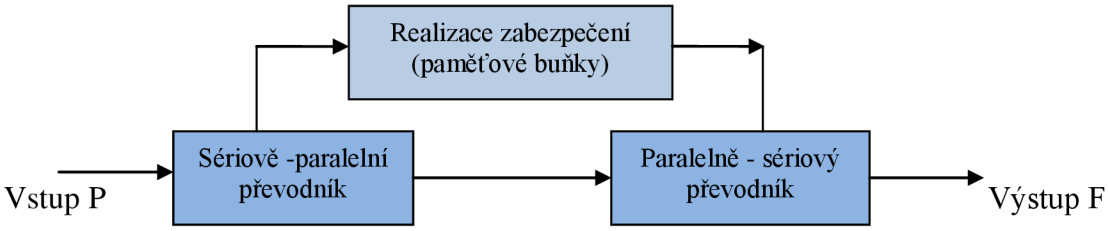
Zabezpečení konvolučního kódu probíhá ve vlastním kodéru. Příchozí bitová nezabezpečená posloupnost P(x) vstupuje do bloku sério – paralelního převodníku, kde se data převádí ze sériové posloupnosti na posloupnost paralelních větví, které následně vstupují do bloku vlastní kodéru. Tady se za pomocí paměťových buněk odvodí zabezpečovací prvek, který je v následujícím bloku přidružen k posloupnosti bitů. Tímto krokem docílíme zabezpečení posloupnosti proti chybám.

Samotné zabezpečení probíhá tak, že se rozdělí vstupní tok na délky k_0, které se vloží do vstupní paměti, která má velikost násobků úseku k_0,ze kterých se po naplnění paměti odvozuje zabezpečovací prvky, jak je vidět na Obr 4.13.



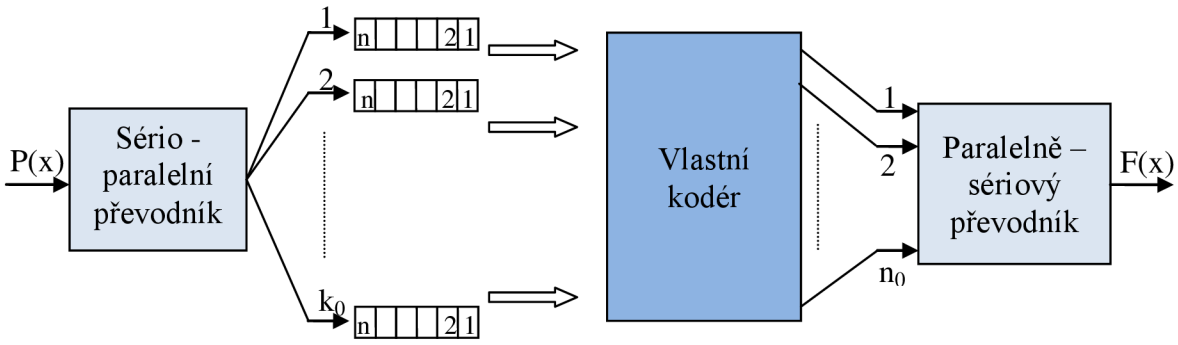
Obr. 4.9 Princip konvolučního kódování

Obecně si můžeme přestavit vlastní dekodér jako převodník kódu ze sériové podoby na podobu paralelní a opět zpět na sériová data. Během převodu je do botového toku vložen zabezpečovací kód, pomocí kterého jsme schopni odvodit, že během přenosu vznikla chyba.



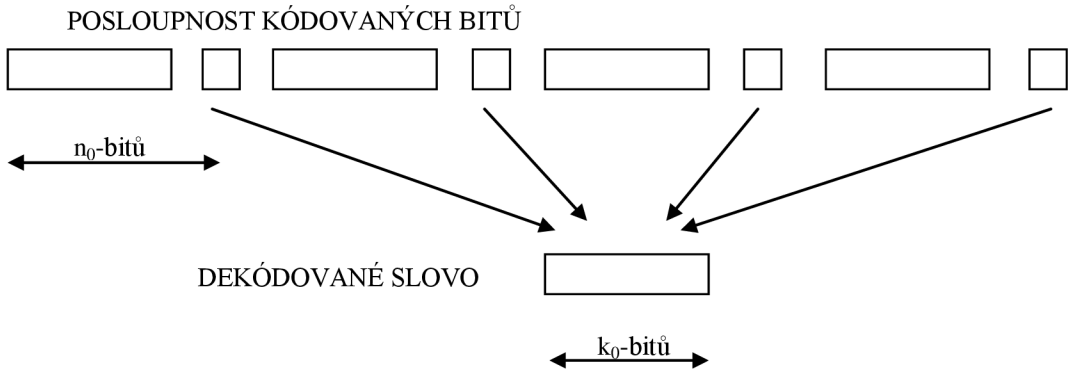
Obr. 4.10 Obecné blokové schéma konvolučního kodéru

Nejvýstižnější popis vlastního kodéru můžeme vidět na Obr. 4.9, kde je celkem jasně, jak se přenášená data větví ze sériového převodníku (demultiplexují) a jednotlivé kanály jsou přiváděny do vlastního kodéru, kde se změní počet výstupů (vytvoří ze další kanály), a opět se všechny větve v převodníku sloučí do jednoto toku (multiplexoru).



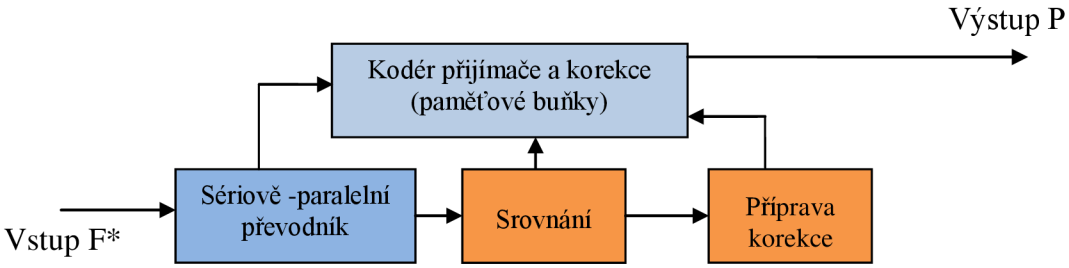
Obr. 4.11 Blokové schéma konvolučního kodéru

Obdobným způsobem pokračujeme při dekódování Obr. 4.12, se ze zabezpečených úseků o délce kódových slov n_0 oddělí zabezpečovací prvek, pomocí kterého jsou pak vytvořena dekódovaná slova o délkách k_0.



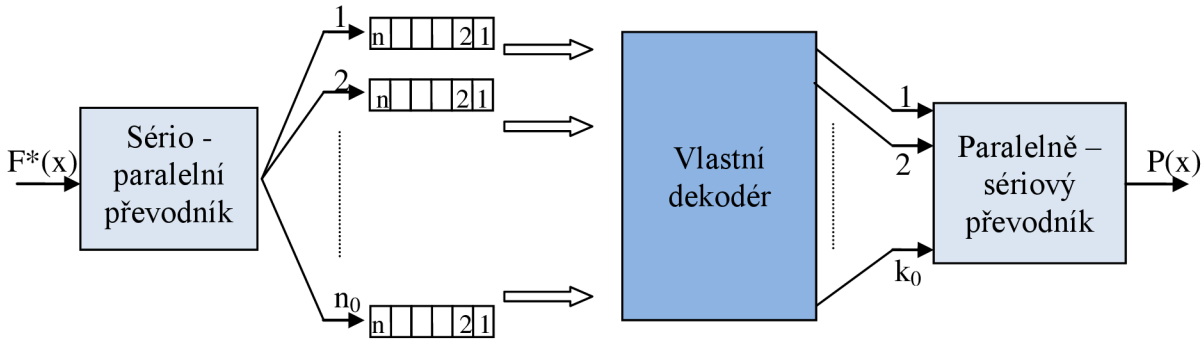
Obr. 4.12 Obecné blokové schéma vlastního dekodéru

Blokově si dekodér můžeme představit jako systém, skládající se ze samostatných částí, kde se v první části opět převedou data ze sériové podoby na paralelní, odtud jednak vstupují do vlastního kodérů, kde se odvodí zabezpečovací prvek a porovná se zabezpečovacím prvkem, který vytvořil vysílač. Následně se oba kódy porovnají a pokud jsou stejné, tak přenos proběhl v pořádku, pokud se liší, tak se provede korekce Obr. 4.13.



Obr. 4.13 Princip konvolučního dekódování

Opět nejvýstižněji je funkce ukázaná na Obr. , kde se převedou přenesené data na z paralelní do sériové podoby, následuje dekodér, kde se oddělí zabezpečovací kanály z toku a následně se data znovu zmultiplexují a vytvoří se bitová posloupnost dat.



Obr. 4.14 Blokové schéma konvolučního dekodéru

Do dekodéru přichází vlivem poruchy vedení již pozměněná posloupnost bitů F*(x).

Vlastní zabezpečení probíhá tak, že se rozdělí vstupní nezabezpečený tok na úseky o délce k_0, který následně vstupují do vstupní paměti a následně do zabezpečovací paměti. Ta je dimenzována m-násobky délky úseku k_0, kde se následně odvodí zabezpečovací bity pro daný úsek viz Obr. 4.14. Z výstupní paměti pak vychází již zabezpečený úsek délky n_0 bitů.

4.2 ZADÁNÍ KONVOLUČNÍHO KÓDU

Konvoluční kód může být zadán dvojím způsobem. Je zapotřebí vyjádřit vztah mezi vstupním a výstupním tokem neboli mezi nezabezpečeným a zabezpečeným bitovým tokem.

4.2.1 Zadání kódu pomocí vytvářecí matice

Vstupní tok zapíšeme jako řádky do matice [P] a do matice [F] zapíšeme po řádcích výstupní tok. Pomocí [G_∞] vytvářecí matice konvolučního kódu, můžeme zapsat vztah. Matice [G_∞] představuje polonekonečnou matice, která je z levé strany ohraničená začátkem kódováním (t = 0), a z druhé strany je ohraničená délkou kódované zprávy $[F] = [P]x[G_\infty]$.

Vytvářecí matice [G_∞] je složena z dílčích vytvářecích matic [G_t], kde rozměr matice je [k x n] neboli počtem řádků *k* a *n* sloupci, za pomocí kterých je možné přepočítat sloupce matice [P] na matici [F] ve stejných časových okamžicích[2].

$$[G_\infty] = \begin{bmatrix} [G_0],[G_1],[G_2], & \cdots & [G_m], & \cdots & \cdots \\ \vdots & [G_0],[G_1],[G_2], & \cdots & [G_m], & \cdots \\ \vdots & \vdots & [G_0],[G_1], & [G_2], & \cdots \\ \vdots & \vdots & \vdots & \downarrow & \vdots \\ \vdots & \vdots & \vdots & \vdots & \downarrow \\ \vdots & \vdots & \vdots & \vdots & [G_0], \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{matrix} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ [G_1],\cdots \\ [G_0],\cdots \end{matrix} \tag{3}$$

Z rovnice (4) vyplývá], že pro popis vstupního bitového toku pomocí nám postačí znát uspořádání matice [G_t] v časových okamžicích pro t = 0, 1, 2 …m, pro bližší informace lit. [2].

$$[G_t] = \begin{bmatrix} g_{(1)}^{(1)}D^t; & g_{(2)}^{(1)}D^t; & \cdots & g_{(n)}^{(1)}D^t; \\ g_{(1)}^{(2)}D^t; \vdots & g_{(2)}^{(2)}D^t; & \cdots & \cdots \\ \vdots & \vdots & \vdots & \cdots \\ g_{(1)}^{(k)}D^t; & g_{(2)}^{(k)}D^t; & \cdots & \cdots \end{bmatrix} \tag{4}$$

4.2.2 Zadání kódu pomocí mnohočlenu matice

Signálové prvky při průchodu posuvným registrem kodéru konvolučního kódu jsou zpožděny a vyjadřujeme je jako operátor zpoždění D za předpokladu, že všechny prvky mají stejnou délku D. Signálové prvky představuje převážně binární signál, který nabývá pouze dvou hodnot (1 a 0). Pokud si stanovíme v čase t_0 nějaký signálový prvek p0, tak za ním následuje v čase t_1 prvek $p_1.D$, dále pak v t_2 prvek $p_2.D^2$, kde můžeme tuto posloupnost vyjádřit jako $p_x.D^x$.
Můžeme tuto posloupnost vstupních signálových prvků vyjádřit vztahem [2].

$$p^{(j)}(D) = p_0^{(j)} + p_1^{(j)} + p_2^{(j)}D^2 + \cdots + p_x^{(j)}D^x \tag{5}$$

Pro vyjádření výstupní posloupnosti signálových prvků pomocí operátoru zpoždění D můžeme napsat[2].

$$F^{(i)}(D) = f_0^{(i)} + f_1^{(i)} + f_2^{(i)}D^2 + \cdots + f_x^{(j)}D^x \tag{6}$$

Pro vyjádření závislosti mezi vstupním a výstupním signálovým tokem platí vztah [2].

$$F^{(i)}(D) = \sum_{j=l}^{k} G_{(i)}^{(j)}(D).p^{(j)}.(D) \tag{7}$$

Pro vytvářecí mnohočlen konvolučního kódu $G_{(i)}^{(j)}$ platí [2].

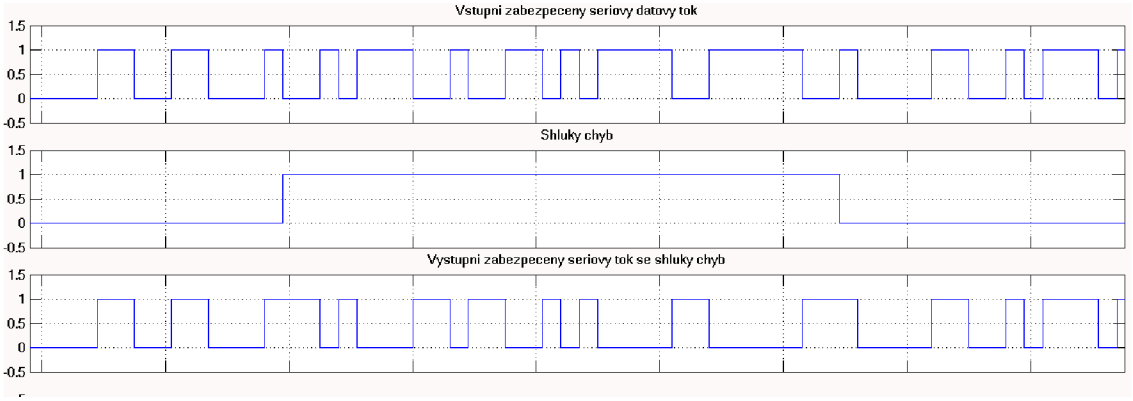
$$G_{(i)}^{(j)}(D) = \sum_{n=1}^{m} g_{(i)}^{(j)}\, u.D^n \tag{8}$$

Při výběru vhodného kódu pro přenos posloupnosti bitů, je zapotřebí, aby byl schopný dostatečně zabezpečit přenášenou informaci. Konvoluční kódy jsou vhodným řešení, protože jsou snadno obvodově řešitelné, a proto můžeme snadno nasimulovat jejich funkci. Zabezpečovací bity jsou vytvářeny vzhledem k informačnímu toku jak u právě přenášeného, tak i u předchozích přenesených bloků.

5 PROKLÁDÁNÍ ZPRÁVY

5.1 SHLUK CHYB

V dnešní době se každoročně velmi rychle zvyšují objemy přenesených dat. To má za následek, že potřebujeme přenášet velmi rychle informace s minimálním časem kladeným na přenos těchto dat mezi vysílačem a přijímačem. Bohužel se začínáme potýkat s chybami, které se shlukují do jednoho celku (nejedná se o jednotlivé ani vícenásobné chyb), které nám značně ovlivňují negativně přenesená data, na která nám už bohužel nestačí běžné zabezpečovací mechanismy. Těmto chybám, které se shlukují do celku a razantně navyšují chybovost spojení říkáme *shluky chyb* Obr. 5.15. Pro běžné zabezpečení se stále častěji používají složitější a složitější zabezpečovací techniky, což má za následek stále náročnější systémy, které potřebují velmi výkonnou techniku na výpočetní operace. A stejně s náročnými zabezpečovacími kódy roste délka zabezpečení, tím rapidně narůstá nadbytečnost kódovaného slova a klesá efektivnost přenosu užitečné informace. Shluky chyb řadíme do závislých chyb, a tudíž je můžeme za různých předpokladů potlačovat. Jedním z nich, je rozprostření signálových prvků nezabezpečených úseků v časové oblasti, kterých můžeme dosáhnout pomocí několika technických postupů.



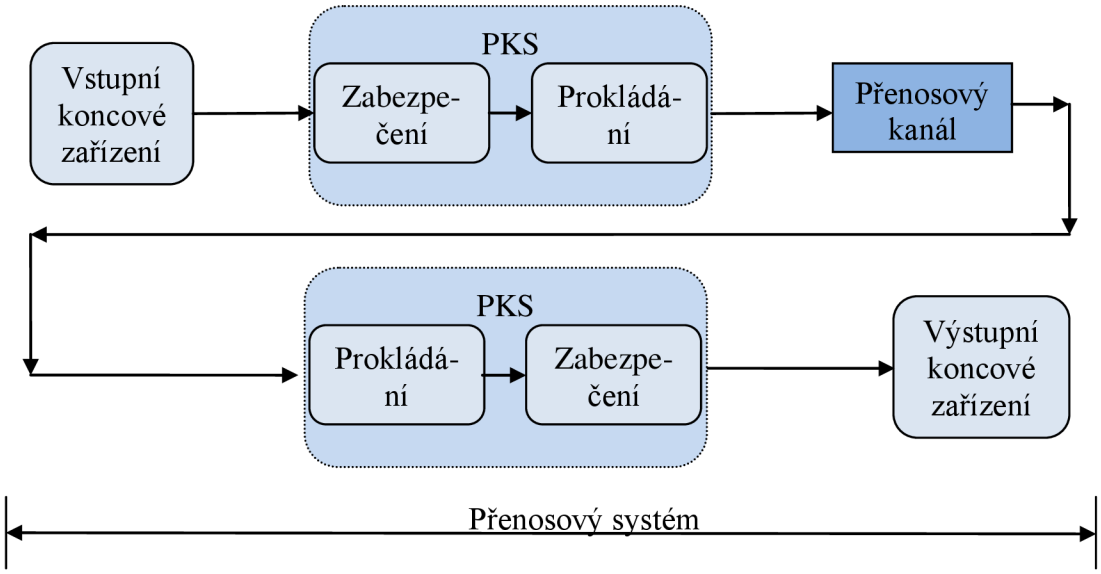
Obr. 5.15 Shluky chyb

Shluky chyb můžeme velmi výrazně potlačit pomocí prokládání zprávy (interleaving), výběrem vhodného konvolučního kódu anebo cyklickými kódy.

5.2 POPIS PROKLÁDÁNÍ

Metoda prokládání zprávy taky často nazývaná jako Interleaving vychází z požadavku rozdělit shluk chyby na příslušný snesitelný počet chyb, které jsme schopni opravit pomocí zabezpečovacích kódů. Děje se to tak, že změníme polohy bitového toků tak, aby se charakter distribuce shluku chyb změnila na distribuci nezávislých chyb, které umíme účinně odstranit či maximálně potlačit.

Prokládání zprávy je systém, který vložíme ve vysílači mezi kodér a přenosové médium a na straně příjemce mezi přenosové vedení a dekodér Obr. 5.16.



Obr. 5.16 Umístění prokládání v PKS

Vycházíme z předpokladu dvoustavového signálu, který je reprezentovaný bitovou posloupností. Zjednodušeně můžeme říct, že prokládání probíhá tak, že vytvoříme matici (vstupní paměť), do které zapisujeme data po řádcích, a jak naplníme celou matici, tak provedeme čtení z matice, ale nikoliv po řádcích, ale po sloupcích. To stejné uděláme v přijímači, kde zapisujeme data do matice po sloupcích a čteme z ní po řádcích. V závislosti na počtu řádků a sloupců matice případně zpoždění docílíme efektu, kdy provedeme změnu časové polohy bitového toku a tím potlačíme shluky chyb. Pomocí prokládání však zvyšujeme zpoždění celého systému.

Zjednodušeně řečeno interleaving přeskládá podle definovaného algoritmu přenášená data a tím rozdělí shuky na malý počet chyb. Interleaving je založený na principu, rozdělení shluku chyb na přijatelný počet, který je následně schopen opravit zabezpečující kód. Z toho vyplývá význam prokládání, je zapotřebí najít flexibilní

prokládací algoritmus, který bude pružný pro široké spektrum použití proti shluku chyb. Při použití prokládání je použit stejný metoda prokládání jak ve vysílači, tak i v přijímači podobně je tomu u kodéru a dekodéru při zabezpečení.

Mezi hlavní parametry prokládání patří:

- *Hloubka prokládání* - čím je větší, tím může být shluková chyba rozprostřena na ojedinělé chyby
- *Rámec vnějšího kódu* – počet bitů, po kterých se bude opakovat ojedinělá chyba
- Možnost prokládat i po kódových slovech např. bytech
- Použití vnitřních a vnějších prokládacích procesů v rámci jednoho zabezpečení (vnitřní na úrovni jednoho bitu, vnější na úrovni bytů)

5.2.1 Blokové prokládání

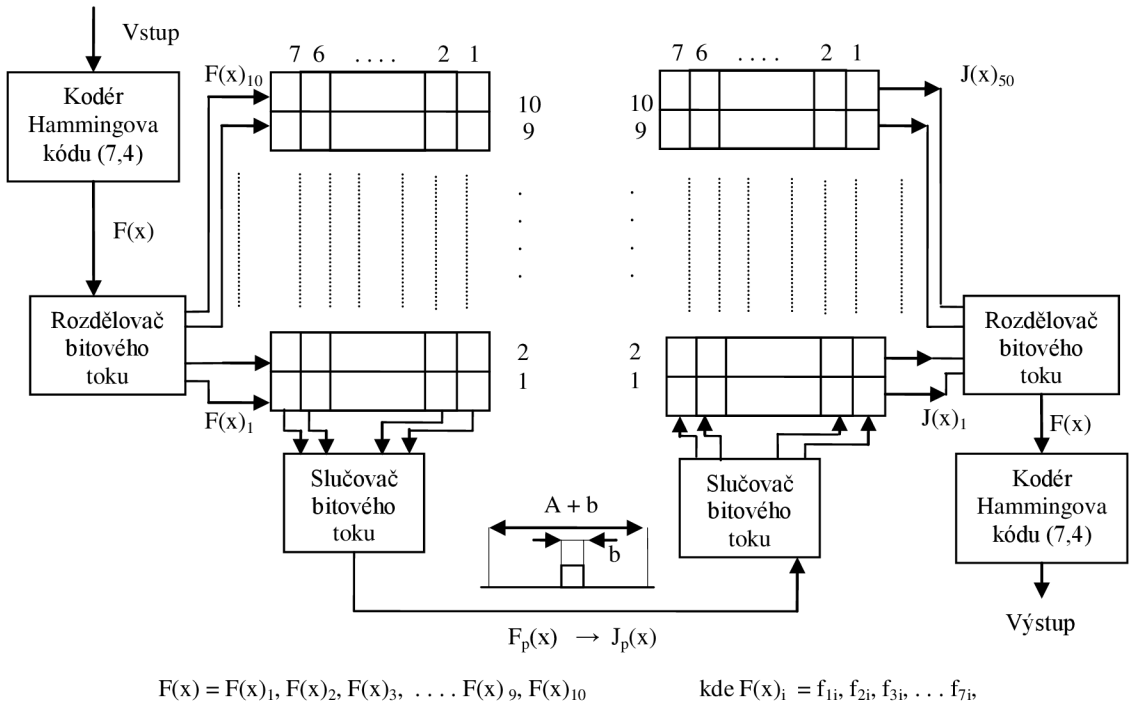
Prokládání bitů ve spojení s korelačními blokovými kódy provádíme pomocí rozdělení chybných bytů, kterými jsou ve shlucích chyb pro definovaný počet kódovaných kombinací. Důsledkem teda je, že počet chyb v jednom kódovém slově je schopný opravit blokový kód.

Na Obr. 4.17 si vysvětlíme činnost prokládání bitů. Na vstup kodéru Hammingova kódu (7,4) vstupuje nezabezpečený tok bitů, kde se k ním přidá (n - k) zabezpečovacích bitů, v našem případě r = 3 bity, aby v dekodéru bylo možné opravit až t chyb (1 chybu). Blok obsahuje paměť (matici) o rozměrech n x j (řádky x sloupce) v našem případě 10 řádku a 7 sloupců, do kterých se postupně ukládají jednotlivé kombinace kódových slov, které se zapisují od spodu do jednotlivých řádků dokud nenaplníme celou paměť. Jakmile ji naplníme, tak musíme přestat plnit paměť, tzn. ukončíme zápis do pamětí a ve stejný okamžik začínáme z paměti číst, ale nikoliv po řádcích, jak byl zápis, ale po sloupcích. Což znamená, že jsme zapisovali po 7dmi znacích do každého řádku a nyní čteme 10 znaků z každého sloupce z paměti. Takže zabezpečený tok vstupuje do paměti po jednotlivých mnohočlenech zprávy $F(x)_i = f_{1i}, f_{2i}, f_{3i}, \ldots f_{7i}$, pro $F(x) = F(x)_1, F(x)_2, F(x)_3, \ldots . F(x)_9, F(x)_{10}$. Tedy od $F(x)_1$ až po $F(x)_{10}$. Takto je proveden zápis.

A pro čtení z paměti použijeme čtení z $F_P(x) = f_{1,1}$, $f_{1,2}$, ..., $f_{1,10}$; $f_{2,1}$, $f_{2,2}$, ..., $f_{2,10}$, $f_{7,1}$, $f_{7,2}$, ..., $f_{7,10}$ neboli první hodnotu z paměti vezmeme z první sloupce, první řádek, následuje první sloupec druhý řádek až přečteme všechny a poslední čtecí hodnota z matice bude sedmí sloupec, sedmí řádek. Tím skončí čtení z pamětí, paměť se vynuluje a můžeme do ní znovu zapisovat. Ten samý mechanismus je použit u přijímače, kde se data prvně nahrají postupně do jednotlivých sloupců, a jak naplníme celou matici, provedeme čtení po řádcích. Po dekódování se korelační kód postará o případné vzniklé chyby, které opraví. Při původní kombinaci pro každé dva byty jsou odděleny (j - 1) bity ostatních kódových kombinací. Pokud je bitový tok napaden shluky chyb, je vzhledem k použití korelačního kódu napadeno všech j kombinací, ale j-krát menším počtem chyb.
Hlavními parametry prokládaní bitů je (n; k) a tzv. prokládací faktor j, který určuje počet kódových kombinací, pro které mají být kódové slova proloženy.
Zpoždění vyplívá z prokládání bitů jsme schopni odvodit pomocí vzorce:

$$Dt = 2.j.n \tag{59}$$

Abychom zpoždění co nejvíce minimalizovali, tak při prokládání bitů je zapotřebí použít paralelní pamětí z toho důvodů, když naplníme matici a má probíhat čtení z matice na straně vysílače či na straně přijímače, tak abychom nemuseli zastavit tok dat a počkat, až se nám vynulují jednotlivé matice. Proto je z technických důvodů lepší zajistit větší kapacitu pamětí, když se jedna matice naplní, tak aby hned převzala další matice funkci na zápis a obráceně.



Obr. 5.17 Schéma PKS s prokládáním bitů

Pro určení paměťového pole matice je potřeba znát několik parametrů, pomocí kterých jsme schopni pole dopočítat. Mezi základní parametry patří minimální délka bezchybného intervalu A [bit] mezi jednotlivými shluky chyb a dále délka shluku chyb b [bit].

Z toho vypočítáme počet řádků matice:

$$j \geq \frac{b}{t} \tag{60}$$

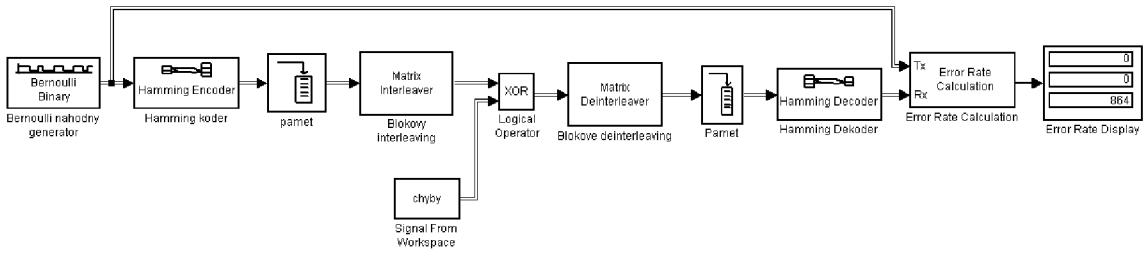
kde t označuje počet nezávislých chyb opravovaných kódem

Pro počet sloupců n hledáme takový kód, který koriguje t chyb v kódové kombinaci délky *n* bit

$$n \leq \frac{A+b}{j} \tag{61}$$

Realizace v matlabu, kde máme na jedné straně generátor binárního signálu s určitou pravděpodobností, kterou si vhodně nastavíme. Následuje blok Hammingův kodér, který jak bylo řečeno, zakóduje nezabezpečený tok o délce *(n – k),* který představuje posloupnost čtyř bitů a přidá k nim tři zabezpečovací bity. Tím se docílí kódu, který je schopen opravit v kódové kombinaci jednu vzniklou chybu. Poté data vstupují do paměti, která představuje blok matice, kam se zapíší jednotlivé řádky ve

vysílači, a po naplnění paměti se provede čtení z matice po řádcích. V bloku XOR se na vedení vloženy shluky chyb, které generujeme pomocí skriptu v programu v Matlabu. Následně provedeme zápis po sloupcích do paměti a hned na to čtení z matice po řádcích.



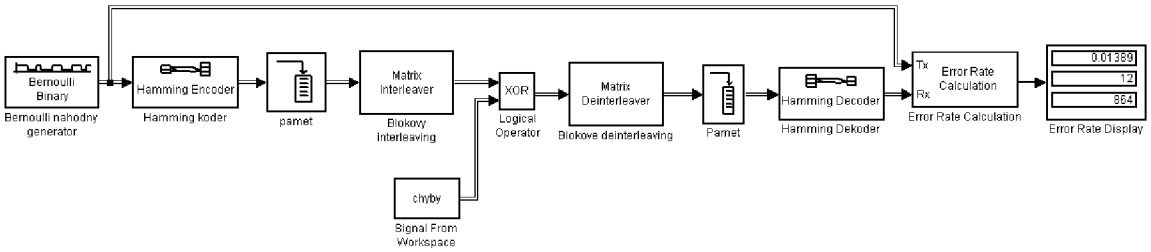
Obr. 5.18 Správně nastavený přenosový systém

Stanovil jsem si podmínku, že budu opravovat shluk chyb délky b = 10 bit. Jelikož zmiňovaný kód opravuje jen jednu chybu, tak je jasné, že shluk chyb musím proložit tak, abych ze shluku chyb 10-ti chyb dostal jen rozdělené chyby. Zvolíme se tedy interval 10 kódových slov, za kterých se může na vedení dostat shluk chyby o délce 10 bitů. Podle rovnice (60 a 61) si vypočítáme paměťový prostor pro matici, máme 7 bitové kódové slovo, shluk chyb je 10 bitů, z toho vyplývá, že budeme potřebovat matici o rozměrech 7x 10 (řádky x sloupce). Po nastavení dílčích bloku vytvoříme skript, pomocí kterého budeme zadávat vstupní hodnoty o shluku chyb. Skript je následovný:

```
errors=zeros(1,10^3);  %° pomocí kterého vytvoříme řadu číslem od nuly do
                       % tisíce, a tu naplníme nulami
n = 1+70*(0:15)        %vypočteme násobky čísla 70, definujeme tak ochranný
errors(n )=[1]         % interval A

errors(n+1 )=[1]
errors(n+2 )=[1]       % na násobky 70, přičteme  intervalu n až n+9, čímž
errors(n+3 )=[1]       % způsobíme shluk chyby
errors(n+4 )=[1]
errors(n+5 )=[1]
errors(n+6 )=[1]
errors(n+7 )=[1]
errors(n+8 )=[1]
errors(n+9 )=[1]
```

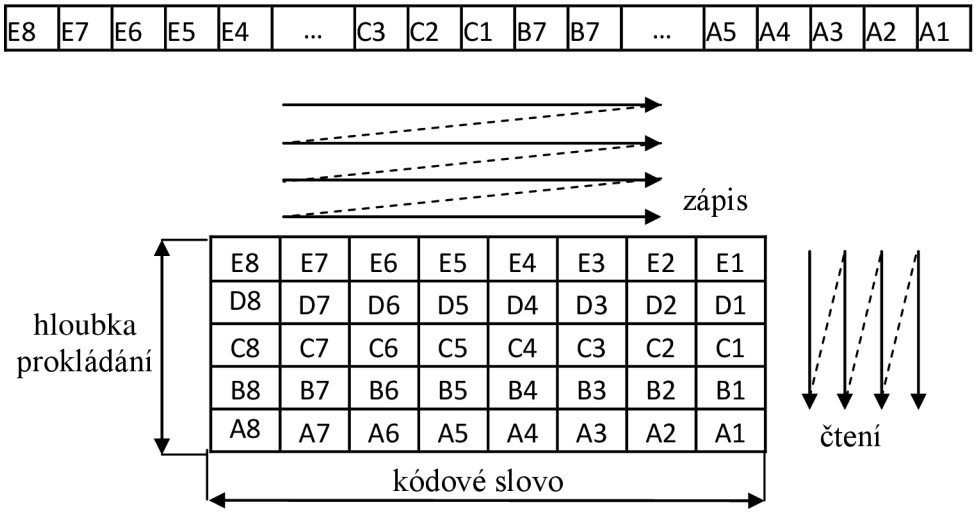
Poté spustíme samotnou simulaci a na Obr. 5.18 vidíme, že simulace proběhla v pořádku. První řádek zobrazuje v procentech poměr přenesených informací a chyb. Druhý řádek zobrazuje počet chybně během přenosu a třetí řádek zobrazuje správně přenesená data. Pokud si nastavíme na generátoru shluk chyb, do skriptu přidáme errors(n+10)=[1], tak na Obr. uvidíme následující.

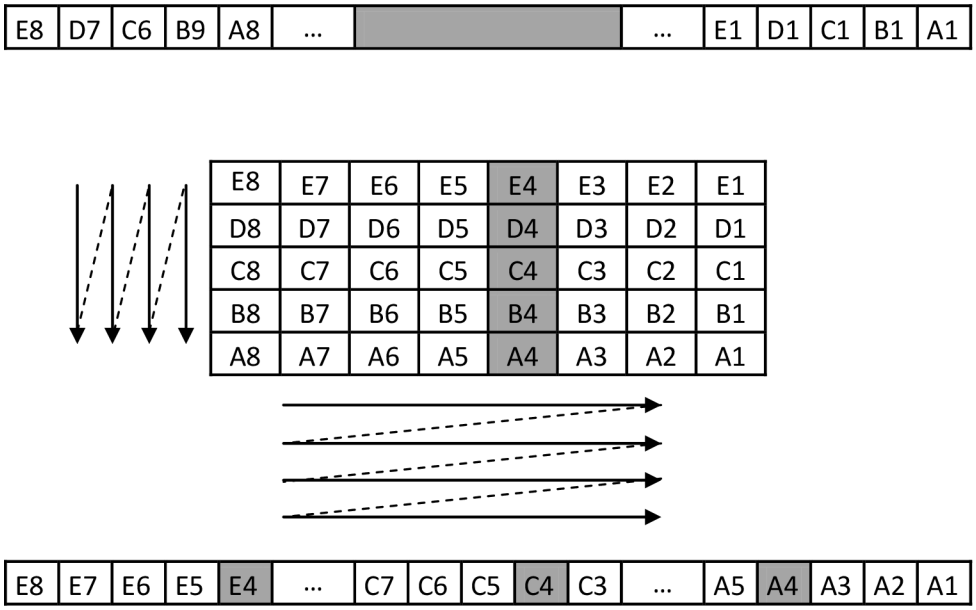


Obr. 5.19 Špatně nastavený přenosový systém

Z Obr. 5.19 je zřejmé, že se během přenosu 1000 bitů vyskytlo 12 chyb.

Funkci prokládání zpráv si můžeme jednoduše vysvětlit na následujícím Obr. 5.20. Budeme mít prokládací matici s počtem sloupců 8 bitů neboli jeden byte. A počtem řádků je roven číslu 5. Bitový tok zapíšeme, jak již bylo řečeno po řádcích do matice a jakmile matici naplníme, tak ji začneme číst po řádcích. Tím docílíme přerušením vazeb mezi kódovanými slovy a přenášíme bity v posloupnosti A1, B1…A2, B2…Při vzniku chyb, jak je ukázáno v prostřední části obrázku, se změní hodnoty přenášených bitů na opačné hodnoty, ale vlivem proložení zprávy v přijímači tento shluk rozdělí na jednotky chyby, na každý řádek, a ten jsme schopni opravit.

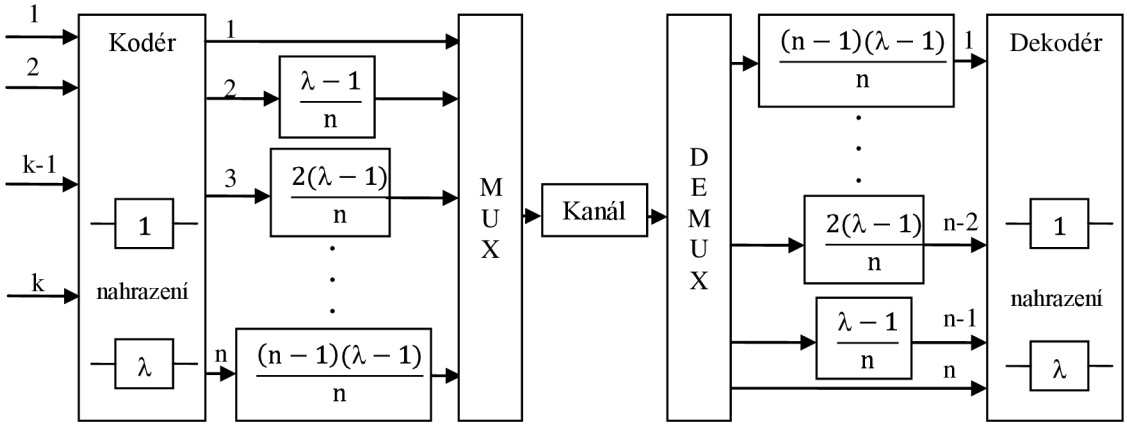




Obr. 5.20 Blokové prokládání

5.1.2. Konvoluční metoda prokládání

U konvolučního prokládání rozděluje tok délky *n*, které se následně ukládají do paměťových buněk konvolučního prokladače. Po naplnění všech větví *n* prokladače se vrací na začátek a v tu chvíli dochází k přímému propojení mezi vstupem a výstupem, což má za důsledek proměnlivého zpoždění různých značek při nulovém zpoždění mezi vstupem a výstupem.

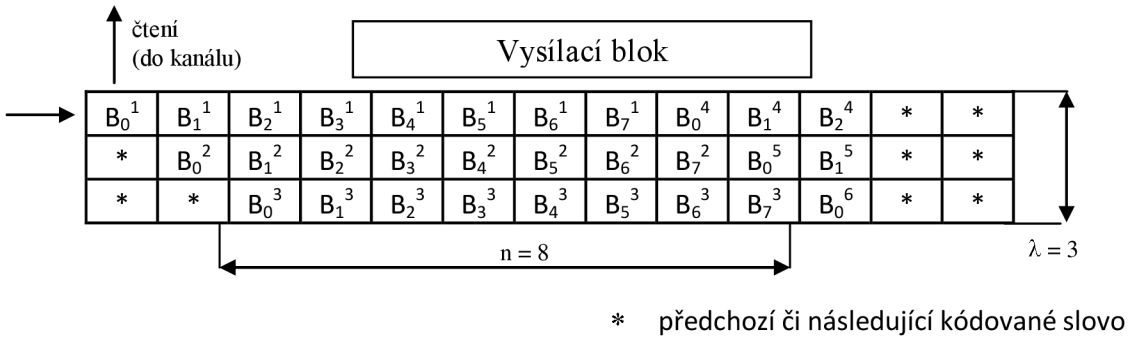


Obr. 5.21 Konvoluční kódovací systém s prokládáním

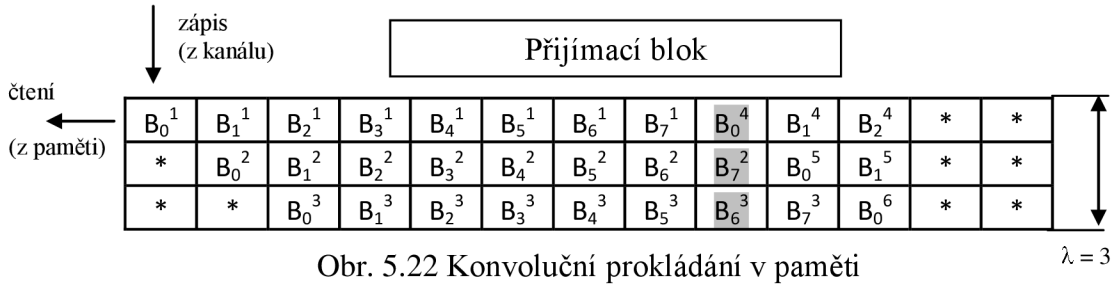
Pro zpoždění konvolučního prokládání platí [9].

$$\frac{\lambda-1}{n} + 2\frac{\lambda-1}{n} + \cdots + (n-1)\frac{\lambda-1}{n} = \frac{(\lambda-1).(n-1)}{2} \tag{62}$$

Představou konvolučního prokládání je multiplexovat výstupy vycházející kodéru, kde se využívá zpoždění každého následujícího vstupu o $\frac{\lambda-1}{n}$, kde λ představuje hloubku prokládání a n délka prokládání. Funkčnost konvolučního kódování je možné pochopit z následujících dvou Obr. 5.21 a Obr. 5.22.



Přenos po vedení se shluky chyb



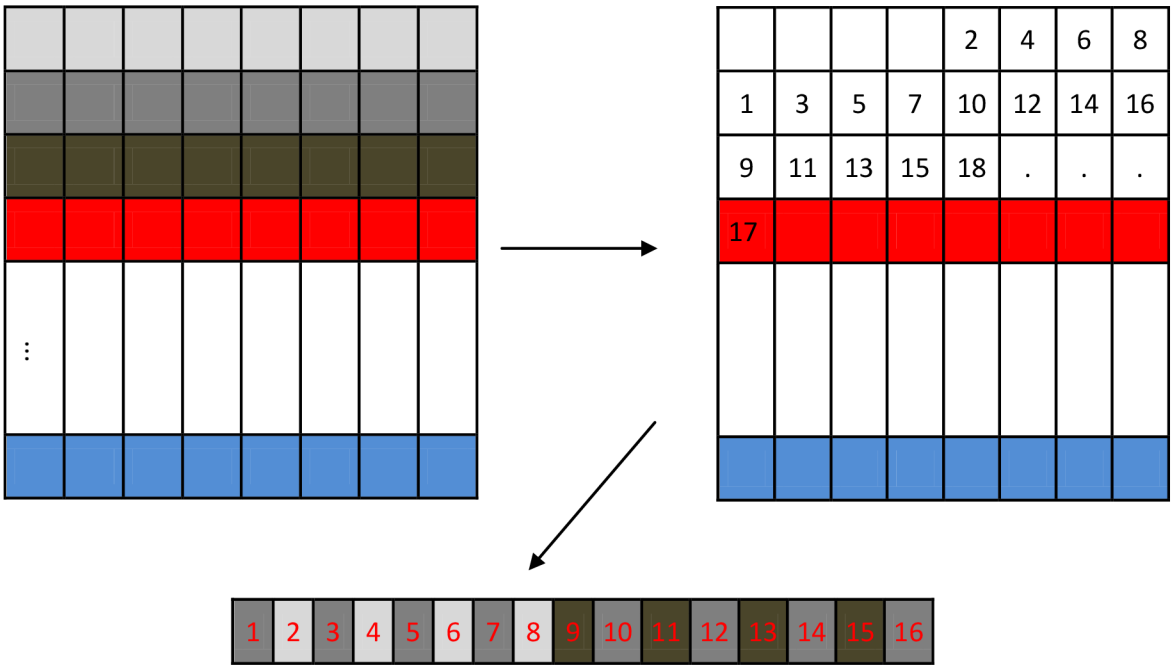
Obr. 5.22 Konvoluční prokládání v paměti

písmenem n označujeme délku kódové kombinace a λ udává hloubku prokládání

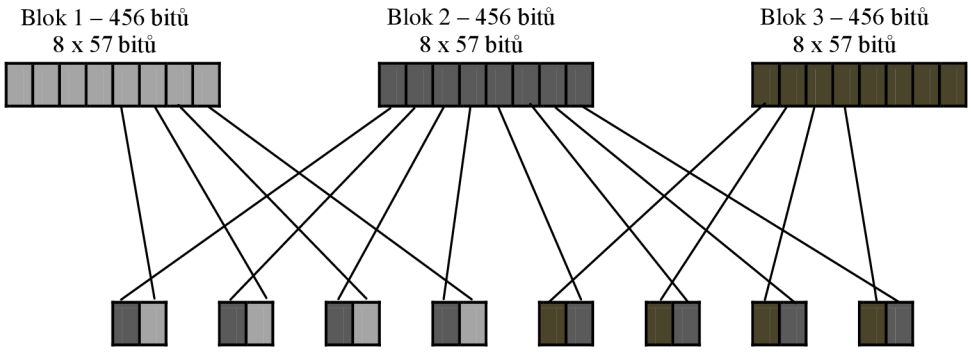
5.2.3 Diagonální prokládání

Používá se například v sítí GSM, kdy je zapotřebí co největší odolnost proti shluky chyb. Tady je bitový tok o délce 456 bitů rozdělen do 8 skupin po 57 bitech. Poté jsou pomocí diagonální metody proloženy s posledními čtyřmi skupinami z předchozího

bloku a prvními čtyřmi skupinami následujícího bloku. Tak se vytvoří blok dvou skupin o délce 57 bitů ale ne už jediného ale dvou bloků viz Obr. 5.23.



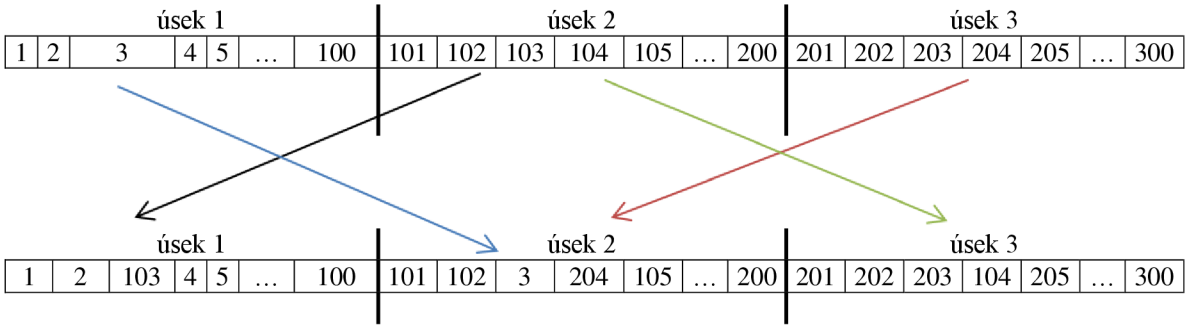
Obr. 5.23 Matice diagonálního prokládání



Obr. 5.24 Diagonální prokládání

5.2.4 Inter – blokové prokládání

Inter-blokové prokládání také nazývané z anglického názvu Inter-block interleaving (zkratka IBP). Tento algoritmus prokládání rozdělí dlouhou sekvenci bitového toku na kratší úseky, které následně prokládá mezi sebou. Funkce IBP prokládání je možné pochopit z Obr. 5.25, kde si rozdělíme datový tok na 3 úseky, každý o délce 100 paměťových míst (bitů) a pak podle algoritmu prokládá části druhého úseku s úsekem prvním a třetím (každý úsek s těmi zbývajícími). Takto dochází k rozložení chyb velmi dlouhých chyb.



Obr. 5.25 Inter-blokové prokládání

6 VÝBĚR TECHNIKY

6.1 Konvoluční kódy proti shluku chyb

Hagelbargův kód je nejstarší z těchto tří popisovaných kódu pro opravu shluku chyb. Pak byl objeven pomocí dvou vědců, kteří objevili zároveň kód Iwadari – Masery, také se podle nich jmenuje. Tento kód je efektivnější, než Hagelbargův kód. A nejmladší z těchto kódů je kód objevený Berlekampem - Preparem, který je z těchto kódů nejefektivnější ve všech směrech (ochranný interval, zpoždění paměťové buňky)

6.2.1 Hagelbargerův kód

Rozměr Hagelbargerova kódu je (n_0, n_0 - 1), který vytvoříme přepsáním blokové matice B_D v dekadickém tvaru do binární matice B_0. Matice je čtvercového typu n_0 x n_0, kterou číslujeme zespodu směrem nahoru. Dekadická čísla zapisujeme po řádcích na vedlejší diagonálu z leva doprava a směrem nahoru po matici. Pokud máme matici o rozměrech 4 x 4 (n_0 x n_0) viz (10). Tak v prvním řádku je zapsané číslo tři, v druhém řádku $n_0 - 1$ je zapsané číslo pět, v předposledním $n_0 - 2$ je sedm a posledním řádku je číslo jednička. Hagelbargerův kód opravuju shluk chyby délky *b* bitů.

$$b \leq n_0 \tag{9}$$

$$B_D = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 7 & 0 \\ 0 & 5 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{bmatrix} \rightarrow B_0 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Přepsání blokové matice v dekadickém tvaru B_D do dvojkového tvaru B_0 realizujeme tak, že dvojková matice bude mít stejný počet sloupců n_0 jako matice B_D,

počet řádků je dán součinem n_0 a počtem potřebných míst k zapsání největšího čísla v B_D. Počet míst dekadického čísla *n* potřebných k zapsání dvojkových číslem, označované také L(n), které vyjadřuje nejmenší celé číslo [11].

$$L(n) \geq 1 + log_2 n \tag{11}$$

Dvojková čísla s nejmenším počtem míst v matici B_0 mají na nepoužívaných místech nuly. První číslo odspodu má vždycky jen dvě dva řádky a je to číslo tři v binární podobě. Nepoužívané pozice v prvním sloupci jsou obsazeny nulovými hodnotami, protože by zbytečně způsobovali v realizaci kodéru časové zpoždění. Naopak poslední řádek obsahuje číslo jedna dekadickém tvaru.
Pro výpočet počtu řádků v matici použijeme vzoreček [11].

$$(n_0 - 1) * L(n) + 2 \tag{12}$$

Jak již bylo zmíněné, tak Hagelbargerův kód opravuje shuky chyb do délky *b* bitů, mezi kterými musí být tzv. *ochranný interval A :*

$$A \geq n_0{}^2 * L(n) - 1 \tag{13}$$

Pomocí vytvářecí dvojkové blokové matice jsme schopni odvodit zapojení jak kodéru, tak i dekodéru pomocí vytvářecích mnohočlenů. Vstupním tokem kodéru jsou první tři sloupce matice a výstupním tokem bude poslední čtvrtý sloupec matice B_0. Dekódování vychází ze znalosti obecného schématu dekodéru se syndromovým dekódováním, kde využíváme přímého převodu vektoru syndromu na chybový vektor. Po přenosu se v dekodéru oddělí zabezpečovací prvek, který se sečte mod2 (XOR funkce) s vypočítaným zabezpečovacím prvkem v dekodéru. Tak vznikne syndromový prvek *s*, pomocí kterého jsme schopni následně najít a opravit vzniklou chybu při přenosu.
Při realizaci tohoto zapojení potřebujeme taky znát údaje, jak budu přenášená bitová posloupnost prvků zpožděna, počet paměťových modulů atd. Hagelbargerův kód způsobuje zpoždění prvků převážně při průchodu paměťovými prvky. V kodéru se vytváří zabezpečovací prvky průběžně a případně paměťové buňky slouží k tomuto účelu. V kodéru tak nenastává žádné zpoždění bitového toku. V dekodéru se již některé paměťové buňky používají k úschově přenášených bitů k případně korekci chybně

přenesených bitů, a proto tu vzniká zpoždění bitového toku. Pomocí následujících vzorečků jsme schopni vypočítat zpoždění přenosového systému platí:

$$Z_K = 0$$
$$Z_D = L(n) * n_0 - 1$$
$$Z = Z_K + Z_D = Z_D = L(n) * n_0 - 1$$

(14, 15, 16)

Konstrukce Hagelbargerova kódu je tak definován jako počet potřebných paměťových buněk s použitím logických funkcí.
Pro výpočet paměťových buněk kodéru platí vztah:

$$S_{PK} = L(n) * (n_0 - 1) + 1$$

(17)

Při výpočtu paměťových buněk dekodéru ho můžeme rozdělit na jednotlivé podbloky, které realizují každý svoji funkci v systému. V dekodéru to jsou 3 bloky a to vlastní dekodér, vlastní korekce chyby a příprava korekce chyby, pro které platí následující vztahy.

$$S_{PD1} = L(n) * (n_0 - 1) + 1$$
$$S_{PD2} = (L(n) * (n_0 - 1)) * (n_0 - 1)$$
$$S_{PD3} = L(n) - 1$$
$$S_{PD} = S_{PD1} + S_{PD2} + S_{PD3} = (L(n) * (n_0 - 1) - n_0 + 1)$$

(18, 19, 20, 21)

Pro celkový počet paměťových buněk jak kodéru, tak i dekodéru platí:

$$S_P = S_{PK} + S_{PD} = \left(L(n) * ({n_0}^2 + n_0 - 1\right) - n_0 + 2) \quad (22)$$

Podobně postupujeme při výpočtu dílčích logických funkcí kodéru a dekodéru jako u paměťových buněk.

$$S_{LK} = 2(\, n_0 - 1)$$

$$S_{LD} = S_{LD1} + S_{LD2} + S_{LD3} = (2(\, n_0 - 1) + (\, n_0 - 1) + (3\, n_0 - 4) = 6(\, n_0 - 1)$$

$$S_L = S_{LK} + S_{LD} = 8(\, n_0 - 1) \quad (23,24,25)$$

6.2.2 Iwadari – Maseryův kód

Rozměr kódu je ($m*n_0$; $m*k_0$) a jako u Hagelbargerova kódu je určen pomocí blokové matice B_0, která má n_0 sloupců. Sloupce popisujeme z leva doprava pomocí abecední řady a k nim příslušný index, udávající číselnou hodnotu pořadí prvku jako např. a_1; b_2 ; c_3 … Řádky blokové matice číslujeme vzestupně od spodu. Na posledním (nejvyšším) řádku vpravo nahoře se nachází jednička, která představuje zabezpečovací prvek. Po ní následují nulové řádky až do počtu n_0. Poté je jednička umístěna o jednu pozici doprava do vedlejšího sloupce. Dále následuje jednička ve stejném sloupci, ale o jeden řádek níže, která má za funkci opravu jednoho bitu. Pro opravu dalších bitů je zapotřebí posunout opět jeden bit o řádek dolů a o jednu pozici doprava, vložením nulového řádku a opět vložením jednoho bitu na stejný sloupec, ale o jeden řádek dolů. Tímto krokem opravíme dva bity, a pokud potřebujeme opravit více bitů, je zapotřebí vkládat mezi dva řádky s jedním bitem stejného sloupce nulové řádky pro opravu až do celkové velikosti n_0. Pochopit tento způsob vytváření blokové matice můžeme pochopit na následující rovnici (26).

$$\begin{bmatrix}
0 & 0 & \cdots & 0 & 0 & 1 \\
0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 \\
0 & 0 & \cdots & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 1 & \cdots & 0 & 0 & 0 \\
\vdots & \ddots & \cdots & \ddots & \ddots & \vdots \\
0 & 1 & \cdots & 0 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 \\
\vdots & \ddots & \cdots & \ddots & \ddots & \ddots \\
0 & 0 & \cdots & 0 & 0 & 0 \\
1 & 0 & \cdots & 0 & 0 & 0
\end{bmatrix}
\begin{array}{l}
\updownarrow\ n_0 \text{ řádků} \\
\hline
\updownarrow\ 2 \text{ řádky} \\
\hline
\updownarrow\ 3 \text{ řádky} \\
\hline
\updownarrow\ n_0\text{-1 řádky} \\
\hline
\updownarrow\ n_0 \text{ řádků}
\end{array}
\qquad (26)$$

Jak již bylo zmíněno, tak vytvářecí bloková matice má tvar B_0

$$B_0 = [\mathrm{m} * \mathrm{n}_0;\ \mathrm{m} * \mathrm{k}_0] \tag{27}$$

kde m a k_0 jsou vyjádřeny:

$$m = \frac{n_0 \cdot (n_0 - 1)}{2} + (2n_0 - 1) \tag{28}$$

$$k_0 = n_0 - 1 \tag{29}$$

m a k_0 definují počty vstupních a výstupních paralelních toků

Iwadari – Maseryův kód umožňuje opravit délku shluku b bitů mezi kterými musí být *ochranný interval A* pro který platí:

$$b \leq n_0$$
$$A \geq n_0 * m - 1 \tag{30, 31}$$

Při překročení mezní zabezpečovací schopnosti tento kód nezpůsobuje nekonečný průnik chyb do informací. Jako u předešlého kódu, tak i u tohoto lze pomocí vytvářecího blokové matice B_0 určené ze vztahu () odvodit schéma zapojení kodéru i dekodéru. Velmi důležitá je syndromový rovnice matice kódu, pomocí které jsme schopni jednotlivé prvky kódu.

Dekódování na straně příjemce je využito pomocí prahového dekódování, kde se k syndromový rovnici pro určitý čas k bitům v ní obsažených je zapotřebí nalézt bity na zabezpečení se podílejících v předchozích okamžicích. Pomocí dvou syndromových rovnic jsme schopni opravit jednu chybu. Pokud bychom chtěli opravit více chyb, tak musíme jít více do minulosti a určit tak jednotlivé další syndromový rovnice. Podle toho musíme taky v dekodéru navrhnout příslušný počet paměťových buněk.

Stejně jako u předešlého kódu celkové zpoždění přenášené informace je způsobené průchodem paměťových buněk v dekodéru, v kodéru se realizuje pomocí buněk jen průběžné zabezpečení.

$$Z_K = 0$$
$$Z_D = m * k_0$$
$$Z = Z_K + Z_D = Z_D = 0{,}5{n_0}^3 + {n_0}^2 - 2{,}5n_0 + 1$$

(32, 33, 34)

Pro výpočet paměťových buněk v kodéru Iwadari – Maseryův kód platí vztah

$$S_{PK} = m * \mathrm{k_0} \tag{35}$$

Počet paměťových buněk v dekodéru jsou buňky použity jen pro vlastní dekodér a přípravu korekce. K bloku vlastní korekci dochází přímo ve vlastním dekodéru, kde jsou zavedeny logické funkce a není třeba tady používat paměťové buňky.

$$S_{PD1} = m * \mathrm{k_0}$$
$$S_{PD2} = \mathrm{n_0} - 1$$
$$S_{PD} = S_{PD1} + S_{PD2} = 0{,}5{n_0}^3 + {n_0}^2 - 1{,}5n_0$$

(36, 37, 38, 39)

Celkové zpoždění paměťových buněk kodéru i dekodéru vypočítáme

$$S_P = S_{PK} + S_{PD} = 2(m * \mathrm{k_0}) + n_0 - 1 = 2{n_0}^3 + 4{n_0}^2 - 9n_0 + 3$$

(40)

Pro použití logických funkcí použité jak v kodéru, tak i v dekodéru v jednotlivých částech platí vztah

$$S_{LK} = 2\,n_0 - 3$$
$$S_{LD} = S_{LD1} + S_{LD2} + S_{LD3} = (2\,n_0 - 3) + (\,n_0 - 1)$$
$$+(2(\,n_0 - 1) + 1)\;5(\,n_0 - 1)$$
$$S_L = S_{LK} + S_{LD} = 7(\,n_0 - 1) - 1$$

(41, 42, 43)

6.2.3 Berlekamp -Preparatův

Berlekamp - Preparát má rozměr kódu (n_0; $n_0 - 1$; m) nezávisle na sobě prokázali kód, který splňuje Gallagerovy meze, ty říkají, že libovolný konvoluční kód o informační rychlosti R má schopnost opravit shluk chyb délky ≤b bitů vzhledem k *ochrannému intervalu* délky *A*, pokud platí

$$\frac{A}{b} \geq \frac{1+R}{1-R} \tag{44}$$

Berlekamp - Preparatův kód je systematický konvoluční, který se používá pro ochranu před shluky chyb omezených do samostatných bloků úměrného ochranného intervalu o délce *m* bezchybného bloku tzn. shluk chyb může nanejvýš ovlivnit jeden blok omezené délky. Berlekamp - Preparatův kód můžeme jako v předchozích dvou případech popsat pomocí vytvářecí blokové matice B_0, která má rozměr matice $n_0 \; x \; 2n_0$ a je složen ze dvou podmatic. První podmatice rozměr $n_0 \; x \; n_0$, má na vedlejší diagonále samé jedničky, druhá podmatice má stejný rozměr, ale jedničky se nachází nad hlavní diagonálou matice, na hlavní diagonále a pod ní jsou umístěny nuly viz ().

$$B_0 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{45}$$

Z rovnice (44) můžeme vyjádřit ochranný interval A po splnění následující podmínky $R = (n_0 - 1)/n_0$ pak platí vztah

$$A = m * n_0 = m * b \tag{46,47}$$

Opět můžeme za pomocí vytvářecí matice B_0 určit schéma zapojení kodéru i dekodéru. Dekódování se provádí pomocí obecné dekódovací techniky pro konvoluční kódy opravující shluky chyb zásluhou Masseyho. Díky tomu nemůže v dekodéru nastat nekonečné šíření chyb.

Při návrhu tohoto kódu je celkové zpoždění systému dáno průchodem informačních bitů přes jednu větev paralelního větvení v dekodéru, kde se vytváří odvození zabezpečení bitů na přijímací straně, pro které platí.

$$Z_K = 0$$
$$Z_D = 2n_0 - 1$$
$$Z = Z_K + Z_D = Z_D = 2n_0 - 1 \quad (48, 49, 50)$$

Ze znalosti vytvářecí blokové matice B_0 můžeme odvodit složitost kodéru vztahem

$$S_{PK} = 2n_0 - 1 \quad (51)$$

U návrhu dekodéru se paměťové buňky používají zejména pro vlastní dekódování a případnou korekci, kde se oprava chybně přeneseného bitu koriguje přímo v dekodéru přijímače.

$$S_{PD1} = 3(2n_0 - 1)$$
$$S_{PD2} = 2n_0 - 2$$
$$S_{PD} = S_{PD1} + S_{PD2} = 8(n_0 - 5)$$

(52, 53, 54)

Pro celkové zpoždění přenosového systému můžeme napsat vztah:

$$S_P = S_{PK} + S_{PD} = 10n_0 - 6 \quad (55)$$

Při výpočtu počtu použitých logických operátorů, použité v blocích přípravy korekce chyby a korekce chyby, platí následující rovnice:

$$S_{LK} = 2(n_0 - 1)$$

$$S_{LD} = S_{LD1} + S_{LD2} = 4(n_0 - 1) + (n_0 - 1) = 5n_0 - 2$$

$$S_L = S_{LK} + S_{LD} = 7n_0 - 4 \quad (56, 57, 58)$$

Pří výčtu jednotlivých rovnic pro dané kódy jsme schopni si vypočítat přesný počet paměťových buněk, počty logických funkcí a hlavně zpoždění systému, které je poměrně dost důležitým parametrem. Pokud si stanovíme potřebný počet chyb, který je

7 REALIZACE PROVEDENÍ

7.1 UVEDENÍ DO PROBLEMATIKY

Po uvedení do problematiky můžeme přejít přímo k realizaci přenosového systému se shlukem chyb. Bylo zapotřebí si stanovit vstupní požadavky systému, které budou pro všechny tři realizace kódu stejné. Shluky chyb budou mít délku $n_0 = 10$, a informační rychlost $R = 0,9$. Budeme porovnávat výsledky Interleavingu&Konvoluční kódy po přenosu informace po přenosovém kanálu, na který se nám naindukuje shluk chyb délky 10-ti bitů. Budeme jejich parametru a podle nasimulovaných hodnot odvodíme závěr, pro oboje metody ochrany proti průniku shluku chyb do samotné přenášené informace.

Pro realizaci nasimulování přenosového systému je zapotřebí si vybrat vhodnou softwarovou aplikaci, na které budeme provádět realizaci. Je zapotřebí, aby prostředí bylo přehledné, a co možná nejbližší reální situaci přenosového řetězce, aby rozdíl mezi praxi a provedenou simulací byl co možná minimální. Vhodných programem pro simulaci je použití softwarové aplikace Matlab, který je svým spektrem možností velmi vhodnou aplikací. Jedná se o program, určený pro technické výpočty a simulaci dějů v různých technických oborech. Součásti aplikace Matlab je program Simulink, který je vhodný pro simulaci a modelování dynamických systémů. Pomocí tohoto programu budu provádět realizaci zapojení dílčích bloků přenosového řetězce.

Realizace shluku chyb budu simulována dvěma metodami prokládání (*blokové – konvoluční prokládání*) a na třech zabezpečovacích konvolučních kódech určených proti shluku chyb (*Hagelbargerův, Iwadari – Maseryův a Berlekamp – Preparatův kód*).

Při pohledu na Obr. 7.26 modelu přenosového systému si můžeme tento systém rozdělit na tři části, a to na vysílací část, vedení a přijímací část. Každou část tohoto systému reprezentují určité bloky. Vysílací část je zastoupena zdrojem digitálního signálu neboli posloupnosti bitového toku, který potřebujeme přenést skrz pomyslnou přenosovou cestu k přijímači. Zdroj je reprezentován jako Bernoulli generátor, který generuje náhodné hodnoty od nuly do jedné (bitový tok „0“ a „1“).

Přenosový systém je založen na konvolučním zabezpečovacím kódu pro opravu shluku chyb. Do tohoto systému je navíc vložen blok *Prokládání*, který se používá taky pro potlačení shlukový chyb. Můžeme říct, že Prokládání je taková nadstavba přenosového

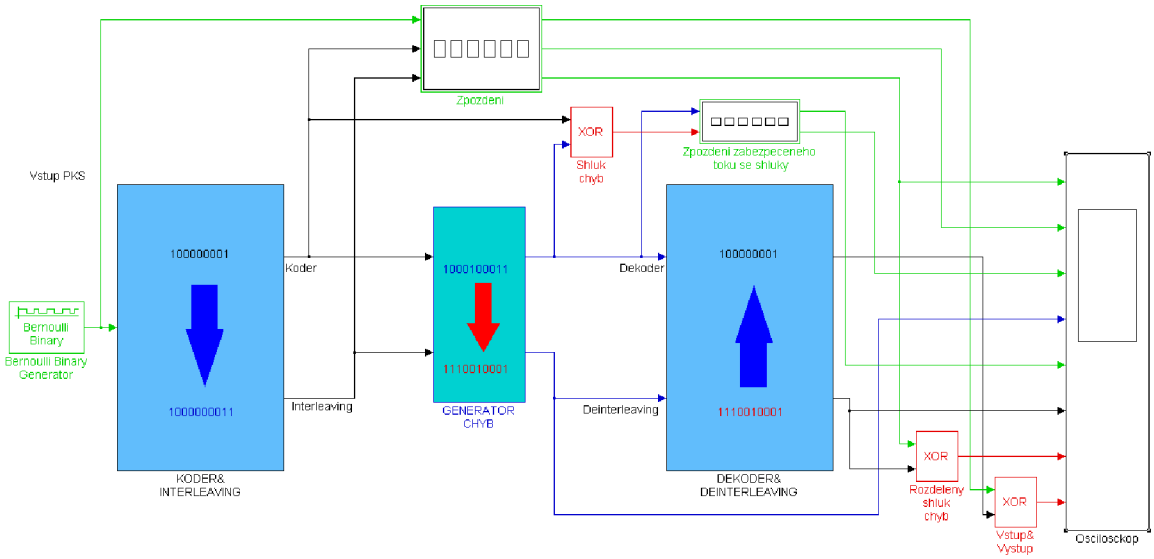
systému, která nám rozdělí vzniklý shluk na menší části chyb, které jsme pak schopni dále opravit pomocí zabezpečovacích kódů.

Dalším blokem v modelu se nachází *generátor chyb*, který nám realizuje přirozené prostředí během přenosu a tak jsme schopni pomocí tohoto bloku simulovat nahodilý jev.

Pomocí zpožďovacích bloků si „zpomalíme“ přenos signálů při zobrazování výsledků, jelikož při průchodu signálu paměťovými prvky dochází k jeho zpoždění. Zpoždění signálů je převážně způsobeno bloku *prokládání* a také v *dekodérech*.

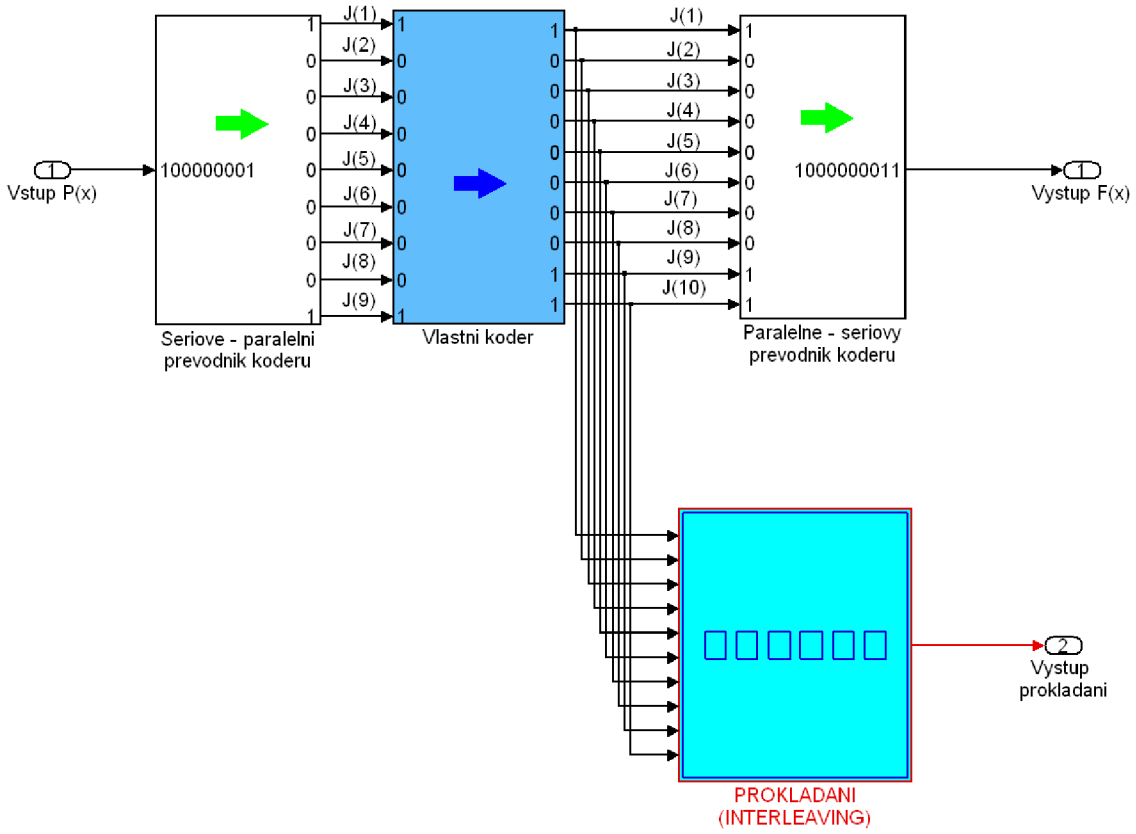
V bloku osciloskopu si můžeme přehledně zobrazit průběhy signálů při průchodu jednotlivými bloky.

Pro lepší pochopení problematiky si blíže popíšeme celý systém.



Obr. 7.26 Model přenosového systému

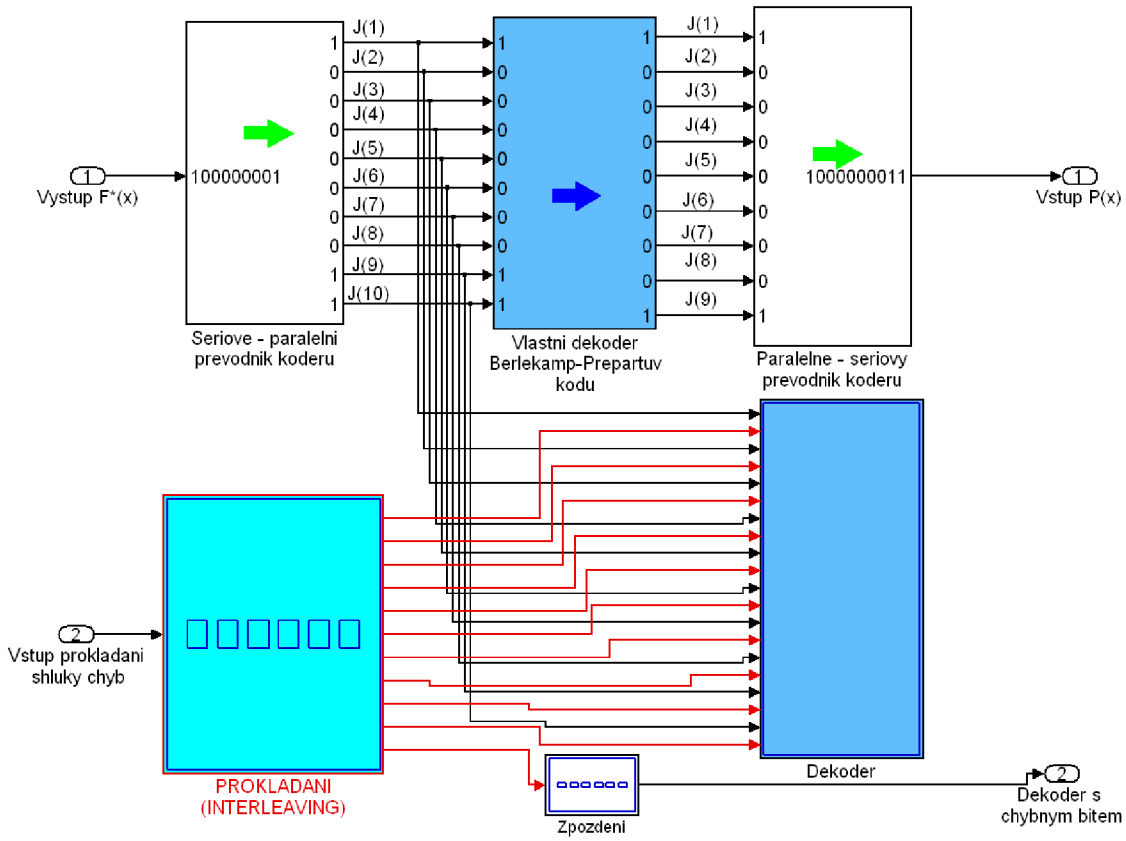
Jak již bylo zmíněno, celý systém je založený na konvolučním zabezpečovacím kódu schopném opravovat shluky chyb do určité délky. Ze vstupních parametrů, které jsme si zadali při sestavování zabezpečovacího systému. A to, že systém bude schopen opravovat shluk do délky 10chyb. Prvně si tedy popíšeme konvoluční zabezpečovací systém a pak až prokládání. Budou popsány tři konvoluční kódy, které se úspěšně používají pro opravu shluku chyb.



Obr. 7.27 Kodér s prokládáním

Při návrhu přenosového systému pomocí použití konvolučních kódů, bude pro všechny tři realizace konvolučních kódu stejné nastavení a funkční bloky sério – paralelní převodníků a paralelně – sériových převodníků tzn., stačí popsat tyto dva prvky jednou a jejich budou totožná ve všech třech nastaveních. Jediné, co se budeme měnit, tak je na straně vysílače schéma zapojení kodéru a na straně příjemce zapojení dekodéru. Pro každý kód platí vlastní zapojení, které bude uvedené v příloze.

Z předchozího textu už víme, že při procesu zabezpečení se sériová podoba přenášených dat převádí na data paralelní v bloku *sério - paralelním převodníku*, kde si tento blok můžeme představit jako *Demultiplexor*. Data pokračují po paralelních větvích a vstupují do bloku vlastní kodéru, kde probíhá odvození z nezabezpečeného toku zabezpečovací data. Z vlastního kodéru data vstupují do bloku paralelně – sériového převodníku, kde se převádí z paralelní podoby přenášené informace do podoby sériové. Tento blok si můžeme představit jako blok multiplexor, který převádí jednotlivé příspěvkové kanály do jednoho společného kanálu. Pomocí tohoto celého procesu jsme převedli nezabezpečený bitový tok na zabezpečený proti vzniku chyb.

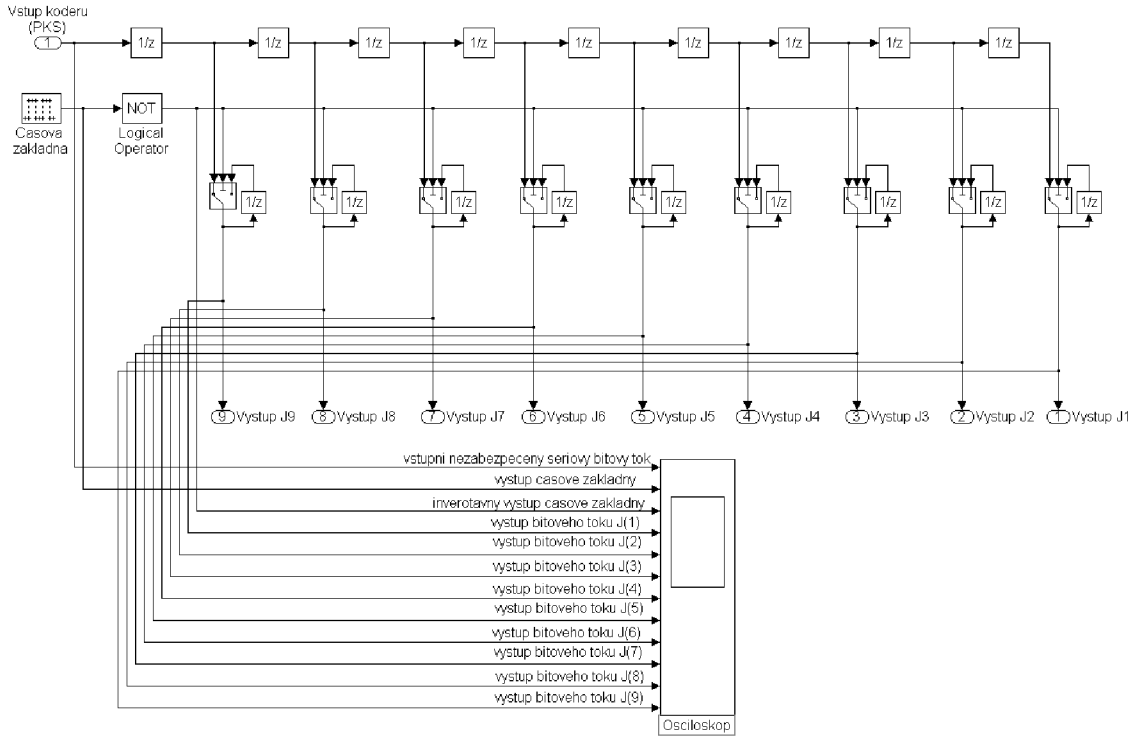


Obr. 7.28 Dekodér s prokládáním

Tento proces probíhá v bloku dekodér stejně, akorát v opačné znění a to tak, že zabezpečená data se převedou ze sériového toku na paralelní, v dekodéru se vypočítá zabezpečovací prvek a porovná se zabezpečovacím prvkem vytvořený ve vysílači. Pokud prvky jsou stejné (je roven logické úrovni nule), tak se nic neprovádí, pokud je však jeden z nich roven „jedničce“, tak došlo během přenosu k chybě a provádí se pomocí korekčních obvodů k nápravě.

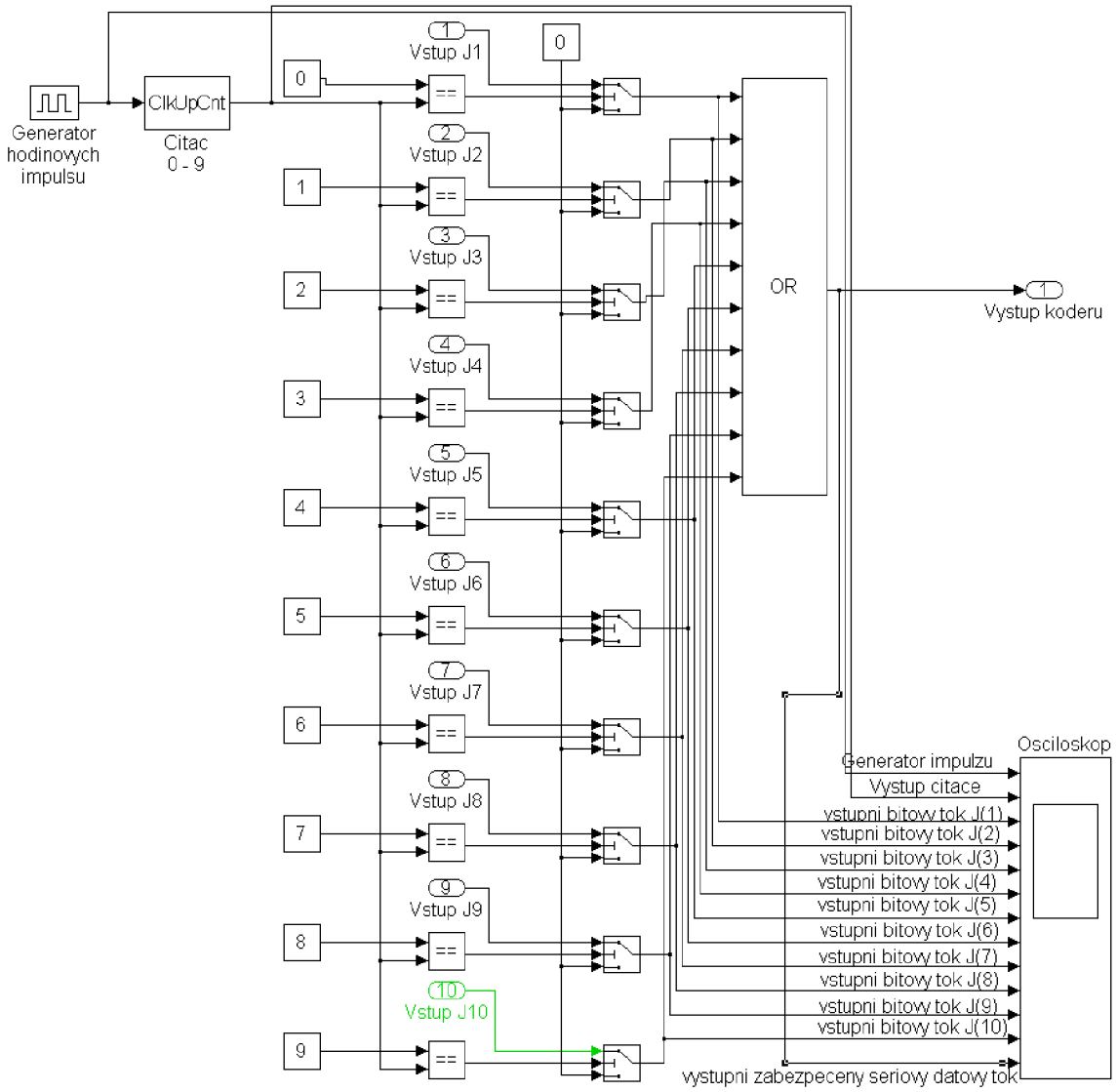
Nyní si blíže popíšeme funkci jednotlivých převodníků, které v zapojeních používáme. Jenda s funkcí převodníku je převod rychlosti nezabezpečeného toku v_1 na tok rychlosti zabezpečeného v_2. V Našem případě se bude jednat o poměr těchto rychlostí $v_1/v_2 = 9/10$, jelikož do vlastního kodéru vstupují data rychlostí v_1 pomocí devíti paralelní větví a vystupují z něj deseti větvemi v_2. Vstupní signál z generátoru je pomoc devíti zpožďovacích článků, které vytváří posuvný registr, postupně přiváděn na jednotlivé vstupy přepínačů., které jsou ovládané pomocí bloku časová základna. Ta ovládá spínání těchto přepínačů na určitou dobu, po kterou se odebere vzorek z posuvného registru a prostřednictvím zpožďovacího článku se zpětnou vazbou do přepínače je tento vzorek

uchován po dobu, než se opět znovu sepne spínač a odebere další vzorek. Takto převedeme sériový bitový tok na paralelní a přivádíme tyto vzorky na výstup. Plní funkci demultiplexoru.



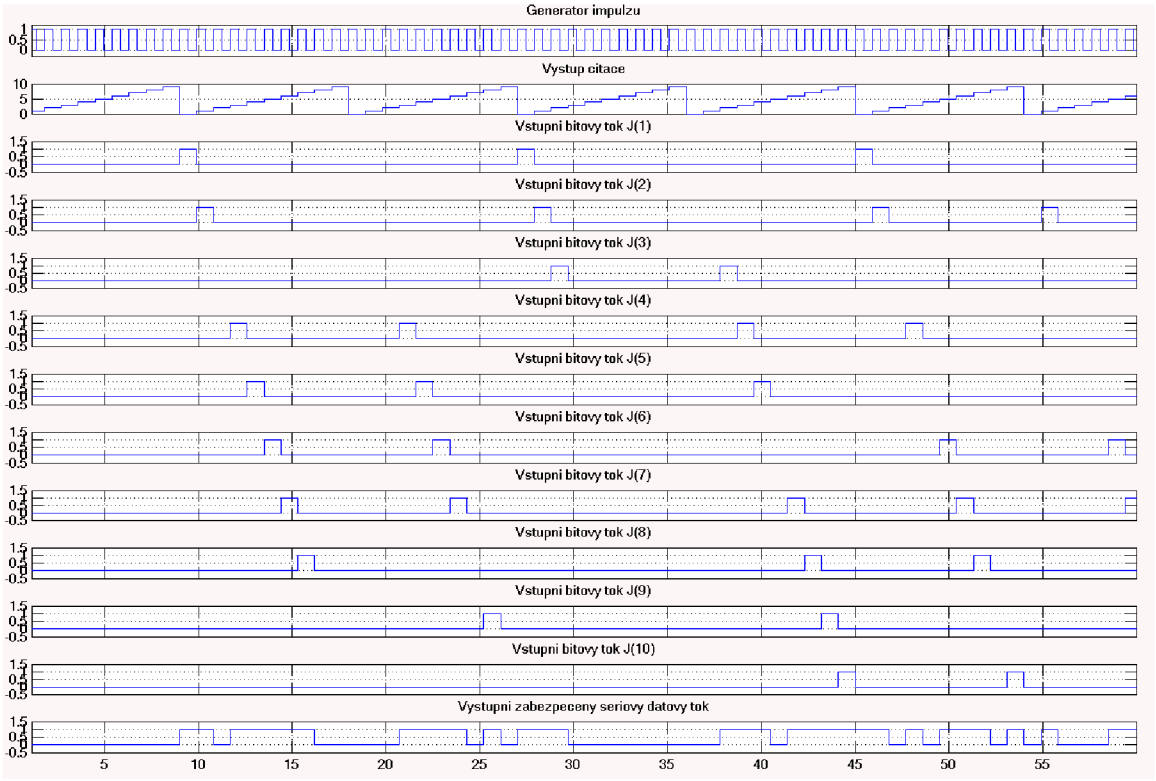
Obr. 7.29 Sério – paralelní převodník

Obr. 7.30 Průběh na sério – paralelním převodníku



Obr. 7.32 Paralelně – sériový převodník

Převodník Obr. 7.32 má opačnou funkci, kdy z paralelního bitového toku převádí datový tok do sériové podoby, tzn. plní funkci multiplexoru a to tak, že pomocí čítače, který má nastaveno desek kroků, které jsou cyklicky nastavené, takže „běží “ pořád do kola. Na jednotlivé přepínače jsou přiváděny signály, a tan vždycky sepne na 1/10 časové základny. Pomocí komparátoru == máme nastaveno, že jednotlivé přepínače budou spínat periodicky od vstupního J1, J2 … J10. Za pomocí logického členu OR sdružujeme odebrané vzorky na výstup. Na výstupy z jednotlivých bloků převodníku se můžeme podívat na Obr.



Obr. 7.32 Průběh na paralelně - sériovém převodníku

Tím jsme prošli funkci dílčích bloků a můžeme se podívat na návrh kodéru a dekodérů konvolučních kódů. Pro zabezpečení před shluky chyb používáme následující tři konvoluční kódy.

7.2 HAGELBAGERUV KOD

Ze znalosti podmínek jsme schopni vypočítat prvky přenosového systému. Z rovnice (9) vyplývá, že n_0 je rovna b_0 a tady $n_0 = 10$. Při znalosti n_0 určíme blokovou vytvářecí matici B_D, která má rozměr n_0 x n_0 ze které dalšími výpočty (11) a (12) vypočítáme počet potřebných řádků pro zapsání dekadického čísla do dvojkové matice B_0.

$$B_D=\begin{bmatrix}0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&19&0\\0&0&0&0&0&0&0&17&0&0\\0&0&0&0&0&0&15&0&0&0\\0&0&0&0&0&13&0&0&0&0\\0&0&0&0&11&0&0&0&0&0\\0&0&0&9&0&0&0&0&0&0\\0&0&7&0&0&0&0&0&0&0\\0&5&0&0&0&0&0&0&0&0\\3&0&0&0&0&0&0&0&0&0\end{bmatrix} \tag{21}$$

Nejmenší celé číslo potřebné k zaspání dekadického čísla do dvojkové matice, které vyhovuje nerovnosti z rovnice (11) je L(n) = 5. Při dosazení do rovnice (12) vypočítáme potřebný počet řádků pro zapsání všech dekadických čísel do dvojkové matice B_0. Počet řádků matice B_0 je roven 47. Bloková dvojková matice bude mít rozměr (10x47), pro její velikost matice nebudu ji uvádět celou.

Pro stanovení ochranného intervalu A (13) před shluky délky 10-ti bitů je zapotřebí interval délky 499 bitů.

$$B_0=\begin{bmatrix}0&0&0&0&0&0&0&0&0&1\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0\end{bmatrix}$$

číslo 1

číslo 19

číslo 17

číslo 5

číslo 3

(21)

$$g_1^{(10)} = D^{45} + D^{46}$$
$$g_2^{(10)} = D^{40} + D^{42}$$
$$g_3^{(10)} = D^{35} + D^{36} + D^{37}$$
$$g_4^{(10)} = D^{30} + D^{33}$$
$$g_5^{(10)} = D^{25} + D^{26} + D^{28}$$
$$g_6^{(10)} = D^{20} + D^{22} + D^{23}$$
$$g_7^{(10)} = D^{15} + D^{16} + D^{17} + D^{18}$$
$$g_8^{(10)} = D^{10} + D^{14}$$
$$g_9^{(10)} = D^{5} + D^{8} + D^{9}$$

(10)

Pro dekódování se využívá syndromového dekódování s přímým převodem [E] →[S]. Podmínkou správného dekódování je dodržení maximální délky shluku chyb n_0 = b a délky ochranného intervalu A.

Při příchodu chybného bitu je bit jednak přiveden přímo na výstup vlastního kodéru, a taky projde příslušnými paměťovými buňkami podle umístění $J^{(n)}$. Ve zpožďovacích buňkách se odvodí zabezpečení a porovná v bloku Generátor syndromu se zabezpečením vytvořené v kodéru vysílače. Pokud by výsledek v Generátory syndromu byl roven nule, tak by byl přenos bez chyby, ale my víme, že byl chybně přenesený úsek, tak na výstupu generátoru syndromu je „jednička“. Proto vzorek vstupuje do bloku Převodníku [S] → [E], kde se lokalizuje chybný prvek, a za pomocí paměťových buněk se zpozdí a následně vloží zpátky do paralelní větve pomocí logického členu XOR.
Schéma zapojení Hagelbargova kodéru a dekodérů najdeme v příloze.
Abych při zobrazení výsledků měli všechny výsledky rovnoměrně pod sebou a nebyli posunuty po ose x, ať už na jakoukoliv stranu, je zapotřebí zpozdit vstup vůči výstupu, protože v dekodéru vlivem vypočítávání zabezpečovacího kódu, porovnáním z přeneseného zabezpečení a případná korekce vzniklé chyby nám vkládá zpoždění celého systému. Je tedy zapotřebí vstup kodéru posunout o tuto hodnotu, která je rovna 45taktům. Ale jelikož se v Matlabu provádí různé kroky, musíme nakonec posunout vstup o 57 taktů, abychom docílili správného zobrazení grafů.

7.3 IWADARI - MASERYUV KOD

Z blokové matice (26), odvodíme velmi důležitou syndromovou rovnici, pomocí které jsme schopni sestavit schéma zapojení kodéru i dekodéru.

$$s_{64} = a_1 + a_{10} + b_{11} + b_{19} + c_{20} + c_{27} + d_{28} + d_{34} + e_{35} + e_{40} + f_{41} + f_{45} + g_{46} + g_{49} + h_{50} + h_{52} + ch_{53} + ch_{54} + i_{64} \quad (63)$$

Pokud vyjdeme z předpokladu, že v samotném kodéru nevznikají chyby, tam můžeme psát rovnici

$$s_{64} = 0 \quad (64)$$

Můžeme pro odvození prvku i_{64} napsav vztah

$$i_{64} = a_1 + a_{10} + b_{11} + b_{19} + c_{20} + c_{27} + d_{28} + d_{34} + e_{35} + e_{40} + f_{41} + f_{45} + g_{46} + g_{49} + h_{50} + h_{52} + ch_{53} + ch_{54} \quad (65)$$

kde nám tato rovnice () udává jednotlivá místa, odkud budeme brát signál pro realizaci zabezpečení. Pro výpočet paměťových míst kodéru platí vztah $n_0 .k_0$ v našem případě to bude 64 míst. Mezi paměťovými místy, odkud dostáváme signál pro zabezpečení se nachází logické členy XOR, které slouží jako spínač pro odebrání v určitý stanový okamžik signálů pro zabezpečení.

Při dekódování se využívá mechanismu prahového dekódování, kde hledáme vztah ortogonality prvku, který se po přenosu kontroluje. Pro korekci chyb v dekodéru nás nezajímá bit i_{64}, který nemá vliv na přenášenou zprávu. Zajímají nás hlavně pozice bitu ch_{54}, h_{50}, g_{46}, f_{41}, e_{35}, d_{28}, c_{20}, b_{11}, a_1 tedy informační bity.

Ze znalosti konvolučních kódů víme, že se při přenosu prvně přenáší informační a pak teprve zabezpečovací bity. Pro návrh dekodéru se využívají syndromové rovnice, kde počet rovnic určuje potřebný požadavek přinejmenším dvou ortogonálních součtů pro jeden opravený bit, tzn. v obou rovnicích se vyskytuje jen opravovaný bit a žádný jiný. Pokud chceme opravit bit v paměťové buňce ch_{53} potřebujeme znát syndromovou rovnici s_{63}, pro kterou platí:

$$s_{63} = a_0 + a_9 + b_{10} + b_{18} + c_{19} + c_{26} + d_{27} + d_{33} + e_{34} + e_{39} + f_{40} + f_{44} + g_{45} + g_{48} + h_{49} + h_{51} + ch_{52} + ch_{53} + i_{63} \quad (66)$$

Pomocí těchto rovnic () a () jsme schopni až předpokladu opravit chybně přenesený bit v buňce ch_{53} pokud

$$s_{64} \,.\, s_{63} = 0 \wedge s_{64} \,.\, s_{63} = 1, \quad (67)$$

kde je výsledek rovnice () roven nule, tak je bit v buňce správný, ale pokud je bit roven nule, tak nastala chyba během přenosu. Pokud chceme opravovat více bitů, je třeba jít více do minulosti a určit tak další potřebné syndromové rovnice pro opravu bitu h_{50}, g_{46}, f_{41}, e_{35}, d_{28}, c_{20}, b_{11}, a_1 platí vztah

$$s_{62} = a_{-1} + a_8 + b_9 + b_{17} + c_{18} + c_{25} + d_{26} + d_{32} + e_{33} + e_{38} + f_{39} + f_{43} + g_{44} + g_{47} + h_{48} + h_{50} + ch_{51} + ch_{52} + i_{62}$$

$$s_{61} = a_{-2} + a_7 + b_8 + b_{16} + c_{17} + c_{24} + d_{25} + d_{31} + e_{32} + e_{37} + f_{38} + f_{42} + g_{43} + g_{46} + h_{47} + h_{49} + ch_{50} + ch_{51} + i_{61}$$

$$s_{60} = a_{-3} + a_6 + b_7 + b_{15} + c_{16} + c_{23} + d_{24} + d_{30} + e_{31} + e_{36} + f_{37} + f_{41} + g_{42} + g_{45} + h_{46} + h_{48} + ch_{49} + ch_{50} + i_{60}$$

$$s_{59} = a_{-4} + a_5 + b_6 + b_{14} + c_{15} + c_{22} + d_{23} + d_{29} + e_{30} + e_{35} + f_{36} + f_{40} + g_{41} + g_{44} + h_{45} + h_{47} + ch_{48} + ch_{49} + i_{59}$$

$$s_{58} = a_{-5} + a_4 + b_5 + b_{13} + c_{14} + c_{21} + d_{22} + d_{28} + e_{29} + e_{34} + f_{35} + f_{39} + g_{40} + g_{43} + h_{44} + h_{46} + ch_{47} + ch_{48} + i_{58}$$

$$s_{57} = a_{-6} + a_3 + b_4 + b_{12} + c_{13} + c_{20} + d_{21} + d_{27} + e_{28} + e_{33} + f_{34} + f_{38} + g_{39} + g_{42} + h_{43} + h_{45} + ch_{46} + ch_{47} + i_{57}$$

$$s_{56} = a_{-7} + a_2 + b_3 + b_{11} + c_{12} + c_{19} + d_{20} + d_{26} + e_{27} + e_{32} + f_{33} + f_{37} + g_{38} + g_{41} + h_{42} + h_{44} + ch_{45} + ch_{46} + i_{56}$$

$$s_{55} = a_{-6} + a_1 + b_2 + b_{10} + c_{11} + c_{18} + d_{19} + d_{25} + e_{26} + e_{31} + f_{32} + f_{36} + g_{37} + g_{40} + h_{41} + h_{43} + ch_{44} + ch_{45} + i_{55} \quad (68, 69, 70, 71, 72, 73,74,75)$$

Pro správně či špatně přenesený bit h_{50}, g_{46}, f_{41}, e_{35}, d_{28}, c_{20}, b_{11}, a_1 platí vztah

$$s_{64} \,.\, s_{62} = 0 \wedge s_{64} \,.\, s_{62} = 1$$

$$s_{64} \,.\, s_{61} = 0 \wedge s_{64} \,.\, s_{61} = 1$$

$$s_{64} . s_{60} = 0 \wedge s_{64} . s_{60} = 1$$
$$s_{64} . s_{59} = 0 \wedge s_{64} . s_{59} = 1$$
$$s_{64} . s_{58} = 0 \wedge s_{64} . s_{58} = 1$$
$$s_{64} . s_{57} = 0 \wedge s_{64} . s_{57} = 1$$
$$s_{64} . s_{56} = 0 \wedge s_{64} . s_{56} = 1$$
$$s_{64} . s_{55} = 0 \wedge s_{64} . s_{55} = 1 \qquad (76,77,78,79,80,81,82,83)$$

Kde opět platí pravidlo, že pokud je výsledek roven nule, je příslušný bit správný a pokud je na místě součinu dvou syndromových rovnic, tak se vyskytla chyba. V Našem případě je třeba použít devět syndromových paměťových buněk pro správnou korekci pro případnou opravu deseti chyb viz příloha. Ze znalosti syndromových rovnic (63) a (66) jsme schopni odvodit schéma zapojení dekodéru.

Schéma zapojení Iwadariho kódu i dekodéru nalezneme v příloze

Abychom dosáhli přehledných výsledků při zobrazení průběhu na osciloskopu, je zapotřebí posunout i tento vstup vůči výstupu o 576 taktů. Ale vlivem výpočetních operací v programu Matlab je zapotřebí tento vstup posunout o 601 taktů.

7.4 BERLEKAMP – PREPARATUV KOD

Jak již bylo zmíněno, rozměr vytvářecí matice tohoto je (n_0 x $2n_0$). Pro opravu deseti chyb z rovnice (45) vyplývá, že budeme potřebovat matici o rozměrech 10 x 20 prvků viz ().

$$B_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Ze znalosti vytvářecí matice jsme schopni odvodit schéma zapojení kodéru i dekodéru pomocí generačního polynomu

$$g_1^{(10)} = D^9 + D^{11} + D^{12} + D^{13} + D^{14} + D^{15} + D^{16} + D^{17} + D^{18} + D^{19}$$
$$g_2^{(10)} = D^8 + D^{12} + D^{13} + D^{14} + D^{15} + D^{16} + D^{17} + D^{18} + D^{19}$$
$$g_3^{(10)} = D^7 + D^{13} + D^{14} + D^{15} + D^{16} + D^{17} + D^{18} + D^{19}$$

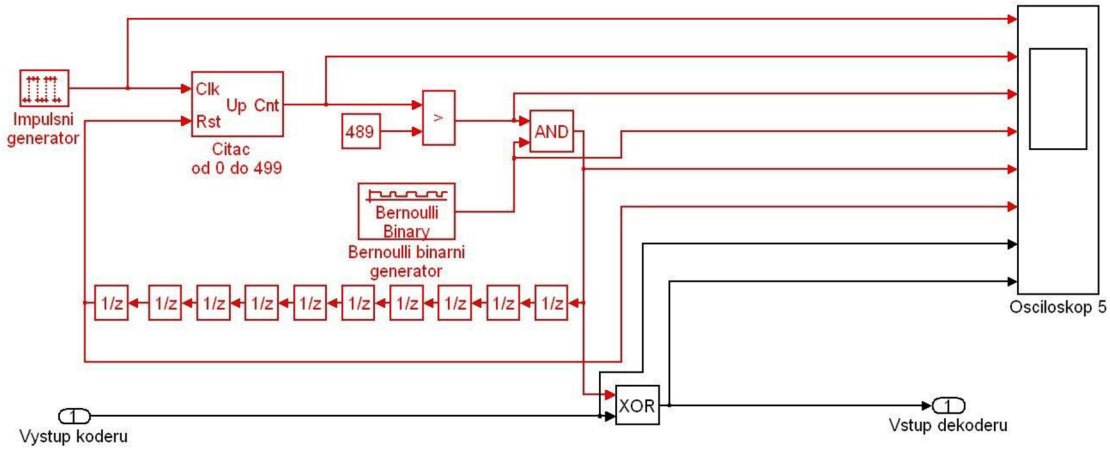
$$g_4^{(10)} = D^6 + D^{14} + D^{15} + D^{16} + D^{17} + D^{18} + D^{19}$$
$$g_5^{(10)} = D^5 + D^{15} + D^{16} + D^{17} + D^{18} + D^{19}$$
$$g_6^{(10)} = D^4 + D^{16} + D^{17} + D^{18} + D^{19}$$
$$g_7^{(10)} = D^3 + D^{17} + D^{18} + D^{19}$$
$$g_8^{(10)} = D^2 + D^{18} + D^{19}$$
$$g_9^{(10)} = \mathrm{D} + D^{19} \quad (45)$$

Schéma zapojení kodéru a dekodéru najdeme v příloze diplomové práce.

Dekódování je pomocí obecné techniky dekódování pro konvoluční kódy. Opět je zapotřebí posunout vstup systému o vypočtených 19 taktů, v praxi je to však 21.

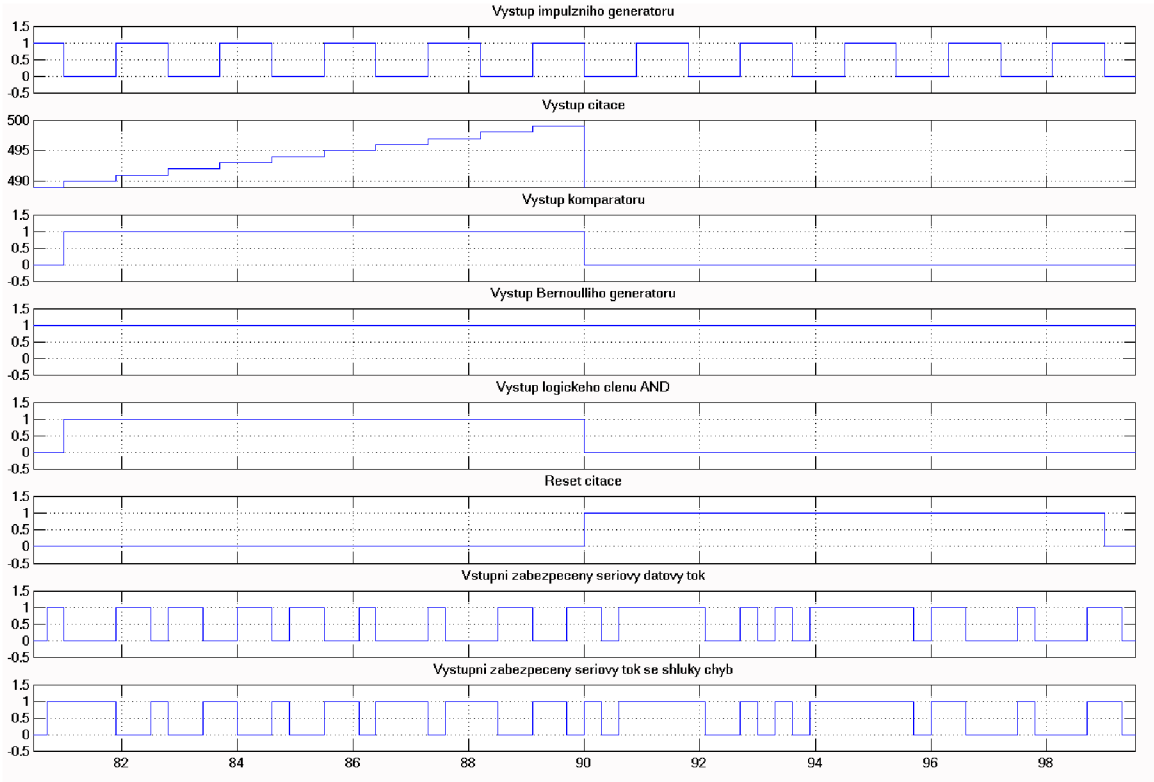
7.5 GENERÁTOR CHYB

Ze znalosti ochranných délek konvolučních kódu jsme schopni odvodit a zapojit jednotlivé bloky tak, abychom docílili požadovaného shluku na délku intervalu *A*. Pro Hagelbargerův kód je délka ochranného intervalu rovna 499-ti bitům, na které připadá shluk *b* roven 10-chybám. Generátor kódu funguje tak, že máme nastavený čítač, aby počítal v cyklu „nula“ – „499“, což je délka ochranného intervalu, kde porovnávač „>“ při hodnotě 490 a větší má na výstupu logickou úroveň jedna a v následujícím logickém členu je na výstup vyslán shluk chyb, který je generován v bloku Bernoulliho binárního generátoru, který má nastavený jako výstup samé jedničky.

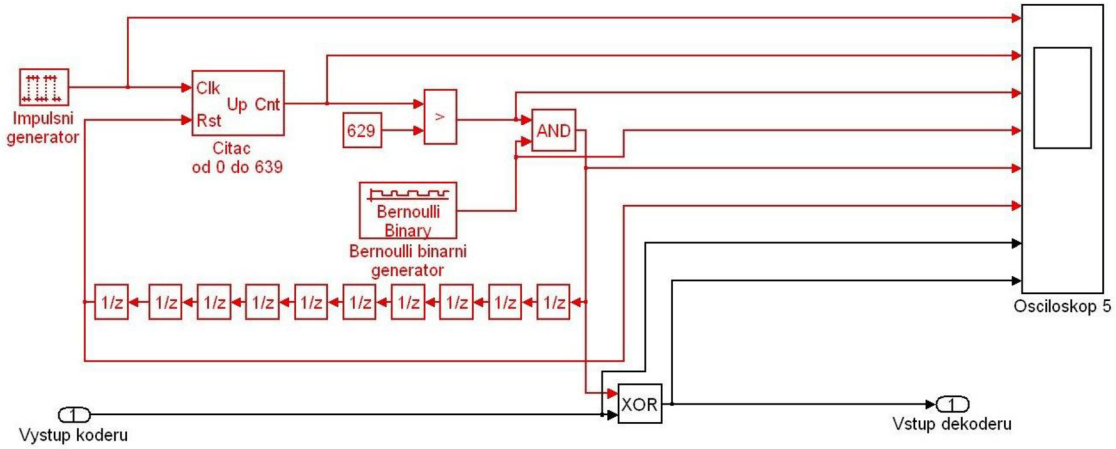


Obr. 7.33 Generátor Hagelbargerových shluku chyb

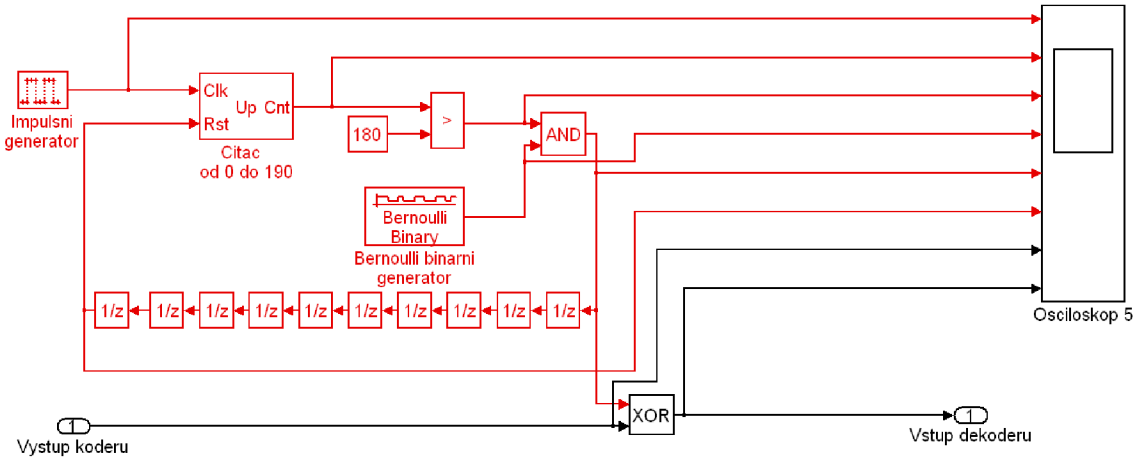
Výstup jednotlivých bloků generátoru chyb je vidět na Obr. 7.34.



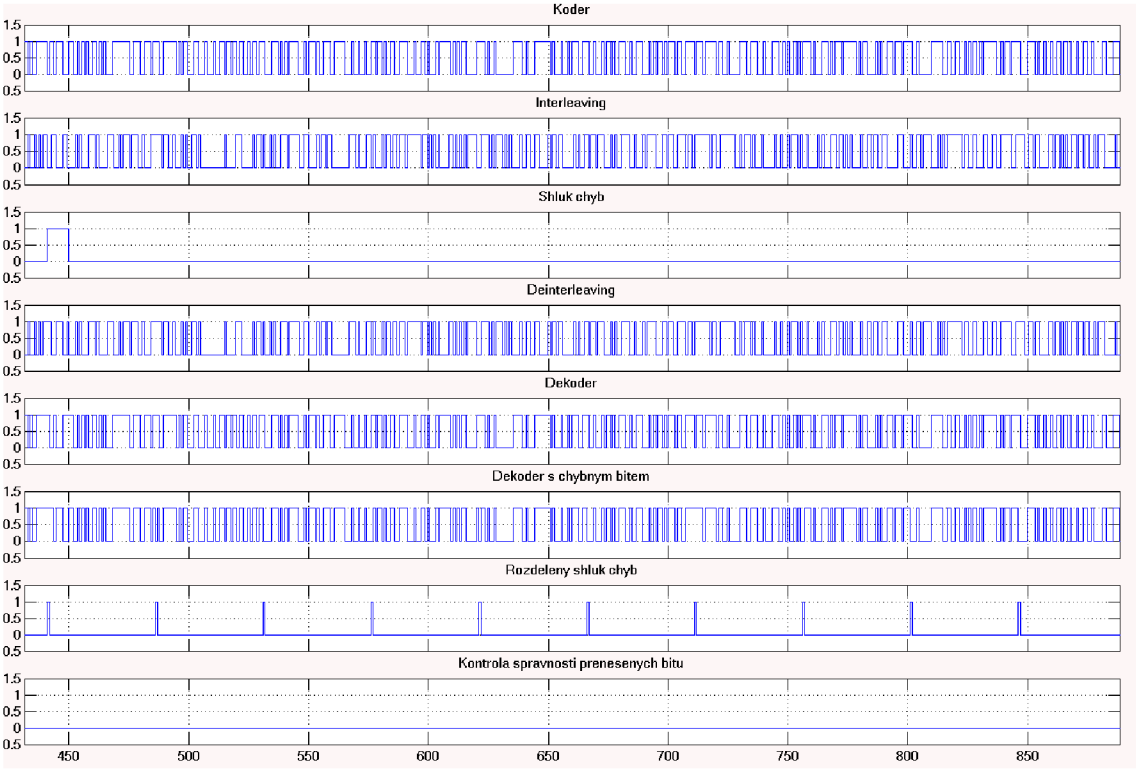
Obr. 7.34 Průběhy Hagelbargerova generátoru chyb



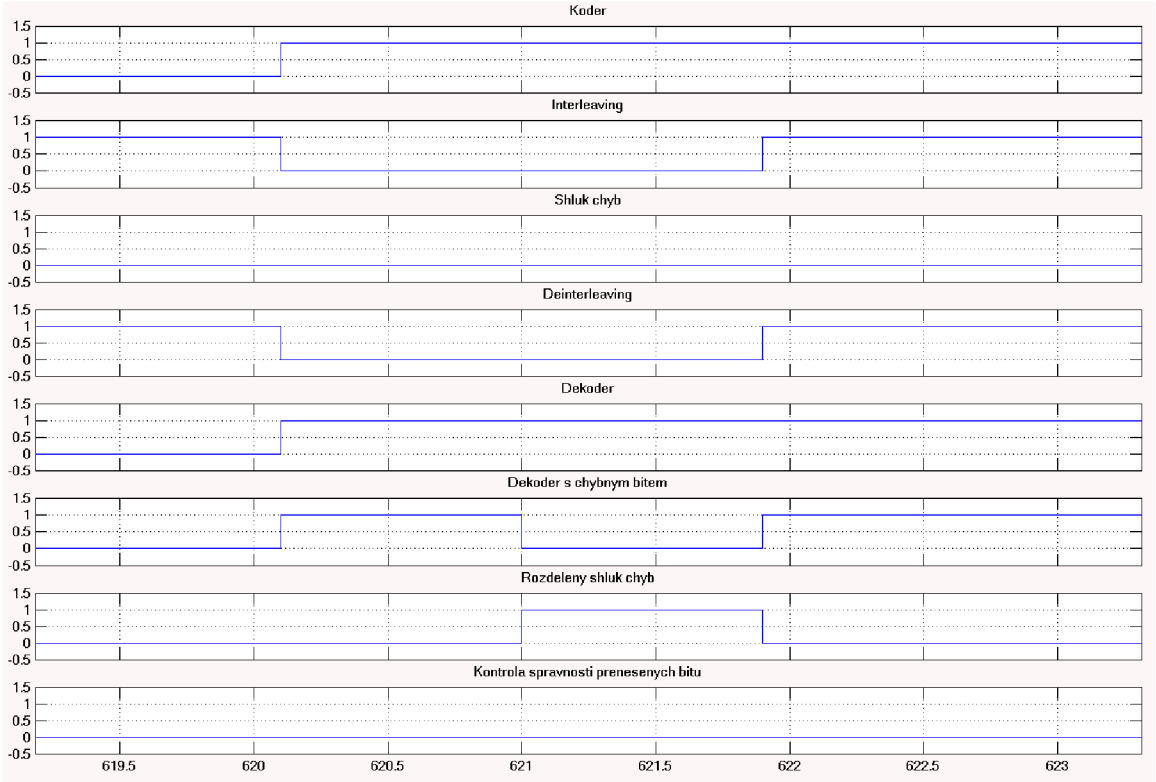
Obr. 7.35 Generátor chyb Iwadari – Maseryův kód



Obr. 7.36 Generátor chyb Berlekampův – Preparatův kód



Obr. 7.37 Simulace Habelbargova kódu



Obr. 7.38 Simulace Habelbargova kódu detail

7.5 Blokové prokládání

Do již existujícího navrhnutého systému vložíme systém s prokládáním zprávy. Interleaving vložíme do přenosového řetězce mezi kodér a vedení na straně vysílače a na straně příjemce mezi vedení dekodér. Budeme simulovat dvě metody prokládání, a to blokové prokládání, které je díky svému značnému zpoždění nepoužitelné a konvoluční prokládání.

Blokové prokládání, jak již bylo napsáno, tak používá pro svoje prokládání matici (paměťová místa), do kterých postupně po řádcích zapisuje posloupnost bitového toku, který po naplnění matice čte nikoliv po řádcích, ale po sloupcích. Tím docílíme proložení zprávy a rozložení případných vzniklých shluků chyb.
Hagelbagberův kód proložení.

Při použití blokového prokládání u tohoto kódu máme vstupní informaci, že *ochranný interval A* je roven 499 bitům a shluk chyb jsme si stanovili na počet 10-ti

chyb. Pomocí vzorečku (60) a (61) si vypočítáme paměťové místo, pro použití blokového prokládání. Pro počet řádků matice

j≥b/t kdy budeme opravovat pouze jednu chybu v řádku, tak vyplývá, že j = 10 a při dosazení pro počet sloupců bude zapotřebí n ≤ 51 paměťových míst. Neboli matici o rozměrech 51 x 10 paměťových míst. Z toho vyplývá zpoždění prokládacího sytému o délce Dt = 2jn neboli 1020 ICZ.

Pro Iwadariho kód, který má *ochranný interval A* roven 639, je zapotřebí použití pamětí o rozměrech 65 x 10 PB. Zpoždění zabezpečovacího systému bude Dt = 1300 ICZ.

Poslední konvoluční kód při použití blokového prokládání bude při *A = 190,* potřebovat prokládací matici o rozměrech 20 x 10 s potřebným zpožděním 400 ICZ.

Jelikož je zpoždění blokového prokládání poměrně značné, tak se tako technika nepoužívá.

7.6 Konvoluční prokládání

Při prokládání zprávy pomocí konvolučních kódů se využívá proměnné zpoždění jednotlivých paralelních větví a tím docílíme poměrně dost snížení zpoždění ve srovnání mezi blokovým a konvolučním prokládáním.

Pro výpočet konvolučního prokládání použijeme již zmiňovaný vzoreček $\frac{(\lambda-1).(n-1)}{2}$, kde *n* vyjadřuje počet řádků konvoluční matice a λ znázorňuje poměr mezi *ochranným intervalem A* a *shluky chyb b*. Po dosazení do vzorečků pro jednotlivé konvoluční kódy dostaneme následující hodnoty.

Hagelbargerův kód by s délkou *ochranné intervalu A* potřeboval $\frac{(50-1).(10-1)}{2}$ po vypočtení vycházelo zpoždění 221 ICZ na jeden prokladač, na přenosový systém, který obsahuje jak interleaving, tak i deinterleaving, by bylo zpoždění dvojnásobné, a to 442 ICZ a celkem 1000PB paměťových buněk.

Iwadariho kód by potřeboval pro *ochranný interval A* celkové zpoždění 568 a pro uložení bitového toku je zapotřebí paměťový prostor o velikosti 1280 PB.

A v neposlední řadě *Berlekampův – Preparatův kód* by potřeboval s délkou *A = 190* bitů celkem 380 PB se zpožděním 162 ICZ na celý přenosový systém.

7. 7 Porovnání metod

V našem případě jsme měli délku shluku chyb „pouze“ délky 10 bitů. Což je pro praxi poměrně málo a během přenosu se může objevit i několikanásobně delší. Při průchodu shluku chyb navrženým prokládacím systémem přerozdělíme vzniklý shluk na jednotlivé chybné úseky o délce 1 bit. Pomocí propočtů jsme schopni vypočítat a rozdělit i vyšší počet shluk, který bychom třeba rozdělili z délky x bitů na jednotlivé chyby o délce 5-ti bitů například. Je to v závislosti, jaký systém navrhujeme, jaké máme vstupné předpoklady, parametry přenosového kanálů a taky schopnost zabezpečovacího kódu.

Tab. 2 Porovnání konvolučních kódů

		zpoždění	počet paměťových buněk PB			logických	Ochranný
		kódu ICZ	kodér	dekodér	celkem	členů	interval A
Konvoluční kódy	Hag	45	46	455	501	72	499
	Iwadari	576	576	586	1161	62	639
	Ber-Prep	19	19	189	208	66	190

Tab. 3 Porovnání prokládání konvolučních kódů

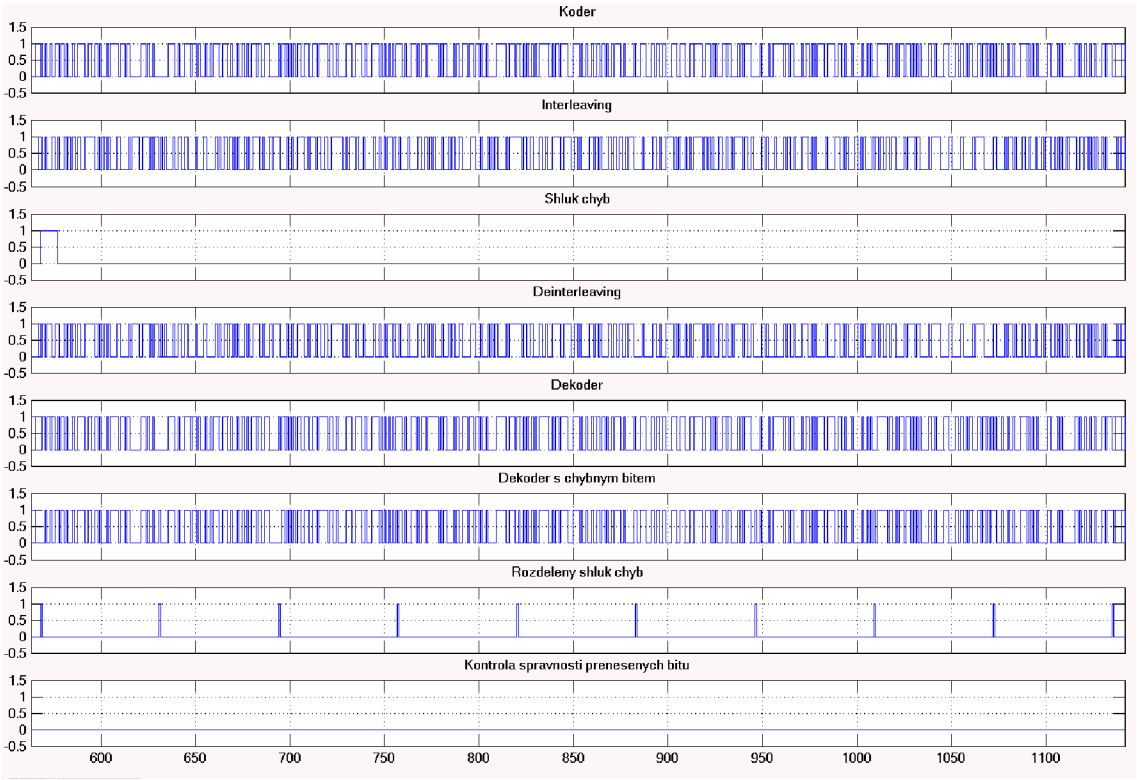
Metoda	Konvoluční kód	Rozměr matice		Paměťových buněk PB			zpoždění	zpoždění
		řádek	sloupec	vysílač	přijímač	celkem	kódu ICZ	celkem ICZ
Prokládání blokové	Hag.	51	10	510	510	1020	510	1020
	Iwadari	65	10	650	650	1300	650	1300
	Ber-Prep.	20	10	200	200	400	400	800
Konvoluční blokové	Hag.	50	10	500	500	1000	221	442
	Iwadari	64	10	640	640	1280	284	568
	Ber-Prep.	19	10	190	190	380	81	162

Pokud srovnáme blokové a konvoluční prokládání podle paměťových buněk je to skoro stejný počet, ale při hodnocení zpoždění prokládacího systému už je na první pohled zřejmé, že u konvolučního prokládání je 2-3x menší zpoždění, což je znatelný rozdíl. Pokud budeme srovnávat konvoluční kód – prokládání, tak pouze Iwadariho konvoluční kód, který má nejdelší *ochranný interval A* je překonán konvolučním pokládáním. Ale k čase prokládání musíme ještě připočíst čas dekódování kódu, který rozložil shluk chyb z 10 chyb na jednu chybu.

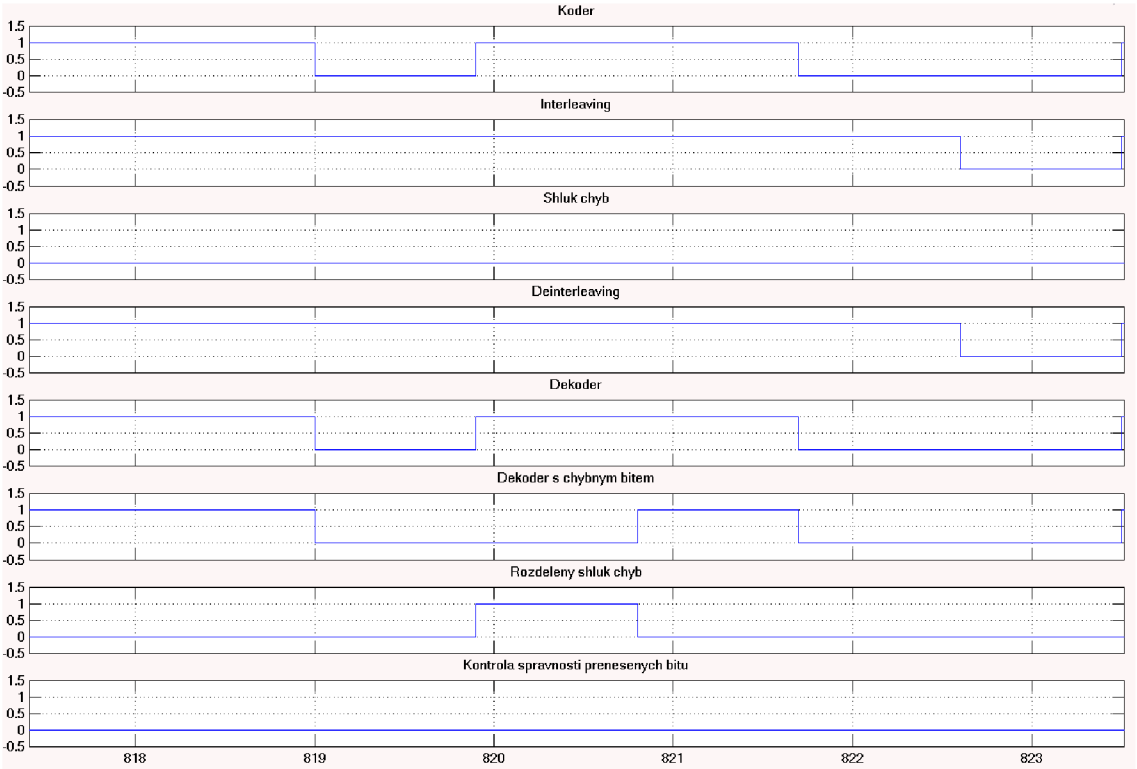
Pokud bychom hodnotili samotný konvoluční kód, tak nejlepší zástupcem z těchto tří zmíněných kódu je jednoznačně Berlekamp – Preparatův kód, krytý mě jednak

nejkratší *ochranný interval*, ale hlavně je zde zpoždění pouhých 19 ICZ, a taky potřebuje pro svoji realizaci pouze 208 paměťových buněk.

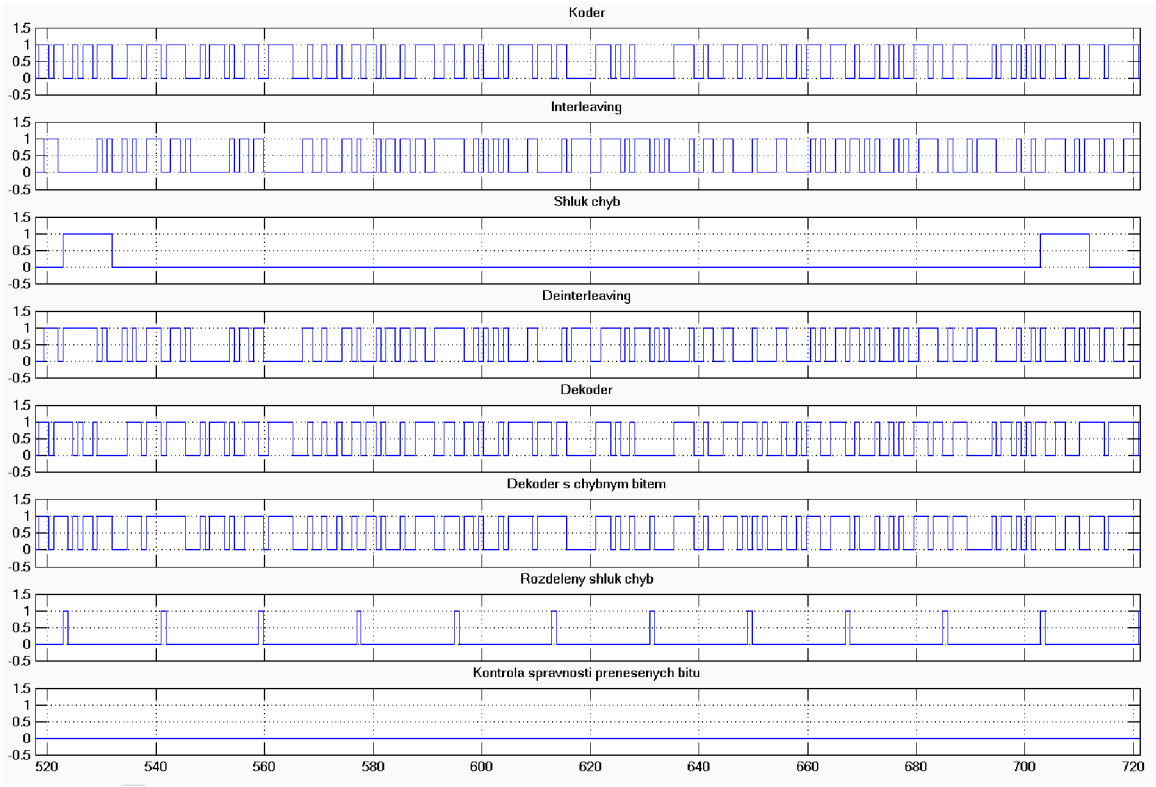
Pokud bychom zkoumali vliv delší chyby, třeba 200 bitů, tak by se tu významně projevili klady konvolučního prokládání. Na následujících třech obrázcích se můžeme podívat na průběhy signálu, jak jsou každým blokem upraveny. Snažil jsme se signál posunou tak, aby všechny tři průběhy od každého kódu byly tak zhruba na stejném rozsahu, aby byl vidět případný rozdíl. Na všech třech obrázcích je vidět průchod signálů celým systémem, a co se signálem ději. V první části výstupu ze simulace je zakódovaný signál pocházející z nezabezpečeného toku dat z Bernoulliho binárního generátoru. Tan vstoupil do kodéru, kde se provedlo zabezpečené a výstup je vidět. Na dalším obrázku je průběh po proložení zakódovaného signálu. V třetím průběhu můžeme vidět shluk chyb, který generujeme pomocí bloku generátor, pro každý kód zvlášť, protože musíme velmi přesně dodržovat délku shluku, a hlavně *ochranný interval A*. Následuje průběh, kdy máme proložená data, na která se nám naindukoval shluk chyb. Na začátku tohoto průběhu je vidět, jak se nám inverzně změnili přenášené informace v grafu Deinterleaving. V dalším průběhu vidíme průběh před dekodérem konvolučního kódu s příměsí shluku chyb. Graf s názvem Dekodér s chybným bitem nám představuje posloupnost zabezpečeným tokem, na kterém je již rozdělena chyba do délky jednoho bitu.



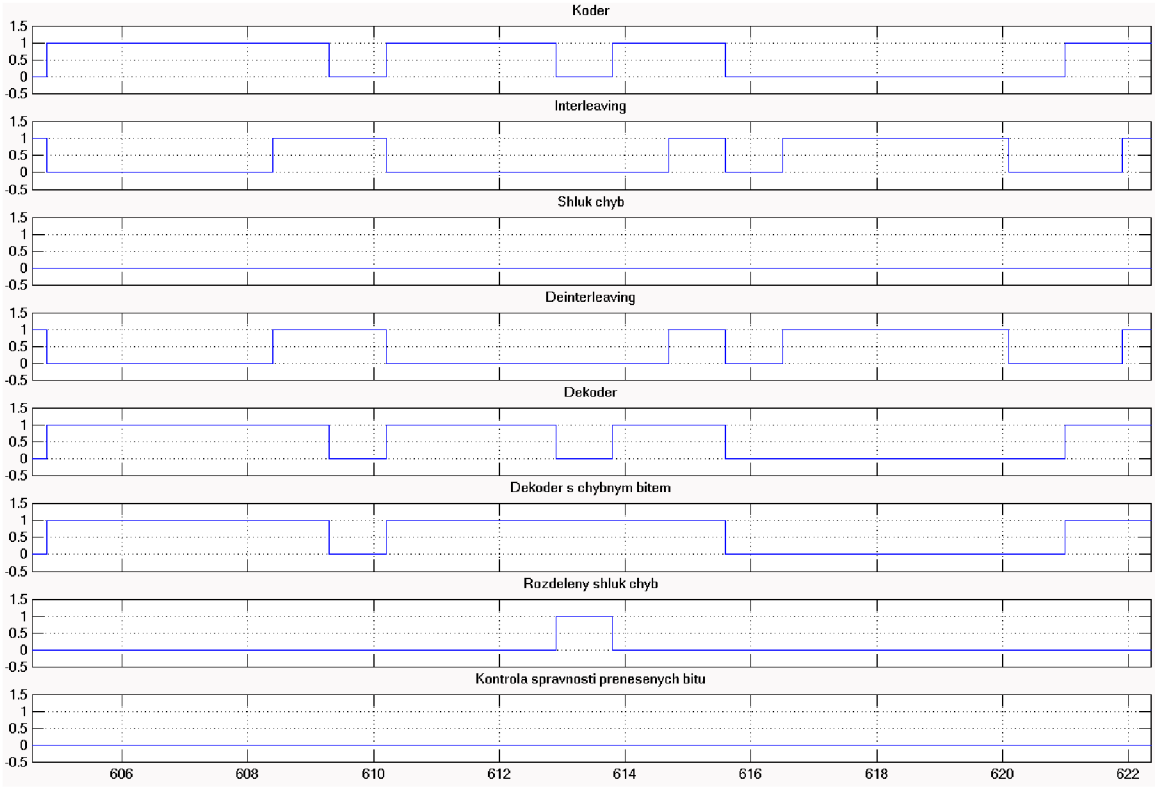
Obr. 7.39 Simulace Iwadariho kódu



Obr. 7.40 Simulace Iwadariho kódu detail



Obr. 7.41 Simulace Berlekampův-Preparatův kód



Obr. 7.42 Simulace Berlekampův-Preparatův kód detail

6. Závěr

Při přenosu informací je kladený velký důraz na kvalitu spojení, aby docházelo během komunikace k minimální změně přenášeného signálů a tím i k efektivnějšímu přenosu informací. Bohužel nikdy nemáme ideální stav, a tak během přenosu se vyskytují jak náhodné, tak i shlukují se chyby. Pokud budeme schopni včasně reagovat na dané podmínky, které se mění, jak v průběhu času, tak i lokalitou přenosových médii, tak dosílíme kvalitnějšího přenosu s minimální chybovostí komunikačního kanálů. Vzniklé chyby mají za následek, že musíme používat nejrůznější technologie, které omezují maximální vytížení přenosového módu. Ať už to jsou např. mechanismy pro opakování přenosu, když dojde ke špatně doručení zprávy, či nejrůznější zabezpečovací techny, případně jejich kombinace. Kdybychom nemuseli používat tyto technologie, tak bychom vytěžovali linky s větší účinností. Ale bohužel musíme používat techniky na zabezpečení přenosu či opakování chybně přeneseného bloku zpráv, protože bez těchto mechanismů bychom nemohli přenášet data na větší vzdáleností či s takovou informační rychlostí jako je přenášíme. Zabezpečovací techniky záměrně vkládají do nezabezpečeného toku dat vhodné kombinace znaků, tím se záměrně zvyšuje nadbytečnost informace, ale díky ní jsme schopni odhalit vzniklé chybně přenesené bity v datovém toku a vhodně na ně reagovat. Využíváme dva mechanismy reagující na chybné bloky. Buď detekovatelní systémy, kde se pouze identifikuje porušený řetězec dat.Druhým způsobem jsou korekční mechanizmy, které umí odhalit vzniklou chybu a pomocí matematických operací a zabezpečovacího kódu vhodně reagovat a porušené úseky opravit na správnou hodnotu. Tyto systémy jsou však velmi náročné na výpočetní techniku reagující na vzniklou situaci, i přesto jsou však nejpoužívanější technikou pro přenos informací. Mají kladený velký důraz na časové zpoždění, které je přísně definované a výpočetní mechanizmy musí stanovené výpočetní aplikace do tohoto intervalu splnit Existuje několik zabezpečovacích kódů, které jsou vhodně použita během přenosu, když se mění situace přenášených dat. Pokud přenášíme informační data po komunikačním kanále různým prostředím (radiové, optické, metalické přenosové médium), tak pro různá prostředí jsou i různá kvalita na přenos signálů. Někde je vznikl chybovosti nízký a někde je až příliš vysoká. Na takový přenos je potřeba využívat několik pružných kódů, které vhodně reagují na danou situaci a jsou schopné se okamžitě této situaci přizpůsobit. Zabezpečovací techniky pomocí zvyšování nadbytečnosti, vkládá do nezabezpečeného bloku dat zabezpečovací kódy a to tak, jak je třeba daný tok zabezpečit. Pokud je při

přenosu minimální vznik chyby, tak se vkládá na stejnou délku malé množství zabezpečovacích dat. Ale pokud je však velká chybovost na vedení, tak je potřeba použít velmi odolné zabezpečovací kódy a tím potlačit vzniklé chyby. Zabezpečovací poměr můžeme vyjádřit jako poměr délky nezabezpečeného bloku délky k ku celkové délce včetně zabezpečení n. Pokud bude poměr příliš malí, tak přenášíme malý objem informačních bitů a velký zabezpečovacích bitů a naopak. Nejideálněji by bylo, kdyby se přenášely dlouhé úseky informačních bitů a minimum zabezpečovacích symbolů. Při vzniků dlouhých úseků chyb nastává ta pravá chvíle pro použití. Prokládání není vhodné pro nenáročni použití, protože by se po finanční stránce „neuživil,, jelikož je drahý na pořízení a taky nám vkládá patřičné zpoždění, které potřebuje pro výpočetní operace. Využití Interleavingu bych viděl zejména v náročných sítích, kde je potřeba maximálně potlačit vznik jakýkoliv chyb. Jak bylo zjištěno při výpočtech, tak metoda prokládání začínala mít smysl, při použití dlouhého ochranného úseku např. u Iwadariho konvolučního systému. Tam již konvoluční metoda prokládání předčila přenosové parametry konvolučního kódu určených k opravě shluku chyb. A zase obráceně, použití konvolučních kódů pro ochranu před shluky je vhodné použít v malých sítích, kde je riziko vzniků chyb malé a tak se tyto kódy snad no „uživí“. Dobrou volbou je i kombinace těchto systémů. Ne vždy však je to možné. Proto je potřeba se tomuto oboru věnovat a získávat stále novější flexibilní kódy, které jsou schopné rychle reagovat na vzniklé změny v přenosu informací, a které budou jednoduché na zhotovení, matematické zpracovaní, minimum paměťových buněk, ekonomické možnosti, a tak dále.

7. Použitá literatura

[1] Němec, K.: *Protichybové kódové systémy*. Publikace v internetovém magazínu Elektrorevue, http://www.elektrorevue.cz/clanky/01027/

[2] NĚMEC,K. *Stromové zabezpečovací kódy I*. Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/clanky/02027/index.html.

[3] NĚMEC,K. *Stromové zabezpečovací kódy II*. Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/clanky/03020/index.html.

[4] NĚMEC,K. *Výběr optimálního protichybového kódu*. Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/cz/download/vyber-optimalniho-protichyboveho-kodu/

[5] NĚMEC,K. *Princip zabezpečení zpráv proti chybám pomocí kódů*. Publikace v internetovém magazinu Elektrorevue, www.elektrorevue.cz/clanky/00029/index.htm.

[6] NĚMEC, K. *Datová komunikace* .2000. Brno: VUTIUM. ISBN 80-214-1652-1

[7] Němec, K.: *Realizace procesu protichybového zabezpečení*. Publikace v internetovém magazínu Elektrorevue, http://www.elektrorevue.cz/clanky/01027/

[8] Němec, K.: *Řešení požadavků kladených na protichybový kódovaný systém*. Publikace v internetovém magazínu Elektrorevue, http://www.elektrorevue.cz/clanky/06017/index.html

[9] Shu, L.; Costello, D.: Error Control Coding - Fundamentals and Applications. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1983. 603s. ISBN 0-13-283796-X.

[10] Purser, M.: Introduction to Error-Correcting. Codes.British Library,London 1995 ISBN 0-89006-784-8.

[11] Křivánek, V., Číka, P.: *Výběr nejvhodnějšího konvolučního kódu - I*. Publikace v internetovém magazinu Access Server, http://access.feld.cvut.cz/view.php?nazevclanku=vyber-nejvhodnejsiho-konvolucniho-kodu-i&cisloclanku=2007020002

[12] Křivánek, V., Číka, P.: Výběr nejvhodnějšího konvolučního kódu - II. Publikace v internetovém magazinu Access Server http://access.feld.cvut.cz/view.php?cisloclanku=2007020003.

[13] Křivánek, V., Číka, P.: *Ochrana dat před shluky chyb, Berlekamp-Preparatův kód.* Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/cz/download/ochrana-dat-pred-shluky-chyb--berlekamp-preparatuv-kod/

[14] Křivánek, V., Číka, P.: *Návrh kodérů a dekodérů umožňující opravu shlukových chyb a jejich simulace*. Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/clanky/06008/index.html

[15] Křivánek, V., Číka, P.: *Simulace shlukových chyb v Matlabu*. Publikace v internetovém magazinu Elektrorevue, http://www.elektrorevue.cz/clanky/06036/index.html#kapitola3

Přílohy:

Příloha 1	Schéma zapojení kodéru Hagelbargerova kódu
Příloha 2	Schéma zapojení dekodéru Hagelbargerova kódu
Příloha 3	Schéma zapojení kodéru Iwadari – Maseryova kódu
Příloha 4	Schéma zapojení dekodéru Iwadari – Maseryova kódu
Příloha 5	Schéma zapojení kodéru Berlekamp - Preparatova kódu
Příloha 6	Schéma zapojení dekodéru Berlekamp - Preparatova kódu