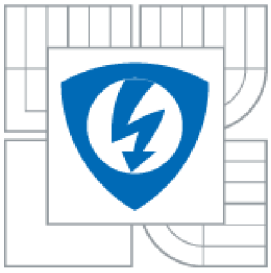
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÍ PŘENOSU DAT BCH KÓDY

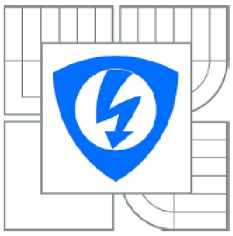
DATA TRANSMITION SECURITY WITH BCH CODES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — JAKUB FROLKA
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL NĚMEC, CSc.
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Jakub Frolka	***ID:***	110408
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Zabezpečení přenosu dat BCH kódy

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte BCH kód, který zabezpečí přenos dat proti t = 4 nezávislým chybám, při informační rychlosti R ? 0,5. Pro tento kód vypracujte podrobný návrh realizace kodeku tohoto kódu. V návrhu realizace využijte skutečnost, že protichybový kodek bude součástí protichybového kódového systému. Ověřte funkční schopnosti tohoto kodeku metodou, kterou považujete pro tento návrh za nejvhodnější.

DOPORUČENÁ LITERATURA:

[1] ADÁMEK,J. Kódování. Nakladatelství technické literatury, Praha 1989.
[2] HOUGHTON,A. Error Coding for Engineers. Kluwer academic Publishers, Boston, Dordrecht, London. 2001.
[3] NĚMEC,K. Protichybové kódové zabezpečení s Bose-Chauhury-Hocquenhemovými kódy. Publikace v internetovém magazínu Elektrorevue, http://www.elektrorevue.cz/clanky/05015/index.htm.

Termín zadání:	29.1.2010	***Termín odevzdání:***	2.6.2010

Vedoucí práce: doc. Ing. Karel Němec, CSc.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce Zabezpečení přenosu dat BCH kódy se zabývá skupinou cyklických kódů, které jsou schopny zabezpečit data v binární podobě proti nezávislým chybám. BCH kódy využívají algebraickou strukturu zvanou Gailosova tělesa. Kódování je stejné jako u cyklických kódů a může být použit kruhový posuvný registr. Dekódování je složitější a lze využít několik algoritmů, v této práci jsou uvedeny tři: Petersonův algoritmus, maticová metoda a Berlekamp-Massey algoritmus. V této práci je uveden popis vlastností BCH kódů, jejich použití ve spojovacích zařízeních a jejich možná realizace těchto kódů. Je vytvořen příklad BCH kódu, který opravuje čtyři nezávislé chyby a je použit pro předvedení způsob kódování a dekódování. Na konci práce je popsána realizace protichybového kodeku pomocí FGPA obvodů.

KLÍČOVÁ SLOVA

BCH kód, kodér, dekodér, oprava chyb, přenos dat, dekódování, FPGA

ABSTRACT

The bachelor thesis Data transmission security with BCH codes deals with class of random error correction cyclic codes which are able to protect data in binary form. BCH codes operate over algebraic structures called Galois fields. Encoding of these codes is same as encoding cyclic codes and can be used cyclic shift register, but decoding is more complex and can be done with many algorithms, in this thesis are mention three: Peterson algorithm, Matrix method and Berlekamp-Massey algorithm. In this thesis are described characteristics BCH codes, their usage in communication devices and their posibility implementation. An example BCH code which is able to correct four independent errors is created. This example is used for presentation of encoding and decoding methods. In the end is described implementation of data protection codec for FGPA devices.

KEYWORDS

BCH code, encoder, decoder, error correction, data transmission, decoding, FPGA

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Zabezpečení přenosu dat BCH kódy jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

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

V Brně dne ....

(podpis autora)

FROLKA, J. *Zabezpečení přenosu dat BCH kódy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 49 s. Vedoucí bakalářské práce doc. Ing. Karel Němec, CSc.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V dnešní době, kdy je potřeba komunikovat, je zapotřebí, aby komunikace byla rychlá a spolehlivá. Komunikací rozumíme předávaní zpráv mezi informačními systémy, nacházející se na různých místech. Do komunikace můžeme zahrnout i ukládání informací a zpráv do paměti nebo jiných záznamových zařízení a jejich pozdější čtení. Ve všech případech komunikace je třeba data přenášet přes nějaké fyzické prostředí (přenosovým kanálem), který často bývá zdrojem rušení (chyb). Takto vzniklé chyby způsobují kvalitu přenášených dat nebo jejich úplné znehodnocení. V této době, kdy se komunikace rozvíjí přes mobilní, satelitní nebo počítačové sítě u kterých jsou vyžadovány velké přenosové rychlosti je také nutné, aby kvalita přenášených dat byla co nejvyšší. Ve většině případů přenosové podmínky nejsou ideální, hlavně u bezdrátového přenosu nebo u metalického vedení, je nutné tyto přenosy zabezpečit proti chybám. Například: Mějme vysokorychlostní kanál s přenosovou rychlostí 1Gb/s který má chybovost 0,001% to by znamenalo, že každou sekundu přijdeme o 10 000 bitů. Pokud bychom přes něj posílali zprávy např. o velikosti 42 000 bitů, tak bychom přibližně skoro každé 4 sekundy ztratili jednu zprávu. Z tohoto příkladu jde vidět, že u vysokorychlostních přenosových kanálů i s malou chybovostí můžeme za určitou dobu ztratit spoustu dat.

Vydáním práce „Matematická teorie sdělování“ v roce 1948, matematikem Claudie Shannonem [1], kde uvedl možnosti vytvoření komunikačního systému, který přenese data efektivně s nulovým počtem chyb. Shannon přišel k závěru, že pokud informační zdroj bude posílat data nižší rychlostí než je komunikační kanál schopen přenášet, tak můžeme přidat ke zprávě několik speciálních bitů, které nám pomůžou snížit pravděpodobnost chybného přenosu [2]. Tato práce způsobila velký vývoj v oblasti teorie informace a vzniku řady kódů, které bylo možné použít na právě vyvíjených prvních elektronických počítačích, nebo rozvoje teorie [1].

Technologický rozmach v oblastech výpočetní číslicové techniky umožňuje využití složitějších algoritmů na protichybové zabezpečení dat, které v dřívějších dobách nešly realizovat. Otevřely se tak nové možnosti pro ověření dříve popsaných teoretických postupů, které dříve nebylo možno realizovat z důvodu nedostatku výpočetního výkonu.

Nejdříve uvedeme několik nejzákladnějších pojmů:

Komunikační kanál – můžeme si ho představit jako nějakou trubku, přes kterou posíláme informace – např. pokud jsou informace text, obrázky nebo video může být kanálem počítačová síť jako Internet.

Kapacita kanálu – maximální objem dat, který můžeme přenést přes komunikační kanál za určitou jednotku času. Např. u ADSL linky maximální kapacita kanálu je 2048 kbit/s – to znamená, že je to maximální možná rychlost přenosu dat, často je opravdová rychlost menší a maximální rychlosti se využívá pouze občas.

Informační zdroj – opět jich může být více druhů např. každá prohlížená webová stránka je uložená na webovém serveru (informačním zdroji). Tento webový server může poskytovat data rychlostí několik Mbit/s, ale jelikož kapacita kanálu je většinou menší než rychlost zdroje dat, proto data nebudeme přijímat tak rychle.

Šum – může způsobit změnu přenášených dat a snížit jejich kvalitu nebo je dokonce znehodnotit. Existuje několik druhů šumu (uměle vytvořený, nebo daný odporem materiálu).

• Cíl práce a jeho rozdělení do kapitol

Cílem této práce je návrh BCH kódu, který bude schopen zabezpečit přenos dat proti $t = 4$ nezávislým chybám, při informační rychlosti $R \geq 0{,}5$. Dále pro tento kód vypracovat podrobný návrh realizace kodeku tohoto kódu. V návrhu realizace využít skutečnosti, že protichybový kodek bude součásti protichybového kódového systému. Ověřit funkční schopnosti tohoto kodeku metodou, považovanou za nejvhodnější pro tento návrh.

V první kapitole je popsána bloková struktura komunikačního systému a popis vlastností jednotlivých bloků. Popsány jsou základní způsoby detekce a opravy chyb.

V Druhé kapitole je rozebrán popis BCH kódů, jejich použití v současných spojovacích zařízeních, zařazení těchto kódů do skupiny korekčních kódů a vysvětlení jednotlivých pojmů. Dále je uveden přehled nezbytných matematických znalostí.

V třetí kapitole je návrh BCH kódu a popis jeho parametrů, který souvisí s vytvořením vytvářecího mnohočlenu.

Ve čtvrté kapitole je popsán kodér BCH kódu a jeho možná realizace.

V páté kapitole je rozebrán dekodér, jeho princip kontroly přenášených dat a jejich následná oprava, pokud došlo při přenosu k chybě.

V šesté kapitole jsou uvedeny způsoby možné realizace kodeku BCH kódu.

V sedmé kapitole jsou popsány prostředky pro návrh a způsoby popisu funkce požadovaného obvodu.

V osmé kapitole je vytvořen příklad BCH kódu opravující $t = 4$ chyby s informační rychlostí $R \geq 0{,}5$.

Poslední kapitola se věnuje návrhu protichybového kodeku v sofwaru Quartus II a jeho následnému ověření funkčnosti. A vytvoření desky plošného spoje.

1 STRUKTURA KOMUNIKAČNÍHO SYSTÉMU

V této kapitole popíšeme blokově strukturu komunikačního systému a následně si popíšeme jednotlivé bloky. Na obr.1.1 je uvedeno blokové schéma struktury komunikačního systému, vycházíme z informací uvedených v [3]. Touto částí poskytneme základní přehled a úvod do částí, které budou dále podrobněji popisovány.

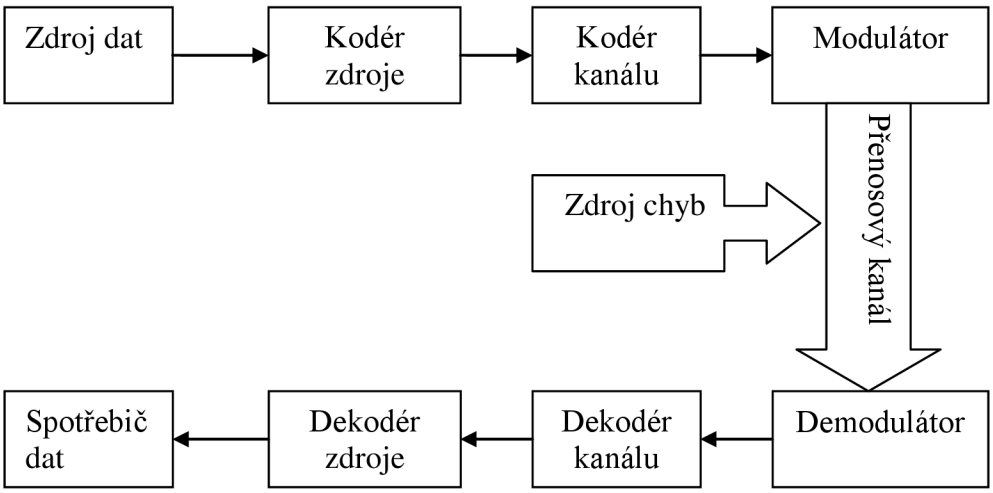
Systém začíná blokem „Zdroj dat“. Zdrojem dat je např. člověk nebo stroj, který potřebuje komunikovat se spotřebičem dat na opačném konci řetězce a vytváří zprávy, které jsou později zpracovávány dalšími bloky, než přijdou ke spotřebiteli dat.

„Kodér zdroje“ – převádí data zdroje na řetězec binárních znaků. Správně navržený kodér zdroje sníží nebo úplně odstraní ze zpráv nadbytečnost a zbytečnost, tím se docílí ke zvýšení efektivity komunikace. Z kodéru zdroje vstupují převedená data v binární podobě do tzv. kodéru kanálu.

„Kodér kanálu“ – převádí řetězec binárních znaků, přidáním řetězce speciálních bitů, na zabezpečenou posloupnost znaků, tuto vzniklou posloupnost bitů často nazýváme kódové slovo [3]. Podle použitého systému zabezpečení se liší vytvoření kódového slova, pro jeho vytvoření mohou, být použity lineární, blokové nebo cyklické kódy a jiné, které popíšeme v další kapitole. Kódové slovo následně vchází do bloku „modulátor“, kde je upravováno (přizpůsobováno) pro přenos přes přenosový kanál nebo pro uložení na paměťové medium.

Na přenosový kanál nám působí rušivé vlivy, označíme je jako šum, které nám mohou přenášené data v kanálu pozměnit podle charakteru a síly šumu. Podle druhu použitého přenosového kanálu (prostředí) má rušení charakteristické vlastnosti. Nejběžnějším přenosovým prostředím jsou např. metalické vedení nebo radiové prostředí. Přítomnost šumu je ve schématu znázorněna blokem „zdroj chyb“.

Po přechodu dat přes přenosový kanál a demodulátoru jsou v dekodéru kanálu detekovány chyby (detekční kódy) nebo chyby opravovány (korekční kódy). Je důležitý výběr správného kodeku pro zabezpečení dat v závislosti na prostředí (přenosovém mediu) přes které budou data přenášena a tím dosáhnout efektivního přenosu. V této práci se zaměříme na využití BCH kódů, které patří do skupiny korekčních kódů pro zabezpečení dat.



Obr. 1.1: Struktura komunikačního systému

1.1 Způsoby detekce a oprav chyb.

Pro zajištění kvalitního a bezchybného přenosu přes komunikační kanál existují dvě hlavní metody:

ARQ – Automatic repeat request – u přijaté zprávy (spotřebičem dat) dochází pouze k detekci, zda nedošlo k poškození dat při přenosu a na základě této detekce se rozhoduje, zde je nutné opakovat přenos požadovaných dat. Pokud byly detekovány chyby, přijímací strana pošle žádost o opakování poškozených dat. Kladnou stránkou této metody je, že nemusíme implementovat složité korekční algoritmy, protože detekce je jednodušší, než korekce dat. Naopak nevýhodou je, že musíme mít možnost obousměrné komunikace – musíme zaslat požadavek od přijímací strany a tím se zvyšuje počet přenesených dat (požadavek o opakované přeposlání + opakované poslání dat.

FEC – Forward error correction – na přijímací straně dochází k detekci i opravě chyb, které nastaly při přenosu přes komunikační kanál. Na přijímací straně (v dekodéru kanálu) je implementován korekční algoritmus. Tento způsob je využíván v případech, kdy není možné komunikovat obousměrně, nebo chceme snížit množství přenášených dat. Do této skupiny patří i skupina korekčních kódů zvané BCH kódy, které v této práci zde budou popisovány.

Tyto uvedené metody mohou být kombinovány, podle potřeb a požadavků na přenosový systém, pro zvýšení jeho efektivity.

2 OBECNÝ POPIS BCH KÓDŮ

V této kapitole se budeme věnovat popisu tzv. BCH kódů, jejich vlastností, využití v současných spojovacích zařízení.

Bose – Chaudriho – Hocquenghem kódy jsou pojmenovány podle jmen jejich autorů, používáme pro ně zkratku BCH kódy. Tyto kódy jsou řazeny mezi lineární blokové cyklické kódy založeny na Hammingových kódech, oproti nim BCH kódy opravují více chyb. BCH kódy byly objeveny panem Hocquenghemem v roce 1959 a nezávisle Bosem a Chaudhurim v roce 1960 [4]. Cyklická struktura těchto kódu byla dokázána panem Petersonem v roce 1960. Nejjednodušší varianta těchto kódů je binární, díky ní mohou být BCH kódy jednoduše implementovány v digitálních zařízeních. Mezi jejich dobré vlastnosti patří velká volitelnost jejich parametrů, dobrý vztah mezi počtem informačních znaků a počtem opravovaných chyb a detailně vypracované dekódovací metody. [1]. První dekódovací algoritmus byl navržen W. W. Petersonem v roce 1960. Následně byly vytvořeny další dekódovací algoritmy pány Berlekampem, Chienem, Forneyem, Masseyem a dalšími [4].

Speciálním případem BCH kódů jsou RS kódy (Reed – Solomonovy kódy). Tyto kódy jsou významné u velkých abeced (nejsou nikdy binární). Jsou důležité např. proto, že pomocí nich lze konstruovat velmi účinné binární kódy [1].

2.1 Použití BCH kódů v současných spojovacích zařízeních

Jak už bylo psáno výše, BCH kódy jsou korekční kódy, které se používají pro zabezpečení dat při jejich přenosu. Jejich použití je jednoznačné, mohou se použít v zařízeních, která potřebují komunikovat a přenášet mezi sebou data. Díky BCH kódům se zajistí oprava chyb vzniklých při přenosu dat přes přenosový kanál nebo ukládání na zálohovací média.

V dnešní době se nejčastěji setkáme při zabezpečování dat se zvláštním případem BCH kódů nazývanou Reed-Solomonovy kódy, které nezabezpečují zprávy po jednotlivých bitech, ale celých bytech.

BCH kódy se používají např. pro zabezpečení ukládaných dat na optických discích CD, DVD a Blu-ray. V diskových polích v zapojení jako RAID 6. Také v modemech xDSL. V digitální satelitní televizi (standard DVB_S2). Pro kódování videa ve videokonferencích (doporučení ITU-T H.261).

2.2 Zařazení BCH kódů a význam jednotlivých pojmů

Jak už bylo uvedeno dříve, BCH kódy řadíme mezi skupinu lineárních blokových cyklických kódů, pracující s binárními daty a jsou účinné zabezpečovací kódy přenosu v komunikačních systémech. Vycházíme z informací uvedených v [3].

Blokové kódy

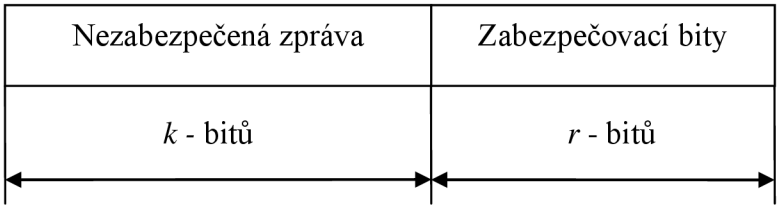
Mají přesně dané rozložení informačních a zabezpečovacích míst v kódové kombinaci. Je-li kódová kombinace systematického kódu o délce n prvcích obsahuje k informačních a $r = n - k$ zabezpečovacích prvků, mluvíme o $(n; k)$ kódu, jeho struktura je uvedena na obr. 2.1. Vždy platí, že $n > k$.

Lineární kódy

Libovolnou kódovou kombinaci lineárního kódu můžeme odvodit jako lineární kombinaci z ostatních kódových informací (využívá se lineární algebry). Bývají zadány pomocí vytvářecí matice G o k řádcích a n sloupcích. Jako řádky matice G jsou použity libovolné lineárně nezávislé kombinace.

Cyklické kódy

Je to druh lineárních kódů $(n; k)$, u kterých ve vytvářecí matici G jsou jednotlivé řádky obsahující stejné prvky, jen posunuty o jedno místo v jednom směru. Tyto kódy mohou být zadány tzv. vytvářecím (generujícím) mnohočlenem G(x). Řád mnohočlenu určuje počet zabezpečovacích prvků $r = (n - k)$. V kódové kombinaci délky n prvků mají cyklické kódy na prvních k místech prvky nezabezpečené zprávy a na zbývajících r místech zabezpečovací prvky.



Obr. 2.1: Rozložení bitů v kódovém slově

2.3 Přehled nezbytných znalostí z lineární algebry

V této části kapitoly budeme vycházet z informací uvedených v [1], [3], [5].

Operace spojené se zabezpečením nezabezpečených signálových prvků BCH kódy se provádí pomocí prvků tzv. Galoisova tělesa $GF(p^r)$, kde p je základ číselné soustavy (odpovídá počtu stavů signálu) a r je stupeň rozšíření Galoisova tělesa. Je tvořeno konečným počtem prvků $n = p^r$, což odpovídá celkovému počtu kódových prvků v mnohočlenu zabezpečené zprávy a vznikne rozšířením konečného tělesa Z_p.

2.3.1 Konečné těleso Z_p

je algebraická struktura určená počtem svých prvků a to je mocnina prvočísla p. Tato struktura je tvořena zbytky po dělení celých kladných čísel prvočíslem p, kde p určuje základ soustavy. V praxi je významné konečné těleso Z_2 tvořené množinou prvků

{0, 1}. Algebraická operace součtu „+“ je známá operace nonekvivalence (XOR) a násobení „·“ jako logický součin (AND). V tělese Z_2 je součet i rozdíl dvou prvků stejná operace. Operace XOR a AND jsou definovány v tabulce 2.1.

Tab. 2.1: XOR a AND v tělese Z_2

XOR		=
0	0	0
0	1	1
1	0	1
1	1	0

AND		=
0	0	0
0	1	0
1	0	0
1	1	1

2.3.2 Galoisovo těleso GF (p^r)

vznikne rozšířením konečného tělesa stupněm *r*. Rozšířením vznikne množina vektorů o *r* prvcích, kde každý prvek je ze Z_p. Celkový počet prvků Galoisova tělesa je p^r. Pro binární kódy se používá Galoisovo těleso GF(2^r), které vznikne rozšířením tělesa Z_2. Toto těleso můžeme vyjádřit pomocí zbytků po dělení mnohočlenu ze Z_2 určitým mnohočlenem (vytvářecím mnohočlenem), pro který platí:

- Je nerozložitelný v uvažovaném okruhu mnohočlenů
- je stupně *r*
- dělí beze zbytku ($x^n - 1$) a žádný jiný dvojčlen nižšího stupně tohoto tvaru.

Např.: Galoisovo těleso GF (2^4) je množina čtveřic prvků ze Z_2. Určíme ji pomocí mnohočlenu z rozkladu dvojčlenu

$$(2^{15} + 1) = (x + 1) \cdot (x^2 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) \cdot (x^4 + x + 1) \cdot (x^4 + x^3 + 1)$$

Při určování Galoisova tělesa GF (2^4) se hodí pouze mnohočleny stupně *r* (zde *r* = 4). Zvolený mnohočlen se nazývá vytvářecí mnohočlen GF (2^4) a zpravidla se označuje $G(x)_{GF}$.

Algebraické operace nad Galoisovým tělesem:

- Součet „+“ se provádí jako sčítání po členech mnohočlenu: $A(x) \oplus B(x) = C(x)$, $c_i(x) = a_i(x) \oplus b_i(x)$, kde $A(x), B(x), C(x)$ jsou mnohočleny vytvářející prvky GF (p^r).
- Násobení „·“ je určeno jako zbytek po dělení součinu dvou prvků vytvářecím mnohočlenem GF (2^4), tj $G(x)_{GF}$.

V Galoisově tělese se nenulové prvky vyjadřují ve tvaru mocnin α^0, α^1, α^2,…, α^{r-1}. Mocnina určuje prvek buďto přímo, nebo prvek vyjádříme pomocí zbytku R[i] po dělení tohoto prvku s „velkým“ exponentem vytvářecím mnohočlenem tělesa $G(x)_{GF}$ za předpokladu, že prvek α je primitivní. Tato podmínka je zajištěná, použijeme-li pro generování $GF(2^r)$ primitivní mnohočlen.

3 NÁVRH BCH KÓDU

Pro návrh BCH kódu potřebujeme nejdříve navrhnout vytvářecí mnohočlen (polynom). Při návrhu vytvářecího mnohočlenu se podle [4] vychází z Bose-Chaudhuriho teorému.

3.1 Sestavení vytvářecího mnohočlenu G (x)

1. Nejdříve je zvolen primitivní mnohočlen a sestrojeno Galoisovo těleso GF (q^m)
2. Dále musí být nalezeny minimální mnohočleny $m_j(x)$ pro α^j, kde $j = 1,2,\ldots,2t$, a α je symbol Galoisova tělesa.
3. Vytvářecí mnohočlen je dán nejmenším společným násobkem (LCM) minimálních mnohočlenů jak je uvedeno v rovnici (3.1)

$$G(x) = \text{LCM}\{m_1(x) + m_2(x) + \cdots + m_{2t-1}(x)\} \quad (3.1)$$

Kód BCH (n; k) Podle [4] má následující parametry

maximální délka kódového slova v bitech $n = 2^{\text{m}} - 1$ (3.2)

počet informačních bitů k v kódovém slově $k \geq n - mt$ (3.3)

minimální Hammingova vzdálenost $d_{min} = 2t + 1$ (3.4)

Důležitým parametrem kódu je také infomační rychlost kódu definovaná jako

$$R = \frac{k}{n} \quad (3.5)$$

V příloze A uvádíme všechny existující binární BCH kódy délky n, převzato z [1]

$n = 2^m - 1 = 7, 15, 31, 63$ a 127

Pokud chceme navrhnout BCH kód, který opravuje t chyb, z rovnice (3.4) vypočítáme Hammingovu vzdálenost d_{min}. Z přílohy A zjistíme počet informačních znaků a délku kódového slova. Z rovnice (3.3) zjistíme řád primitivního mnohočlenu i stupeň Galoisova tělesa m. Prvky Galoisových těles generovány jednotlivými primitivními mnohočleny jsou uvedeny v tabulkách ve spoustě literatur zabývajících se kódováním informace např. [1], [4]. V příloze B je přiložena tabulka prvků Galoisova tělesa $GF(2^6)$ vytvořené mnohočlenem $G(x)_{GF} = x^6 + x + 1$, kterou později použijeme pro vytvoření konkrétního příkladu.

Minimální mnohočleny $m_j(x)$ a jejich kořeny pro jednotlivé Galoisova tělesa jsou uvedeny např. [4]. Podle [5] pokud bychom použili minimální mnohočlen m_1, který má kořeny α^1, α^2, α^4, α^8 a mnohočlen m_2 (x) s kořeny α^3, α^6, α^9, α^{12}, pak jejich součin

$G(\text{x}) = m_1(x) \cdot m_2(x)$ má kořeny $\alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^6, \alpha^8, \alpha^9, \alpha^{12}$. Musíme proto vytvořit vytvářecí mnohočlen $G(x)$ součinem několika nerozložitelnými polynomy $m_j(x)$ takových, aby výsledný mnohočlen $G(x)$ měl po sobě jdoucích kořenů, jak vyžaduje Bose-Chaudhuriho teorém tj. $2t$ po sobě jdoucích kořenů.

4 KODÉR BCH KÓDU

Jak už bylo psáno výše, BCH kódy patří do skupiny cyklických kódů, proto jejich kodér můžeme realizovat obvyklým způsobem pro cyklické kódy, tj. pomocí děličky mod $G(x)$ [1], [2], ve které se odvodí zabezpečovací část pro mnohočlen zabezpečené zprávy $F(x)$ viz [3].

BCH kód je vyjádřen vytvářecím (zabezpečovacím) mnohočlenem $G(x)$.

Označme:

$P\ (x)$ – mnohočlen nezabezpečené zprávy.

$G\ (x)$ – vytvářecí (generující) mnohočlen označující zabezpečovací kód.

$M(x)$ – mnohočlen podílu.

$R\ (x)$ – mnohočlen zbytku (po dělení).

$F\ (x)$ – mnohočlen zabezpečené zprávy.

$J\ (x)$ – mnohočlen přenesené zprávy.

$E(x)$ – chybový mnohočlen.

Podle [3] můžeme způsob zabezpečení matematicky popsat

$$\frac{P(x)\cdot x^{(n-k)}}{G(x)} = M(x) + \frac{R(x)}{G(x)} \tag{4.1}$$

Rovnici (4.1) jednoduchou úpravou přepíšeme na

$$P(x)\cdot x^{(n-k)} = M(x)\cdot G(x) + R(x) \tag{4.2}$$

Přepsáním pomocí algebry mod2 dostaneme

$$F(x) = P(x)\cdot x^{(n-k)} + R(x) = M(x)\cdot G(x) \tag{4.3}$$

Z rovnice (3.7) vidíme, že $F(x)$ je dělitelná $G(x)$ bezezbytku. Pokud přenesená zpráva se rovná zprávě zabezpečené (vyslané) můžeme zapsat.

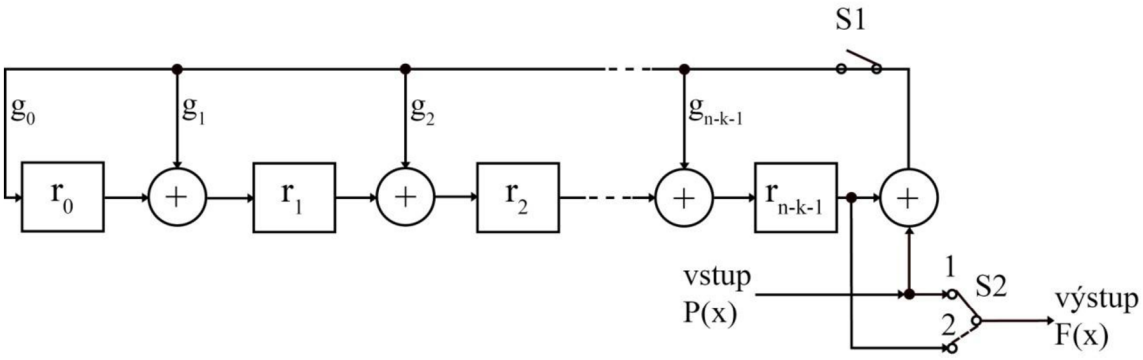
$$J(x) = F(x) \tag{4.4}$$

Tento postup můžeme shrnout do tří kroků:

1. Vynásobíme mnohočlen nezabezpečené zprávy $P(x)$ členem $x^{(n-k)}$, neboli ke zprávě $P(x)$ přidáme $(n-k)$ nul (tím zvýšíme $P(x)$ řád).
2. Dále takto upravený mnohočlen vydělíme vytvářecím mnohočlenem $G(x)$, zbytek po dělení je $R(x)$.
3. Zbytek po dělení přičteme k mnohočlenu získaného v bodě 1. $P(x)\cdot x^{(n-k)}$ tím získáme mnohočlen zabezpečené zprávy $F(x)$.

4.1 Realizace Kódéru pomocí děličky $G(x)$

Podle [3] výše popsaný postup zabezpečení, který je založen na dělení vytvářecím mnohočlenem $G(x)$. Uskutečňuje se v děličce mod $G(x)$, která je tvořena kruhovým registrem se zpětnými vazbami se sčítačky modulo2 vhodně umístněny podle vytvářecího mnohočlenu $G(x)$ vybraného kódu. Příklad zapojení posuvného registru je uveden na obr. 4.1, základní podoba převzata z [4], [6].



Obr. 4.1: Dělička mod $G(x)$

Posuvný kruhový registr obsahuje $(n - k)$ paměťových buněk $(r_0 \dots r_{n-k-1})$. Potřebný počet těchto buněk můžeme zjistit z řádu vytvářecího mnohočlenu $G(x)$.

Z vytvářecího mnohočlenu $G(x)$ zjistíme, před kterými členy je znaménko „+“ a před stejné členy v posuvném registru vložíme sčítačku mod2. Nejvyšší řád vytvářecího mnohočlenu již nemá svoji buňku v posuvném registru a sčítačka, která mu patří, je umístěna před buňkou r_0.

Popis funkce kodéru uvedeného na Obr. 4.1:

Na začátku do kodéru vstupuje k informačních bitů. Spínač S2 je v poloze 1, to znamená, že vstupující informační bity se dostávají i na výstup v nezměněné podobě. Spínač S1 je po dobu vstupu informačních bitů sepnut. Díky tomuto nastavení spínače S1 se data dostávají do jednotlivých buněk posuvného registru. Tento děj se opakuje od 1 do k cyklů.

V cyklech od $k + 1$ do n probíhá vyprazdňování posuvného registru a všechna data jsou přenášena na výstup kodéru. To znamená, že spínač S1 je rozepnut a přepínač S2 je v poloze 2. Celkový počet cyklů posunutí v posuvných registrech je n. Po provedení celého postupu je na výstupu připraveno kódové slovo (zabezpečená zpráva $F(x)$). Dále stejný postup se opakuje s další bitovou posloupností. Výsledek neboli zabezpečené kódové slovo $F(x)$ má strukturu jak bylo znázorněno na obr. 2.1.

5 DEKODÉR BCH KÓDU

V této kapitole budou popsány metody pro dekódování BCH kódu. Proces dekódování BCH kódu je na rozdíl oproti kódování výpočetně náročnější. U dekódování tzv. korekčního kódu, ke kterým patří i námi rozebírané BCH kódy je myšleno vytvoření z přijaté zprávy, ve které mohlo dojít vlivem rušení ke změnám, zprávu stejnou s vyslanou zprávou. Za předpokladu, že nebyla překročena zabezpečovací schopnost kódu. BCH dekodér pro kód s plánovanou vzdáleností (často bývá stejná jako Hammingova minimální vzdálenost uvedená v rovnici (3.4)) [1], dokáže opravit až *t* – násobně chyb.

Budeme používat stejné označení, které bylo uvedeno v kapitole 4. Mějme vyslané slovo $F(x)$, které bylo při přenosu zatíženo chybou. Příjmem tedy pozměněné kódové slovo $J(x)$. Potom podle [3] můžeme napsat vztah:

$$J(x) = F(x) + E(x) \tag{5.1}$$

Pokud tedy přijatý mnohočlen $J(x)$, který obsahuje *t* nebo méně chyb, snažíme se najít chybový mnohočlen $E(x)$. Pokud v přijatém mnohočlenu zprávy $J(x)$ je více chyb, než *t* byla překročena zabezpečovací schopnost kódu a tím proces dekódování skončí neúspěšně. Dále je pak už zapotřebí jak jsme uváděli v kapitole 1.3 využít např. ARQ pro zažádání o znovu poslání zprávy.

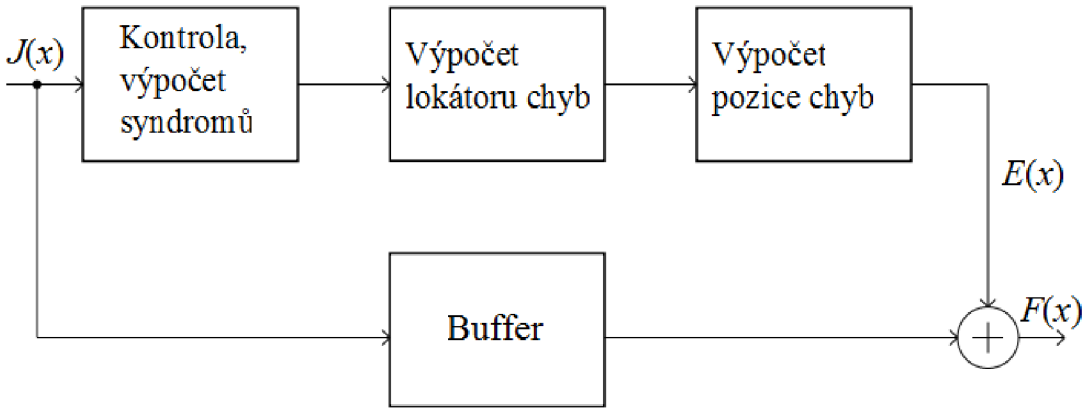
Jestliže počet chyb nepřekročil zabezpečovací schopnosti kódu, můžeme popsat dekódování ve čtyřech bodech [1] a [5].

1. Kontrola správnosti a výpočet syndromů
2. Výpočet mnohočlenu pozice chyb – lokátor chyb
3. Výpočet pozice chyb
4. Oprava chyb

Tyto body můžeme znázornit obr.5.1

Jak už bylo popsáno v kapitole 2, pro dekódování bylo navrženo hodně dekódovacích algoritmů (např. Petersonův algoritmus, Maticová metoda a Berlekamp-Massey algoritmus), ty se z pravidla liší ve druhé fázi a tou je stanovení mnohočlenu pozice chyb. Dále popíšeme metody:

1. Petersonův algoritmus
2. Maticová metoda
3. Berlekamp-Massey algoritmus



Obr. 5.1 Schéma dekódování

5.1 Kontrola správnosti

Vyjdeme ze vztahu (5.1) [4], kdy přijaté slovo můžeme zapsat ve tvaru

$$J(x) = J_0 + J_1 x + \ldots + J_{n-1} x^{n-1} \tag{5.2}$$

Nejdříve musíme zjistit, zda vůbec k nějakým chybám během přenosu došlo. To můžeme zjistit tak, že postupně dosazujeme kořeny vytvářecího mnohočlenu $G(x)$ do mnohočlenu přijaté zprávy $J(x)$. Tím získáme syndromové rovnice S_j.

$$S_j(\alpha^j) = S_{j+1}(\alpha^{j+1}) = \ldots = S_{2t}(\alpha^{2t}) \tag{5.3}$$

Pokud při přenosu, nedošlo k chybám je výsledek každé rovnice S_j nulový. Pokud k chybám došlo, vznikne nám soustava syndromových rovnic. Pro BCH kód opravující t-chyb bude $2t$ rovnic. Z kterých je v následujících krocích možné stanovit počet a polohu chyb a následně je opravit.

5.2 Výpočet mnohočlenu pozice chyb – lokátoru chyb

5.2.1 Petersonův algoritmus

V této fázi dekódování musíme převést soustavu syndromů S_j na jeden mnohočlen – tzv. lokátor chyb.

Podle [5] a [7] dosadíme hodnoty S_j do soustavy vazebních rovnic v následujícím tvaru

$$S_j \cdot z_v + S_{j+1} \cdot z_{v-1} + \ldots + S_{j(v-1)} \cdot z_1 + S_{j+v} = 0, \tag{5.4}$$

kde všechny S_j jsou známy pro $m_0 \leq j \leq (m_0 + (2\_ - 1) - v)$, (5.5)

kde v je počet chyb, které skutečně nastaly. Pro začátek zvolíme $v = t$.

Vztahem (5.5) je také určen počet vazebních rovnic.
V této soustavě vazebních rovnic se určí počet lineárně **nezávislých** po sobě jdoucích rovnic, jejich počet určí hodnotu v (počet chyb, které skutečně nastaly).

Rovnice jsou lineárně **závislé**, pokud můžeme jednu rovnici vytvořit z druhé lineární operací. (např. vynásobením nějakým prvkem). Posouzení lineární závislosti této soustavy není přímo možné, proto se používá, sčítání jednotlivých rovnic (tím vypadávají neznámé z_v, např. z_1 atd.).

K posouzení lineární závislosti lze využít matic, po přepsání soustavy rovnic do matice. Pokud v matici budeme používat vhodné úpravy a některý řádek se nám vynuluje, to znamená, že řádek je lineárně závislý.

Pokud $v < t$ tak v soustavě rovnic položíme veličiny $z_{v+1}; z_{v+2}; \ldots; z_t$ rovny nule a řešením prvních v rovnic získáme veličiny $z_1; z_2; \ldots; z_v$. (K řešení soustavy rovnic se můžou opět použít matice a pomocí Gaussovy eliminační metody získat jednotlivé veličiny $z_1; z_2; \ldots; z_v$). Tyto veličiny určují mnohočlen chyb, položíme-li jej roven nule, obdržíme rovnici polohy chyb.

$$X^v + z_1 \cdot X^{v-1} + \ldots + z_v = 0 \tag{5.6}$$

5.2.2 Maticová metoda

podle [1] má chybový mnohočlen tvar

$$k(x) = (1 - k_1 x) \cdot (1 - k_2 x) \cdot \ldots \cdot (1 - k_p x), \tag{5.7}$$

kde p je počet chyb, které nastaly v přijaté zprávě. Tento polynom přepíšeme podle [1] na polynom

$$k(x) = k_0 + k_1 x + k_2 x^2 + \ldots + k_p x^p, \tag{5.8}$$

první koeficient $k_0 = 1$ (protože $k\,(0) = 1$).

Pro koeficienty $k_1, k_2, \ldots, k_p$ platí matice

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ S_2 & S_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ S_4 & S_3 & S_2 & S_1 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{2p-2} & S_{2p-1} & \cdots & \cdots & \cdots & \cdots & S_p & S_{p-1} \end{bmatrix} \times \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ \cdots \\ k_p \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ \cdots \\ S_{2p-1} \end{bmatrix}. \tag{5.9}$$

Tuto matici však můžeme sestrojit, pouze pokud známe počet chyb p, které při přenosu nastaly. V [1] je uvedeno tvrzení, že pro každé $q = 1, 2, \ldots, t+1$ platí následující matice

$$M_q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ S_2 & S_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ S_4 & S_3 & S_2 & S_1 & 1 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{2q-2} & S_{2q-1} & \cdots & \cdots & \cdots & \cdots & S_q & S_{q-1} \end{bmatrix}, \tag{5.10}$$

která má determinant roven nule, pokud $q > p + 1$, ale matice M_p i M_{p+1} mají nenulový determinant. Po výpočtu determinantů je tedy jasné, kolik chyb ve skutečnosti vzniklo.

Podle [1] lze koeficienty lokátoru chyb $k(x)$ nález řešením soustavy

$$M_q = \begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ \cdots \\ k_q \end{bmatrix} = \begin{bmatrix} S_1 \\ S_3 \\ S_5 \\ \cdots \\ S_{2q-1} \end{bmatrix}, \tag{5.11}$$

kde q je největší z čísel 1,2,…, t *takové*, že $\det M_q \neq 0$. To znamená, že nejprve položíme $q = t$ a potom:

- Je-li $\det M_q \neq 0$, řešíme uvedenou soustavu a dostaneme koeficienty lokátoru chyb.
- Je-li $\det M_q = 0$, položíme $q = t - 2$ a postupujeme stejně, tzn. $\det M_{t-2} \neq 0$, řešíme uvedenou soustavu a pokud $\det M_{t-2} = 0$, položíme $q = t - 4$, atd.
- Jestliže všechny determinanty vyjdou nulové a syndrom je nulový, pak došlo k velké chybě, kterou dekodér není schopen rozpoznat.

5.2.3 Berlekamp-Massey Algoritmus

Tato metoda využívá iterační algoritmus pro výpočet lokátoru chyb (chybového mnohočlenu). Chybový mnohočlen $k(x)$ je nalezen po $t - 1$ rekurzivních iterací. Během každé iterace *r*, je řád chybového mnohočlenu $k(x)$ inkrementován o 1. Touto metodou je řád $k(x)$ počtem chyb, které jsou označeny kořeny $k(x)$. Počet iterací *r* je vždy $r \geq v$, kde v značí počet chyb, které skutečně nastaly při přenosu a odchylka d_r (uváděna v [4], [7] jako discrepancy) je většinou rovna nule. Pokud odchylka vypočítaná vztahu podle [8]

$$d_r = \sum_{i=0}^{t} k_i \cdot S_{2r-i} \tag{5.12}$$

není nulová pro $r < v$ je v dalším kroku inkrementován řád chybového mnohočlenu $k(x)$ a přepočítávají jeho koeficienty. Označíme-li řád $k(x)$ *i*-tého kroku iterace l_i a vyjde odchylka v dalších $t - l_i - 1$ krocích nulová je l_i konečný řád $k(x)$ [4]. Na tomto je založeno: pokud je $v \leq t$, tak pouze $t + v$ kroků iterace je potřeba pro nalezení chybového mnohočlenu $k(x)$. [4]

Pokud po skončení výpočtů je řád $k(x)$ větší než *t,* přijaté slovo $J(x)$ obsahuje více chyb než je kód schopen opravit a proces dekódování skončí neúspěšně [4].

Podrobnější popis Berlekamp-Massey algoritmu je uveden v [4], [7] a [8]. Pro jednodušší implementaci je v [8] uveden popis v programovacím jazyku Pascal.

5.2.4 Výpočet pozice chyb

Výpočet pozice chyb se provádí postupným dosazováním nenulových prvků z Galoisova tělesa $GF(q^m)$, (které bylo použito pro tvorbu vytvářecího polynomu), do rovnice pozice chyby. Pokud rovnice pozice chyby bude splněna, ty kořeny označují svým exponentem pozici chyby a tedy i chybový mnohočlen $E(x)$. Tento proces se nazývá Chien search. [4]

5.2.5 Oprava chyb

Je uskutečňována součtem mod2 přeneseného mnohočlenu $J(x)$, který je uložen v paměti dekodéru (na Obr. 5.1 označena jako Buffer) s chybovým mnohočlenem $E(x)$.

Zjednodušeně řečeno chybné bity v $J(x)$, na pozicích, které byly zjištěny v kapitole 5.3 invertujeme.

6 MOŽNÉ REALIZACE KODEKU BCH

V předchozích kapitolách jsme se věnovali především teoretickému popisu BCH kódů a jejich návrhu. V této kapitole uvedeme možnosti realizace kodéru a dekodéru BCH kódu součastnými technickými prostředky.

6.1 Rozdělení možných realizací

Nejjednodušším rozdělením realizace kodeku může být:

- Softwarová realizace
- Hardwarová realizace

Obě možnosti mají své výhody a nevýhody, které se budeme dále snažit přiblížit.

Pokud je potřeba vybrat způsob realizace, je třeba znát, pro jaký účel bude kodek použit, jaký typ přenosu dat bude kód zabezpečovat (druh přenosového kanálu).

V závislosti na těchto informacích může být výhodnější realizace hardwarová, pokud bychom např. chtěli integrovat zabezpečovací systém (kodek) do jiného hardwarového celku. Pod hardwarovou realizací si v dnešní době představíme spíše návrh ve specializovaném hardwarově orientovaném jazyce jako je např. HDL (Hardware Description Language) a následnou implementaci do specializovaného hardwaru, nežli zapojováním diskrétních součástek.

6.1.1 Softwarová realizace

Pod softwarovou realizaci si můžeme představit naprogramování kodéru a dekodéru BCH kódu ve vyšším programovacím jazyce např. C nebo C++. Jeho možným použitím by pak bylo např. pro zabezpečení komunikace přes sériový port, nebo zabezpečení ukládaných dat.

Výhody:

- Jednoduchá úprava programu v průběhu testování
- Není nutné vytvářet hardware

Nevýhody:

- Pro realizaci nutno PC (rozměry)

6.1.2 Hardwarová realizace

Informace převzaté z [9]

- Zapojení tvořené číslicovými integrovanými obvody
- Mikrokontroléry
- Aplikačně specifické integrované obvody (ASIC)
- Programovatelnými logickými obvody

Zapojení tvořené číslicovými integrovanými obvody

Tato realizace pomocí číslicových obvodů byla rozšířena v 70. letech 20. století, kdy vznikali první integrované obvody. Pomocí těchto obvodů lze realizovat malý rozsah jednodušších funkcí. Pro realizaci BCH kodeku by tato metoda byla velice složitá.

Výhody:

- Rychlá reakce
- Malá spotřeba – pro obvody CMOS
- Snadná dostupnost

Nevýhody:

- Velké množství pouzder – pro složité obvody nevhodné
- Změna funkce – složitá, při úpravě zapojení nutná změna plošného spoje

Tato možnost realizace kodeků je nejméně vhodná pro testování z důvodu velké složitosti změn v obvodech (funkcích).

Použití mikrokontrolérů

Mikrokontroléry, jinak též označovány jako jednočipové mikropočítače, obsahují v jediném pouzdře několik základních částí:

- Řadič s aritmetickou jednotkou. Obvyklá délka slova 4,8,16 nebo 32 bitů.
- Paměť programu – většinou typu EPROM nebo Flash. Pro neměnné použití (sériovou výrobu) se používá paměť typu ROM
- Paměť dat – typu Read/Write
- Periferní obvody pro vstup a výstup dat.

Výhody oproti využití obvodů základních řad:

- Lze realizovat složité algoritmy
- Univerzálnost dovolující hromadnou výrobu procesorů
- Možnost měnit vykonávanou funkci bez nutnosti měnit zapojení (plošného spoje)

Nevýhody:

- Nižší rychlost reakce – musí proběhnout několik instrukčních cyklů – nevhodné pro řízení v reálném čase
- Složitost – některé procesory nutno programovat v programátoru.

Obvody ASIC

Představují vyšší stupeň realizace číslicových subsystémů vhodných pro hromadnou výrobu. Jsou levnější než programovatelné obvody, pokud rozpočítáme větší počáteční náklady, jejich příprava výroby je delší a složitější. Jsou vzájemně kompatibilní s obvody PLD a FPGA.

Programovatelné obvody

jsou podobné jako číslicové obvody, jsou však realizovány na jednom čipu. Jejich funkci můžeme změnit přeprogramováním. Reakční rychlost mají jako číslicové obvody, ale oproti mikrokontrolerům mají menší univerzálnost.

Rozdělení programovatelných obvodů:

- PLD – (Programmable Logic Devices) – Programovatelné logické obvody: Základem makrobuněk je dvoustupňová struktura (AND a OR) používaná při realizaci kombinačních logických funkcí zapsaných ve tvaru součtu součinů, doplněná přídavnými prvky, jako jsou klopné obvody a podobně. Jsou vhodné pro jednodušší až středně složité aplikace.
- CPLD – (Complex PLD) – obsahují několik (2 až 16) bloků složených z makrobuněk. Bývají vybaveny mnoha dalšími přídavnými prvky, které umožnují další funkce. Představují mezistupeň mezi obvody PLD a FPGA.
- FPGA – (Field Programmable Gate Arrays) – tvořeny polem konfigurovatelných bloků, které můžeme přirovnat k malým blokům obvodů CPLD, obsahující malé generátory logických funkcí s pamětmi, klopnými obvody a spoustou jiných specializovaných prvků, které slouží k efektivnímu vytvoření často používaných logických buněk do celků pro vytvoření složitějších kombinačních funkcí. Mají nejvíce obecnou strukturu – umožněno realizovat nejsložitější číslicové systémy.

Výhody:

- Rychlá reakce
- Snadná modifikace – návrh v HDL jazyce, jednoduchá změna funkce obvodu
- Univerzálnost
- Práce ve vývojovém prostředí usnadňující návrh obvodu

Nevýhody:

- Vyšší cena vývoje
- Znalost speciálních nástrojů

7 PROSTŘEDKY PRO NÁVRH S OBVODY FPGA

Pro výhody uvedené v předchozí kapitole byla pro návrh kodeku BCH kódu zvolena implementace do obvodů FPGA, v této kapitole popíšeme obecný postup a prostředky při návrhu aplikací pro obvody FPGA.

7.1 Způsoby popisu aplikace

V dnešní době se používá více způsobů popisů funkcí aplikace. Jedním způsobem je popis tzv. pomocí schématu. Kdy se využívá editoru schémat, který bývá součástí vývojových prostředí, nebo nějakého obecně rozšířeného editoru, jako je např. OrCAD [9]. Tento způsob popisu má nevýhodu pro jeho špatnou přenositelnost (je více druhů popisu pro vstup schémat).

Nerozšířenějším způsobem popisu jsou jazyky HDL. Dva z nich VHDL a Verilog jsou standardy IEEE [9]. V současné době jsou některé systémy schopny zpracovat i popis pomocí jazyka C.

Jazyky HDL umožňují popsat funkci obvodu v několika úrovních abstrakce, jako jsou například

- Úroveň RTL – u synchronních subsystémů je jejich struktura složena z registrů, mezi nimiž jsou kombinační bloky. Charakteristické je, že lze u tohoto popisu rozeznat uvedenou strukturu.
- Úroveň behaviorální

Podle [9] HDL popis můžeme také rozdělit

- Behaviorální styl popisu – chápeme tak popis vyjadřující popisovaného objektu, např. inkrementaci čítače popíšeme zápisem $cnt <= cnt + 1$.
- Strukturální styl popisu – popis propojení komponentů (čítač se skládá z klopných obvodů – popisujeme jejich propojení)

Při složitějších konstrukcích jsou jazyky HDL (textový popis) doplňovány pro přehlednost grafickými. Tyto způsoby popisu jsou ve stylu např. blokových schémat, stavových diagramů a tabulkových editorů.

7.2 Typický postup návrhu

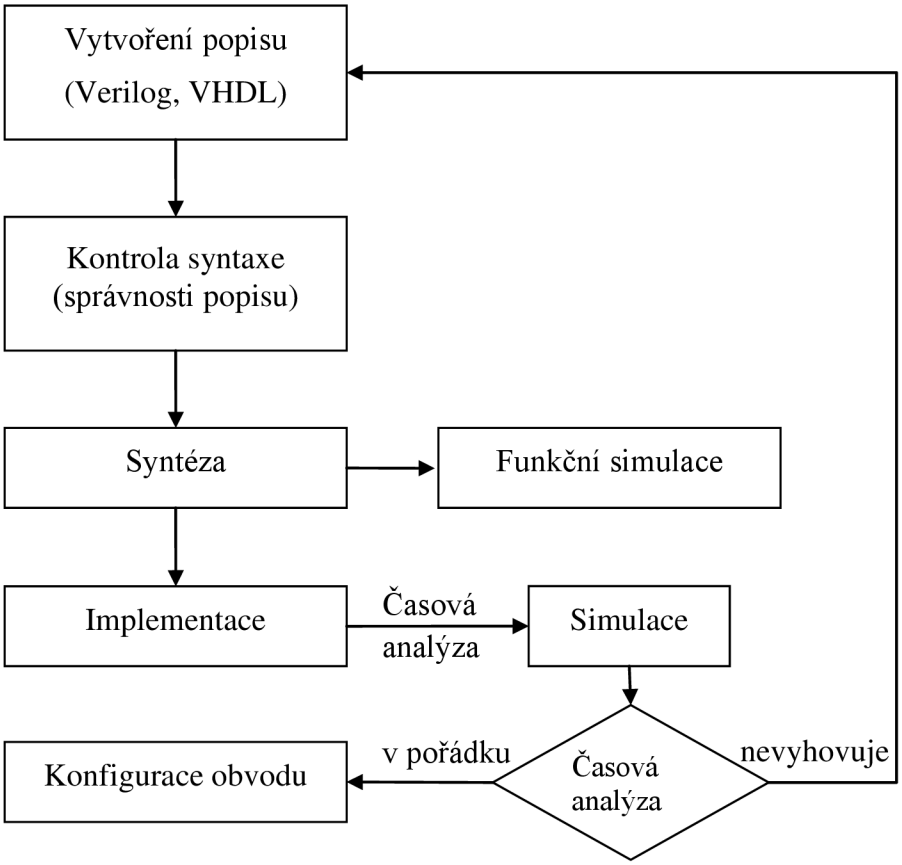
Při návrhu aplikací pro programovatelné logické obvody je postup znázorněn na obr. 7.1.

Popis jednotlivých kroků

1. Vytvoření popisu vyvíjené aplikace – návrh modelu obvodu s použitím nějakého HDL jazyka (např. Verilog nebo VHDL), nebo schematického editoru.
2. Kontrola syntaxe – kontrola správnosti popisu.
3. Syntéza – v tomto kroku se vytváří (tzv. netlist) zapojení obvodových prvků,

jako jsou logické členy, registry atd., které vykonávají vyžadovanou funkci. Provádí se i minimalizace (obdoba Karnaghových map) a optimalizace.

4. Funkční simulace – provádí se po syntéze, kdy se kontroluje funkčnost návrhu (bez kontroly časových parametrů – požadavků)
5. Implementace - provádí se několik kroků
    a. Mapping – přiřazení obvodových prvků (výsledky syntézy) konkrétním prvkům v cílovém obvodu.
    b. Place-and-Route – přiřazení obvodových prvků k strukturám, které budou vytvořeny v cílovém obvodu. Následně se provede jejich propojení.
6. Časová analýza – funkční simulace s kontrolou časových parametrů jako jsou zpoždění průchodu signálu. Pokud výsledky časové analýzy splňují požadavky výsledné aplikace, následuje konfigurace obvodu.
7. Konfigurace obvodu – v dnešní době nejčastěji se využívá tzv. in system programming – programování přímo v systému, obvod obsahují malý jednoúčelový programátor integrovaný přímo na čipu a je řízen z počítače, pomocí programovacího kabelu. Takto programovatelné obvody se označují zkratkou ISP [9].



Obr. 7.1: Blokové schéma návrhu

8 NÁVRH BCH KÓDU

Podle zadání zde bude navržen BCH kód opravující $\boldsymbol{t = 4}$ chyby s informační rychlostí $\boldsymbol{R \geq 0,5}$. Následně na něm bude ukázán způsob kódování a dekódování.

8.1 Sestavení vytvářecího mnohočlenu

Pro kód opravující $\boldsymbol{t = 4}$ vypočítáme ze vztahu (3.4) minimální Hammingovu vzdálenost $d_{min} = 2t + 1 = 2 \cdot 4 + 1 = 9$

V tabulce přehledu BCH kódu v příloze A najdeme BCH kód s vypočítanou minimální Hammingovou vzdáleností $d_{min} = 9$

vidíme, že existují dva BCH kódy splňující tuto vlastnost a to kódy

- BCH (63, 39)
- BCH (127,99)

Existuje jich i více, ale pro naše potřeby postačí tyto.

Ze vztahu (3.5) vypočítáme jejich informační rychlost *R*

- Pro BCH (63, 39) $R = 0{,}619.$
- Pro BCH (127,99) $R = 0{,}779.$

Vidíme, že požadavky zadání splňují oba uvedené kódy. Pro další pokračování vybereme kód BCH (63,39) z důvodu jednoduší implementace.

Parametry BCH (63,39) jsou

délka kódového slova $n = 63$

počet informačních bitů $k = 39$

Dále v návrhu z rovnice (3.3) zjistíme řád primitivního mnohočlenu a zároveň stupeň Galoisova tělesa $m \geq 6$

z toho můžeme usoudit, že budeme potřebovat Galoisovo těleso GF (2^6)

jeho prvky jsou uvedeny v tabulce v příloze B

z přílohy B. 1, kde jsou uvedeny nerozložitelné mnohočleny nad Galoisovým tělesem GF (2^6) použijeme nerozložitelných mnohočlenů, kolik vyžaduje Bose-Chaudhuriho teorém (s 2*t* po sobě jdoucích kořenů). Použitím vztahu (3.1) dostaneme vytvářecí mnohočlen

$$G(x) = m_1(x) \cdot m_2(x) \cdot m_3(x) \cdot m_4(x),$$

který rozepíšeme

$$G(x) = (1 + x + x^6) \cdot (1 + x + x^2 + x^4 + x^6) \cdot (1 + x + x^2 + x^5 + x^6) \cdot (1 + x^3 + x^6)$$

roznásobením mnohočlenů dostaneme

$$G(x) = x^{24} + x^{23} + x^{22} + x^{20} + x^{19} + x^{17} + x^{16} + x^{13} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1 \quad (8.1)$$

8.2 Ukázka způsobu zakódovaní zprávy

Kód opravuje 4 nezávislé chyby. Mějme nezabezpečený mnohočlen zprávy

P (x) = $x^6 + x^3 + x + 1$ = [P] = [00000000000000000000000000000001001011]

vytvářecí mnohočlen $G(x)$ z (8.1) přepíšeme

G = [11101101100100111011101111]

Dále pokračujeme podle postupu uvedeného v kapitole 4

$$P(x) \cdot x^{(n-k)} = x^{30} + x^{27} + x^{25} + x^{24} = [P] = [10010110000000000000000000000000] \quad (8.2)$$

Dále je třeba získat zbytek po dělení R (x) mnohočlen z (8.2) vydělíme (8.1)

Jeden způsob dělení tzv. postupným sčítáním, který je popsán v [3] je ukázán dále

00000000000000000000000000000001001011000000000000000000000000
11101101100100111011101111
01111011100100111011101110
11101101100100111011101111
00011010101101001100110010 00
11101101100100111011101111
00111000001101011101111110 0
11101101100100111011101111
00001101010001001100010 11

$[R] = [00011010100010011000 1011] =$

$$R(x) = x^{20} + x^{19} + x^{17} + x^{15} + x^{11} + x^8 + x^7 + x^3 + x + 1 \quad (8.3)$$

podle vztahu (4.3) sečteme získaný mnohočlen (8.2) s (8.3) tím dostaneme zabezpečenou zprávu $F(x)$

$F(x) = P(x) \cdot x^{(n-k)} + R(x) =$

$$= x^{30} + x^{27} + x^{25} + x^{24} + x^{20} + x^{19} + x^{17} + x^{15} + x^{11} + x^8 + x^7 + x^3 + x + 1 \quad (8.4)$$

= [F] = [00000000000000000000000000000001001011000110101000100110001011]

8.3 Ukázka dekódování zprávy

Předpokládejme, že chyba vznikla na sedmém, třináctém, dvacátém prvním a dvacátém šestém místě. Chybový mnohočlen je pak $E(x) = x^{26} + x^{21} + x^{13} + x^7$, z rovnice (5.1) můžeme napsat $J(x) = F(x) + E(x) =$

= $x^{30} + x^{27} + x^{26} + x^{25} + x^{24} + x^{21} + x^{20} + x^{19} + x^{17} + x^{15} + x^{13} + x^{11} + x^8 + x^3 + x + 1$=

[J] = [00000000000000000000000000000001001111001110101010100100001011]

8.3.1 Kontrola správnosti

Vytvářecí mnohočlen $G(x)$ má kořeny α, α^2, α^3, α^4, α^5, α^6, α^7, α^8.

Tyto kořeny, budou dosazovány do mnohočlenu přijaté zprávy, viz kapitola 5.1. K výpočtu budeme používat tabulku prvků Galoisova tělesa z přílohy B.

Na ukázku uvedeme pár kompletních výpočtu syndromů, kvůli rozsáhlosti budeme dále uvádět pouze výsledek.

$S_1 = 1 + (\alpha^1)^1 + (\alpha^1)^3 + (\alpha^1)^8 + (\alpha^1)^{11} + (\alpha^1)^{13} + (\alpha^1)^{15} + (\alpha^1)^{17} + (\alpha^1)^{19} + (\alpha^1)^{20} + (\alpha^1)^{21} + (\alpha^1)^{24} + (\alpha^1)^{25} + (\alpha^1)^{26} + (\alpha^1)^{27} + (\alpha^1)^{30} =$
$= \alpha + \alpha^3 + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^5 + \alpha + \alpha^3 + \alpha^3 + \alpha^5 + \alpha + \alpha^2 + \alpha^5 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + 1 + \alpha + \alpha^3 + \alpha^4 + \alpha^5 + 1 + \alpha^4 + \alpha + \alpha^5 + 1 + \alpha + \alpha^2 + \alpha + \alpha^2 + \alpha^3 + 1 + \alpha + \alpha^4 + \alpha^5 = \alpha^4 + \alpha^5 = \alpha^{10}$

$S_2 = 1 + (\alpha^2)^1 + (\alpha^2)^3 + (\alpha^2)^8 + (\alpha^2)^{11} + (\alpha^2)^{13} + (\alpha^2)^{15} + (\alpha^2)^{17} + (\alpha^2)^{19} + (\alpha^2)^{20} + (\alpha^2)^{21} + (\alpha^2)^{24} + (\alpha^2)^{25} + (\alpha^2)^{26} + (\alpha^2)^{27} + (\alpha^2)^{30} =$
$= 1 + \alpha^2 + \alpha^6 + \alpha^{16} + \alpha^{22} + \alpha^{26} + \alpha^{30} + \alpha^{34} + \alpha^{38} + \alpha^{40} + \alpha^{42} + \alpha^{48} + \alpha^{50} + \alpha^{52} + \alpha^{54} + \alpha^{60} =$
$= 1 + \alpha^2 + 1 + \alpha + 1 + \alpha + \alpha^4 + 1 + \alpha^2 + \alpha^4 + \alpha^5 + 1 + \alpha + \alpha^2 + 1 + \alpha + \alpha^4 + \alpha^5 + \alpha^2 + \alpha^5 + 1 + \alpha + \alpha^3 + \alpha^4 + 1 + \alpha + \alpha^2 + \alpha^3 + \alpha^5 + \alpha + \alpha^3 + \alpha^4 + \alpha^5 + 1 + \alpha^2 + \alpha^3 + \alpha^2 + \alpha^4 + \alpha^5 + 1 + \alpha^2 + \alpha^4 + 1 + \alpha + \alpha^2 + \alpha^4 + 1 + \alpha^3 + \alpha^4 + \alpha^5 = \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 = \alpha^{20}$

$S_3 = 1 + (\alpha^3)^1 + (\alpha^3)^3 + (\alpha^3)^8 + (\alpha^3)^{11} + (\alpha^3)^{13} + (\alpha^3)^{15} + (\alpha^3)^{17} + (\alpha^3)^{19} + (\alpha^3)^{20} + (\alpha^3)^{21} + (\alpha^3)^{24} + (\alpha^3)^{25} + (\alpha^3)^{26} + (\alpha^3)^{27} + (\alpha^3)^{30} =$
$= 1 + \alpha^3 + \alpha^9 + \alpha^{24} + \alpha^{33} + \alpha^{39} + \alpha^{45} + \alpha^{51} + \alpha^{57} + \alpha^{60} + \alpha^{63} + \alpha^9 + \alpha^{12} + \alpha^{15} + \alpha^{18} + \alpha^{27} =$
$= 1 + \alpha^3 + \alpha^3 + \alpha^4 + 1 + \alpha^4 + \alpha + \alpha^4 + \alpha + \alpha^2 + \alpha^4 + \alpha^5 + 1 + \alpha^3 + \alpha^4 + 1 + \alpha + \alpha^3 + \alpha^5 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + 1 + \alpha^3 + \alpha^4 + \alpha^5 + 1 + \alpha^3 + \alpha^4 + 1 + \alpha^2 + \alpha^3 + \alpha^5 + 1 + \alpha + \alpha^2 + \alpha^3 + \alpha + \alpha^2 + \alpha^3 = \alpha^2 + \alpha^5 = \alpha^{34}$

$S_4 = \alpha^{40}$ $S_5 = \alpha^{31}$ $S_6 = \alpha^5$ $S_7 = \alpha^{25}$ $S_8 = \alpha^{17}$

Pro přehlednost opíšeme výsledky Syndromů

$S_1 = \alpha^{10}$; $S_2 = \alpha^{20}$; $S_3 = \alpha^{34}$; $S_4 = \alpha^{40}$; $S_5 = \alpha^{31}$; $S_6 = \alpha^5$; $S_7 = \alpha^{25}$; $S_8 = \alpha^{17}$

8.3.2 Výpočet mnohočlenu pozice chyb

Pro získání mnohočlenu pozice chyby použijeme Petersonův algoritmus popsaný v kapitole 5.2.1.

Kód opravuje $t = 4$ chyb proto i počet vazebních rovnic je $v = 4$, tím získáme soustavu vazebních rovnic tvaru (5.4)

$$S_1 z_4 + S_2 z_3 + S_3 z_2 + S_4 z_1 + S_5 = 0$$
$$S_2 z_4 + S_3 z_3 + S_4 z_2 + S_5 z_1 + S_6 = 0$$
$$S_3 z_4 + S_4 z_3 + S_5 z_2 + S_6 z_1 + S_7 = 0$$
$$S_4 z_4 + S_5 z_3 + S_6 z_2 + S_7 z_1 + S_8 = 0$$

do této soustavy dosadíme hodnoty syndromů získaných v kapitole 7.3.1, a zároveň

samostatné prvky GF (2^6) převedeme na pravou stranu rovnice

$$\alpha^{10} z_4 + \alpha^{20} z_3 + \alpha^{34} z_2 + \alpha^{40} z_1 = \alpha^{31}$$
$$\alpha^{20} z_4 + \alpha^{34} z_3 + \alpha^{40} z_2 + \alpha^{31} z_1 = \alpha^{5}$$
$$\alpha^{34} z_4 + \alpha^{40} z_3 + \alpha^{31} z_2 + \alpha^{5} z_1 = \alpha^{25}$$
$$\alpha^{40} z_4 + \alpha^{31} z_3 + \alpha^{5} z_2 + \alpha^{25} z_1 = \alpha^{17}$$

Nyní je třeba zjistit, kolik soustava rovnic obsahuje po sobě jdoucích lineárně nezávislých rovnic. Rovnice jsou lineárně **závislé**, pokud můžeme jednu rovnici vytvořit z druhé lineární operací nad GF (např. vynásobením nějakým prvkem). Posouzení lineární závislosti této soustavy není přímo možné, proto se používá sčítání jednotlivých rovnic (tím vypadávají neznámé např. z_1).

Zkoušením různých lineárních operací zjistíme, že není možné z naší soustavy rovnic vytvořit jednu rovnici z jiné, proto usuzujeme, že všechny rovnice jsou lineárně nezávislé. Z toho můžeme říci, že nastaly čtyři chyby, tj. $v = 4$.

Vyřešením této soustavy rovnic o čtyřech neznámých, dostaneme hodnoty

$z_1 = \alpha^{10}$, $z_2 = \alpha^{26}$, $z_3 = 1$, $z_4 = \alpha^4$

z těchto hodnot můžeme sestavit mnohočlen polohy chyb. Položíme-li jej roven nule, dostaneme rovnici udávající polohy chyb, tvaru rovnice (5.6).

$$X^4 + \alpha^{10} X^3 + \alpha^{26} X^2 + 1\, X + \alpha^4 = 0 \qquad (7.5)$$

8.3.3 Výpočet pozice chyby

Podle kapitoly 5.3 hledáme kořeny rovnice (7.5), do které budeme dosazovat jednotlivé prvky Galoisova tělesa GF (2^6). Zjistíme, že jsou to prvky:

α^7, α^{13}, α^{21}, α^{26}

Z toho můžeme usoudit, že chyby vznikly na místech: 7.; 13., 21.; 26. bitu

8.3.4 Oprava chyb

Máme přijatou právu zatíženu chybou

[J] = [00000000000000000000000000000001001$\underline{1}$1100$\underline{1}$1101010$\underline{1}$01001$\underline{0}$0001011]

Postupujeme, jak je popsáno v kapitole 5.4. Místa 7.; 13., 21.; 26. v $J(x)$ invertujeme.

[J] = [00000000000000000000000000000001001$\underline{0}$1100$\underline{0}$1101010$\underline{0}$01001$\underline{1}$0001011]

9 NÁVRH REALIZACE KODEKU

Na trhu existuje více výrobců FPGA obvodů. Dva z hlavních jsou firmy Xilinx a Altera. Oba výrobci nabízejí vývojové prostředí obsahující kompletní software pro návrh.

Pro realizaci byla vybrána firma Altera s vývojovým prostředím Quartus II Web Edition (v9.1), které je po zaregistrování zdarma ke stažení na webových stránkách firmy Altera. Tato verze je omezená pouze na návrh CPLD obvodů (řady MAX), jednodušších obvodů FPGA (řady Cyclone), středních (řady Arria) a několik málo high-end FPGA (řad Stratix). Pro náš návrh je toto prostředí dostačující, jelikož obsahuje všechny potřebné nástroje pro úspěšné vytvoření návrhu.

Podle [10] má popis v jazyku VHDL dvě základní části

- Deklaraci entity – zde se popisují vstupy a výstupy objektu. Entitu si můžeme představit jako součástku, která má definované parametry vstupů a výstupů obvodu.
- Tělo architektury – definuje funkce (chování) entit

V popisu VHDL využíváme klíčových slov, které zapisujeme velkými písmeny. Jejich seznam a syntaxe je uvedená např. viz [11].

Při návrhu kodeku je nutné brát v úvahu, že protichybový kodek bude součástí protichybového kódového systému. Tzn kodek musí být schopen komunikovat s protichybovým systémem – musí být schopen synchronizace, řízení a práce podle externího hodinového signálu.

9.1 Návrh kodéru

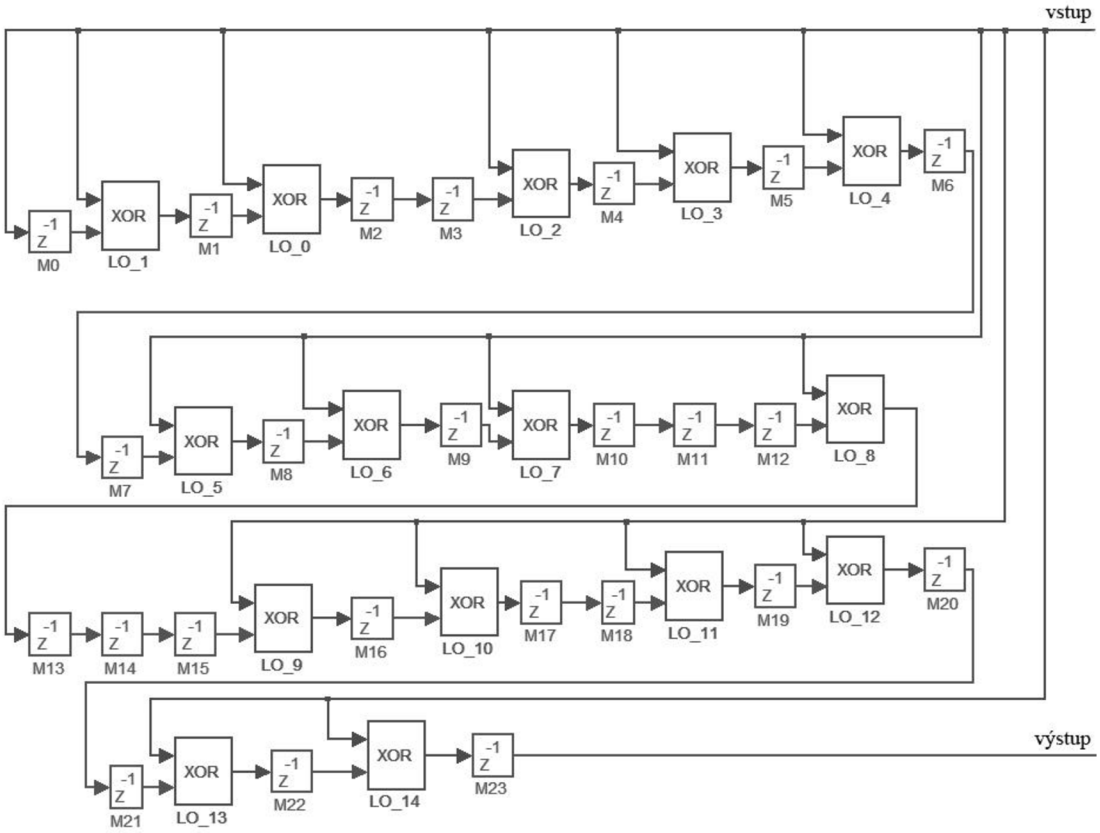
Při návrhu kodéru BCH kódu se vychází z teorie uvedené v kapitole 4, kdy kodér tvoří kruhový posuvný registr. Jeho zapojení odvodíme podle vytvářecího mnohočlenu $G(x)$ navrženého v kapitole 8.

Pro kód BCH (63, 39) je jeho vytvářecí mnohočlen

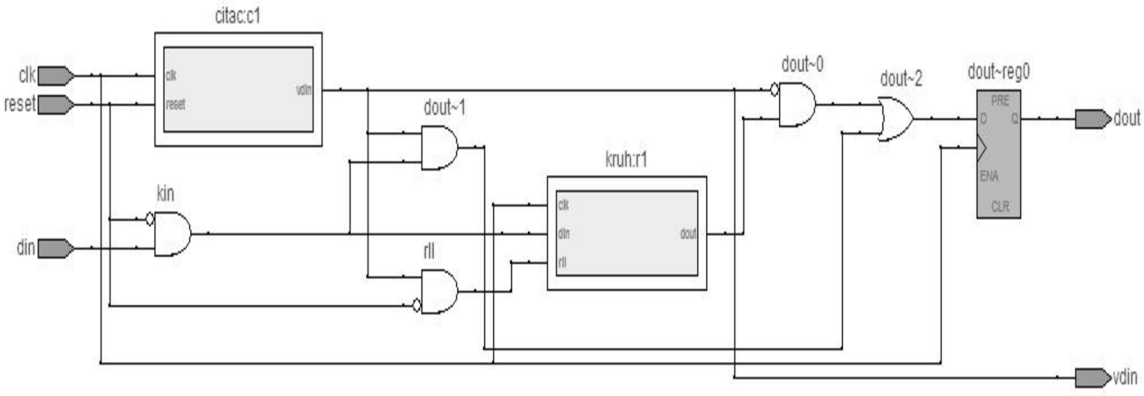
$$G(x) = x^{24} + x^{23} + x^{22} + x^{20} + x^{19} + x^{17} + x^{16} + x^{13} + x^{10} + x^{9} + x^{8} + x^{6} + x^{5} + x^{4} + x^{2} + x + 1$$

Na obr.9.1 je znázorněno zapojení děličky kruhového registru pro náš vytvářecí mnohočlen. Členy M0-M23 představují jednotlivé x vytvářecího mnohočlenu a funkci součtu zajišťují členy LO_1 – LO_14 (funkce XOR).

Dále zde popíšeme návrh ve VHDL jazyce. Kodér se skládá ze tří entit „kruh“, „citac“ a „koder“. Entita „kruh“ definuje vstup architektury „kruh_a“, která slouží k popisu zapojení jednotlivých částí posuvného kruhového registru (podle obr.9.1), část této architektury je uvedena v příloze C.1. Entita „citac“ a její architektura, nám plní úlohu přepínačů S1 a S2 (viz obr.4.1 z teoretické části), čítač spolu s klopným obvodem, který se současně nastavuje (pokud se čítač rovná nule) a resetuje (pokud se čítač rovná k. Výstup klopného obvodu je v jedničce pokud má čítač hodnotu $\leq k - 1$, jinak je v nule.



Obr. 9.1: Zapojení děličky mod $G(x)$, podle vytvářecího mnohočlenu



Obr. 9.2: Blokové vnitřní zapojení kodéru

Na obr.9.2 je znázorněno blokové vnitřní zapojení kodéru, kde můžeme vidět entity „citac“ a „kruh“.

Popis vstupů a výstupů kodéru

- Clk – vstup pro hodinový signál
- Reset – reset kodéru za účelem synchronizace
- Din – data input – vstup pro nezabezpečenou posloupnost dat
- Dout – data out – výstup z kodéru

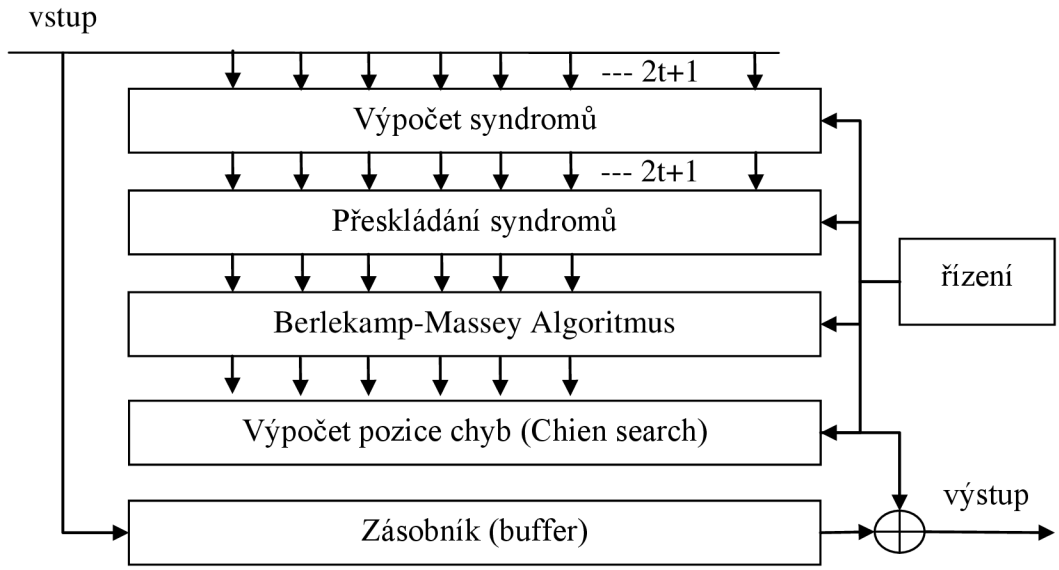
- Vdin – valid data in – slouží k řízení vstupu dat, je nastaven do 1 při vstupu dat, které jsou zároveň posílány na výstup (dout). Po vstupu *k* informačních bitů je vdin nastaven do 0 (data nejsou na vstupu přijímána) a na výstup jsou posílány zabezpečovací bity.

9.2 Návrh dekodéru

Návrh dekodéru byl proveden ve VHDL jazyce. Dekódovací proces rozdělen do 3 základních celků

- Výpočet syndromů – jak bylo vysvětleno v kapitole 5.1 a předvedeno v kapitole 8.3.1
- Berlekamp-Massey Algoritmus – který byl popsán v kapitole 5.2.3
- Výpočet pozice chyb – popsán v kapitole 5.3

K těmto částem dekodéru jsou vyžadovány další entity, které slouží k řízení, zásobník (buffer), přeskládání syndromu a jiné.

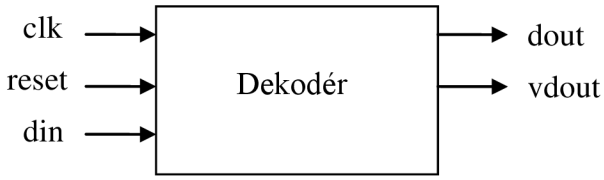


Obr. 9.3 Blokový diagram obvodu dekodéru

Z důvodu složitosti vnitřní struktury dekodéru je zde vložen pouze blok znázorňující vstupy a výstupy (obr. 9.4)

Popis vstupů a výstupů dekodéru

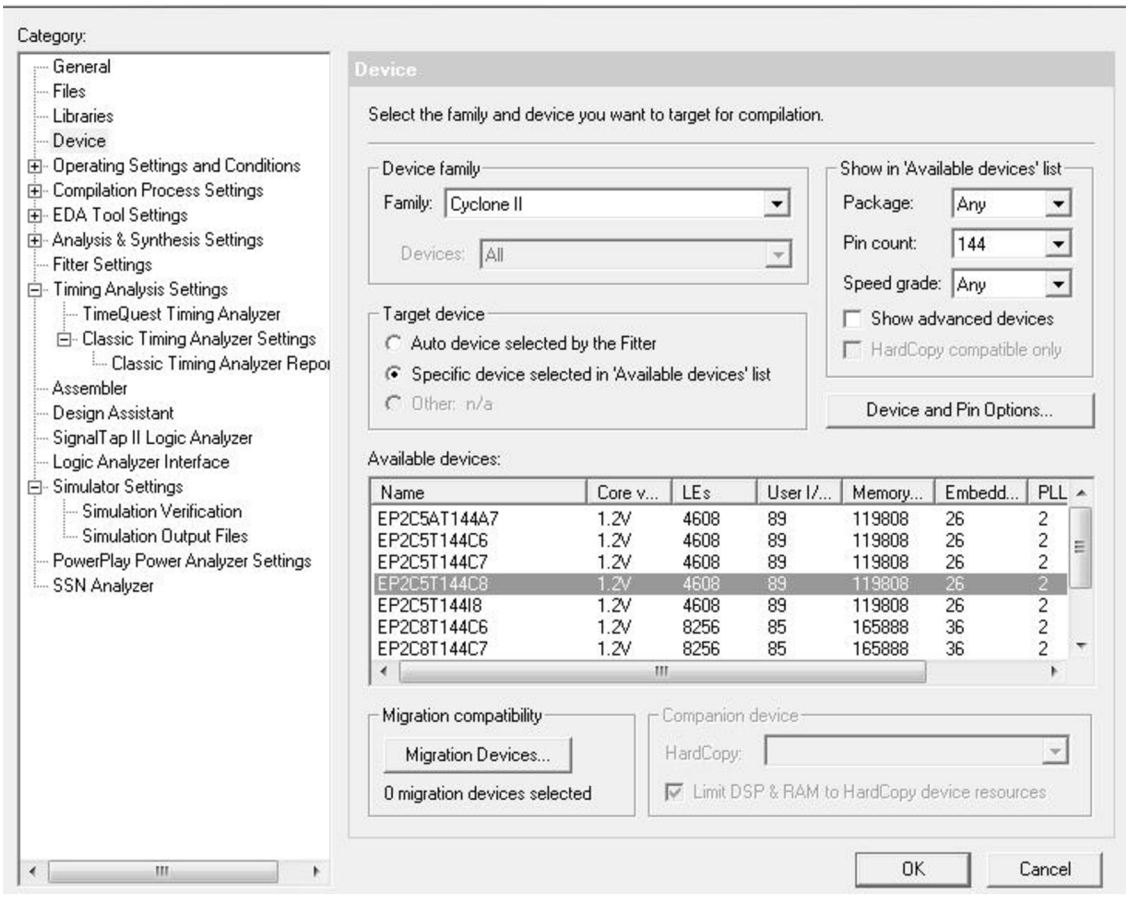
- Clk – vstup pro hodinový signál
- Reset – reset kodéru za účelem synchronizace
- Din – data input – vstup pro nezabezpečenou posloupnost dat
- Dout – data out – výstup z dekodéru (opravená data)
- Vdout – valid data out – slouží jako řízení výstupu dat, při nastavení do 1 jsou opravená data posílána na výstup (dout). Zároveň mohou vstupovat nová data do dekodéru.



Obr. 9.4: Vstupy a výstupy dekodéru

9.3 Návrh ve vývojovém softwaru Quartus II

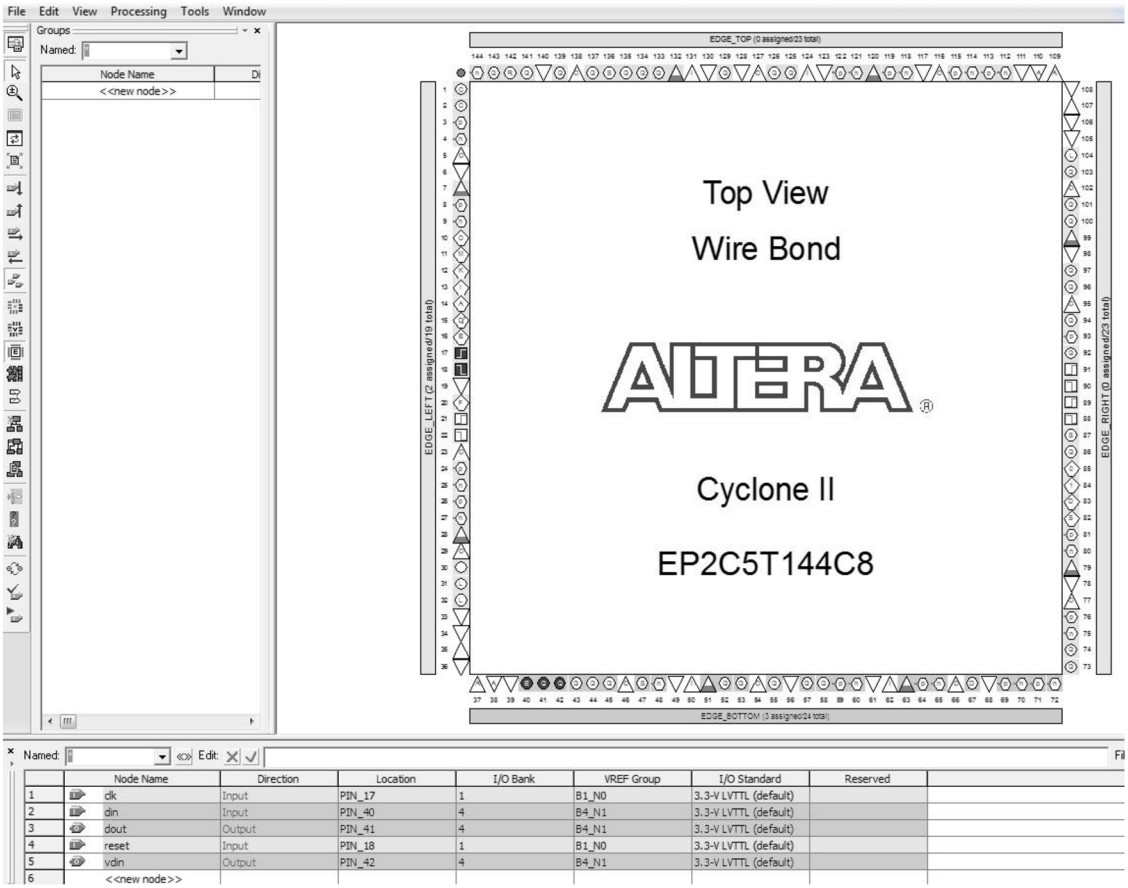
Celý návrh koderu i dekodéru probíhal v programu Quartus II Web Edition (v9.1), kdy na začátku je založen nový projekt a pomocí pomocníka (wizardu) jsou nastaveny všechny potřebné parametry návrhu. Pokud už máme navrhovaný obvod popsán jazykem VHDL a načtenou hlavní entitu je zde možné vybrat cílový obvod pro který bude návrh optimalizován. K seznámení s programem Quartus II je možné pomocí uživatelského manuálu [12]. Výběr cílového obvodu je zobrazen na obr.9.5. Můžeme si vybrat pomocí řady obvodu (Device family), podle pouzdra, počtu pinů cílového obvodu.



Obr. 9.5: Výběr požadovaného cílového obvodu

Pro náš návrh byl vybrán FPGA obvod z řady Cyclone II s označením EP2C5T144C8 (patřící do skupiny levných (low cost) FPGA) a pouzdrem typu TQPF a 144-mi piny. Jeho další parametry a vlastnosti jsou uvedeny v dokumentaci [13].

Po návrhu funkce obvodu je spuštěna kompilace, kdy se provádějí kroky popsané v kapitole 7.2. Pokud proběhla kompilace bez chyb je možné přiřadit vstupní a výstupní piny našeho návrhu na piny cílového obvodu, k tomu nám může posloužit tzv pin planer, který je zobrazen na obr.9.6.

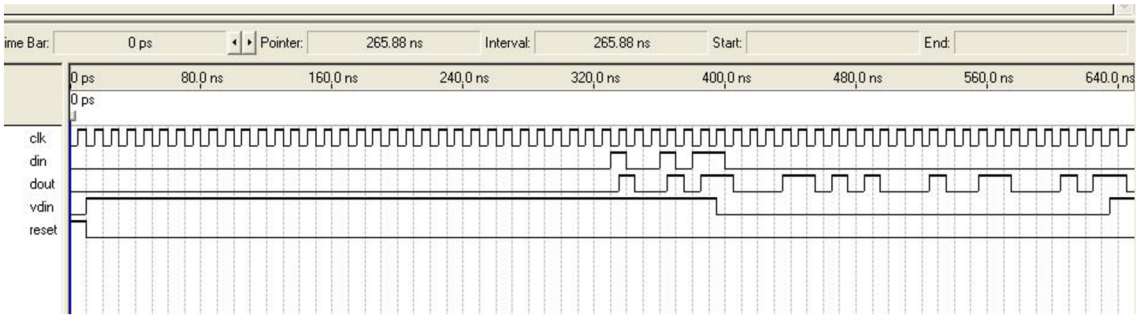


Obr. 9.6: Přiřazení pinů pomocí Pin planeru

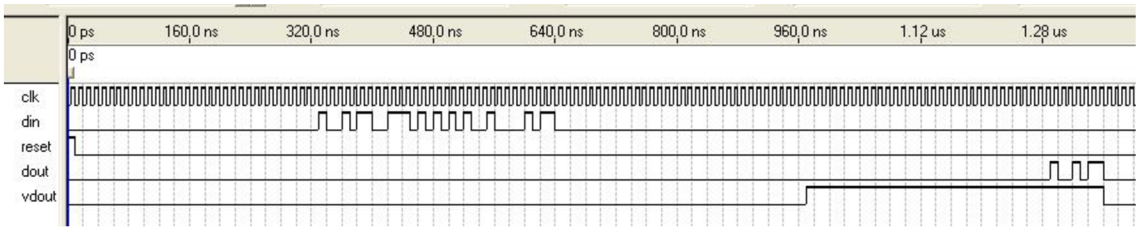
9.4 Simulace

Pomocí simulace můžeme snadno ověřit funkci celého návrhu. Součástí programu Quartus je i Simulator tool, pomocí kterého můžeme simulovat funkční i časové vlastnosti našeho návrhu.

Pro simulaci byl použit příklad vstupního slova použitého pro ukázku kódování v kapitole 8.2. Perioda vstupního hodinového signálu (clk), byla pro simulaci zvolena 10ns. Na obr.9.7 je zobrazen výstup simulace kodéru (zakódování vstupní posloupnosti). Na obr.9.8 je zobrazen výstup simulace dekodéru – se vstupní posloupností obsahující 4 chyby. Porovnáním vstupu din obr.9.7 a výstup dout obr.9.8 můžeme usoudit, že námi navržený kodek funguje. Z obr.9.8 můžeme určit dobu dekódovacího procesu (doba mezi posledním vstupním bitem a prvním výstupním) a ta je přibližně 320ns.



Obr. 9.7: Výstup simulace kodéru



Obr. 9.8: Výstup simulace dekodéru

9.5 Konfigurace FPGA obvodu

Konfigurace FGPA obvodu je možná pomocí standardizovaného rozhraní JTAG (programování v systému) s využitím programovacího kabelu např. USB-Blaster, specifikace, které jsou uvedeny v [14] tohoto kabelu využijeme pro zapojení konektoru námi navrhovaného plošného spoje.

9.6 Návrh desky plošného spoje pro navržený kodek

Pro návrh desky plošného spoje byl použit návrhový software Eagle verze 5.8.0 light edition. Vlastnosti jednotlivých pinů a jejich požadavky pro zapojení jsou uvedeny v [14]. V příloze D je uvedeno schéma zapojení pro tvorbu plošného spoje. Přílohy D.1, D.2, D.3 obsahují desky plošného spoje a osazení plošného spoje, které nejsou v měřítku M1:1, z důvodu rozměrů součástek a vodivých cest, byly tyto přílohy zvětšeny. Příloha D.4 obsahuje seznam použitých součástek.

Námi vybraný FPGA obvod potřebuje ke své činnosti napájení 1,2V a 3,3V, to zajišťují stabilizátory IC2 a IC3, jejich napětí je nastaveno rezistory R1, R2, R3. Vstupní napětí (připojuje se ke konektoru SV3) navrženého obvodu může být v rozmezí 7–15V (doporučené) 7V minimální pro zaručení funkčnosti, maximální hranice pro vstupní napětí může být až 40V, avšak při tomto napětí už by musely být ke stabilizátorům připevněny chladiče. Kondenzátory C1 a C4 plní funkci filtru napětí a musí být umístěny co nejblíže IC1. Rezistory R4–R8 jsou zapojeny ke konfiguračním pinům obvodu a zabraňují vzniku neurčitých stavů. Konektor SV2 slouží ke konfiguraci FPGA obvodu pomocí programovacího kabelu USB-Blaster. Konektor SV1 slouží k připojení námi navrženého kodeku. Jelikož námi navržený kodér a dekodér se liší pouze v programu nahraného do FPGA, jsou jejich desky plošných spojů totožné.

10 ZÁVĚR

Tato práce se zabývá zabezpečením dat pomocí binárních BCH kódů. Tyto kódy jsou schopny zabezpečit binární data proti vzniku nezávislých chyb a řadí se do skupiny lineárních blokových cyklických kódů. U této skupiny kódů je výhodou dobrá volitelnost parametrů – např. kódového slova nebo počet opravitelných chyb. Zabezpečení přenášených dat se realizuje přidáním zabezpečovacích bitů k informačním bitům (přenášená data). Takto vytvořené kódové slovo se přenáší přes přenosový kanál.

První kapitola je úvodem do komunikace a je v ní popsána struktura komunikačního systému a možné způsoby detekce a oprav chyb. V druhé kapitole je rozebrán popis BCH kódů, jejich použití v současných spojovacích zařízení, které jsou např. pro zabezpečení přenosu videa ve videokonferencích (doporučení ITU-T H. 261), zařazení těchto kódů do skupiny korekčních kódů a vysvětlení jednotlivých pojmů. Dále je uveden přehled nezbytných znalostí, týkající se Galoisových těles, které jsou u BCH kódů využívány. V další kapitole je popsán návrh BCH kódu, jehož základem je vytvoření vytvářecího mnohočlenu. Ve čtvrté kapitole je uveden matematický popis pro kódování BCH kódem, dále je uvedena možná realizace BCH kodéru. Při kódování se vychází ze zapojení kruhového posuvného registru se zpětnými vazbami a sčítačkami modulo 2 zapojenými podle vytvářecího mnohočlenu. V páté kapitole je uveden postup pro dekódování přijatého kódového slova, které je výpočetně složitější než kódování. Dekódování dělíme do čtyř kroků: kontrola kódového slova na výskyt chyb a nalezení syndromů, nalezení mnohočlenu pozice chyby (lokátor chyby), výpočet pozice chyby a nakonec oprava chyb. Nejsložitější částí dekódování je určení lokátoru chyb, pro tuto část dekódování, byly v této práci popsány: Petersonův algoritmus, maticová metoda a Berlekamp-Massey algoritmus. Po nalezení pozice chyb je jejich následná oprava jednoduchá, jelikož se jedná o binární data, stačí chybné bity invertovat. V šesté kapitole jsou uvedeny způsoby možné realizace kodeku BCH kódu. Nejméně vhodná je realizace pomocí číslicových integrovaných obvodů zejména z velkého množství pouzder (složitosti zapojení) a špatné modifikace funkce. Další možností realizace je softwarové naprogramování aplikace na PC, která by například zabezpečovala přenos dat přes sériový port, nebo ukládaných dat. Pro realizaci lze využít i mikrokontrolery, kde mohou být využita většina výpočetních algoritmů pro dekódování a opravu chyb, ale jelikož jejich nevýhodou je pomalá reakce (nutno provést velký počet instrukcí) je toto řešení nevhodné pro vysokorychlostní přenosy. Pomocí mikrokontroléru by šla zabezpečit komunikace dvou systémů, u kterých není potřeba rychlé výměny dat např. zabezpečení dat (např. řídících příkazů). Poslední zde uvedenou možností realizace je využití programovatelných obvodu např. FPGA, kde je výhodou jejich univerzálnost a rychlost reakce. Další výhodou je, že programátor nemusí mít speciální znalosti na úrovni hradel, jelikož popis funkce obvod se provádí pomocí některého jazyka HDL, vnitřní zapojení obvodu může být ponecháno na kompilátoru (vývojovém prostředí).

Pro výhody uvedené v kapitole 6 jsem pro návrh protichybového kodeku BCH kódu zvolil implementaci do obvodů FPGA, a proto v sedmé kapitole jsou popsány prostředky pro návrh a způsoby popisu funkce požadovaného obvodu. V osmé kapitole jsem vytvořil příklad BCH kódu opravující $t = 4$ chyby s informační rychlostí $R \geq 0{,}5$. Tyto podmínky splňovaly dva BCH kódy a to BCH (63,39) a BCH (127,99). Pro další práci jsem vybral kód BCH (63,39), protože vyhovoval zadání a přitom jeho realizace byla méně složitá. Dále je popsán na vytvořeném příkladu způsob kódování a dekódování.

Poslední kapitola se věnuje návrhu protichybového kodeku v sofwaru Quartus II a následnému ověření jeho funkčnosti. Z výsledků simulace (obr. 9.8) můžeme také kromě funkčnosti určit dobu, po kterou v dekodéru probíhá proces dekódování. Ta při použitém časovém signálu (clk) s periodou 10 ns se rovnala přibližně 320 ns.

Při návrhu desky plošného spoje jsem využíval návrhového programu Eagle 5.8.0 light edition. Jelikož se kodér a dekodér liší pouze nahraným programem do obvodu FPGA, jsou jejich desky plošných spojů totožné. Na plošném spoji jsou umístěny konektory:

- pro napájení (konektor SV3, napájecí napětí v rozsahu 7–15V (doporučené) 7V minimální pro zaručení funkčnosti, maximální hranice pro vstupní napětí může být až 40V, avšak při tomto napětí už by musely být ke stabilizátorům připevněny chladiče).
- pro konfiguraci FPGA obvodu (konektor SV2 zapojen pro konfiguraci pomocí kabelu USB-Blaster).
- pro připojení navrženého kodeku (konektor SV1).

11 POUŽITÁ LITERATURA

[1] Adámek, Jiří. *Kódování a teorie informace.* Praha : skriptum ČVUT, 1991. ISBN: 80-01-00661-1.

[2] Wallace, Hank. *Error Detection and Correction Using the BCH Code.* [Online] 2001. [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.aqdi.com/bch.pdf>.

[3] Němec, Karel. *Datová komunikace.* místo neznámé : skriptum VUT-BRNO, 2007.

[4] Costello, Daniel J. and LIN, SHU. *Error Control Coding: Fundamentals and Aplications.* s.l. : Prentice-Hall, 1983. ISBN: 0-13-283796-X.

[5] Němec, Karel. *Protichybové kódové zabezpečení s Bose-Chauhury-Hocquenhemovými kódy. Publikace v internetovém magazínu Elektrorevue.* [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.elektrorevue.cz/clanky/05015/index.htm>.

[6] Číka, Petr. *Protichybové zabezpečení BCH kódem. Publikace v internetovém magazínu Elektrorevue.* [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.elektrorevue.cz/clanky/06015/index.html>.

[7] Peterson, William Weasley a Weldon, E. J. *Error-Correcting Codes.* SECOND EDITION. Cambridge : MIT Press, 1996. ISBN: 0 262 16 039 0.

[8] Sweeney, Peter. *Error control coding from theory to practice.* Guildford : John Willey & Sons, 2002. ISBN 0 470 84356 X.

[9] Kolouch, Jaromír. *Programovatelné logické obvody.* Brno : skriptum VUT, 2007.

[10] Musil, Vladislav, Fujcik, Lukáš a Kolouch, Jaromír. *Návrh digitálních integrovaných obvodů VLSI a jazyk VHDL.* Brno : skriptum VUT, 2007.

[11] VHDL Syntax (IEEE Std 1076-2002). [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.iis.ee.ethz.ch/~zimmi/download/vhdl02_syntax.html>..

[12] ALTERA. Quartus II Handbook. [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf>.

[13] ALTERA. Cyclone II device Handbook. [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf>.

[14] ALTERA. USB-blaster download cable- user guide. [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.altera.com/literature/ug/ug_usb_blstr.pdf>.

[15] ALTERA. Cyclone II EP2C5 Device pin-out. [Online] [Citace: 30. 4 2010.] Dostupné z WWW: <http://www.altera.com/literature/dp/cyclone2/ep2c5.pdf>.

SEZNAM ZKRATEK

ARQ	Automatic repeat request
BCH	Bose Chaudhuri Hocquenghem
FEC	Forward Error Correction
CD	Compact Disc
DVD	Digital Versatile Disc/ Digital Video Disc
RAID	Redundant Array of Independent Disks
DVB_S2	Digital Video Broadcasting _ Satellite – Second generation
ITU-T	International Telecommunication Union Standardization
mod2	modulo 2
HDL	Hardware Description Language
LSFR	Linear Feedback Shift Register
ISP	In System Programming

SEZNAM PŘÍLOH

A PŘEHLED BCH KÓDŮ

Tabulka přehled BCH kódů převzata z [1]

Délka n	Minimální vzdálenost d_{min}	Informačních znaků k
7	3	4
	7	1
15	3	11
	5	7
	7	5
	15	1
31	3	26
	5	24
	7	16
	11	11
	15	6
	31	1
63	3	57
	5	51
	7	45
	9	39
	11	36
	13	30
	15	24
	21	18
	23	16
	27	10
	31	7
	63	1
127	3	120
	5	113
	7	106
	9	99
	11	92
	13	85
	15	78
	19	71
	21	64
	23	57
	27	27
	31	43
	31	36
	43	29
	47	22
	55	15
	63	8

B GALOISOVO TĚLESO GF (2^6)

Tabulka Galoisova tělesa GF (2^6) převzata z [1]

Vyjádření prvků Galoisova tělesa GF (2^6) generováno G (x) = $x^6 + x + 1$

exp	Zbytkem po dělení	Binárně
0	0	000000
1	1	000001
α	α	000010
α^2	α^2	000100
α^3	α^3	001000
α^4	α^4	010000
α^5	α^5	100000
α^6	$1+\alpha$	000011
α^7	$\alpha+\alpha^2$	000110
α^8	$\alpha^2+\alpha^3$	001100
α^9	$\alpha^3+\alpha^4$	011000
α^{10}	$\alpha^4+\alpha^5$	110000
α^{11}	$1+\alpha+\alpha^5$	100011
α^{12}	$1+\alpha^2$	000101
α^{13}	$\alpha+\alpha^3$	001010
α^{14}	$\alpha^2+\alpha^4$	010100
α^{15}	$\alpha^3+\alpha^5$	101000
α^{16}	$1+\alpha+\alpha^4$	010011
α^{17}	$\alpha+\alpha^2+\alpha^5$	100110
α^{18}	$1+\alpha+\alpha^2+\alpha^3$	001111
α^{19}	$\alpha+\alpha^2+\alpha^3+\alpha^4$	011110
α^{20}	$\alpha^2+\alpha^3+\alpha^4+\alpha^5$	111100
α^{21}	$1+\alpha+\alpha^3+\alpha^4+\alpha^5$	111011
α^{22}	$1+\alpha^2+\alpha^4+\alpha^5$	110101
α^{23}	$1+\alpha^3+\alpha^5$	101001
α^{24}	$1+\alpha^4$	010001
α^{25}	$\alpha+\alpha^5$	100010
α^{26}	$1+\alpha+\alpha^2$	000111
α^{27}	$\alpha+\alpha^2+\alpha^3$	001110
α^{28}	$\alpha^2+\alpha^3+\alpha^4$	011100
α^{29}	$\alpha^3+\alpha^4+\alpha^5$	111000
α^{30}	$1+\alpha+\alpha^4+\alpha^5$	110011
α^{31}	$1+\alpha^2+\alpha^5$	100101

exp	Zbytkem po dělení	Binárně
α^{32}	$1+\alpha^3$	001001
α^{33}	$\alpha+\alpha^4$	010010
α^{34}	$\alpha^2+\alpha^5$	100100
α^{35}	$1+\alpha+\alpha^3$	001011
α^{36}	$\alpha+\alpha^2+\alpha^4$	010110
α^{37}	$\alpha^2+\alpha^3+\alpha^5$	101100
α^{38}	$1+\alpha+\alpha^3+\alpha^4$	011011
α^{39}	$\alpha+\alpha^2+\alpha^4+\alpha^5$	110110
α^{40}	$1+\alpha+\alpha^2+\alpha^3+\alpha^5$	101111
α^{41}	$1+\alpha^2+\alpha^3+\alpha^4$	011101
α^{42}	$\alpha+\alpha^3+\alpha^4+\alpha^5$	111010
α^{43}	$1+\alpha+\alpha^2+\alpha^4+\alpha^5$	110111
α^{44}	$1+\alpha^2+\alpha^3+\alpha^5$	101101
α^{45}	$1+\alpha^3+\alpha^4$	011001
α^{46}	$\alpha+\alpha^4+\alpha^5$	110010
α^{47}	$1+\alpha+\alpha^2+\alpha^5$	100111
α^{48}	$1+\alpha^2+\alpha^3$	001101
α^{49}	$\alpha+\alpha^3+\alpha^4$	011010
α^{50}	$\alpha^2+\alpha^4+\alpha^5$	110100
α^{51}	$1+\alpha+\alpha^3+\alpha^5$	101011
α^{52}	$1+\alpha^2+\alpha^4$	010101
α^{53}	$\alpha+\alpha^3+\alpha^5$	101010
α^{54}	$1+\alpha+\alpha^2+\alpha^4$	010111
α^{55}	$\alpha+\alpha^2+\alpha^3+\alpha^5$	101110
α^{56}	$1+\alpha+\alpha^2+\alpha^3+\alpha^4$	011111
α^{57}	$\alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^5$	111110
α^{58}	$1+\alpha+\alpha^2+\alpha^3+\alpha^4+\alpha^5$	111111
α^{59}	$1+\alpha^2+\alpha^3+\alpha^4+\alpha^5$	111101
α^{60}	$1+\alpha^3+\alpha^4+\alpha^5$	111001
α^{61}	$1+\alpha^4+\alpha^5$	110001
α^{62}	$1+\alpha^5$	100001
α^{63}	1	000001

B.1 Nerozložitelné mnohočleny nad GF (2^6) a jejich kořeny

Tabulka nerozložitelných mnohočlenů nad GF (2^6) převzata z [4]

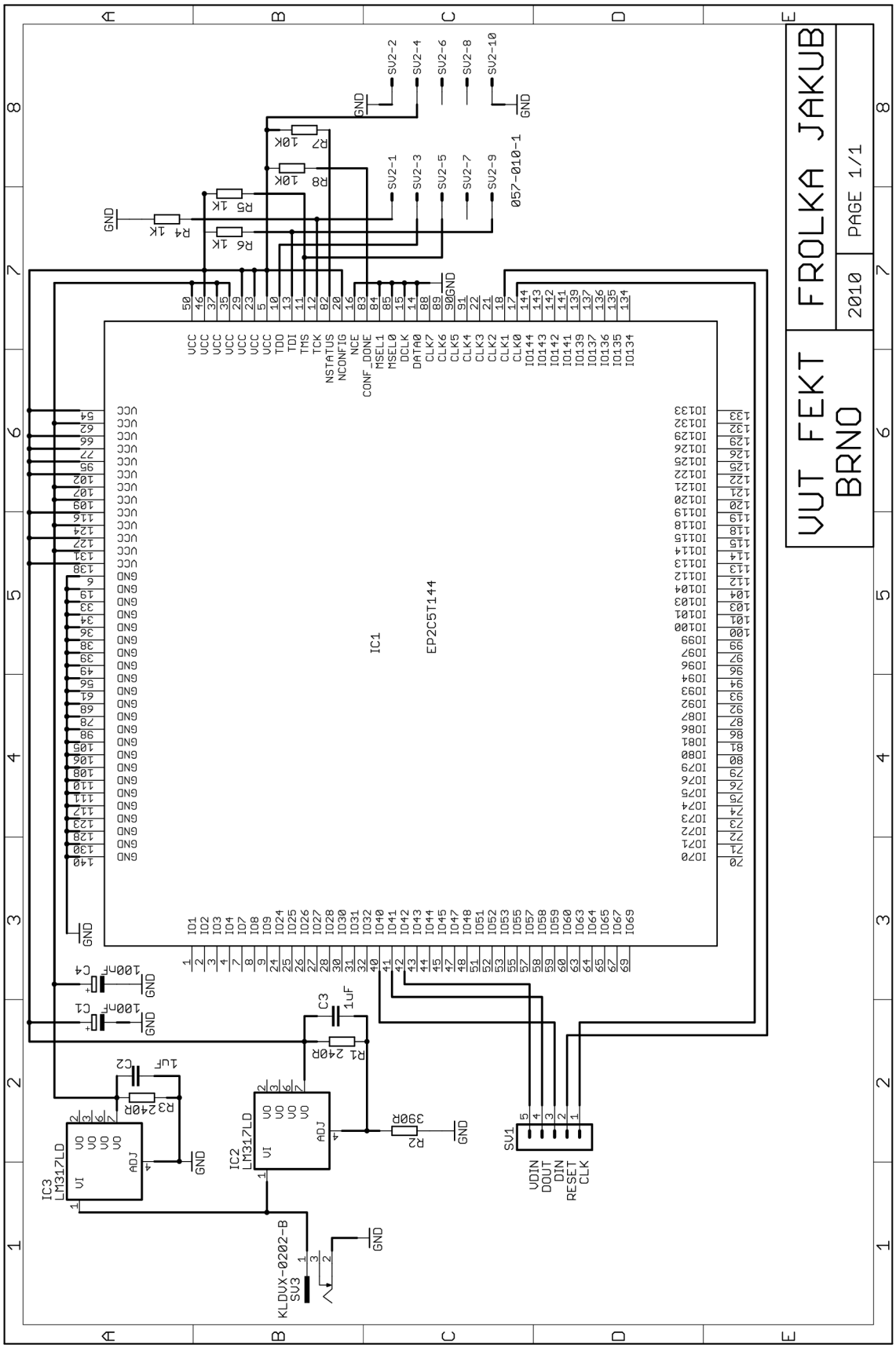
j	Nerozložitelný mnohočlen $\mathbf{m}_j$	Kořeny mnohočlenu
1	$1 + x + x^6$	$\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}$
2	$1 + x + x^2 + x^4 + x^6$	$\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{48}, \alpha^{33}$
3	$1 + x + x^2 + x^5 + x^6$	$\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40}, \alpha^{17}, \alpha^{34}$
4	$1 + x^3 + x^6$	$\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{56}, \alpha^{49}, \alpha^{35}$
5	$1 + x^2 + x^3$	$\alpha^9, \alpha^{18}, \alpha^{36}$
6	$1 + x^2 + x^3 + x^5 + x^6$	$\alpha^{11}, \alpha^{22}, \alpha^{44}, \alpha^{25}, \alpha^{50}, \alpha^{37}$
7	$1 + x + x^3 + x^4 + x^6$	$\alpha^{13}, \alpha^{26}, \alpha^{52}, \alpha^{41}, \alpha^{19}, \alpha^{38}$
8	$1 + x^2 + x^4 + x^5 + x^6$	$\alpha^{15}, \alpha^{30}, \alpha^{60}, \alpha^{57}, \alpha^{51}, \alpha^{39}$
9	$1 + x + x^2$	α^{21}, α^{42}
10	$1 + x + x^4 + x^5 + x^6$	$\alpha^{23}, \alpha^{46}, \alpha^{29}, \alpha^{58}, \alpha^{53}, \alpha^{43}$
11	$1 + x + x^3$	$\alpha^{27}, \alpha^{54}, \alpha^{45}$
12	$1 + x^5 + x^6$	$\alpha^{31}, \alpha^{62}, \alpha^{61}, \alpha^{59}, \alpha^{55}, \alpha^{47}$

C ČÁST ZDROJOVÝCH KÓDŮ

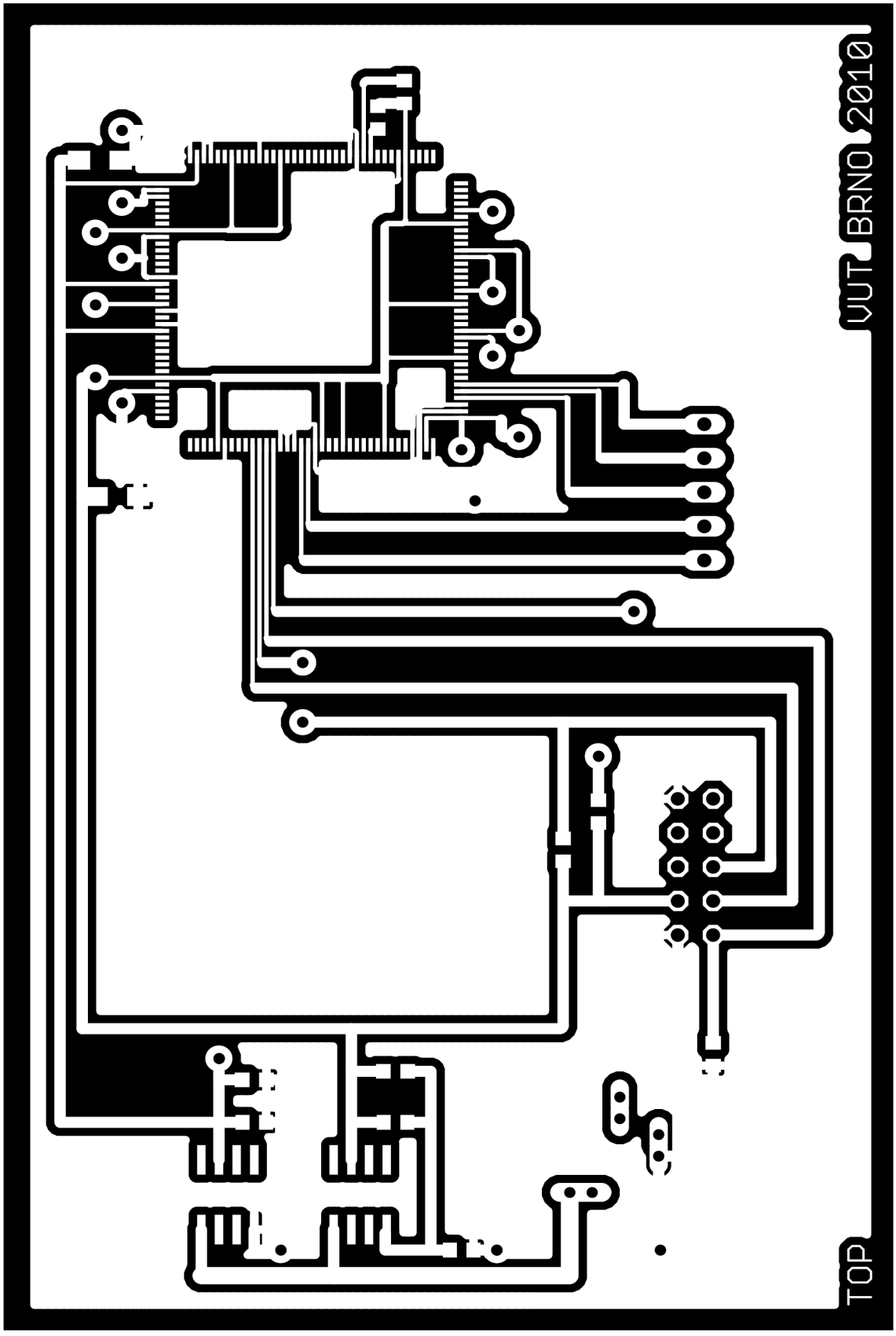
C.1 Kodér: architerktura kruh_a

```
ARCHITECTURE kruh_a OF kruh IS
        SIGNAL kin, kout: BIT_VECTOR(0 TO nk-1); -- kruhovy registr
        SIGNAL kin_0: BIT;
 BEGIN
        dout<= kout(nk-1);                              -- (nk-1)
        kin_0 <= (din XOR kout(nk-1)) AND rll;
        kin(0)<= kin_0;
        kin(1)<= kout(0) XOR kin_0;
        kin(2)<= kout(1) XOR kin_0;
        kin(3)<= kout(2);
        kin(4)<= kout(3) XOR kin_0;
        kin(5)<= kout(4) XOR kin_0;
        kin(6)<= kout(5) XOR kin_0;
        kin(7)<= kout(6);
        kin(8)<= kout(7) XOR kin_0;
        kin(9)<= kout(8) XOR kin_0;
        kin(10)<= kout(9) XOR kin_0;
        kin(11)<= kout(10);
        kin(12)<= kout(11);
        kin(13)<= kout(12) XOR kin_0;
        kin(14)<= kout(13);
        kin(15)<= kout(14);
        kin(16)<= kout(15) XOR kin_0;
        kin(17)<= kout(16) XOR kin_0;
        kin(18)<= kout(17);
        kin(19)<= kout(18) XOR kin_0;
        kin(20)<= kout(19) XOR kin_0;
        kin(21)<= kout(20);
        kin(22)<= kout(21) XOR kin_0;
        kin(23)<= kout(22) XOR kin_0;
```

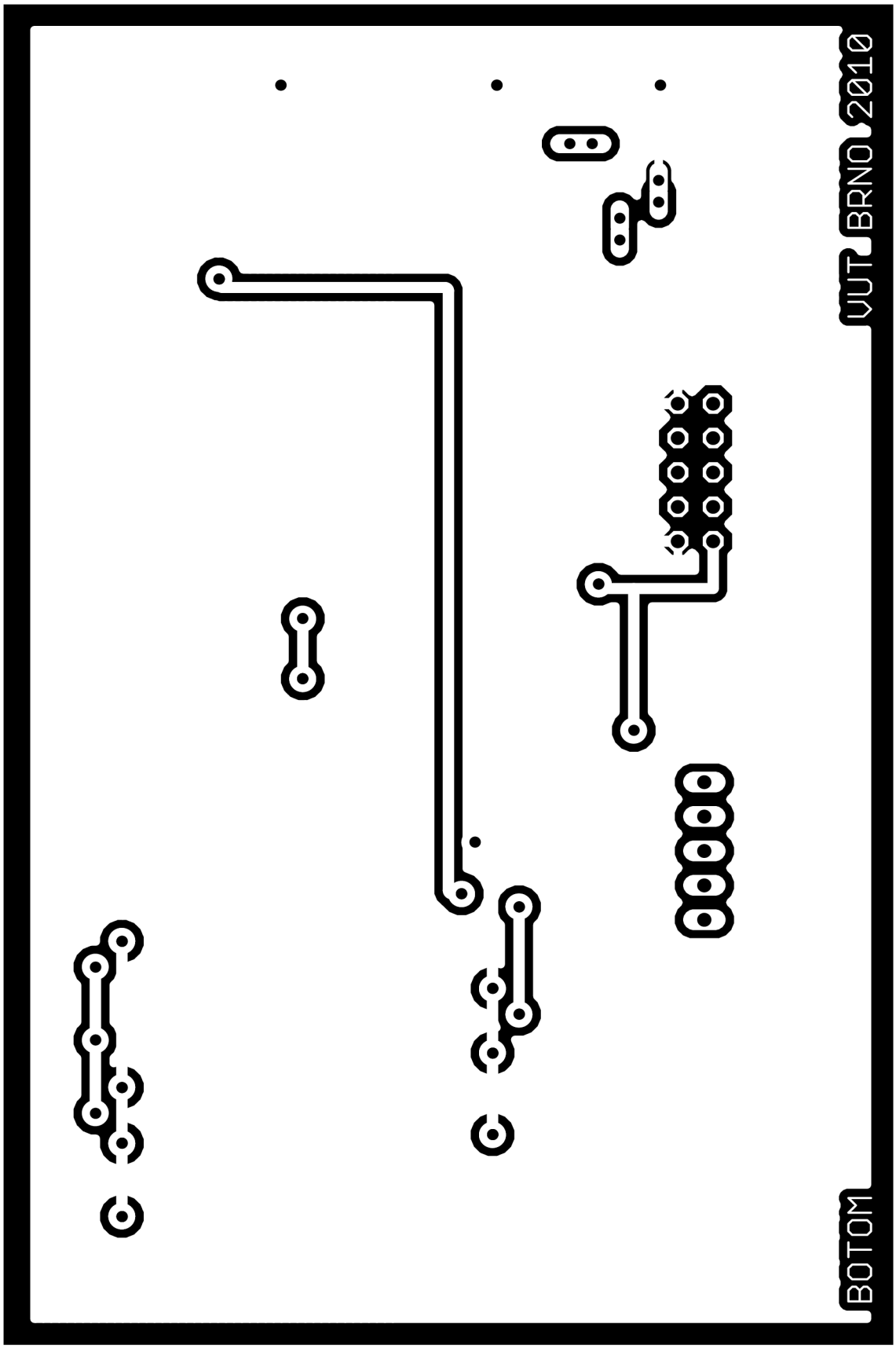
D SHÉMA ZAPOJENÍ PLOŠNÉHO SPOJE



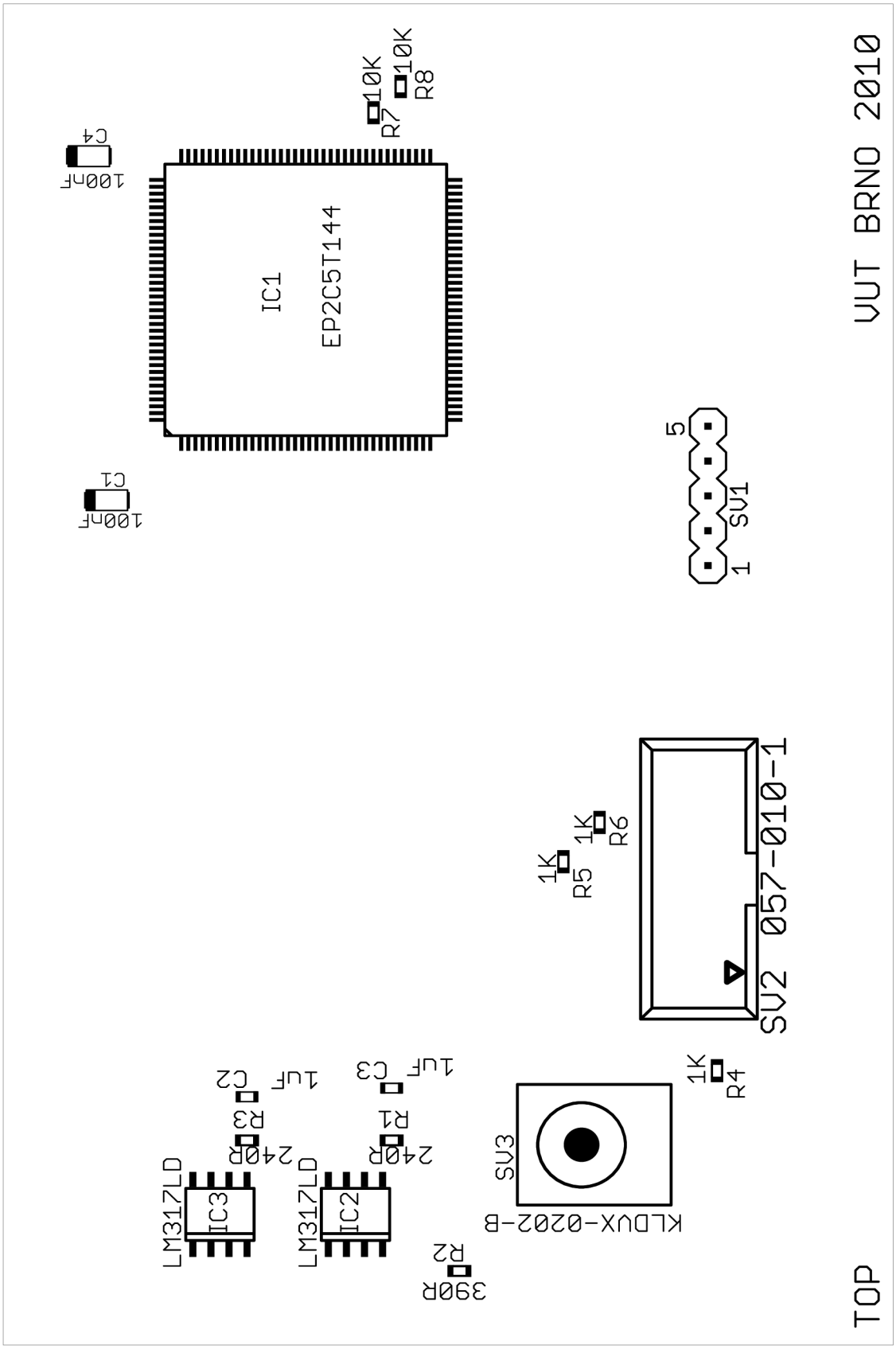
D.1 Deska plošného spoje TOP není v M1:1



D.3 Osazení plošného spoje BOTOM není v M1:1



D.3 Osazení plošného spoje TOP není v M1:1



D.4 Seznam použitých součástek

Označení	Součástka typ / hodnota	Pouzdro
C1	Tantalový kondenzátor / 100nF	SMD_A
C2	Keramický kondenzátor / 1uF	C0603K
C3	Keramický kondenzátor / 1uF	C0603K
C4	Tantalový kondenzátor / 100nF	SMD_A
IC1	FPGA EP2C5T144C8	TQFP144
IC2	Stabilizátor LM317LD	SO08
IC3	Stabilizátor LM317LD	SO08
R1	Rezistor / 240R	R0603
R2	Rezistor / 390R	R0603
R3	Rezistor / 240R	R0603
R4	Rezistor / 1K	R0603
R5	Rezistor / 1K	R0603
R6	Rezistor / 1K	R0603
R7	Rezistor / 10K	R0603
R8	Rezistor / 10K	R0603
SV1	Konektor MA05-1	
SV2	Konektor 057-010-1	
SV3	Konektor KLDVX-0202-B	