|  |  |
| --- | --- |
| Žilinská univerzita v Žilinežilinská univerzita v žiline  Fakulta riadenia a informatikyfakulta riadenia a informatiky | |
| diplomová práca  Študijný program:  **Aplikované sieťové inžinierstvo**  Podnázov práce | |
| **Bc. Branislav Juriš**  Hardvérová a aplikácia pre monitorovanie a bezpečnosť  vysokorýchlostných sieti  Vedúci práce: Ing. Petr Ivaniga, PhD.  Registračné číslo: 180/2017  Žilina, 2017Bakalárska práca | |
|  |  |

|  |  |
| --- | --- |
| Žilinská univerzita v Žilinežilinská univerzita v žiline  Fakulta riadenia a informatikyfakulta riadenia a informatiky | |
| diplomová práca  Študijný program:  **Aplikované sieťové inžinierstvo**  Podnázov práce | |
| **Bc. Branislav Juriš**  Hardvérová a aplikácia pre monitorovanie a bezpečnosť  vysokorýchlostných sieti  Žilinská univerzita v Žiline  Fakulta riadenia a informatiky  Katedra informačných sietí  Žilina, 2017  Bakalárska práca | |
|  |  |

Zadanie temy

ČESTNÉ VYHLÁSENIE

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí. Všetky zdroje použité v bakalárskej práci som uviedol v súlade s predpismi.

Súhlasím so zverejnením práce a jej výsledkov.

V Žiline, dňa 24.04.2015 Branislav Juriš

poďakovanie

Chcem sa touto cestou poďakovať predovšetkým môjmu vedúcemu práce Ing. Petrovi Ivanigovi, PhD. Za ochotu a množstvo času, ktorá mi venoval pri vytváraní tejto práce. Za jeho cenné rady a pripomienky, ktorými ma vždy dokázal naviesť správnym smerom. Rovnako patrí veľká vďaka mojej rodine a blízkym za obrovskú morálnu podporu z ich strany.

Abstrakt

JURIS, Branislav: *Hardvérová a aplikácia pre monitorovanie a bezpečnosť vysokorýchlostných sieti* [diplomová práca] – Žilinská univerzita v Žiline. Fakulta riadenia a informatiky; Katedra informačných sietí. – Vedúci: doc. Ing. Petr Ivaniga, PhD. – Stupeň odbornej kvalifikácie: Inžinier v odbore Aplikované sieťové inžinierstvo. Žilina: FRI ŽU v Žiline, 2017. – 67 s.

Cieľom diplomovej práce je oboznámenie sa z monitorovaním vysokorýchlostných sieti a možnými útokmi, ktoré nastávajú v sieti. Súčasťou práce je získanie znalostí o ochrane vysokorýchlostných sieti, ale aj práca z FPGA obvodmi pre potrebu odskúšania ochranných kódov na NetFPGA zariadení. V práci sú stručne zhrnuté znalosti z oblasti vysokorýchlostných sieti ale aj postup tvorby FPGA návrhu a popísané HW komponenty NetFPGA karty. Časť práce je venovaná popisu problémov vo vysokorýchlostných sieťach a spôsoby odstraňovania týchto problémov. V samostatnej kapitole je podrobne popísaná NetFPGA karta, ktorá je použitá v tejto práci. Práca uvádza popis vývoja implementácie Reed-Solomon kódu pre NetFPGA kartu v jednotlivých krokoch. Na základe výsledkov realizácie a testovania projektu sú zhrnuté výhody a nevýhody použitej technológie implementácie projektu v počítačových sieťach.

Kľúčové slová: NetFPGA, FPGA, HDL programovanie, HDL Code, IP Core, počítačová sieť.

Abstract

JURIS, Branislav: Hardware application for monitoring and safety of high-speed networks [Master thesis] – The University of Žilina. Faculty of Management Science and Informatics; Department of information networks. – Tutor: doc. Ing. Petr Ivaniga, PhD. – Qualification level: Master in field Applied network engineering. Žilina: FRI ŽU in Žilina, 2017. – 67 pages.

The aim of the thesis is the acquaintance with the monitoring of high-speed networks, possible attacks that could happen inside a network, acquirement of knowledge about high-speed network security, and learning how to work with FPGA circuits in case of testing the security codes on a NetFPGA device. The thesis contains briefly summarized information from the field of high-speed networks and the process of creating FPGA design and the description of HW components of NetFPGA card. Part of the thesis is focused on the description of problems which occur in high-speed networks and means of eliminations of these problems. There is a dedicated chapter to the thorough description of the FPGA card, which was used in the thesis. The thesis states the step by step process of creating an implementation of the Reed-Solomon code for the NetFPGA card. The summary of advantages and disadvantages of used technology and implementation in computer networks is based on the results of realization and testing of the project.

Key words: NetFPGA, FPGA, HDL programming, HDL Code, IP Core, computer network.

**Obsah**

[Zoznam použitých obrázkov 10](#_Toc470620814)

[Zoznam použitých tabuliek 11](#_Toc470620815)

[Zoznam použitých skratiek 12](#_Toc470620816)

[Úvod 14](#_Toc470620817)

[1. Úvod do monitorovania vo vysokorýchlostných sieťach 15](#_Toc470620818)

[1.1 Analýza monitorovania vysokorýchlostných sieti 15](#_Toc470620819)

[1.2 Potencionálne útoky v sieti 16](#_Toc470620820)

[1.2.1 Šum 17](#_Toc470620821)

[1.2.2 Skreslenie 18](#_Toc470620822)

[1.2.3 Rušenie 18](#_Toc470620823)

[1.2.4 Útok na synchronizačný kód 19](#_Toc470620824)

[1.3 Možností hardvérových možností monitorovania vysokorýchlostných sieti. 19](#_Toc470620825)

[1.3.1 Softvérové monitorovanie 19](#_Toc470620826)

[1.3.2 Hardvérové monitorovanie 20](#_Toc470620827)

[2. Metódy zabezpečenia prenosu 22](#_Toc470620828)

[2.1 Automatic repeat request (ARQ) 22](#_Toc470620829)

[2.2 Forward error correction 22](#_Toc470620830)

[2.3 Blokové kódy 23](#_Toc470620831)

[2.4 Konvolučné kódy 23](#_Toc470620832)

[2.5 Paritný bit 24](#_Toc470620833)

[2.6 Kontrolné sumy 24](#_Toc470620834)

[2.7 Cyklický redudančný súčet 25](#_Toc470620835)

[3. Reed-solomon samopravný kód 26](#_Toc470620836)

[3.1 Čo je Reed-Solomon kód ? 26](#_Toc470620837)

[3.2 Princíp fungovania Reed-Solomon kódu 27](#_Toc470620838)

[3.3 Encoder 28](#_Toc470620839)

[3.4 Decoder 28](#_Toc470620840)

[4. FPGA zariadenie dostupné na KIS 30](#_Toc470620841)

[4.1 Špecifikácia Karty 30](#_Toc470620842)

[4.2 Zdroje napájania 31](#_Toc470620843)

[4.3 Oscilátory 31](#_Toc470620844)

[4.4 Pamäte 31](#_Toc470620845)

[4.4.1 BPI-Flash pamäť 31](#_Toc470620846)

[4.4.2 Block RAM (BRAM) 32](#_Toc470620847)

[4.4.3 DDR3 pamät 32](#_Toc470620848)

[4.4.4 QDRII+ pamäť 32](#_Toc470620849)

[4.4.5 SD karta 32](#_Toc470620850)

[4.5 Ostatné periférie 33](#_Toc470620851)

[4.5.1 Ethernet PHY čipy 33](#_Toc470620852)

[4.5.2 PCI-express rozhranie 33](#_Toc470620853)

[4.5.3 PMOD a FMC rozširujúce konektory 33](#_Toc470620854)

[4.5.4 PIC Mikroradič 34](#_Toc470620855)

[4.5.5 LED diódy a tlačidlá 34](#_Toc470620856)

[5. Postup vývoja a práce. 35](#_Toc470620857)

[5.1 Použité vývojové prostredia 35](#_Toc470620858)

[5.2 Mathworks Simulink 35](#_Toc470620859)

[5.2.1 Vytvorenie simulačného modelu v Simulinku 36](#_Toc470620860)

[5.2.2 Overenie funkčnosti 37](#_Toc470620861)

[5.3 Simulácia HDL kódu vo Vivade 39](#_Toc470620862)

[5.4 Tvorba IP modulu 40](#_Toc470620863)

[5.4.1 RSEncoder ver. 1.0 40](#_Toc470620864)

[5.5 Návrh IP modulov na FPGA kartu 42](#_Toc470620865)

[5.6 Tvorba a genrovanie VHDL kódu 42](#_Toc470620866)

[Záver 43](#_Toc470620867)

[Zoznam použitej literatúry 44](#_Toc470620868)

[Prílohy 45](#_Toc470620869)

Zoznam použitých obrázkov

[Obrázok 1.0.1 Komunikačný systém 17](#_Toc470620870)

[Obrázok 3.1 Postup pri dekódovaní správy 28](#_Toc470620871)

[Obrázok 4.1 FPGA 1G - CML 30](#_Toc470620872)

[Obrázok 5.1 Simulačný model so Simulinku 36](#_Toc470620873)

[Obrázok 5.2 Simulácia RS kódera v Logic Analyzéry 38](#_Toc470620874)

[Obrázok 5.3 Simulácia RS kódera vo Vivade 40](#_Toc470620875)

[Obrázok 5.4 Tvorba IP modulu 41](#_Toc470620876)

[Obrázok 5.5 Výsledný IP modul HDLEncoder 42](#_Toc470620877)

Zoznam použitých tabuliek

[Tabuľka 3.1 Prvky poľa pre GF(16) z polynómu p(x)= x4 + x + 1 22](#_Toc470010518)

[Tabuľka 5.1 Tabuľka parametrov nastavených v Simulinku 30](#_Toc470010519)

Zoznam použitých skratiek

BER Bit Error Rate

BPI Byte Peripheral Interface

CPLD

CRC Cyclic Redundacy Check

DCM Digital Clock Manger

DDR3 Double Data Rate

ECC Error-Correcting Code

FEC Foward Error Correction

FMC FPGA Mezzanine Card

FPGA Field Programmable Gate Array

GF Galoins Field

HDL Hardware Description Language

IP Intellectual Property Core

LDPC Low-Density Parity-check Code

LFSR Linear Feedback shift register

MIG Memory Interface Generator

SONET Synchronous Optical Network

PMOD Peripheral Module

QDRII Quad Data Rate 2

QoS Quality of Service

RAM Random Access Memory

RS Reed-Solomon

SNR Signál to Noise Ratio

SRAM Static Random Access Memory

TCP Transmission Control Protocol

UART Universal asynchronous receiver and transmitter

# Úvod

Hlavným znakom dnešnej doby zrýchľujúci rozvoj vo informačný a komunikačných technológiách. Tento trend je možné pozorovať najmä pri výpočtovej technike, ktorá predstavuje jednu z najviac rozvíjajúcich oblastí tejto doby. Do tejto kategórie môže zaradiť aj oblasť sieťovej komunikácie a do nej spadajúca celosvetová sieť (internet). Internet neslúži iba ako lacná a rýchla sieť ale aj súži na poskytovanie širokého sortimentu služieb pre rôznych užívateľov. Prevádzka na internete je rôzneho druhu, ktorý výrazne rastie. Jedná sa o populárne sociálne siete a multimediálny obsah vysokej kvality ako je napríklad rôzne videá a internetové televízie.

Význam počítačových sietí, aj napriek ich obrovskému rozšíreniu, naďalej

neúprosne rastie. Pribúda počet zariadení schopných sieťovej komunikácie. Jedná sa hlave o mobilné telefóny, ktoré sú čoraz častejšie online. Ďalšou záťažou je zväčšenie kvality prenášaných obrazových materiálov a teda aj logický nárast prenášaných dát. Tieto trendy majú za následok exponenciálny nárast objemu dát prechádzajúcich sieťou za posledných päť rokov vzrástol objem sieťovej komunikácie viac ako12 násobne. Dôsledkom uvedeného nárastu je neustála potreba operátorov sprostredkujúcich internet rýchlejšej a najmä výkonnejšej sieťovej infraštruktúry na zabezpečenie prenosu dát.

V dnešnej dobe sa preto vo vysokorýchlostných sieťach sa stávajú štandardom nie rýchlosti 1 a 10 Gb/s ale zariadenia ktoré využívajú 40 a 100 Gb/s technológie.

Cieľom mojej diplomovej práce je sa oboznámiť sa z monitorovaním vysokorýchlostných sieti, možnými útokmi ktoré v sieti nastávajú. Úlohou je získanie znalostí ochrany vysokorýchlostných sieti. ale aj naučiť sa pracovať z FPGA obvodmi. Pre potrebu odskúšania ochranných kódov na NetFPGA zariadení je potrebné oboznámenie sa z vývojovým prostredím pre NetFPGA karty. Následný návrh a implementácia zabezpečovacích kódov na FPGA obvod.

Práca pozostáva zo xy kapitol. popis kapitol

1. Úvod do monitorovania vo vysokorýchlostných sieťach

Postupom času ako sa internet rozširuje a zvyšuje sa každoročný nárast prevádzky v internetových sieťach ale aj v lokálnych sieťach je potrebné si určiť priority, ktorí dátový tok má prednosť, aké veľké linky treba použiť ako treba dimenzovať kapacitu siete. Preto je dobré vedieť čo sa reálne po sieťach a v akom množstve posiela. Vysokorýchlostné siete považuje dátové linky ktoré majú kapacitu viac ako 40 Gb/s. Pri takýchto rýchlostiach vznikajú rôzne anomálie rôzneho druhu ale aj vplyvu. Preto je potrebné zabrániť vzniku chýb pri prenose prípadne rôznym útokom o ktorých píšem v ďalšej kapitole .

Analýza monitorovania vysokorýchlostných sieti

Existuje viac druhov a pohľadov na analýzu monitorovania vysokorýchlostných sieti. Potrebné si je určiť priority, ktoré potrebujeme zistiť a sú potrebné pre našu ďalšiu prácu. Monitorovanie siete môžeme prevádzať dvomi spôsobmi:

* pomocou softvérovej sondy
* pomocou hardvérovej sondy

Voľba správneho monitorovacieho zariadenia alebo softvéru môžeme zistiť rôzne údaje o dátovom prenose. Softvérové sondy sa najčastejšie používajú pri malých prenosových rýchlostiach, kde nie je potrebné spracovať až taký objem dát. Najčastejšie sa jedná o monitorovanie domácich sieti. Prípadne zariadení pripojených 1 Gb linkou do siete. Pri väčších rýchlostiach sa kladie dôraz na spracovanie a na informácie zobrazované v reálnom čase. Najčastejšie sa využívajú rôzne sondy. Veľa výrobcov implementuje takéto sondy do svojich zariadení napríklad sa jedná o brány (firewall-y). Prípadne existuje ďalšia možnosť a tou sú vysokorýchlostné sondy založené na FPGA poliach. Tieto zariadenia sú určené na spracovanie veľkého objemu dát v reálnom čase z minimálnym oneskorením.

Správnym spracovaním dát môžeme zisti nielen objem toku dát v sieti ale aj napríklad či nedochádza k stratám paketov alebo opakovaným žiadostiam o odoslanie predchádzajúcej správy. Žiadosť o opakované odoslanie správy signalizuje, že paket vôbec neprišiel tento spôsob nastáva hlavne pri TCP spojeniach. Druhou možnosťou je poškodenie paketu pri prenose sieťou.

Potencionálne útoky v sieti

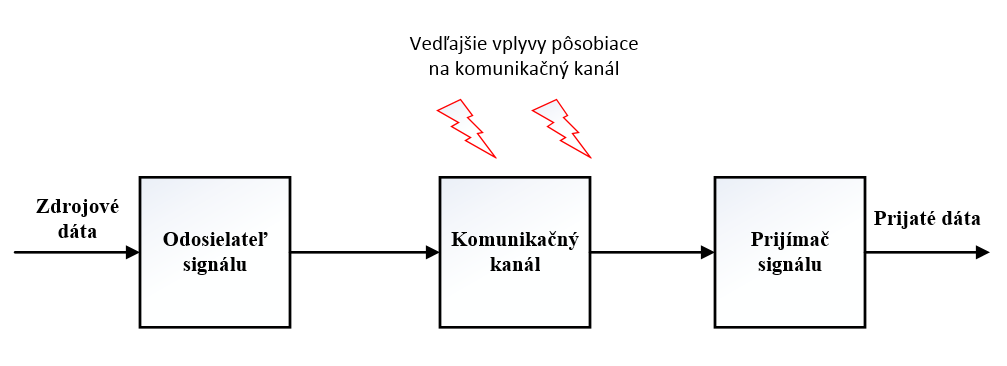
Útoky v sieti môžu byť definované rôznym spôsobom. Záleží od spôsobu prevedenia útoku, rozsahu útoku a podobne. Každý výrobca sieťových zariadení ale aj poskytovatelia služieb sa snažia zabránim rôznim typom útokov. Preto môžeme útoky rozdeliť do dvoch základných skupín:

* cielené útoky spôsobené človekom
* útoky spôsobené vplyvom vonkajšieho prostredia na komunikačný kanál.

Cielené útoky spôsobené človekom sú najčastejším javom dnešnej doby. Pre tento spôsob útokov sa najčastejšie využívajú rôzne možnosti zahltenia siete. Jedná sa najmä o rôzne spôsoby DDoS útokov. Tento druh útoku spôsobuje abnormálne zaťaženie siete a tým pádom zariadenia, ktoré obsluhujú sieť nestíhajú dané požiadavky vybavovať a pre posielať. Pri správnom nastavení filtrov dochádza k zahadzovaniu takýchto správ. Takýto spôsob riešenia problémov pomáha udržať priepustnosť siete.

V mojej práci som sa skôr zameral na útoky spôsobené vplyvom vonkajšieho prostredia na komunikačný kanál. Najčastejšie sa jedná o útoky, ktoré spôsobujú poveternostné vplyvy. Na zistenie takýchto útokov sa najčastejšie používa výpočtový spôsob nazývaný Bitová chybovosť (Bit error rate)

Bitová chybovosť (BER) je pomer počtu správne prijatých k celkovému poštu prijatých bitov. Tento pomer vyjadruje spoľahlivosť celého komunikačného systému. Medzi základné merateľné charakteristiky výkonnosti komunikačného systému patria prijaté a odoslané bity („bits out“ a „bits in“). Ako je ukázané na obrázku 1 komunikačný systém sa skladá z odosielateľa bitov (Transmiter), ktorý konvertuje surové bity alebo tiež sekvencia binárnych číslic. Konvertovaný signál spĺňa formát prispôsobený vlastnostiam komunikačného kanála (Channel). Tento signál následné príjme prijímač signálu (Receiver), ktorý obnoví prijatý signál na binárne číslice a následne sú tieto údaje spravované konečným cieľom. Kanál je fyzické médium slúži na odoslanie signálu od vysielača k prijímači. Médium môže byť vzduch, drôtené linky, optické vlákna a podobne. Počas prenášania komunikačným kanálom môže dôjsť k poškodeniu signálu. Základný vplyv na poškodenie signálu má tepelný šum generovaný elektronickým zariadením, elektrické výboje blesku počas búrky alebo elektromagnetické žiarenie rôznych zariadení.



Obrázok 1.0.1 Komunikačný systém

Pomocou BER je možné odhaliť, že sieť je vystavovaná vysokej vonkajšej intervencii a teda je potrebné riešiť zabezpečenie siete. Na tieto účely slúžia rôzne metódy čiastkovej opravy alebo ochrany prenášaných údajov. Tieto metódy v krátkosti pospisujem v nasledujúcej kapitole.

Okrem rôznych vonkajších poveternostných vplyvov môžu na komunikačný kanál vplývať cielené útoky. Ich najčastejšou úlohou je znemožniť dátový prenos po komunikačnom kanále spôsobom, že prijaté dáta nebudú čitateľné prijímačom a teda budú považované za neplatné. Spôsoby cielených útokov:

* šum
* skreslenie
* rušenie
* útok na synchronizačný kód

V nasledujúcich podkapitolách popíšem v krátkosti jednotlivé druhy útokov.

Šum

V informačných technológiách a elektronike je šum považovaný ako náhodné kolísanie elektrického signálu. Tento jav nastáva vo všetkých elektronických obvodoch ale aj pri prenosových kanáloch založených na metalickej prenosovej sústave. Šum, ktorý najčastejšie vydávajú elektronické zariadenia je veľmi rozdielny a dokonca môžeme mať rozličný vplyv na prenosový kanál. V telekomunikačných systémoch, je šum ako veličina a veľmi dôležitá informácia. Šum je súčtom nechcenej alebo rušivej energie z umelých alebo prírodných zdrojov. Najčastejšie vyjadrenie šumu je ako pomer signálu a šumu (SNR)

Skreslenie

Skreslenie signálnu je zmena tvaru signálu pre prenose alebo transformácii. Existujú dva hlavné prejavy narušenia kanálu pomocou veľkosti a fázového skreslenia signálu. Okrem tohto problému v rádiovej komunikácii existuje viaccestný efekt. Tento efekt prenáša signál odlišnými cestami pre prijímač. Dôsledok tohto je viac verzií signálu s rôznym oneskorením a útlmom. Skreslenie potom môže vytvoriť veľkú deformácii a výsledkom toho je znížená kvalita komunikácie alebo dokonca môže dôjsť k rozpadu procesu komunikácie[2]. Skreslenie sa označuje ako SINAD (Signal to noise and distortion).

Rušenie

V telekomunikáciách a elektrotechnike je za rušenie považované všetko čo modifikuje alebo inač narúša signál prechádzajúci medzi zdrojom a príjemcom správy. Za rušenie sa považuje pridanie nežiaducich signálov do prenosového kanála. Existuje veľa druhov a prípadov použitia takéhoto rušenia. Intersymbolové rušenie (ISI) je spôsob skreslenia pri ktorom pomocou jedného symbolu môžeme vyrušiť niekoľko nasledujúcich symbolov. Šírenie takéhoto impulzu mimo jeho vymedzený časový interval môže vyrušiť celú správa. Preto je potrebné aby pri dvojcestných komunikáciách dochádzalo k synchronizácii prenosových signálov.

Útok na synchronizačný kód

Za synchronizačný kód sa považuje unikátny dekódovateľný kód, ktorým sa začína tok dát prenášaných po prenosovom kanáli. Útok spočíva v zmene synchronizačných kódov alebo rôzne spôsoby dostráňovania týchto kódov. Ak by došlo k zlúčeniu dvoch dátových tokoch a tým porušeniu synchronizačných kódov potom je prijatá správa považovaná za neplatnú. Vo všeobecnosti každý synchronizačný kód sa nachádza ako prefix pred dátami ale nie všetky prefixy môžeme považovať za synchronizačné kódy.

Možností hardvérových možností monitorovania vysokorýchlostných sieti.

Dnešná doba ponúka rôzne možnosti monitorovania sietí. Potrebné je však rozlišovať o aké druhy monitorovania ide. Je potrebné si určiť čo chceme monitorovať a v akom rozsahu je potrebné monitorovať. Na monitorovanie vysokorýchlostných sieti môžeme využiť dva druhy monitorovania.

* Softvérové monitorovanie
* Hardvérové monitorovanie

Softvérové monitorovanie

Každý spôsob monitorovania má svoje výhody. Pri softvérom monitorovaní medzi výhody patrí ľahká dostupnosť rôznych softvérov, vysoká modularita a možnosť na sadania v rôznych prostrediach. Naopak medzi hlavné nevýhody patrí rýchlosť spracovania údajov pokiaľ sa jedná o vysokorýchlostné siete. Ak hovoríme o softvérovom monitorovaní prevažne sa jedná o sondy ktoré dokážu monitorovať dátovú prevádzku v lepšom prípade údaje každého paketu alebo rámca. Monitorovanie týchto vymenovaných vecí veľmi zaťažuje hardvér hosťujúceho zariadenia následne vznikajú dosť výrazné oneskorenia. Ďalšie sondy dokážu monitorovať hardvérové činnosti sieťových uzlov ako sú teploty výpočtových procesorov, využite pamäte alebo využite sieťových kariet a podobne. V tomto prípade až taký problém z oneskorením nevzniká. Možnosť softvérového monitorovania ponúka rôzne spôsoby analýzy monitorovaných dát.

Hardvérové monitorovanie

Hardvérové monitorovanie sa v dnešnej dobe využíva hlavne kvôli svojej rýchlosti spracovania údajov. Nevýhodou sú vysoké vstupné náklady na vytvorenie alebo zakúpenie takýchto zariadení. Ďalšou podstatnou nevýhodou je fyzické umiestnenie takejto sondy čo môže byť za určitých podmienok veľmi náročné. V dnešnej dobre začínajú výrobcovia stavať univerzálne zariadenia na monitorovanie siete ale pridávajú im ďalšie funkcie aby boli pre zákazníkov výhodnejšie. Ako hardvérové sondy na monitorovanie dátových sieti sa v dnešnej dobe využívajú najmä zariadenia Firewall-y a FPGA sondy.

**Firewall-y**

Ich prvotnou úlohou je filtrovať sieť a zabezpečiť ju pred možnými útokmi. Doplnkovou funkciou je však zber dát, ktoré si zariadenie ukladá. Následne sa tieto dáta dajú pomocou dodatočného softvéru vyhodnotiť. Toto sa javí ako výhodné riešenie z pohľadu prevádzkovateľa tejto siete. Keďže firewall-y sa najčastejšie nasadzujú na vstupné a výstupné linky do siete a tým pádom nie je možný náhľad a monitorovanie liniek, ktoré sa nachádzajú vo vnútri siete. Trendom dnešnej doby je však vytvárať virtuálne firewall-y, ktoré bežia na serveroch z inými funkciami a týmto sa snažia šetriť svoje náklady na údržbu a rozširovanie siete. Lenže virtuálizovaním takýchto zariadení, ktoré slušia aj na monitorovanie siete dochádza k zväčšeniu latencie teda oneskorením dát v sieti. Pri virtuálnych firewall-och už nemôžeme hovoriť o hardvérovom monitorovaní.

**FPGA sondy**

Druhou a veľmi zaujímavou alternatívou sú práve FPGA sondy. Tieto sondy pracujú ako integrované číslicové obvody a teda nie sú závislé od rôznych materských operačných systémov. Ich výhodou je rýchle spracovanie údajov a pri vhodnej implementácii je takáto sonda obohatená o automatické spracovanie a vyhodnocovanie údajov. Vyhodnotené údaje môže pre posielať administrátorovi alebo dokonca automaticky môže prispôsobovať nastavenia siete. Pod pojmom nastavenia siete sa myslia najmä QoS a ochrany pred útokmi. Takéto úpravy fungujú v reálnom čase. Dôsledkom čoho je zlepšenie a z efektívnenie priepustnosti siete . Takéto sondy je možné nasadiť v rôznej časti topológie site. Len je potrebné zabezpečiť pre takúto sondy vhodné hostiteľské prostredie. Najčastejšie túto funkciu zastávajú počítače alebo moduly pre rozšírenie serverov.

# Metódy zabezpečenia prenosu

Počas prenosu v reálnych prostrediach dochádza k takzvanému šumu z rôznych príčin. Tento šum spôsobuje, že počas prenosu správa dôjde k chybe. Prijatá správa potom nie je úplne totožná s vyslanou správou. Môžu nastať dva druhy chýb. Prvá z nich je označovaná ako „vznik“ znaku, aj keď nebol vyslaný, alebo opačný prípad „pohĺtenie“. Tieto chyby sa však vyskytujú veľmi zriedka. Skôr som sa zameral na druhý druh chyby pri prenose a to je zámena vyslaného znaku za iný znak.

Na prenos dát sa používajú rôzne metódy zabezpečenia, ktoré sa snažia aspoň čiastočne zabezpečiť dáta. V súčasnej dobe sa používajú rôzne zabezpečovacie metódy. Každá z nich ma svoje výhody aj nevýhody. Preto záleží pri akých dátových tokoch sa používa. Tento problém riešia rôzne štandardy a inštitúcie ako sú ITU (International Telecommunication Union) alebo IETF (Internet Engineering Task Force). Všeobecne sa takéto kódy označujú ako bezpečnostné kódy alebo tiež ako samoopravné kódy. Tieto kódy sú o čosi dlhšie pretože obsahujú aj určité množstvo prídavných informácii. Jednotlivé spôsoby zabezpečenia popíšem v ďalšej časti.

Automatic repeat request (ARQ)

Automatická požiadavka na opakovanie alebo tiež občas nazývaná spätná oprava chýb. Táto technika opravy chýb v kombinácii so žiadosťou o opakovanie prenosu chybných dát. Každý blok prijatých dát sa kontroluje pomocou kódu na detekciu chýb. Ak kontrola zlyhá, je požadované opakované vysielanie dát, toto opakovanie môže byť vykonané viac krát, kým sa dáta nedajú overiť.

Forward error correction

Forward error correction (FEC)alebo tiež kanálové kódovanie je technika používaná pre kontrolu chýb v prenose dát menej spoľahlivé alebo hlučné komunikačné kanály. Myšlienkou tohto spôsobu je že, odosielateľ kóduje správu redundantným spôsobom pomocou error-correcting code (ECC). Redundancia umožňuje, aby prijímač detekoval obmedzený počet chýb, ktoré sa môžu vyskytnúť počas prenosu v správe. Často tieto chyby opraví bez potreby opätovného prenosu. Pomocou FEC má prijímač schopnosť opraviť chyby bez potreby žiadosti o opätovné poslanie dát, pri ktorých by tiež mohlo nastať poškodenie. FEC je vhodné používať pri prenosoch, kde sú opakované prenosy nákladné alebo nemožné ako napríklad jednosmerné komunikačné linky alebo pri prenose pre viacerých prijímačov teda multicast-ové správy. FEC sa delí do dvoch základný kategórii:

* Blokové kódy
* Konvolučné kódy

Blokové kódy

Blokové kódy posielajú správu cez rozdelené a oddelené bloky dát. Postupnosť symbolov vstupujúca do kódera je delená na bloky o veľkosti *k* bitov. Každý blok má vlastnú ochranu pomocou parity. Parita je pridaná na koniec bloku a skladá sa z kombinácie kódovaného slova. Všeobecne používaný systematický kód, znamená že proces kódovanie nemení symboly správy ale symboly ochrany sú pridané ako samostatná časť bloku. Kódy, pomocou ktorého chceme opraviť všetky chyby nesmú obsahovať poškodenú časť parity. Najjednoduchšie kódy patria kódy s kontrolnou paritou a kódy z opakovaním.

Konvolučné kódy

Konvolučné kódy sú rozšírené v mnohých súčasných aplikáciách. Ich cieľom je dosiahnuť spoľahlivého dátového prenosu. Tento druh kódov sa používa hlavne pri digitálnom prenose ako je video, rádio, mobilná a satelitná komunikácia – štandardy (IEEE 802.11 a IEEE 812.16). Konvolučné kódysú založené na spracovaní bitov sériovým spôsobom, nie je potrebné využitie veľkých vyrovnávacích pamätí. Dĺžka vstupných informačných blokov *k* je zvyčajne malé číslo ( 1, 2, 3,... bity), pričom n-bitové slová závisia nielen na aktuálnej *k* –tici informačných bitov, ale aj na predchádzajúcich *k*-ticiach.

Špeciálna forma konvolučných kódov, nazývané tiež systematické konvolučné kódy. Tento spôsob využíva výstupné bity, ktoré obsahujú ľahko rozpoznateľne postupnosti vstupných bitov . Systematické kódy sú častejšie preferovane oproti nesystematickým. Dôvodom takéhoto využitia je že umožňujú rýchlejší pohľad. Tiež sú menej náročné na hardvér pri kódovaní. Medzi dôležité vlastnosti systematických kódov je, že nie sú „katastrofické“, čo znamená že sa chyby nemôžu

množiť kaskádovým spôsobom. Všetky vlastnosti sú veľmi žiaduce pre kvalitu prenosu. Vlastnosť ochrany chýb majú však systematické kódy ale aj nesystematické majú rovnakú.

Paritný bit

Jedná sa o jednoduchá kontrolný súčet určený na ochranu integrity dátového slova. Zvyčajne sa pridáva na koniec dátového slova. Tento bit vyjadruje či v dátovom slove sa nachádza párny alebo nepárny počet logických jednotiek. V prípade nepárneho počtu logických jednotiek v danom dátovom slove, je paritný bit nastavený na logickú jednotku. Analogicky to funguje aj pri párnom slove, kde je paritný bit nastavený na logickú nulu. Teda súčet dátového slova plus paritný bit musí byť parné číslo. Druhý pohľad na paritu môže byť vzájomný XOR všetkých bitov v slove, alebo tiež súčet všetkých bitov v slove upravených modulom 2. Párna parita sa dá vyjadriť aj ako CRC, tá je generovaná polynómom x+1.

Kontrolné sumy

Kontrolný súčet správy je modulárny aritmetický súčet kódových slov správy o pevnej dĺžky slov napríklad ako bitová hodnota slova. Súčet môže byť negovaný pomocou operácie doplnku pred prenosom správy. Výsledkom doplnku je nulová správa.   
Kontrolné sumy zahŕňajú paritné bity, kontrolné číslice a číslice na pozdĺžnu kontrolu redundancie. Medzi najznámejšiu kontrolnú sumu patrí MD5. Táto funkcia sa už ale považuje za definitívne degradovanú pre bezpečnosť kvôli publikovaní metódy pre rýchle hľadanie konkrétnych kolízii.

Cyklický redudančný súčet

Cyklický redudančný súčet (CRC) je hešovacia bezpečnostná funkcia na detekciu náhodných chýb v digitálnych dátach v počítačovej sieti. Na základe jednotlivých bitov sa pomocou hešovania vypočítava zabezpečovací údaj. Ten sa porovnáva na konci celého bloku so zabezpečovacím údajom. Tento zabezpečovací údaj si podľa rovnakých pravidiel vypočítal odosielateľ a pripojí ho k prenášanému bloku dát. Ak prijímateľ porovná tieto dva údaje a nastane zhoda, potom prenesený blok s vysokou pravdepodobnosťou môžem považovať za správny. Pre svoju jednoduchosť a dobré matematické vlastnosti je veľmi rozšírený. Detekcia chýb záleží od správnej voľby kľúča pre hešovaciu funkciu ale vo všeobecnosti platí, dlhšie kľúče majú lepšiu detekciu chýb.

# Reed-solomon samopravný kód

Reed-Solomon kód je lineárny kód (pridaním dvoch kódových slov vytvárajúci iné kódové slovo) a cyklický (cyklický posun symbolov v kódovom slove vytvára iné kódové slovo ). Tento spôsob patrí ku skupine Bose-Chaudhuri-Hocquenghem (BCH) kódom ale vyznačuje sa multi-bitovými symbolmi. Tieto kódy sú obzvlášť výhodné pokiaľ sa jedná o jednu chybu v blokovej správe ktoré dokážu opraviť. Ak by došlo k veľkej chybovosti správy, teda viac ako polovica bitov by bola chybná, neexistuje taký kód, ktorý by ich dokázal opraviť. Vyberaním rozličných parametrov pre kódy, vznikajú rozdielne stupne ochrany ale zároveň to má zásadný vplyv na zložitosť implementácie.

Čo je Reed-Solomon kód ?

RS kód je mapovanie z vektorového priestoru o veľkosti *m* do konečného poľa *K* vo vektorovom priestore väčšej dimenzie *n* v rovnakom poli za predpokladu, že nedochádza pri prenose chybám, ktorých je viac ako *(n - m)/2*. Potom existuje postup na dekódovania, ktorý obnovuje chyby úplne. Reed-Solomov kód môže byť popísaný ako *( n , k )* kód, kde *n* je dĺžka bloku v symboloch v našom prípade v bitoch a *k*je počet informačných symbolov a *2t = n - k* je to počet paritných znakov. Každý symbol obsahuje s bitov, kde:

alebo

Z matematického vzťahu vyplýva, že pre jedno kódové slovo môže použiť maximálne *2s – 1* rôznych symbolov okrem samých núl. Maximálny počet chybných symbolov, ktoré môžu byť opravené v RS je dané hodnotou  *t* , čo je polovičná veľkosť parity. V mojom prípade som použil RS(255,239) definovaný pre optický prenos po sieti špecifikáciou OTN G.709 [1]. Veľkosť kódového slova *n*  je 255 symbolov, počet dátových symbolov *k* je 239 symbolov. Maximálny počet chýb , ktoré môžu byť opravené je 8 symbolov. Každý symbol obsahuje 8 bitov.

Princíp fungovania Reed-Solomon kódu

Funkčnosť a správnosť výpočtov je založená na Galoinsových poliach (GF). Tento spôsob výpočtu je mimoriadne vhodný pre logické obvody a programovateľné polia pretože je založený na matematickej násobenie. Takýmto postupom dokážu veľmi rýchlo logické obvody a programovateľné polia spracovať veľké množstvo dát. Galoinsové pole je konečné pole, ktoré obsahuje sadu nenulových prvkov tvoriacich cyklickú skupinu násobenia. Každý nenulový prvok môže byť vyjadrený ako mocnina primitívneho prvku.

Na vytvorenie GF je potrebné poznať ireducibilný primitívny polynóm *p(x)*. Proces vzniku takého poľa je založený na násobení dvoch elementov z poľa medzi sebou. Preto dôležité aby kóder aj dekóder používali rovnaký ireducibilný primitívny polynóm. Pre príklad uvediem GF(16), ktoré je zložené so 4 bitových slov (24). Toto pole môže využívať polynóm *p(x) = x4 + x + 1*

Kompletné GF(16) je zobrazené v tabuľke

|  |  |  |  |
| --- | --- | --- | --- |
| index | polynomická forma | binárna forma | decimálna forma |
| 0 | 0 | 0000 | 0 |
| α0 | 1 | 0001 | 1 |
| α1 | α | 0010 | 2 |
| α2 | α2 | 0100 | 4 |
| α3 | α3 | 1000 | 8 |
| α4 | α+ 1 | 0011 | 3 |
| α5 | α2 + α | 0110 | 6 |
| α6 | α3 + α2 | 1100 | 12 |
| α7 | α3 + α+ 1 | 1011 | 11 |
| α8 | α2 + 1 | 0101 | 5 |
| α9 | α3 + α | 1010 | 10 |
| α10 | α2 + α+ 1 | 0111 | 7 |
| α11 | α3 + α2 + α | 1110 | 14 |
| α12 | α3 + α2 + α + 1 | 1111 | 15 |
| α13 | α3 + α2 + 1 | 1101 | 13 |
| α14 | α3 + 1 | 1001 | 9 |

Tabuľka 3.1 Prvky poľa pre GF(16) z polynómu p(x)= x4 + x + 1

Encoder

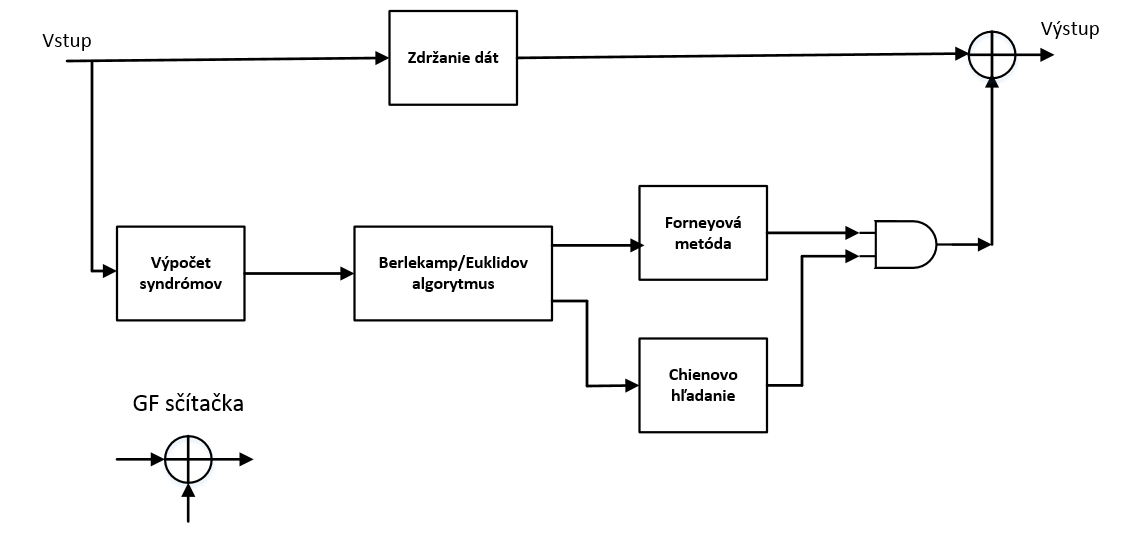
Kódovanie pomocou RS kódu je jednoducho založené na mapovaní správy o veľkosti *k* na kódové slovo veľké *n*. Mapovanie sa uskutočňuje pomocou generického polynómu. Pri RS (255,239) je využité GF(256), Toto pole je generované ireducibilný primitívny polynóm:

*p(x)*=*x16 + 59x15 +13 x14 + 104x13 + 189x12 +68 x11 + 209x10 + 30x9 + 8x8 + 163x7 + 65x6 + 41x5 + 229x4 + 98x3 + 50x2 + 36x+39*

Tento polynóm je implementovaný pomocou posuvných registroch s lineárnou spätnou väzbou (LFSR) . Pomocou GF(256) dokáže FPGA obvod pomerné rýchlo dopočítať paritu. Spôsob výpočtu je založený na delení pôvodnej správy polynómom pričom zvyšok po delení je práve ochranná parita o veľkosti 16 symbolov (16 x 8 bitov). Následné je celá správa poslaná odosielaciemu modulu.

Decoder

Na rekonštrukciu prijatej správy je potrebný trošku komplikovanejší spôsob výpočtov oproti kódovaní správy. Pre správne dekódovanie je potrebné správne poradie algoritmov. Na obrázku je znázornený postup pri dekódovaní správy.



Obrázok 3.1 Postup pri dekódovaní správy

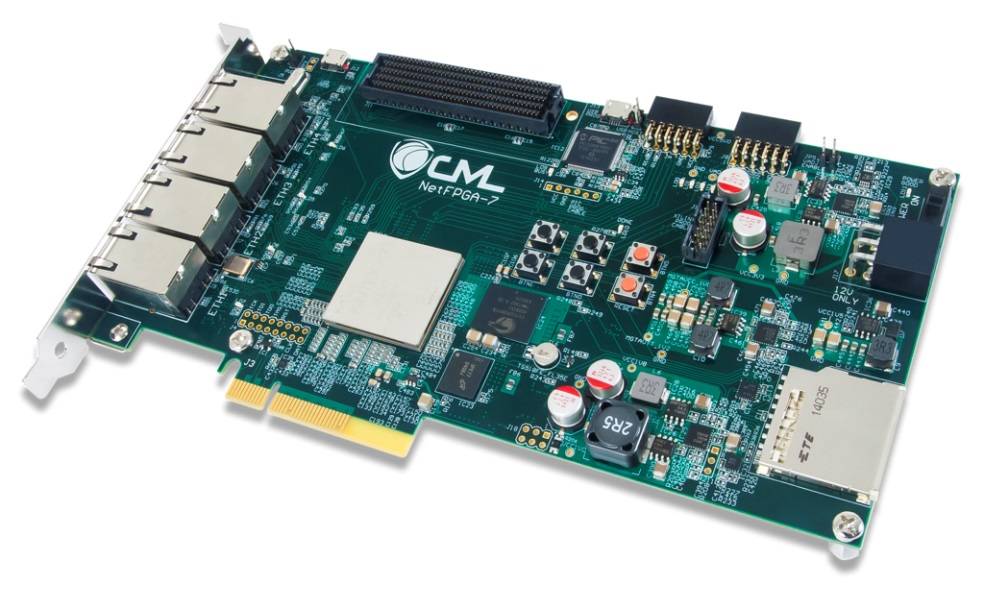
Prvým krokom je výpočet syndrómov. Prijaté kódove slovo *c(x)* je násobkom generačného polynómu, ktorý je zobrazený v predchádzajúcej podkapitole. Ak pri delení prijatého slova generačným polynómom je výsledok iný ako 0 potom je veľká pravdepodobnosť, že pri prenose nastalo poškodenie slova. Teda c(x) je delený každým jedným koreňom g(x) a výsledkom tohto delenia je slovo r(x). Ak je súčet všetkých zvyškov rovný nule tak prijaté slovo je bez chyby.

Druhým krokom je Berlekamp

Dorobiť

# FPGA zariadenie dostupné na KIS

Na overenie funkčnosti a potvrdenie správnosti mojej práce, som využil zariadenie dostupné na KIS. V tejto kapitole sa zameriavam na všeobecný popis základných parametrov karty. Jedná sa najmä o možnosti konektivity a možnosti pripojenia rozličných prídavných modulov. Jedná sa o kartu ktorá dovoľuje rozličné možnosti implementácie najmä pri práci s ethernetom. Na špecializovaných webových stránkach sa nachádzajú Open-source projekty ako napríklad softvérová implementácia smerovača, prepínača alebo sieťovej karty ako poznáme z počítačov.



Obrázok 4.1 FPGA 1G – CML

Túto kartu je však potrebné rozšíriť o Pmod USBUART, ktorý slúži ako prevodník medzi UART portom a USB portom. Celý rozširovací modul robí kros-konverziu pomocou FTDI FT232RQ (označenie čipu priamo určeného na túto konverziu). Užívateľovi umožňuje posielať dáta v oboch smeroch a na PMOD port prijímať konvertované dáta. Náhľad modulu je na obrázku



Obrázok 4.2 Pmod UARTUSB

Špecifikácia Karty

Názov modelu karty je NetFPGA-1G-CML. Výrobca karty Digilent využil ako hlavný logický obvod systému FPGA modul z označením Kintex-7 XC7K325T. Táto karta je osadená štvoricou sieťovými portmi s maximálnou rýchlosťou do 1 Gb/s. Karta využíva rôzne pamäťové moduly, ktoré dopĺňa možnosť pripojenia externej SD karty. Medzi dynamické pamäte patrí napríklad DDR3 RAM o veľkosti 512 MB a taktiež aj QDRII+SRAM o veľkosti 4,5MB. Pre potrebu uloženia konfigurácie a zároveň ako zdroj FPGA obvodu slúži BPI flash pamäť, ktoré je veľká 128 MB. Táto karta tak isto obsahuje PCIe x4 Gen 2 konektor, ktorý sa používa vo väčšine dnešných stolových počítačov. Na pripojenie rôznych rozšíriteľných modulov slúžia konektory ako napríklad FMC a PMOD.

Zdroje napájania

Napájanie karty je zabezpečené pomocou 6-pinového molex konektora. Tento istý typ konektora sa používa aj pri napájaní výkonných grafických kariet. Parametre napájacieho konektora musia byť 12V a 5A, najčastejšie sa jedná o ATX zdroje, ktoré sa používajú v počítačoch. Vedľa portu na napájanie karty sa nachádza ochranný mechanický prepínač napájania, tento treba prepnúť do polohy Zapnutý alebo tiež POWER ON.

Oscilátory

Každý logický obvod potrebuje pre svoju činnosť takzvaný clock (hodiny). Tieto hodiny zabezpečujú synchronizáciu prvkov FPGA obvodov a ostatných periférii. Na vytáranie hodinového signálu sa využívajú oscilátory. Hlavné systémové hodiny sú riadené cez diferenčný oscilátor, ktorý má dva piny positive a negative clock. Tento riadi DCM (Digital Clock Manger) bloky a má frekvenciu 200 MHz. Pre riadenie ethernetových radičov (PHY) sa využívajú oscilátor z frekvenciu 125 MHz..

Pamäte

Táto karta obsahuje rôzne pamäťové radiče, ktoré slúžia na ukladanie konfigurácie FPGA karty, rôzne zásobníky ukladania dočasných súborov a údajov alebo na kartu môže byť pripojená aj externá SD karta

BPI-Flash pamäť

Jedná s o non-volatile pamäť Mumonyx BPI flash memory s kapacitou 128 MB. Hlavným využitím pamäte je vysokorýchlostná konfigurácia FPGA obvodu. Vďaka svojej veľkosti sa do pamäte zmestí viacero konfiguračných súborov. Tieto sa dajú dynamicky riadiť pomocou externých zariadení napríklad mikroprocesor alebo CPLD. Toto zabezpečuje nakonfigurovanie karty vždy podľa aktuálnych potrieb. Po nakonfigurovaní sa dá pamäť využiť na čítanie a zápis údajov ktoré potrebujeme.

Block RAM (BRAM)

Tento druh pamäte sa využíva ako FIFO štruktúra vhodná na prenos dát medzi funkčnými blokmi. Táto karta poskytuje 1,78 MB pamäte teda 445 kusov BRAM o veľkosti 36 Kb. Každú z týchto BRAM pamätí je možné ešte dodatočne rozdeliť na dve 18 Kb RAM pamäte. Nastavenia pamätí je možno robiť cez prostredie Vivado.

DDR3 pamät

Na karte sa nachádza aj pamäť Micron MT41K512M8 DDR3 SRAM o veľkosti 512MB a pracovnou frekvenciou 800 MHz, ktorá pracuje z 8 bitovou dátovou zbernicou. Hlavným využitím je napríklad uloženie paketov vo frontoch, v našom prípade sa skôr jedná o dátový tok z ktorého vytvárame rámce. Jej nesporná výhoda je až 1600 MT/s (Million(s of) Transfers per Second) priepustnosť. Nástroj MIG slúži na automatickú konfiguráciu FPGA obvodu a prepojenie z AXI4-Systémovou zbernicou.

QDRII+ pamäť

Funkcia tejto pamäte spočíva v tom , že dokáže prenášať údaje cez READ aj WRITE porty na oboch hranách hodinového signálu. Celková rýchlo potom môže byť až 3,6Gb/s. Táto karta disponuje veľkosťou QDRII+ pamäte 4,5MB CY7C2263KV18 QDRII+ SRAM o frekvencii 450 MHz a priepustnosti 900 MT/s. Túto pamäť je možné využiť pre FIFO front ale aj Look-up tabuľky pre jej nízke oneskorenie a vysokú prenosovú rýchlosť. Celá konfigurácia tejto pamäte je vykonávaná pomocou MIG nástroja.

SD karta

Karta obsahuje slot pre SD kartu a tým je zabezpečená podpora fyzickej vrstvy. Pomocou SD karty môžeme rozšíriť NetFPGA kartu o dodatočnú non-volatile pamäť. Využitie tejto SD karty môže byť napríklad miesto pre ukladanie rôznych logovacích súborov.

Ostatné periférie

Medzi ostatné periférie patria rôzne porty na pripojenie externých zariadení alebo na priamu komunikáciu z počítačom. Môžme tam zaradiť aj rozhrania ktoré komunikujú z užívateľom teda ako napríklad led diódy a tlačidlá.

Ethernet PHY čipy

Jedná sa o štyri ethernetové porty ktoré označuje výrobca ako PHY. Porty na sieťovú komunikáciu využívajú RJ-45 konektory. Súčasťou sú pre každý port aj dve LED diódy. ktoré sa správajú rovnako ako u bežnej sieťové karty. PHY sa programujú pomocou spoločnej Management Data Input/Output (MDIO) zbernicu. Každý PHY má funkciu auto-negotation, ktorá dokáže nezávisle prispôsobovať svoju dátovú rýchlosť v rozpätí 10/100/1000 Mb/s.

Každý PHY majú vlastné adresy označené ako ETH1, ETH2, EHT3, ETH4. Prenos údajov zabezpečuje Reduced Gigabit Media Independent Interface (RGMII). Toto rozhranie používa frekvenciu 125 MHz. Pre použitie týchto konektorov je potrebné pre každý jeden pridať v návrhu už predpripravené Ethernet MAC Xilinx IP jadro.

PCI-express rozhranie

Základným rozhraním každej karty je PCIe rozhranie, ktorým sa karta priamo pripája do zbernice počítača a je určená na priamu komunikáciu FPGA obvodu a CPU počítača. Na túto komunikáciu sú určené štyri vysoko-rýchlostné sériovo zapojené GTX vysielače/príjmače, ktoré tvoria štyri samostatné Gen 2 PCIe komunikačné linky.

PMOD a FMC rozširujúce konektory

Na pripojenie dodatočných funkcionalít, ktoré sú potrebné pre daný návrh môžeme využiť PMOD a FMC konektory. Funkcionalitu PMOD karty PmodUSBUART využijem v mojej práci. Jedná sa o konverziu z UART (Universal Asynchronous Receiver and Transmiter)

PIC Mikroradič

Tento mikroradič je nastavený pre riadenie USB On-The-GO (OTG), kde môže fungovať ako Slave alebo Master. Táto funkcia mu umožňuje načítať konfiguračný súbor cez Micro USB port. Ďalej sa dá tento radič využíva na správu Real-Time hodín a na zabezpečenie bitstream súboru (konfiguračný súbor) pred neoprávneným čítaním. Táto funkcia sa nazýva Secure Storage. Tento mikro radič môže byť preprogramovaný používateľom ak je potrebné využiť iné komponenty pripojené k mikroradiču cez Micro USB port.

LED diódy a tlačidlá

Programátor má možnosť na karte použiť štyri LED diódy, ktoré sú voľne programovateľné. Prípadne môže použiť štyri voľné tlačidlá. Okrem toho sa tam nachádzajú červené tlačidlá z definovanou funkciou Reset (reset celého návrhu) a Program (vyvoláva konfiguračný proces).

# Postup vývoja a práce.

Prvým krokom bolo vytvorenie simulácie pre lepšie oboznámenie a overenie teórie funkčnosti RS(255,239). Tento druh samo opravného kódu by mal vedieť opraviť až 8 chýb. Druhým krokom bolo vytvorenie IP Core, ktoré Vivado používa pre vygenerovanie základnej šablóny kódu. Tento Blok bol pripojený k ostatným blokom navrhnutým pre komunikáciu z PC pomocou UART a Ethernetového portu pre odosielanie správ smerom do siete. Tretím krokom bolo stránenie všetkých chýb a odladením časovania karty pre správnu funkčnosť karty. Posledným a teda štvrtým krokom bolo testovanie a overenie správneho fungovania karty. Podrobný a detailný postup je popísaný v nasledujúcich podkapitolách.

Použité vývojové prostredia

Pri mojej práci som využil dve aplikácie, ktoré mi umožnili vytvoriť a implementovať RS kodér a taktiež mi pomohli lepšie pochopiť fungovanie FPGA obvodov a prácu s nimi. Ako vývojové prostredia som použil:

* Mathworks Simulink
* Xilinx Vivado

Obe prostredia ponúkajú možnosť simulácie zostaveného programu. V podkapitolách popíšem základné veci ktoré boli potrebné spraviť.

Mathworks Simulink

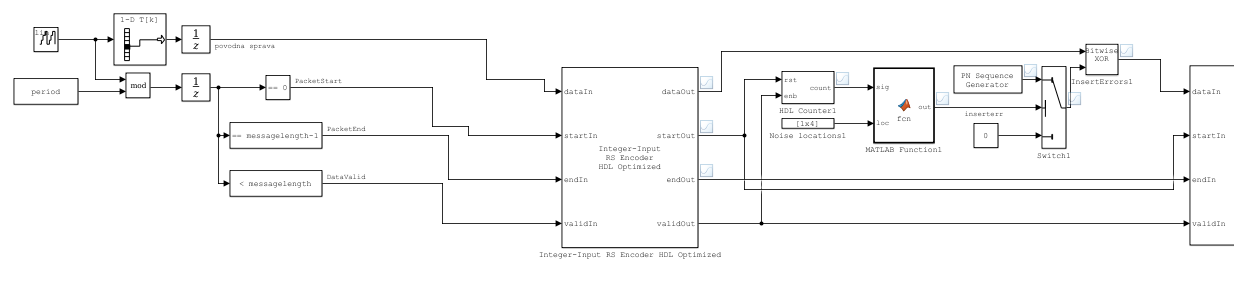
Od Firmy Mathworks je známy a často používaný program MATLAB, ktorý som doplnil o trialovu (časovo obmedzenú na 30 dní) verziu Simulinku. Následne som zistil že, Simulink podporuje hdl programovací jazyk, tak som sa pokúsil vytvoriť simuláciu z hdl dostupných modulov. Pre správnu kompiláciu a analýzu takejto simulácie bolo potrebné doinštalovať ešte moduly:  
Requires Signal Processing Toolbox

* Requires Fixed-Point Designer
* Requires DSP System Toolbox
* HDL Coder
* HDL Verifier

Pri doplnku HDL Coder bolo potrebné požiadať o licenciu e-mailom Firmu ktorá distribuuje MATLAB pre Slovensko.

Vytvorenie simulačného modelu v Simulinku

Na obrázku je zobrazený HDL Encoder a ostatné prvy na vytvorenie správy alebo vytvorenie šumu.



Obrázok 5.1 Simulačný model so Simulinku

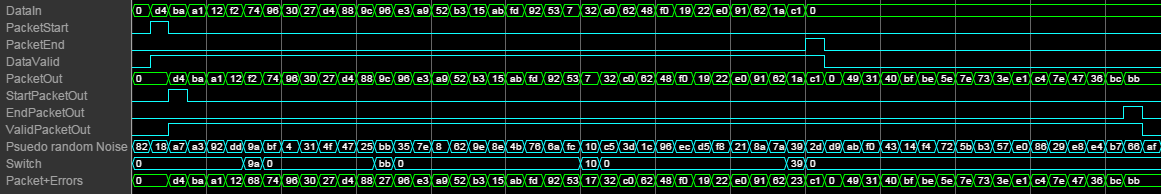
V ľavej časti sa nachádzajú prvky na vytvorenie správy a synchronizáciu vstupu a prvého a posledného bajtu. Stredný blok je RS Encoder optimalizovaný pre HDL jazyky. V pravej časti sa nachádza generátor náhodného toku a funkcie ktorá určuje pozície na ktorej sa má vykonať logická operácia XOR so pôvodnou správou s paritou a náhodným tokom. Následne sme si nastavili parametre:

|  |  |  |
| --- | --- | --- |
| **Názov parametra** | **Hodnota** | **Dátový typ** |
| K | 239 | double(auto) |
| N | 255 | double(auto) |
| message | [212 186 161 18 242 116 150 48 39 212 136 156 150 227 169 82 179 21 171 253 146 83 7 50 192 98 72 240 25 34 224 145 98 26 193 0] | double(auto) |
| messagelength | 36 | double(auto) |
| period | 236 | double(auto) |

Tabuľka 5.1 Tabuľka parametrov nastavených v Simulinku

Parametre K a N vyžaduje RS Encoder na presné nastavenie a správne fungovanie. Parameter K určiuje maxinmálnu veľkosť správy. Parameter N určuje celkovú dĺžku odoslanej správy aj z paritou ktorá v tomto prípade bude vždy 16 bajtov. Message je mnou definovaná správa a však v simuláciách bude zobrazované v Hexadecimálnom tvare. Správa v hexadecimálnom vyzerá takto: D4 BA A1 12 F2 74 96 30 27 D4 88 9C 96 E3 A9 52 B3 15 AB FD 92 53 7 32 C0 62 48 F0 19 22 E0 91 62 1A C1 0. Počet čísel je 36. Podľa dokumentácie dĺžka periody na generovanie dát má byť 236. Je zložená z ochranného intervalu ktorý ma byť najmenej 200 a dĺžky generovanej správy.

Overenie funkčnosti

Pri overení funkčnosti sme využili nástroj Logic Analyzer, ktorý ponúka MATLAB od verzie 2016b. Tento nástroj umožňuje sledovať priebeh vstupných aj výstupných signálov a tým overiť správnu funkčnosť celého systému. Na obrázku zobrazený priebeh simulácie .  


Obrázok 5.2 Simulácia RS kódera v Logic Analyzéry

* DataIn – parameter reprezentuje vstupný tok dát do RS Encodera. Do kóder prichádzajú postupne v časovom slede všetky bajty správy.
* PacketStart - signál určuje začiatok vstupu správy.
* PacketEnd – označuje kedy vstupuje do kódera posledný bajt správy
* DataValid – musí byť na stavený na signalovú 1 počas celej dĺžky vstupu dát to kódera.
* PacketOut – je výstupná správa ktorá na konci obsahuje už vypočítanú paritu, ktorá je pripojená na konci pôvodnej správy.
* StartPacketOut – je výstupný signál, ktorý indikuje začiatok výstupu zakódovaného paketu z kódera.
* EndPacketOut- je výstupný signál, ktorý indikuje koniec výstup zakódovaného paketu z kódera. Teda označuje úplne posledný bajt ktorý patrí do správy.
* ValidPacketOut – výstupný signál, ktorý je nastavený sa signálovú 1 počas celej doby výstupu dát z kódera.
* Psusedo random Noise – sú náhodne generované bajty, ktoré simulujú šum v prenosovom kanály. Následne vykonaný logický XOR na presne definovaných miestach a teda spravím kontrolované zašumenie správy. Takto poškodená je následne poslaná do RS Dekódera na ktorom sa overuje správna funkčnosť systému.
* Switch - určuje presné miesta, kde má nastať k zašumeniu správy.
* Packet-Errors – je finálna správa odoslaná do dekódera vrátane cieleného šumu.

Simulácia HDL kódu vo Vivade

Využil som možnosť vygenerovať HDL kód a súbory TestBench pomocou modulu HDL Coder, ktorý obsahuje MATLAB a Simulink. Najprv je však potrebné aby bola simulácia v poriadku a minimálne overená správnosť signálov v Logic Analyzéry. Následne existujú dve možnosti ako vytvoriť HDL kód. Prvá z možností je v zadanie príkazov v konzolovom riadku Matlabu. Potrebné je zadať nasledovné príkazy:

checkhdl('RSHDLZaloha/RSDecoder','TargetDirectory','workingdir');

makehdl('RSHDLZaloha/RSDecoder','TargetDirectory','workingdir');

makehdltb ('RSHDLZaloha/RSDecoder','TargetDirectory','workingdir');

Príkaz:

* checkhdl – skontroluje kompatibilitu modelu so syntaxou HDL jazyka,
* makehdl – vytvorí súbory z príponou vhdl, ktoré obsahujú všetky potrebné prvky systému vrátane konfiguračných riadkov pre použitie v iných simulátoroch
* makehdltb - vytvorí súbory TestBench, ktoré sú vhodné na simuláciu v programe Vivado.

Parametre príkazov:

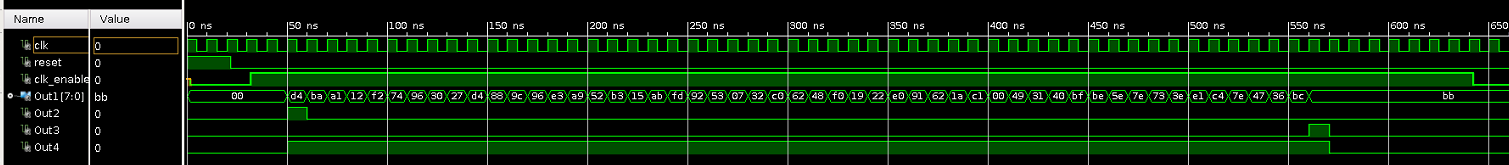
* RSHDLZaloha/RSDecoder – je názov projektu a konkrétnej oblasti ktorú chceme skontrolovať/vygenerovať. Pričom RSDecoder je názov konkrétnej časti. Túto časť nazýva Matlab ako subsystém.
* TargetDirectory – je parameter za ktorým nasleduje cieľový priečinok kam sa majú vygenerované súbory uložiť.
* workingdir – je nastavenie ukladacieho priestoru. V tomto prípade ide o priečinok v ktorom je aktuálne otvorený Matlab.

Následne som pripojil vygenerované súbory do nového projektu cez záložky: *Add Source -> Add or create simulation source -> Add Directories*

Do zoznamu som pridal všetky súbory, ktoré vygeneroval Matlab. Potom som spustil simuláciu pomocou záložky simulácia:

*Simulation -> Run Simulation -> Run Behavioral Simulation.*

Na obrázka je možné vidieť priebeh simulácie a konkrétne dáta, ktoré vychádzajú na porte *Out1* teda výstupnom porte z RS kódera.



Obrázok 5.3 Simulácia RS kódera vo Vivade

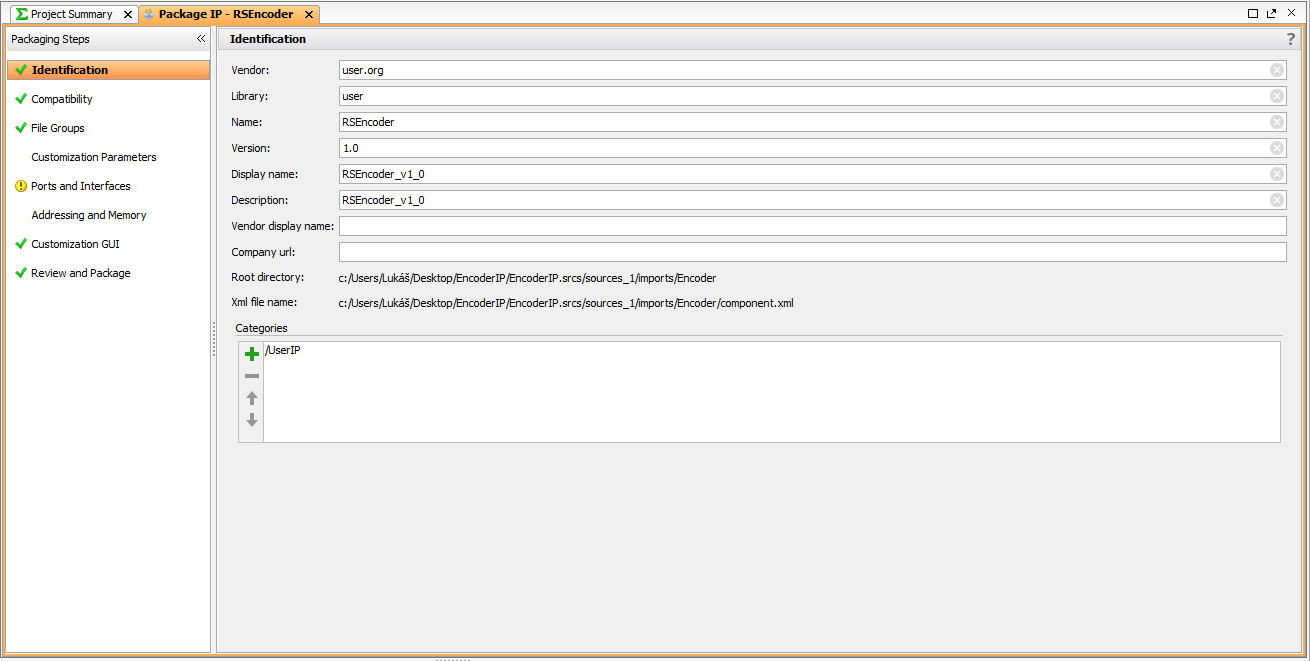
Zobrazené signály ukazujú na správny priebeh simulácie a teda môžeme predpokladať že vygenerovaný HDL kód bude fungovať aj po vytvorení IP modulu, ktorý bude mať rovnaké vstupné a výstupné porty ako sa nachádzajú v tejto simulácii. Pri nábežnej hrane signálu *clk\_enable* začína RS kóder spracovávať prijaté dáta a z oneskorením približne 20ns sa začínajú zobrazovať na výstupnom rozhraní z názvom *Out1.* Následne z portu vychádzajú postupne prijaté dáta. Pre lepšiu prehľadnosť som vstupných dátach som na koniec nastavil ako poslednú hodnotu 0. Táto hodnota nie je pravidlo ale grafickom vyjadrení je lepšie vidieť, kde končí pôvodná správa (D4 BA A1 12 F2 74 96 30 27 D4 88 9C 96 E3 A9 52 B3 15 AB FD 92 53 7 32 C0 62 48 F0 19 22 E0 91 62 1A C1 0) a začína dopočítaná parita (49 31 40 BF BE 5E 7E 73 3E E0 C4 7E 47 36 BC BB). Výstupné signály sa zhodujú so simulácii v Matlabe aj vo Vivade.

Tvorba IP modulu

Ďalším krokom v mojej práci bolo vytvorenie IP modulu pre jednoduchšiu implementáciu RS kódera vo VHDL. Takto pripravený modul môže byť nie len vo viacerých verziách programu. Dokonca môže byť tento modul použitý v iných vývojových a simulačných prostriedkoch pre HDL jazyky. Mojou úlohou bolo vytvoriť, vyskúšať a overiť funkčnosť.

RSEncoder ver. 1.0

Ako kostru pre vytvorenie tohto modulu som zvolil vygenerovaný RS kóder, ktoré ho zdrojový kód som už odskúšal vo Vivade pomocou TestBench modulov. Tento spôsob tvory som si zvolil preto aby som ušetril značné množstvo času. Pretože vytvorenie zdrojového kódu pre takýto modul je veľmi náročné. Pomocou sprievodcu na vytváranie nových IP modulov som si pripojil zdrojové kódy vygenerovaného Enkódera. Sprievodcu na vytvorenie IP modulov nájdeme: Tools *->* Create and Package IP. Po úspešnom prejdení sprievodcu sa nám zobrazí ponuka ako je na obrázku



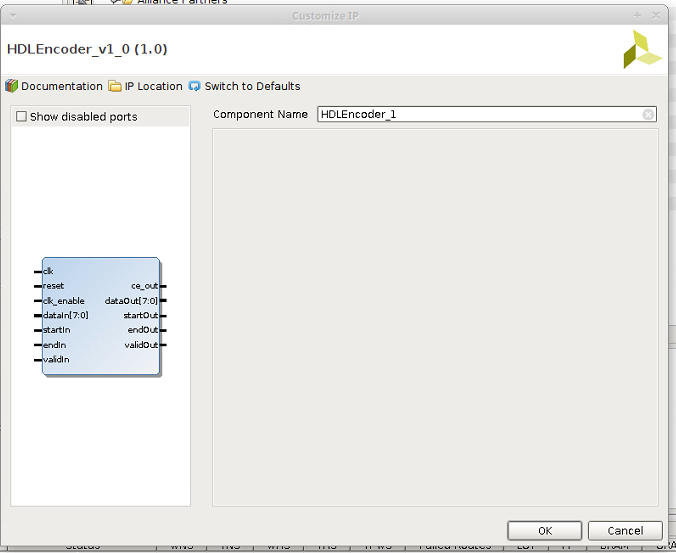
Obrázok 5.4 Tvorba IP modulu

Na tejto záložke je možnosť upraviť náš RSEncoder. Potrebné sú najmä úpravy vstupných a výstupných rozhraní. V nasledujúcej tabuľke tieto rozhrania popisujem a uvádzam všetky potrebné údaje potrebné pre správnu funkčnosť.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Interface Mode** | **Direction** | **Size** | **Type Name** |
| clk | slave |  |  |  |
| reset | slave |  |  |  |
| clk\_enable |  | in |  | std\_logic |
| dataIn |  | in | [7:0] | std\_logic\_vector |
| startIn |  | in |  | std\_logic |
| validIn |  | in |  | std\_logic |
| ce\_out |  | out |  | std\_logic |
| dataOut |  | out | [7:0] | std\_logic\_vector |
| startOut |  | out |  | std\_logic |
| endOut |  | out |  | std\_logic |
| validOut |  | out |  | std\_logic |

Tabuľka 5.2 Rozhrania IP modulu

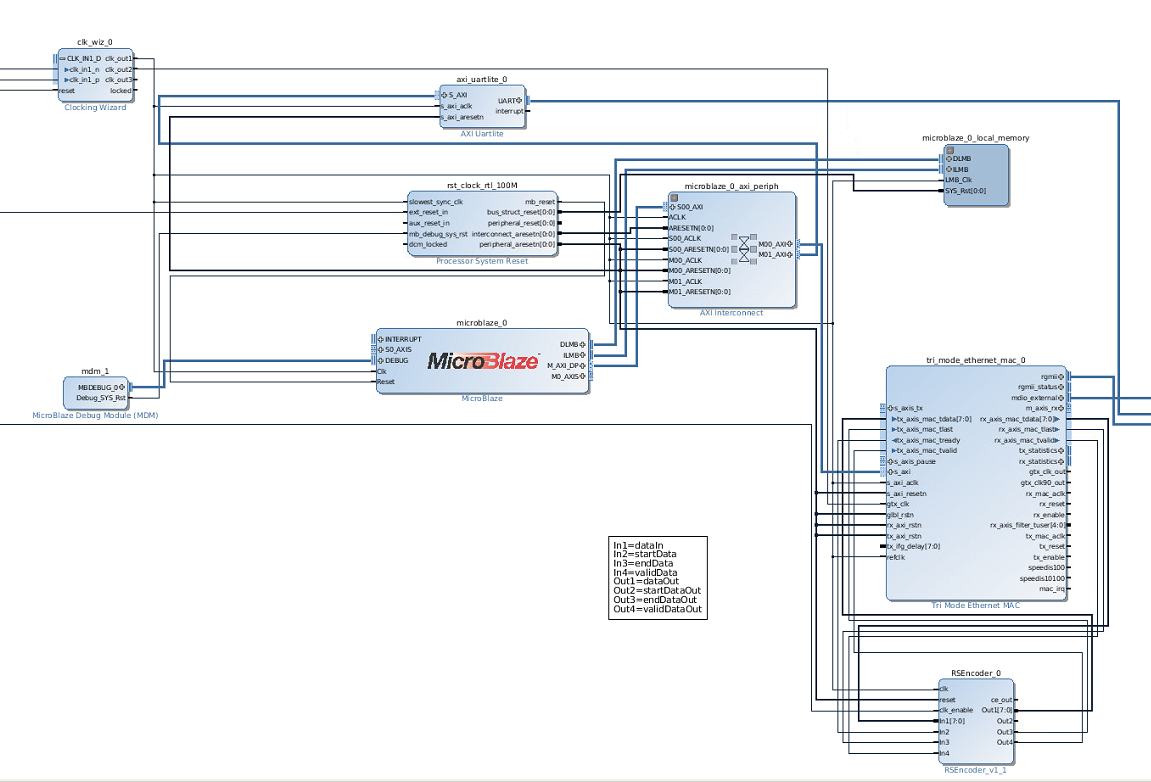
Výsledný IP modul je zobrazený ma obrázku. Ďalej tento modul používam pri vytvorenia schémy z IP modulov, ktoré predstavujú základnú časť implementácie. Tento spôsob vývoja pomáha odhaliť nedostatky hneď na začiatku vývoja.



Obrázok 5.5 Výsledný IP modul HDLEncoder

Návrh dizajnu IP modulov na FPGA kartu

Na obrázku je zobrazená schéma zapojenia funkčného RS Enkódera na rozhrania pomocou IP modulov. Všetky tieto rozhrania sú voľne dostupné od spoločnosti Xilinx. Pomocou týchto modulov som vytvoril návrh implementácie RS kódera na NetFPGA kartu. Tieto moduly je však potrebné prepájať manuálne. V Ďalších podkapitolách stručne popíšem jednotlivé IP moduly.



Obrázok 5.6 Schéma implantácie na NetFPGA kartu

Clocking Wizard

Logické jadro Clocking Wizard verzie 5.3 je určené na vytváranie časovacích signálov pre všetky ostatné IP moduly. Ako vstup príma časovanie od oscilátora karty. Následne dokáže tento signál rôznymi spôsobmi modifikovať. Dokáže upravovať výstupnú frekvenciu, vytvoriť viac rôznych frekvencií a podobne. [3]

Processor System Reset

Logické jadro Processor System Reset verzie 5.0 sa využíva na resetovanie pre celý systém, vrátane procesora, interconnectu ale aj periférnych zariadení. Toto jadro môže využívať interné alebo externé podmienky pre resetovanie systémov. Ďalšou možnosťou je nastavenie automatického resetu po určitom počte cyklov v systéme. [4]

MicroBlaze

Logické jadro MicroBlaze verzie 11.4. Jedná sa o mikroprocesor, ktorý využíva 32-bitové registre pre všeobecné použitie. Architektúra mikroprocesora je založená na Big-Endian formáte (obrátený formát). Tento mikroprocesor je optimalizovaný pre implementáciu v Xilinx. Medzi základné výhody mikroprocesora patrí tridsaťdva 32-bitových registrov. Inštrukcie o dĺžke 32-bitov s tromi operandmi a dva režimy pre adresovanie. Ďalšou výhodou je použitie až 32-bitových adries pre zbernice. [5]

http://ecasp.ece.iit.edu/tutorials/microblaze\_tutorial.pdf

MicroBlaze Local Memory

Logické jadro MicroBlaze Local Memory verzie 3.0 rozširuje správu pamäte pre mikroprocesory. Pomocou tohto jadra sa ovládajú pamäte typu RAM a BRAM. Medzi výhody jadra, patrí efektívna jedna hlavná zbernica (nie je potrebný arbiter). Ďalšie výhody sú separátny zápis a čítanie z dátovej zbernice. [6]

MicroBlaze Debug Module (MDM)

Logické jadro MicroBlaze Debug Module verzie 3.2 podporuje JTAG softvér na debugovanie. Táto podpora umožňuje synchronizované ladenie a ovládanie MiroBlaze procesorov. Tak isto sa tento modul dá využiť pre riadenie UART a AXI4-Lite rozhrania. Možnosťou tohto systému je aj trasovanie programu z pripojených MicroBlaze jadier až po externé rozhrania. [7]

AXI Interconect

Logické jadro AXI Interconnect verzie 2.1. Tento modul umožňuje prepájanie modulov typu master a slave, ktoré sú k nemu pripojené. Tieto moduly sa môžu líšiť od seba pokiaľ ide o šírku dát, hodinový signál. Spôsob prepájania funguje pomocou krížového prepínača, požiadavky pre príslušné jadrá sú automaticky vyhodnocované a prepojené. V rámci prepojovania vykonáva potrebné konverzie. [8]

AXI Uartlite

Logické jadro AXI Uart Lite verzie 2.0. Toto jadro poskytuje konektivitu medzi UART rozhraním a AXI Interconnect. Modul zabezpečuje plne duplexný presun dát. Využíva FIFO fronty pre prijímacie a výstupné porty. Pre správnu funkciu je však potrebné nastaviť počet dátových bitov od 5 do 8 bitov na znak. Ďalšími možnostiami konfigurácie tohto modulu sú voliteľný paritný bit (párny, nepárny alebo žiadny) a nastaviteľná prenosová rýchlosť. [9]

Tri Mode Ethernet MAC

Logické jadro Tri Mode Ethernet Media Access Controller verzie 9.0 riadi ethernetové rozhrania na karte. Tento modul má širokú škálu nastavení rýchlosti 10/100/1000 Mb/s prípadne 2.5 Gb/s, ktoré podporujú half-duplex aj full-duplex mód. Modul má širokú škálu nastavenia typov zberníc (RGMII, GMII, MII) pre univerzálne použitie. V module sú vstavané možnosti filtrov pre rámce z nastaviteľnými tabuľkami a voliteľnými štatistikami. Podpora kontrolných rámcov, virutal LAN (VLAN) a extra veľkých rámcov (jumbo frame) je samozrejmosť. [10]

RSEncoder

Jedná sa o mnou vytvorené jadro určené na kódovanie správy v dátovom prenose. Podrobnejší popis ale aj nastavenia a postup na vytvorenie podobného modulu boli predstavené v predchádzajúcej kapitole.

Tvorba a generovanie VHDL kódu

Po správnej validácii dizajnu IP modulov je potrené vytvoriť Wrapper......

Záver

Zoznam použitej literatúry

[1] International Telecommunication Union. ITU-T G. 709, February 2001 .

[2] <http://www.roma1.infn.it/exp/cuore/pdfnew/ch02.pdf>

[3] <https://www.xilinx.com/support/documentation/ip_documentation/clk_wiz/v5_3/pg065-clk-wiz.pdf>

[4] <https://www.xilinx.com/support/documentation/ip_documentation/proc_sys_reset/v5_0/pg164-proc-sys-reset.pdf>

[5] <https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/mb_ref_guide.pdf>

[6] <https://www.xilinx.com/support/documentation/ip_documentation/lmb_v10/v3_0/pg113-lmb-v10.pdf>

[7] <https://www.xilinx.com/support/documentation/ip_documentation/mdm/v3_2/pg115-mdm.pdf>

[8] <https://www.xilinx.com/support/documentation/ip_documentation/axi_interconnect/v2_1/pg059-axi-interconnect.pdf>

[9] <https://www.xilinx.com/support/documentation/ip_documentation/axi_uartlite/v2_0/pg142-axi-uartlite.pdf>

[10] https://www.xilinx.com/support/documentation/ip\_documentation/tri\_mode\_ethernet\_mac/v9\_0/pg051-tri-mode-eth-mac.pdf

Prílohy