VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

HARDWAROVÁ IMPLEMENTACE CRC PRO VYSOKORYCHLOSTNÍ SÍŤE

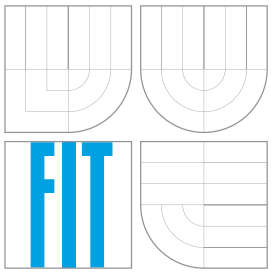
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAELA BELEŠOVÁ

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

HARDWAROVÁ IMPLEMENTACE CRC PRO VYSOKORYCHLOSTNÍ SÍŤE

HARDWARE IMPLEMENTATION OF CRC FOR HIGH SPEED NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAELA BELEŠOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ MATOUŠEK
SUPERVISOR

BRNO 2012

Abstrakt

Tato bakalářská práce řeší problém hardwarové implementace CRC ve vysokorychlostních sítích. V teoretické části popisuje základní informace o CRC a různé metody na jeho výpočet. Z popisovaných metod jsou zvoleny metody využívající Galoisova pole a metody zapojení jednotek počítajících CRC sériově a paralelně. Na základě těchto metod je postavena implementace, která počítá se vstupním slovem šířky 256 b a dodržuje propustnost přes 40 Gb/s. Implementace také řeší problém nezarovnaných začátků a konců rámce a možnosti výskytu částí až dvou rámců uvnitř jednoho slova. Celé řešení je optimalizováno tak, aby zabíralo co nejmenší množství zdrojů.

Abstract

This bachelor's thesis deals with a problem of hardware implementation of CRC for high speed networks. In the first section, there is a description of basic information about CRC and several methods for CRC calculation. From methods described in the first section methods representing Galois fields and methods describing connection of units computing CRC in parallel and in series were chosen. Based on these methods an implementation was created. This implementation expects 256 b input words and achieves throughput over 40 Gbps. This implementation also deals with a problem of unaligned starts and ends of frames and possible occurrence of pieces from two different frames in the same word. The whole solution was designed with minimal resource usage in mind.

Klíčová slova

CRC, FPGA, Galoisova pole, Ethernetový rámec, Tabulková metoda

Keywords

CRC, FPGA, Galois fields, Ethernet frame, table-driven implementation

Citace

Michaela Belešová: Hardwarová implementace CRC pro
vysokorychlostní síťe, bakalářská práce, Brno, FIT VUT v Brně, 2012

Hardwarová implementace CRC pro vysokorychlostní síťe

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod odborným vedením pana Ing. Jiřího Matouška. Informace mi také poskytl Bc. Lukáš Kekely ze sdružení CESNET z. s. p. o.

.....
Michaela Belešová
15. mája 2012

Poděkování

Chtěla bych poděkovat hlavně vedoucímu práce Ing. Jiřímu Matouškovi za jeho odbornou pomoc a rady při psaní této práce. Chtěla bych také poděkovat Bc. Lukášovi Kekelymu za cenné rady.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Počítačové siete sú v dnešnom svete už takmer nevyhnutnosťou a našli si uplatnenie v skoro každom obore ľudskej činnosti. Využívame ich na šírenie informácií, zdieľanie dát, komunikáciu, elektronické bankovníctvo a mnohé iné. Stretneme sa s nimi v práci, pri štúdiu i vo voľnom čase. Medzi kľúčové vlastnosti počítačových sietí, ktoré napomáhajú ich prudkému rozmachu, patrí rýchlosť šírenia informácií, dostupnosť a možnosť posielať a získavať veľa foriem údajov. Neustály nárast ich dôležitosti pre spoločnosť vyžaduje aby sa zlepšovali a prispôsobovali rozličným požiadavkám. Nároky sú kladené najmä na rýchlosť a spoľahlivosť.

Počítačové siete sú zložené z rozličných zariadení a prenosových médií, ktoré majú rôznu spoľahlivosť a rýchlosť. Nad všetkými týmito súčasťami funguje mnoho protokolov a služieb. Počítačové siete bývajú veľmi komplexné, neustále sa do nich pripojujú nové zariadenia a komunikácia, ktorá tu tečie, sa zrýchľuje a zväčšuje objem. Príkladom siete, na ktorú sú kladené vysoké nároky je Internet. Internet je v niektorých literatúrach označovaný ako sieť sietí [1], to znamená, že ho tvoria poprepájane podsiete. Internet je verejná sieť rozkladajúca sa po celom svete. Nájdeme v nej milióny rôznych zariadení pospájaných rôznymi médiami, prenášajúcimi veľa druhov informácií a na jeho jednotlivé časti sú kladené inakšie rýchlostné nároky. Na prenos medzi koncovými zariadeniami vo vnútri malej podsiete môže stačiť aj 100Mb/s. Prepojenia medzi krajinami či kontinentmi podliehajú prísnejším kritériám do výšky jednotiek až desiatok Gb/s. Internet je zložitý a s posielaním dát cez neho sa spúšťa mnoho mechanizmov ktoré posielanie riadia. Tieto mechanizmy musia zvládať obrovské množstvá dát a vysoké rýchlosti, ktoré neustále narastajú a preto všetky procesy kontrolujúce prenos treba neustále a neodkladne vylepšovať. K takýmto procesom patrí aj kontrola, či nedošlo k poškodeniu prenášaných dát. Pre niektoré druhy služieb, ako napríklad služby v bankovníctve, prenášanie súborov či pre posielanie e-mailov, je neprípustné, aby prijímali chybné dáta. Preto existuje mnoho postupov, ako opraviť, alebo aspoň zistiť chybu. Medzi jeden z najrozšírenejších postupov detekcie chýb patrí počítanie cyklického redundantného súčtu [1]. Rýchlemu výpočtu cyklického redundantného súčtu, známeho taktiež pod skratkou CRC, sa venuje aj táto bakalárska práca.

CRC je redundantná kontrolná informácia počítaná pri prenose údajov v počítačových sieťach na strane odosielateľa i príjemcu. Výsledok CRC odosielateľa je odosielaný spolu s dátami príjemcovi, ktorý si vypočíta vlastné CRC a porovná s CRC odosielateľa [1]. Počítanie CRC je možné realizovať softvérovo i hardvérovo. Softvérové riešenia sú väčšinou univerzálnejšie, teda fungujú na rôznych architektúrach. Oproti tomu hlavnou výhodou implementácií pre hardvér je najmä rýchlosť výpočtu. Táto rýchlosť je dosiahnutá optimalizáciou hardvérovej architektúry pre konkrétny účel. Hardvérová implementácia taktiež

prináša urýchlenie vďaka paralelizmu a zreťazeniu. Vďaka svojej hlavnej výhode, hardvérové riešenia nachádzajú v sieťach široké uplatnenie a sú aj predmetom tejto bakalárskej práce.

Táto bakalárska práca sa teda venuje hardvérovej implementácií CRC pre vysokorýchlostné siete. Je v nej popísaný návrh, implementácia, testovanie a optimalizovanie hardvérovej jednotky pre FPGA, ktorá ponúka vysokú priepustnosť a nízky počet využitých zdrojov. Navrhnuté riešenie dodržuje pravidlá Ethernetu [2]. V kapitole č. 2 je popísaná základná teória ohľadne CRC, ktorá zahrňuje dôvody, spôsob používania a matematiku pre jeho výpočet. V 3. kapitole sú popísané rôzne metódy výpočtu CRC pre bežné počítače aj iné zariadenia. V kapitole č. 4 je popis vytvorenia základného riešenia. 5. kapitola sa zaoberá jednoduchým rozširovaním základného riešenia a meraním maximálnej možnej frekvencie a využitia zdrojov pre jednotlivé rozšírenia. Z nameraných hodnôt sú identifikované klady a nedostatky základného riešenia a jeho rozšírení. V 6. kapitole je vytvorené riešenie odstraňujúce identifikované nedostatky z predošlej kapitoly. Tomuto riešeniu je taktiež zmeraná maximálna možná frekvencia a počet využitých zdrojov. 7. kapitola, záver, opakuje výsledky dosiahnuté v tejto práci a popisuje jej možné pokračovanie.

1.1 Cieľ mojej práce

Cieľom mojej bakalárskej práce je implementovať hardvérovú jednotku, ktorá bude schopná počítať CRC vo vysokorýchlostných sieťach. Požadovaný implementačný jazyk je VHDL.

Prvým krokom je naštudovanie metód pre generovanie CRC. Spomedzi týchto metód si vyberiem jednu, ktorá bude naprogramovaná v FPGA, otestovaná, vyhodnotená a vylepšovaná. Vylepšovaním sa rozumie zvyšovanie výkonu rozširovaním vstupného slova. Približná požadovaná priepustnosť na výslednú jednotku je 10Gb/s až 40Gb/s.

Kapitola 2

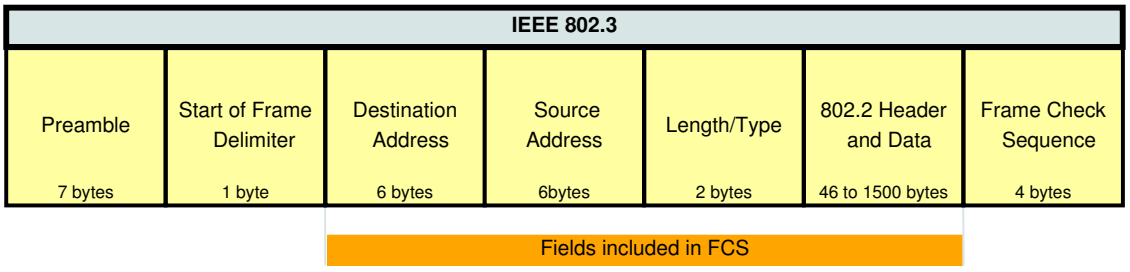
Teoretický úvod do problematiky

2.1 Čo sú to počítačové siete a CRC

CRC je skratka pre cyklický redundantný súčet odvodená z anglického názvu cyclic redundancy check. CRC označuje redundantné dáta používané pre detekciu chýb pri prenose a príjme dát z média alebo pri ukladaní dát.

Počítačové siete sú v dnešnom svete už takmer nevyhnutnosťou v takmer každom obore ľudskej činnosti. Stretneme sa s nimi v práci, na úradoch i pri oddychu. Čo sú to teda siete? Je to prepojenie koncových a sieťových zariadení a komunikácia medzi nimi založená na protokoloch. Medzi koncové zariadenia patria osobné počítače, IP telefóny, servery, tlačiarne a iné. Príkladom sieťového zariadenia sú smerovače a prepínače [1]. Prepojenie môže byť realizované rôznymi fyzickými médiami, ako napríklad optický či koaxiálny kábel, alebo dokonca aj vzduchom. Dôležité je to, že počas prenosu informácií môže dochádzať k chybám. Na siete je však často kladená požiadavka, aby prijímateľovi doručili dáta bez chyby, alebo túto chybu dokázali aspoň zistiť.

Médium pre prenos je teda obecne nespoľahlivé prostredie, v ktorom vznikajú chyby. Signály prenášané po médiu môžu byť rušené, skreslené, prípadne sa môžu úplne stratiť. Toto všetko má vplyv na zmenu posielaných dát reprezentovaných týmito signálmi [1]. Aby prijímateľ dát vedel rozhodnúť, či dáta boli prijaté správne, využívame rôzne metódy, medzi nimi aj CRC.



Obr. 2.1: Formát Ethernetového rámca [1]

Medzi najznámejšie využitie CRC v sieťach patrí počítanie FCS. FCS je skratka pre frame check sequence, ktorá reprezentuje políčko s hodnotou v Ethernetovom rámci. Mechanizmus zisťovania poškodenia dát pri ich posielaní po médiu a čítaní z neho pomocou CRC je následovný: odosielateľ vypočíta kontrolný súčet zasielaných dát a uloží do FCS, rámec je odoslaný vrátane CRC. Prijímateľ po prijatí rámca vypočíta svoj vlastný kontrolný

súčet a porovná [1]. Formát Ethernetového rámca je popísaný na obrázku 2.1. Z obrázku ide vidieť, ktoré políčka rámca sú započítavané do CRC a veľkosť jednotlivých políčok rámca v bajtoch.

Ethernet je v súčasnosti vo svete dominantná LAN technológia. Je zložený poskladaním niektorých protokolov linkovej a fyzickej vrstvy sieťového ISO/OSI modelu. Popisuje adresovanie, zabaľovanie dát do rámca a prijímanie a odosielanie dát na médium. Napriek tomu, že Ethernet podporuje rôzne druhy médií a šírky prenosových pásiem, základný formát rámca a adresná schéma ostáva tá istá. Pre adresovanie Ethernet využíva MAC adresy, ktoré jednoznačne identifikujú zariadenia v rámci siete. Na komunikáciu s vyššími vrstvami sieťového modelu využíva technológiu LLC. Celosvetový rozmach Ethernetu nastal vďaka jeho výrazným výhodám ako napríklad: jednoduchosť údržby, schopnosť prijať nové technológie, spoľahlivosť, nízka cena zavedenia a aktualizácií. Ethernet je pokrytý štandardami IEEE 802.3 [2]. Technológia gigabitový Ethernet je aplikovaná aj za hranice LAN do MAN a sietí založených na WAN [1].

Na záver tejto podkapitoly je dôležité zmieniť že vždy, aj vtedy, keď výpočet kontrolného súčtu a porovnanie prijímateľovi indikuje prijatie rovnakého rámca ako bol odoslaný, existuje malá pravdepodobnosť, že dáta došli v skutočnosti s chybou. Tento fakt je spôsobený schopnosťou chýb sa pri počítaní CRC navzájom vyrušiť. V takomto prípade detekcia chýb závisí na vyšších protokoloch [1].

V mojej bakalárskej práci sa budem zaoberať najmä CRC využívaným v Ethernete.

2.2 Teória výpočtu CRC

Ako som už spomínala v podkapitole 2.1, i v prípade správneho výsledku CRC vždy existuje malá pravdepodobnosť, že sa v dátach vyskytuje chyba. Úlohou algoritmov pre detekciu chyby je túto pravdepodobnosť minimalizovať. Preto kvalitné algoritmy musia spĺňovať aspoň tieto dva aspekty:

1. dostatočná šírka výsledku
2. chaos: predpis, ktorý dáva každému vstupnému bitu schopnosť meniť akékoľvek množstvo bitov výsledného súčtu.

Na splnenie druhého aspektu nestačia napríklad triviálne algoritmy založené na sčítavaní, naopak delenie, ktoré je využívané aj v CRC, je vhodné [3].

Nasledujúci odsek je prebratý z [4].

> Nech originálna správa $M = (m_0, m_1, \ldots m_{k-1})$ obsahuje k číslic a n reprezentuje dĺžku lineárneho blokového kódu $C = (c_0, c_1, \ldots c_{n-1}) \in C^*$. (n, k) lineárny kód C^* je cyklický, ak každý cyklický posuv kódového vektoru patriacemu C^* je opäť kódovým vektorom z C^*. Obecne, základné vlastnosti cyklických kódov sa dajú odvodiť manipulovaním s polynómami reprezentujúcimi M a C. Konkrétne, ak $M(x) = m_0 + m_1 x + \ldots + m_{k-1} x^{k-1}$ je polynóm správy asociovaný s M a $C(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$ je kódový vektor asociovaný C, potom sú vlastnosti cyklického kódu charakterizované príslušným generujúcim polynómom $G(x)$ stupňa $n-k$, teda $G(x) = 1 + g_1 x + \ldots + g_{n-k-1} x^{n-k-1} + x^{n-k}$. Z toho vyplýva, že každý polynóm $C(x)$ v (n, k) cyklickom kóde môže byť vyjadrený ako $C(x) = M(x)G(x)$. V praktických aplikáciách sa však používa systematická reprezentácia uvedeného vzťahu vo forme $C(x) = x^{n-k} M(x) + S(x)$, kde $S(x)$

je zvyšok po delení $x^{n-k}M(x)$ polynómom $G(x)$. Pri takomto prístupe, k číslic vpravo každého kódového vektoru prislúcha nezmenenej originálnej správe a $n-k$ číslic vľavo patrí paritným bitom. Takýto kód korešponduje kódovému vektoru $(s_0, s_1, \ldots, s_{n-k-1}, m_0, m_1, \ldots, m_{k-1})$, ktorý sa dá jednoducho realizovať pomocou deliaceho okruhu. Deliaci okruh je lineárny $(n-k)$-stupňový posuvný register so spätnými väzbami založený na $G(x)$. Je to najtriviálnejšia metóda pre výpočet CRC.

Táto metóda je bližšie popísaná v podkapitole 3.1.

Pri výpočte CRC sa teda využíva polynomiálna aritmetika. Na dáta, deliteľa a výsledok sa nepozerá ako na kladné celé číslo, ale ako na polynómy s binárnymi koeficientmi a to tak, že čísla si treba predstaviť ako reťazce bitov, ktorých bity sú koeficientmi polynómov [3]. Nad polynómami existuje mnoho rôznych aritmetík, pre účely binárnych dát, nad ktorými počítam CRC, využijeme „polynomiálnu aritmetiku mod 2“. To znamená, že koeficienty môžu byť len 0 alebo 1 a prenosy sa neuvažujú.

Bežné operácie v „polynomiálnej aritmetike mod 2“. Sčítavanie a odčítavanie čísel je rovnaké ako v obyčajnej binárnej aritmetike, akurát nepoužívame prenosy. Z uvedeného princípu vyplýva, že obe tieto operácie sú navzájom zhodné a ekvivalentné operácií XOR. V tabuľke 2.1 sú uvedené všetky kombinácie operandov a výsledky sčítavania a odčítavania v popisovanej aritmetike. Násobenie je taktiež veľmi podobné tomu, ako ho poznáme z bežnej binárnej aritmetiky. Jeho výsledkom je sčítanie posúvaného prvého čísla podľa druhého. Pri tomto sčítavaní je využitá „polynomiálna aritmetika mod 2“. To znamená, že ak je číslo A násobkom čísla B, potom je možné vytvoriť A z nuly a rôzne posunutého B aplikáciou operácie XOR. Násobeniu sa venuje aj tabuľka 2.2. Pre delenie je potrebné uviesť nasledujúci fakt platiaci v popisovanej aritmetike: X je väčšie alebo rovné Y, ak pozícia najvyššieho bitu s hodnotou 1 v X je rovnaká alebo väčšia než pozícia najvyššej 1 v Y. Porovnávanie čísel na inom, než prvom bite nemá žiadny vplyv na určenie, ktoré z týchto čísel je väčšie. Tento fakt je daný tým, že operácia sčítania a odčítania je rovnaká. V ostatnom, okrem využívania operácie odčítavania bez prenosov a diskutovaného porovnávania dvoch čísel, by sme opäť dve binárne čísla ručne delili rovnako, ako v aritmetike s prenosom [3].

+/−	0	1
0	0	1
1	1	0

Tabuľka 2.1: Sčítavanie a odčítavanie v „polynomiálnej aritmetike mod 2“

*	0	1
0	0	0
1	0	1

Tabuľka 2.2: Násobenie v „polynomiálnej aritmetike mod 2“

Detekcia chýb pomocou CRC v Ethernete teda funguje následovne: binárna správa, na ktorú sa pozeráme ako na polynóm $M(x) = m_0 + m_1 x + \ldots + m_{k-1}x^{k-1}$, je vynásobená odosielateľom $x_{n-k}M(x)$, pričom $n-k$ je stupeň generujúceho polynómu $G(x)$. Tento krok v skutočnosti urobí to, že správu posunie o počet bitov stupňa generujúceho polynómu a na

nové najnižšie bity vloží nuly. Nasledujúca úloha pri počítaní CRC u odosielateľa je vydeliť $x_{n-k}M(x)$ generujúcim polynómom $G(x)$. Získame zvyšok po delení $R(x)$. Odčítaním $R(x)$ od $x_{n-k}M(x)$ dostávame kódový polynóm $C(x)$ deliteľný $G(x)$. Vieme, že odčítavanie a pričítavanie sú v použitej aritmetike ekvivalentné operácie, posledných $n-k$ bitov $x_{n-k}M(x)$ je nulových a šírka zvyšku neprekoná šírku nulových bitov. To znamená, že odčítavanie $R(x)$ od $x_{n-k}M(x)$ môžeme jednoducho realizovať pripísaním $R(x)$ o šírke $n-k$ za $M(x)$. Novo vzniknuté kódové slovo $C(x)$ je násobkom $G(x)$. Úlohou prijímateľa správy je vydeliť prichádzajúcu správu $G(x)$ a overiť či je zvyšok po tomto delení nulový. V prípade nulového zvyšku vyhodnotíme prijatý Ethernetový rámec za správny, v opačnom prípade predpokladáme chybu [5].

Zhrnutie: Počítanie CRC znamená počítanie zvyšku po delení kontrolovaných dát fixným a známym číslom v „polynomiálnej aritmetike mod 2“.

2.3 Základné vlastnosti operácie XOR

Operácia XOR, často sa stretneme aj s názvom exkluzívny OR, patrí medzi binárne logické operácie. Pre počítanie CRC je veľmi dôležitá. V nasledujúcej tabuľke sú uvedené všetky možné kombinácie dvoch operandov a výsledky, ktoré vzniknú po aplikovaní operácie XOR na príslušnej dvojici.

XOR	0	1
0	0	1
1	1	0

Tabuľka 2.3: Pravdivostná tabuľka operácie XOR

XOR spĺňuje tieto matematické zákony [6]:

- asociatívny: $(A\ XOR\ B)\ XOR\ C = A\ XOR\ (B\ XOR\ C)$
- komutatívny: $A\ XOR\ B = B\ XOR\ A$

Ďalšie dôležité vlastnosti platiace pre XOR:

- $A\ XOR\ A = 0$
- $A\ XOR\ 0 = A$

2.4 Príklad výpočtu CRC

Príklad výpočtu CRC v tejto kapitole je prebratý z [3].

```
Originálna správa          : 1101011011
Generujúci polynóm         : 10011
W(stupeň polynómu)         : 4
Správa po prilepení W núl : 11010110110000

11010110110000 / 10011 =1100001010
10011
-----
 10011
 10011
 -----
  00001
  00000
  -----
   00010
   00000
   -----
    00101
    00000
    -----
     01011
     00000
     -----
      10110
      10011
      -----
       01010
       00000
       -----
        10100
        10011
        -----
         01110
         00000
         -----
          1110 = zvyšok = Kontrolný súčet
```

Prvým krokom delenia 11010110110000 číslom 10011 v „polynomiálnej aritmetike mod 2“, ktorú využíva aj CRC, je porovnanie, či je prvých 5 cifier z delenca väčších alebo rovných než deliteľ. V používanej aritmetike sa deje porovnanie čísel iba na základe porovnania prvej cifry. Obe z porovnávaných čísel majú na prvom mieste jednotku a tak sú vyhodnotené ako rovné. Prvou cifrou výsledku sa stáva jednotka. Od prvých 5 cifier delenca je odčítaný deliteľ a k výsledku je pripísaná 6. cifra z delenca. Nech je vzniknuté číslo označené d. Odčítavanie je v „polynomiálnej aritmetike mod 2“ zhodné s operáciou XOR. Číslo d je porovnané s deliteľom. Obe majú na prvom mieste jednotku a tak je do výsledku pripísaná

ďalšia jednotka. Od d je opäť odčítaný deliteľ a pripísaná 7. cifra delenca. Novovzniknuté číslo c ma na prvom mieste nulu a tak je menšie než deliteľ, ďalšou cifrou výsledku je 0. Operácia XOR je teraz vynechaná. K c je pridaná ďalšia cifra delenca. Postup sa opakuje až pokým nevznikne výsledok.

Vydelením vznikne podiel a zvyšok po delení. Podiel je pre CRC nepodstatný. Zvyšok je pripojený za pôvodnú správu namiesto núl a odoslaný.

2.5 Výber generujúceho polynómu

I keď pravidlá pre výpočet CRC sa nemenia, polynómy pre delenie dát sa môžu protokol od protokolu líšiť. Štruktúra a dĺžka polynómu majú veľký vplyv na schopnosť detekcie chýb a veľkosť redundancie. Použitie CRC v rôznych oblastiach má rôzne nároky na predošlé uvedené. V tabuľke 2.4 sú uvedené niektoré populárne polynómy.

CRC kód	**Generujúci polynóm**
CRC-CCITT (X25)	$x^{16} + x^{12} + x^5 + 1$
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$
CRC-32 (Ethernet)	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
GSM TCH/FS-HS-EFS (Kódovanie hlasovej komunikácie)	$x^3 + x + 1$
GSM TCH/EFS pre-coding	$x^8 + x^4 + x^3 + x^2 + 1$
GSM control channels – FIRE code	$x^{40} + x^{26} + x^{23} + x^{17} + x^3 + 1$

Tabuľka 2.4: Známe generujúce polynómy [7]

2.6 Hammingova vzdialenosť

So schopnosťou kódu objavovať chyby súvisí pojem Hammingova vzdialenosť. Hammingova vzdialenosť je počet znakov, v ktorých sa slova kódu líšia. Hammingova vzdialenosť kódu d je najmenšia z Hammingových vzdialeností všetkých kombinácií slov kódu. Kód odhaľuje t-násobné chyby, pokiaľ Hammingova vzdialenosť kódu $d > t$. Kód opravuje t-násobné chyby pokiaľ $d > 2t$ [8]. Jednotlivé kódy pre detekciu chýb využívajúce rôzne generujúce polynómy sa líšia v Hammingovej vzdialenosti a teda majú rôznu schopnosť detekovať chyby. CRC sú všeobecne veľmi populárne pretože majú lepšiu Hammingovu vzdialenosť než mnoho iných techník kontrolných súčtov [9].

Kapitola 3

Súčasne používané prístupy

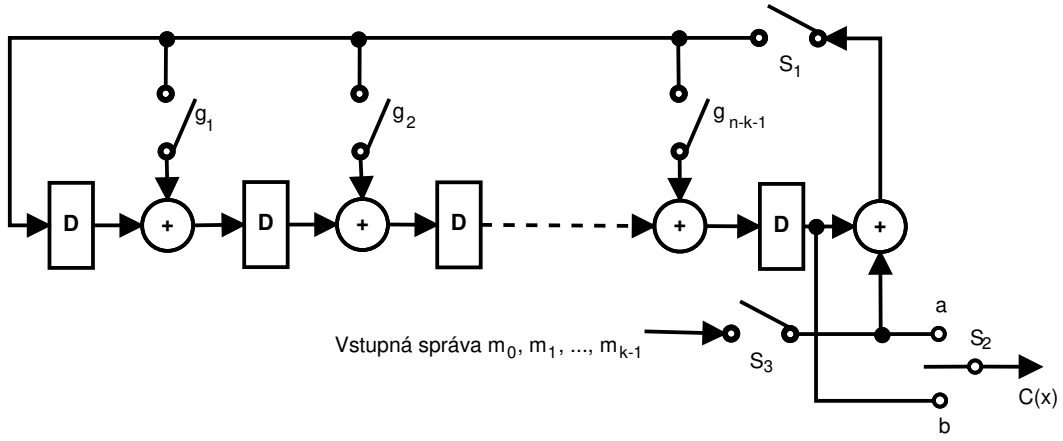
3.1 Priama metóda po bitoch

Priama metóda po bitoch sa pri počítaní kontrolného súčtu presne drží základnej matematiky definovanej nad CRC. V mojej práci je základná matematika popísaná v podkapitole 2.2. Táto metóda delí prichádzajúcu správu, ktorá do systému vstupuje po jednom bite, známym fixným polynómom v „polynomiálnej aritmetike mod 2“. Nie sú tu uplatnené žiadne ďalšie výrazné optimalizácie. To znamená, že tento spôsob je veľmi pomalý a paralelné spracovanie vylúčené. Princíp výpočtu je popísaný algoritmom 1 a jeho implementácia je znázornená na obrázku 3.1.

Algoritmus 1: Priama metóda po bitoch [3]

Inicializuj CRC register R(obvykle samé 0 alebo 1)
Rozšír originálnu správu o w 0, kde w je stupeň polynómu
while *existuje nespracovaný bit rozšírenej správy* **do**
    Posuň R doľava o 1 bit
    Do R vlož nasledujúci bit rozšírenej správy na pozíciu 0
    if *bit vysunutý von z R v predošlom kroku je 1* **then**
        $R = R\ XOR$ generujúci polynóm
    end
end
Register R teraz obsahuje hľadaný kontrolný súčet

Obrázok 3.1 koncepčne zachycuje kódovanie (n, k) cyklického kódu alebo CRC v deliacom okruhu. (n, k) cyklické kódy boli popísané v podkapitole 2.2. Pamäťové prvky (D klopné obvody) sú zapojené tak, aby spolu tvorili posuvný register. Znaky medzi nimi označujú modulo-2 sčítačky/odčítačky, to znamená, že vykonávajú operáciu XOR. Spätná väzba existuje pokiaľ príslušný koeficient g_i z $G(x)$, tzv. generujúci polynóm, je rovný 1. Posúvanie správy $M(x)$ do okruhu sprava je ekvivalentné násobeniu $M(x)$ krát x_{n-k}. Počas vsúvania $M(x)$ sú spínače S_1 a S_3 zavreté a S_2 pripojený k bodu a. Po tom, čo obvod spracuje celú správu, výsledný kontrolný súčet o veľkosti $n - k$ číslic je uložený v okruhu a následne vysunutý vonku otvorením spínačov S_1 a S_3 a prepnutím S_2 do bodu b. Ak D klopné obvody inicializujeme na iné hodnoty, než na 0, jediná zmena, ktorá sa so systémom stane je tá, že sa zmení zvyšok po delení, ktorý tento systém počíta [4].



Obr. 3.1: Hardvérové riešenie priamej metódy po bitoch [4]

3.2 Tabuľková metóda

Priama metóda po bitoch popisovaná v podkapitole 3.1 nie je vyhovujúca dnešným nárokom na rýchlosť výpočtu CRC. Pre zrýchlenie ľudia postupne vymýšľali nové algoritmy, ktoré dokážu v jednom okamihu spracovávať väčší objem dát než 1bit. Obľúbené boli násobky 8 bitov, teda 8 b, 16 b, 32 b, prípadne väčšie. Medzi metódy spracovávajúce väčšiu šírku dát v jednom okamihu patrí aj tabuľková.

Predstavme si, že počítame kontrolný súčet priamou metódou po bitoch [3]. Horných 8 bitov 32-bitového registra má hodnotu:

$$[t_7\ t_6\ t_5\ t_4\ t_3\ t_2\ t_1\ t_0]$$

A najvyšších 8 bitov generátoru polynómu je:

$$[g_7\ g_6\ g_5\ g_4\ g_3\ g_2\ g_1\ g_0]$$

V ďalšej iterácii priamej metódy t_7 rozhodne, či na register použijeme operáciu XOR s generátorom polynómu. Ak $t_7 = 1$, tak XOR prebehne, inak nie. Hodnota vrchných 8 bitov registra teda nadobudne hodnotu:

$$[t_6\ t_5\ t_4\ t_3\ t_2\ t_1\ t_0\ 0] + t_7 * [g_7\ g_6\ g_5\ g_4\ g_3\ g_2\ g_1\ g_0]$$

Výpočet prebehne v „polynomiálnej aritmetike mod 2“, kde platí, že sčítavanie je ekvivalentné operácii XOR a násobenie operácii AND. Nový najvyšší bit má teraz hodnotu $t_6 + t_7 * g_7$ a práve on kontroluje, čo sa stane v nasledujúcej iterácii. Dôležité je to, že všetky potrebné informácie na výpočet nového najvyššieho bitu boli prítomné vo vyšších bitoch pôvodného bitového registra.

Predstavme si, že používame 8 najvyšších bitov registra na výpočet hodnoty najvyššieho bitu registra počas ďalších 8 iterácii. Všimneme si 3 veci [3]:

1. Najvyšší bajt nás ďalej nezaujíma. Bude vysunutý z registra.
2. Zvyšné bajty budú posunuté a na začiatok sa vloží nový.
3. Existuje hodnota, z ktorej použitím XOR s pôvodným registrom vznikne rovnaký efekt, ako keby sme postupovali priamou metódou po 1 bite.

Nasledujúca časť podkapitoly sa venuje odvodeniu tabuľky tabuľkovej metódy pre generátor polynómu CRC-16, ktorý má tvar $x^{16} + x^{15} + x^2 + 1$. Odvodenie tabuľky je prebrané z [10]. Pomocou tejto tabuľky sa bude dať spracovávať 8 bitov v jednom časovom okamihu a bude odvodzovaná z priamej metódy po bitoch. Cieľom je teda dokázať vytvoriť tabuľku, pričom jedno jej použitie bude mať pre výpočet CRC hodnoty rovnaký efekt, ako 8 iterácií pomocou priamej metódy. Odvodzovanie prebieha pomocou zaznamenávania jednotlivých krokov priamej metódy a následným vyhodnotením posledného. Postup je napísaný v tabuľke 3.1 a notácia v nej použitá je:

- Bity sú číslované od 1 a bit 1 je najmenej významný bit
- Stĺpec označený SH určuje, koľký posuv je aktuálny
- Stĺpec IN označuje, aké dáta vchádzajú do systému, M_i je i-ty bit aktuálneho bajtu originálnej správy $M(x)$
- R_i predstavuje i-ty bit CRC registru
- C_i je i-ty bit inicializovaného registra predtým, než s ním boli prevedené akékoľvek posuvy či XOR vzhľadom na aktuálny bajt správy $M(x)$
- Záznamy v stĺpci R_i označujú položky, na ktoré musíme navzájom použiť XOR aby sme dostali novú hodnotu R_i

V tabuľke nie je uvedených všetkých 8 krokov z dôvodu, že ide iba o vysvetlenie princípu. Uvedené sú len kroky prvý, druhý, tretí a posledný.

Z tabuľky 3.1 môžeme pozorovať že obsah CRC registra po ôsmych posunutiach je funkcia obsahujúca operácie XOR na rôzne kombinácie dát vstupného bajtu a predošlého obsahu registra. Táto funkcia môže byť zjednodušená zvážením základných vlastností operácie XOR uvedených v podkapitole 2.3. Po aplikovaní týchto pravidiel môžeme zostaviť tabuľku 3.2.

Definovaním vektoru X, môžeme zjednodušiť zápis výpočtu CRC registra. Vektor X je zložený z 8 položiek (bitov) označených X_i, pričom:

$$X_i = M_i \; XOR \; C_i$$

Vektor X je výsledkom operácie XOR nižšieho bajtu CRC registra a vstupného bajtu. Tabuľku 3.3 dostaneme z tabuľky 3.2 použitím vektoru X.

Z tabuľky 3.4 môžeme pozorovať, že:

- Vyšší bajt CRC registra závisí iba na inicializačnej hodnote spodného bajtu CRC registra a vstupného bajtu správy $M(x)$
- Nižší bajt CRC registra je závislý od inicializačnej hodnoty spodného bajtu CRC registra, vstupného bajtu $M(x)$ a od inicializačnej hodnoty horného bajtu CRC registra

Tieto fakty vedú k záveru, že môžeme v CRC registri posunúť horný bajt do spodného, spodný zahodiť a pridať určitú 16 bitovú hodnotu pomocou XOR.

SH	**IN**	**CRC REGISTER**															
		R_{16}	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1
0		C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1
1	M_1	C_1	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2
		M_1		C_1													C_1
				M_1													M_1
2	M_2	C_2	C_1	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3
		C_1	M_1	C_2	C_1												C_2
		M_1		C_1	M_1												C_1
		M_2		M_1													M_1
				M_2													M_2
3	M_3	C_3	C_2	C_1	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4
		C_2	C_1	M_1	C_2	C_1											C_3
		C_1	M_1	C_3	C_1	M_1											C_2
		M_1	M_2	C_2	M_1												C_1
		M_2		C_1	M_2												M_1
		M_3		M_1													M_2
				M_2													M_3
				M_3													
								⋮									
8	M_8	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9
		C_7	C_6	C_5	C_4	C_3	C_2	C_1	M_1	C_2	C_1						C_8
		C_6	C_5	C_4	C_3	C_2	C_1	M_1	C_3	C_1	M_1						C_7
		C_5	C_4	C_3	C_2	C_1	M_1	M_2	C_2	M_1							C_6
		C_4	C_3	C_2	C_1	M_1	M_2	C_4	C_1	M_2							C_5
		C_3	C_2	C_1	M_1	M_2	M_3	C_3	M_1								C_4
		C_2	C_1	M_1	M_2	M_3	C_5	C_2	M_2								C_3
		C_1	M_1	M_2	M_3	M_4	C_4	C_1	M_3								C_2
		M_1	M_2	M_3	M_4	C_6	C_3	M_1									C_1
		M_2	M_3	M_4	M_5	C_5	C_2	M_2									M_1
		M_3	M_4	M_5	C_7	C_4	C_1	M_3									M_2
		M_4	M_5	M_6	C_6	C_3	M_1	M_4									M_3
		M_5	M_6	C_8	C_5	C_2	M_2	M_5									M_4
		M_6	M_7	C_7	C_4	C_1	M_3										M_5
		M_7		C_6	C_3	M_1	M_4										M_6
		M_8		C_5	C_2	M_2	M_5										M_7
				C_4	C_1	M_3											M_8
				C_3	M_1	M_4											
				C_2	M_2	M_5											
				C_1	M_3	M_6											
				M_1	M_4												
				M_2	M_5												
				M_3	M_6												
				M_4	M_7												
				M_5													
				M_6													
				M_7													
				M_8													

Tabuľka 3.1: CRC register po 8 posuvoch

SH	IN	CRC REGISTER															
		R_{16}	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1
8	M_8	C_8	C_7	C_8	C_7	C_6	C_5	C_4	C_3	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9
		M_8	M_7	M_8	M_7	M_6	M_5	M_4	M_3	C_2	C_1						C_8
		C_7	C_6	C_7	C_6	C_5	C_4	C_3	C_2	M_2	M_1						M_8
		M_7	M_6	M_7	M_6	M_5	M_4	M_3	M_2	C_1							C_7
		C_6	C_5							M_1							M_7
		M_6	M_5														C_6
		C_5	C_4														M_6
		M_5	M_4														C_5
		C_4	C_3														M_5
		M_4	M_3														C_4
		C_3	C_2														M_4
		M_3	M_2														C_3
		C_2	C_1														M_3
		M_2	M_1														C_2
		C_1															M_2
		M_1															C_1
																	M_1

Tabuľka 3.2: Zjednodušenie využitím vlastností operácie XOR

SH	IN	CRC REGISTER															
		R_{16}	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1
8	M_8	X_8	X_7	X_8	X_7	X_6	X_5	X_4	X_3	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9
		X_7	X_6	X_7	X_6	X_5	X_4	X_3	X_2	X_2	X_1						X_8
		X_6	X_5							X_1							X_7
		X_5	X_4														X_6
		X_4	X_3														X_5
		X_3	X_2														X_4
		X_2	X_1														X_3
		X_1															X_2
																	X_1

Tabuľka 3.3: Aplikácia X_i

SH	IN	CRC REGISTER															
		R_{16}	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1
8	M_8	0	0	0	0	0	0	0	0	C_{16}	C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9
		X_8	X_7	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	0	0	0	0	0	X_8
		X_7	X_6	X_7	X_6	X_5	X_4	X_3	X_2	X_1							X_7
		X_6	X_5							X_1							X_6
		X_5	X_4														X_5
		X_4	X_3														X_4
		X_3	X_2														X_3
		X_2	X_1														X_2
		X_1															X_1

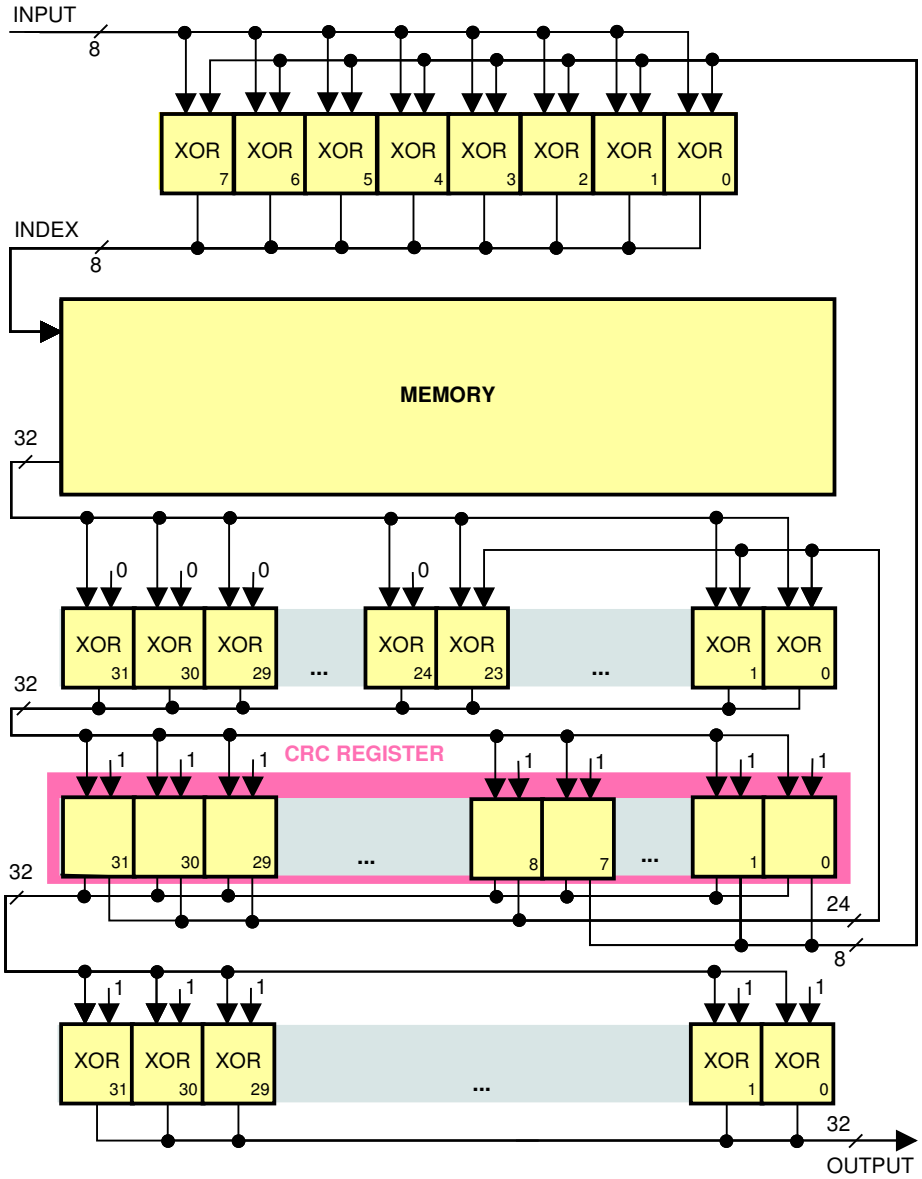
Tabuľka 3.4: Zvýraznenie závislostí nižšieho a vyššieho bajtu

V tomto prípade, kde načítame vstup po 8 bitoch a teda X je 8-bitový vektor, môže nadobudnúť iba 256 rôznych hodnôt. Výsledky pre všetky možné kombinácie vstupných dát dokážeme podľa tabuľky 3.4 vypočítať dopredu a uložiť do tabuľky v pamäti, pričom X bude slúžiť ako index. Dostávame tabuľku, ktorej vytvorenie bol účel tejto časti podkapitoly. Pseudokód použitia tabuľkovej metódy je potom uvedený v algoritme 2.

Algoritmus 2: Tabuľková metóda po bajtoch [10]

Inicializuj CRC register R(obvykle samé 0 alebo 1)
Vytvor a naplň tabuľku T
foreach *bajt b vstupnej správy* **do**
    Vypočítaj X ako XOR b s najnižším bajtom R
    Posuň R doprava o 8 bitov
    $R = R\ XOR\ T[X]$
end
Register R teraz obsahuje hľadaný kontrolný súčet

Na nasledujúcom obrázku 3.2 je schéma hardvérovej realizácie tabuľkovej metódy s čítaním vstupnej správy po 8 bitov. Schéma demonštruje počítanie CRC v Ethernete. To znamená, že dĺžka polynómu je 32 bitov, CRC register je na začiatku inicializovaný samými jednotkami a konečný CRC kontrolný súčet získame z vypočítaného súčtu použitím XOR s vektorom obsahujúcim 32 jednotiek. V pamäti je uložená dopredu pripravená tabuľka, do ktorej je ako index využívaný výsledok XOR aktuálneho bajtu vstupnej správy a najnižšieho bajtu CRC registru. Novú hodnotu CRC registru získame operáciou XOR medzi výstupom tabuľky a vrchnými tromi bajtami CRC registru posunutých o 8bitov doprava.



Obr. 3.2: Hardvérová realizácia tabuľkovej metódy

3.3 Metódy reprezentujúce Galoisové polia

Metódy založené na obvodoch reprezentujúcich Galoisové polia patria medzi paralelné, čiže urýchľujú výpočet CRC hodnoty spracovávaním viacerých bitov vstupnej správy súčasne. Obľúbené dátové šírky vstupujúce do systému naraz nazývame slová. Tieto metódy oproti tabuľkovým prinášajú vylepšenie. Tabuľkovú metódu využívame tak, že máme dopredu vypočítané hodnoty, ktoré využívame pre výpočet, uložené v pamäti. To znamená, že sme výrazne obmedzení veľkosťou pamäte. Takýmto spôsobom nedokážeme zrealizovať ani načítanie vstupnej správy po 32 bitoch, pretože by sme potrebovali 2^{32} záznamov. Problém nedostatočnej pamäte je ľahko riešiteľný. Vráťme sa k rovniciam vypočítaným v tabuľke 3.4. Rozdiel medzi tabuľkovou metódou a metódou založenou na obvodoch reprezentujúcich Galoisové polia spočíva v reprezentácii tých istých rovníc, ale inak. Tabuľková metóda si dopredu vypočíta výsledky pre všetky možné kombinácie vstupov a uloží do pamäte a počas výpočtu CRC k nim pristupuje. Druhá z metód nahradí tieto rovnice logickými členmi a výsledok si vždy znova vypočíta.

Pre vygenerovanie potrebných rovníc teraz uvediem aj iné riešenie než určovanie po krokoch z priamej metódy po bitoch. Nové riešenie je síce zložitejšie na pochopenie a požaduje znalosť Galoisových polí dvoch elementov a násobenie matíc, ale je obecnejšie. Nasledujúce riešenie je prebraté z [5].

Nech je vektor $X(t) = [x_0\ x_1\ \ldots x_{n-1}]$ zvyšok po t deleniach uložený v registri, $G = [g_0\ g_1\ \ldots g_{n-1}]$ je vektor generátoru polynómu, m_t aktuálny bit v správe a x_i je skratka pre $x_i(t)$. Potom pre zvyšok po delení $X(t)$ polynómom G v nasledujúcej iterácii $X(t+1)$ môžeme prehlásiť platnosť rovnosti:

$$X(t+1) = [x_0\ x_1\ \ldots x_{n-2}][0|I_{n-1}]\ XOR\ (x_{n-1}\ XOR\ m_t)G$$

Nech ma celková správa šírku m a slovo prichádzajúce každým taktom je šírky k. Potom na vypočítanie CRC stačí $\frac{m}{k}$ taktov. $M(t) = [m_t\ m_{t+1}\ \ldots m_{t+k-1}]$ reprezentuje vstupnú správu. Register obsahujúci $X(t)$ nadobudne po k vydeleniach polynómom G hodnotu $X(t+k)$, o ktorej platí:

$$X(t+k) = [x_0\ x_1\ \ldots x_{n-k-1}][0|I_{n-k}]\ XOR\ ([x_{n-k}\ x_{n-k-1}\ \ldots x_{n-1}]\ XOR\ M(t))D$$

Kde D je matica s predpisom:

$$D = \begin{bmatrix} G \\ GT^1 \\ \vdots \\ GT^{k-1} \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_0 & g_1 & g_2 & \ldots & g_{n-1} \end{bmatrix}$$

Realizácia obvodu, ktorý počíta s n bitovým generujúcim polynómom a šírkou vstupných slov k, bude následovná. Vypočítame si maticu D, ktorá bude mať k riadkov a n stĺpcov. Do počítacieho obvodu bude vchádzať i vychádzať n signálov. Výstupné signály tvoria hodnotu novo vypočítaného CRC. Potom $n-k$ vstupných signálov tvorí $[x_0\ x_1\ \ldots x_{n-k-1}]$ a k vstupných bitov je $[x_{n-k}\ x_{n-k-1}\ \ldots x_{n-1}]\ XOR\ Z(t)$. Nech y_i je i-ty bit výstupu. Pre y_i je dôležitý i-ty stĺpec tabuľky a v ňom riadky, v ktorých je 1. Ak je na b-tom riadku i-teho stĺpca 1, potom $y_i = y_i\ XOR\ x_{n-k-1+b}\ XOR\ z_{t+b}$. Ak $i \geq k$ počiatočná hodnota y pre každú iteráciu sa rovná x_{y-k}, inak 0, pretože x_{y-k} neexistuje. Vzhľadom na to, že $0\ XOR\ A = 0$, inicializáciu na 0 v obvode realizujeme jednoduchým nezapojením neexistujúceho člena x_{y-k}.

3.4 Rýchle algoritmy zvládajúce ľubovoľnú dĺžku správy

Tabuľková metóda i metódy založené na obvodoch reprezentujúcich Galoisové polia sú veľmi efektívne a v súčasnosti využívané. Pre požiadavky modernej doby na rýchlosť sietí je nevyhnutné, aby bolo spracovávaných v danom okamihu viac než 8 bitov vstupnej správy. Rozšírením vstupných slov vzniká problém s rôznou dĺžkou Ethernetových rámcov. Kritérium na veľkosť Ethernetového rámca je iba ohraničenie počtu bajtov zdola a zhora a pravidlo, že jeho veľkosť musí byť násobkom 8 bitov.

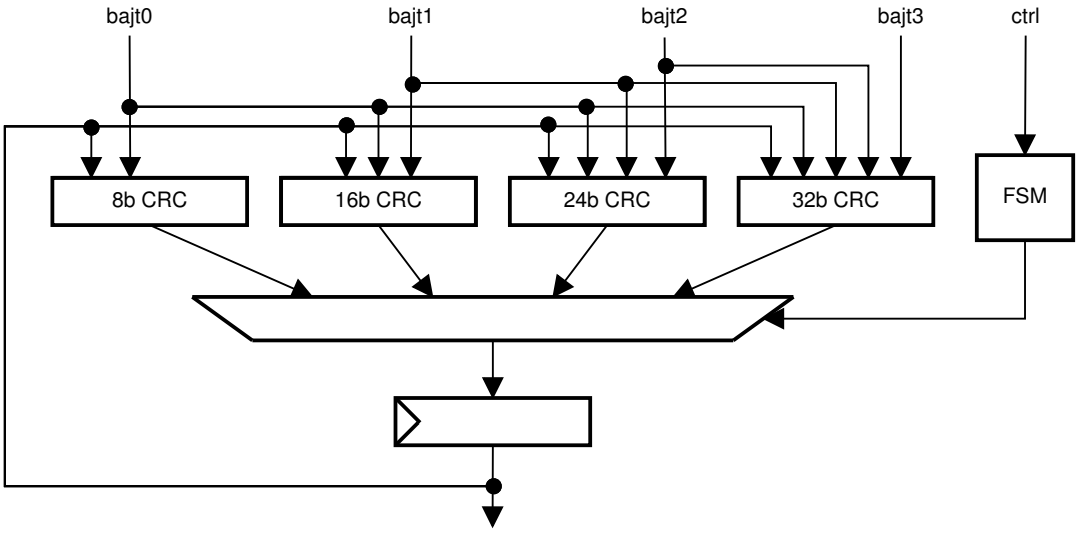
Nech je veľkosť políčok Ethernetového rámca, ktoré sa započítavajú do CRC, 70 bajtov. Štruktúra Ethernetového rámca a vyznačenie políčok, ktoré ovplyvňujú hodnotu CRC je na obrázku 2.1. Ďalej predpokladáme, že používame metódu založenú na obvodoch reprezentujúcich Galoisové polia, ktorá vyžaduje 4 bajty vstupnej správy naraz. Počas počítania kontrolného súčtu prvých 17 iterácií prebehne v poriadku, v poslednej nastane problém.

Ak chceme teda spracovávať viac než 8 vstupných bitov Ethernetového rámca naraz, musíme použiť niektorú z metód umožňujúcich spracovávať ľubovoľnú veľkosť v bajtoch počas poslednej iterácie. Takéto metódy obvykle využívajú algoritmy zavedené už dávnejšie a prinášajú do nich niečo nové. V mojej bakalárskej práci popíšem tri takéto metódy.

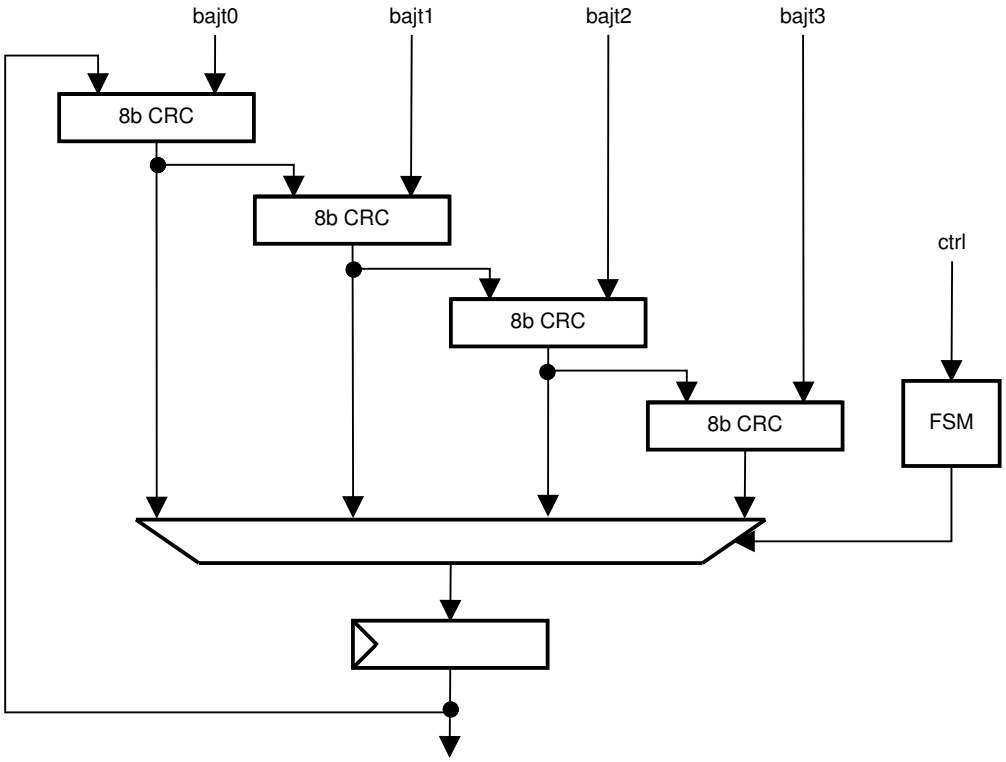
Zapojenie jednotiek paralelne a vyberanie správneho výsledku pomocou vstupných kontrolných signálov. Každá z paralelne zapojených jednotiek počíta CRC pre iný počet vstupných bajtov tak, aby boli pokryté všetky počty bajtov od 1 až po šírku vstupného slova. Výhodou tejto metódy je jej rýchlosť, nevýhodou veľký počet hardvérových súčiastok, ktoré sú potrebné na jej realizáciu. Na obrázku 3.3 je príklad zariadenia, ktoré dokáže spracovávať 32 bitov vstupnej správy naraz. Pri každej iterácii, okrem poslednej, sa ako nový výsledok registra vyberie výsledok jednotky s 32 bitovým vstupom. O výbere výsledku poslednej iterácie spomedzi 4 sa rozhodne pomocou riadiacich signálov. FSM je konečný automat kontrolujúci priebeh výpočtu [11].

Zapojenie jednotiek do série ako je možné vidieť v schéme na obrázku 3.4. Celý výpočet je opäť riadený pomocou konečného automatu FSM a riadiacich signálov. Princíp obvodu spočíva v tom, že ako nový výsledok CRC registra môže byť zvolený výsledok ľubovoľnej z jednotiek zapojených za sebou. Vo všetkých, okrem poslednej iterácie, to býva posledná jednotka. V poslednej iterácii je vybratý výsledok jednej z jednotiek podľa toho, koľko bajtov vstupnej správy ostalo. Novinkou je to, že každá ďalšia jednotka potrebuje výsledok predošlej. Obvod na obrázku 3.4 spracováva 32 bitov vstupnej správy naraz. Zapájanie do série je pomalšie než vyššie uvedené zapojenie paralelné, ale zaberá výrazne menej hardvérových prostriedkov. Spomalenie vyplýva z čakania na výsledok predošlej jednotky [11].

Doplnenie núl na začiatok rámca tak, aby jeho dĺžka bola násobkom dĺžky slova. Táto metóda vyžaduje, aby bola dopredu známa dĺžka rámca, čo je väčšinou realizované odchytením celého rámca do pamäte, zistením jeho veľkosti a až nakoniec spustením výpočtu CRC. Takéto riešenie však mierne spomaľuje proces posielania a vyžaduje pamäť navyše. CRC register musí obsahovať pri začiatku spracovávania pôvodných dát inicializačnú hodnotu, obvykle samé jednotky. S touto podmienkou sa dá vysporiadať pridaním ďalších častí do obvodu, ktoré dokážu inicializovať register tak, aby po spracovaní pridaných núl na začiatok obsahoval požadovanú inicializačnú hodnotu [12].



Obr. 3.3: Schéma zapojenia CRC jednotiek paralelne



Obr. 3.4: Schéma zapojenia niekoľko 8 bitových CRC jednotiek do série

Kapitola 4

Základné riešenie

4.1 Návrh základného riešenia

V rámci vypracovania bakalárskej práce som v kapitole 3 popísala viaceré metódy pre generovanie CRC hodnoty. Každá z uvedených metód má vymenované svoje výhody i nevýhody, na základe ktorých som sa rozhodla, ktoré metódy najlepšie spĺňajú ciele kladené na moju prácu. Cieľu tejto bakalárskej práce sa podrobne venuje kapitola 1.1.

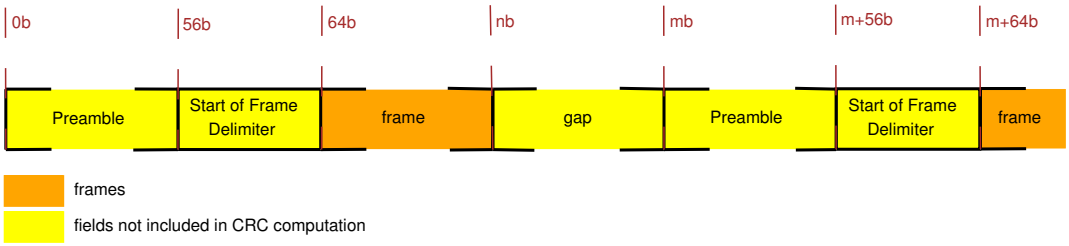
Najdôležitejším cieľom na vyvíjanú jednotku je priepustnosť. Ďalšie nie sú v zadaní špecifikované a preto som si ich vhodne určila ja. Pri určovaní požiadaviek som sa riadila praxou a preto som vychádzala z definície Ethernetu [2]. Predpokladám, že minimálna dĺžka vstupujúceho rámca je 480 b. Dĺžka rámca môže byť rôzna, ale vždy musí byť násobkom 8 bitov. Minimálna medzera medzi po sebe nasledujúcimi rámcami je 160 b, pričom je tvorená samotnou medzerou o najmenšej možnej dĺžke 96 b a ďalej pozostáva z Start of Frame Delimiter a Preamble. Pravidlá odvodené z definície Ethernetu som doplnila vlastnými. Predpokladám, že počas predspracovania sú vylúčené chybné rámce a preto sa na vstup mojej jednotky dostanú iba správne. Od jednotky, ktorá posiela dáta na vstup mojej, ďalej očakávam aj to, že pokiaľ bude veľkosť vstupného slova niekoľko bajtov, nemusí zarovnať začiatok a koniec rámca v rámci slova, ale prvý a posledný bajt môže umiestniť na pozíciu ľubovoľného bajtu v slove. Predspracujúca jednotka môže takto upraviť zarovnanie dát, napriek tomu, že štandard Ethernetu definuje zarovnanie začiatku rámca na hranicu 32 b. Posledným pravidlom, ktoré si zavediem, je, že vo vnútri slova sa po začiatku a pred ukončením rámca nemôže nachádzať kus dát, ktoré rámcu nepatria, s výnimkou neplatných dát o veľkosti násobku celého slova zarovnaných na začiatok slova. To znamená, že dáta rámca musia prichádzať kontinuálne za sebou a nesmie sa v nich vyskytovať medzera. Medzi platné slová rámca však môžu byť zaradené neplatné slová, ktoré budú ignorované.

Uvedené predpoklady z predošlého odseku sú znázornené na obrázkoch 4.1 a 4.2. Žlté políčka tvoria medzery a oranžové patria rámcom. Obrázok 4.1 ukazuje minimálne veľkosti po sebe nasledujúcich rámcov a minimálne veľkosti medzier medzi nimi. Číslo n a m môžu byť ľubovoľné prirodzené čísla, pre ktoré platia podmienky:

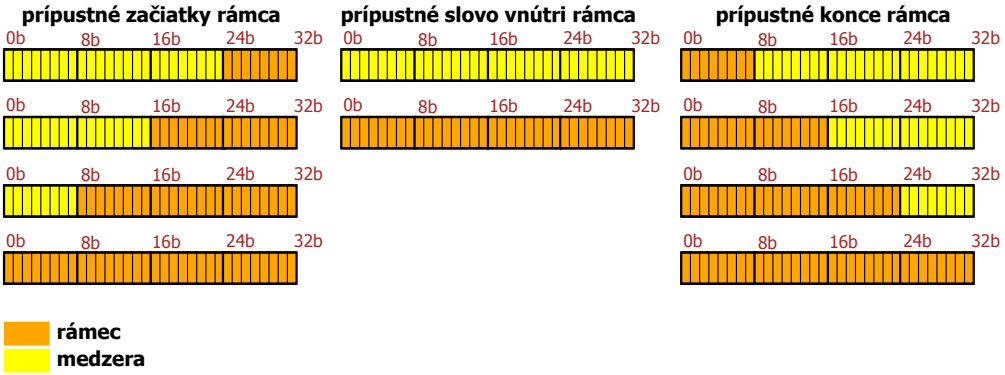
$$n \geq 544$$

$$m - n \geq 96$$

Na 4.2 je na konkrétnom príklade šírky slova 32 b ukázané, na aké pozície v slove môžu byť rámce zarovnané. 4.2 takisto ukazuje povolené neplatné slovo počas spracovávania rámca.



Obr. 4.1: Povolené veľkosti po sebe idúcich rámcov a medzier medzi nimi



Obr. 4.2: Povolené zarovnávanie rámcov vo vnútri 32 b slova

Po zvážení uvedených požiadaviek som si pre základné riešenie jednotky počítajúcej CRC v mojej bakalárskej práci zvolila metódu zapojenia jednotiek paralelne, popísanú v podkapitole 3.4 v kombinácií s metódami reprezentujúcimi Galoisové polia z podkapitoly 3.3. Dôvodom pre výber zapojenia jednotiek paralelne je jeho rýchlosť a zároveň schopnosť počítať CRC pre dáta rôznej dĺžky. Hlavnou nevýhodou tejto metódy, ktorú je potrebné mať na pamäti, je veľký počet hardvérových súčiastok potrebných pre jej realizáciu. Metódy reprezentujúce Galoisové polia som zvolila, pretože mi umožnia paralelné spracovanie viacerých vstupných bitov bez nutnosti využívať obrovské množstvo pamäte.

Do fázy návrhu jednotky patrí aj navrhnutie rozhrania. Jednotka bude ako produkt svojho výpočtu posielať na výstup 32 bitový signál obsahujúci vypočítané CRC a 1 bitový signál slúžiaci ako príznak jeho platnosti. Na vstupe bude jednotka očakávať dáta šírky 32 b a príznak, či majú byť zahrnuté do výpočtu. Ďalšími vstupujúcimi príznakmi budú indikátor prvého slova rámca a indikátor posledného. Okrem spomínaných bude vyvíjaná jednotka požadovať aj signál, ktorý by niesol informáciu o tom, ktoré z aktuálnych dát na vstupe náležia rámcu a teda je z nich počítané CRC. Poslednými vstupnými signálmi budú reset a hodinový signál.

4.2 Realizácia základného riešenia

Metóda zapojenia jednotiek paralelne z podkapitoly 3.4 síce dokáže vypočítať CRC pre vstup o rôznej dĺžke, ktorá je násobkom 8 bitov, ale predpokladá že dáta majú zarovnaný začiatok a len koniec môže byť nezarovnaný. Vytváraná jednotka však musí dokázať spracovať aj nezarovnaný začiatok rámca. Preto musí byť schéma 3.3 z podkapitoly 3.4 znázorňujúca využívanú metódu obohatená o nové prvky. Na výber sa ponúkajú dve možnosti ako vyriešiť problém nezarovnaných začiatkov:

- Prvou možnosťou je pri každom nezarovnanom začiatku urobiť spätnú kalkuláciu a nájsť počiatočnú hodnotu registra tak, aby po prepočítaní CRC dát v slove nepatriacich rámcu, bola hodnota CRC inicializačnou hodnotou [12].

- Druhou možnosťou je využitie jednotiek počítajúcich nezarovnané konce, pričom im na vstup budú privedené iné dáta, než keby sa počítal koniec. Takýto postup je jednoduchší a zaberie menej nových hardvérových súčiastok, ale môže byť využitý iba v prípade, že neexistuje veľa možností, na akej pozícií v slove sa nachádzajú správne vstupné dáta.

Pre svoju jednotku som vybrala druhú z predošlých uvedených možností. Rozhodla som sa na základe požiadaviek uvedených v minulej podkapitole a to konkrétne: minimálnej dĺžky rámca 480 b a existencie iba takých dier vo vnútri rámcov, ktoré zaberajú celé slová. S platnosťou požiadaviek v predošlej vete platí, že pokiaľ šírka slova neprekročí minimálnu dĺžku rámca, nezarovnaný začiatok rámca musí byť vždy umiestnený na ten istý okraj slova. Tento fakt vidno aj na obrázku 4.2, kde všetky prípustné začiatky rámca zaberajú posledný – 31. bit. To znamená, že existuje jediné miesto v slove, kde pre určitú dĺžku začiatku je tento začiatok umiestnený, a to je na konci slova. Podmienka pre použiteľnosť druhej možnosti riešenia problému nezarovnaných začiatkov je splnená. Využila som teda druhú možnosť vzhľadom na jej výhody. Všetky paralelné jednotky počítajúce CRC pre dáta s menšou šírkou, než šírka slova, majú celkovo dve možnosti vstupu, buď sú dáta patriace rámcu umiestnené na jeden alebo druhý okraj slova v závislosti na tom, či sa jedná o začiatok alebo koniec rámca. Správny vstup bude vyberaný jednoducho – pomocou dvojvstupových multiplexorov. Predošlé tvrdenia neplatia pre jedinú z paralelne zapojených jednotiek a to tú, ktorá predpokladá, že všetky bajty vstupného slova sú platné. Táto jednotka multiplexor nepotrebuje, pretože na jej vstup je vždy privádzané celé vstupné slovo a žiadna iná možnosť neexistuje. Zavedené multiplexory sa budú vyskytovať v nasledujúcich schémach s označením MX s číslom, ktoré je násobkom 8, podľa toho, akej šírky v bitoch sú jeho dátové vstupy a výstup.

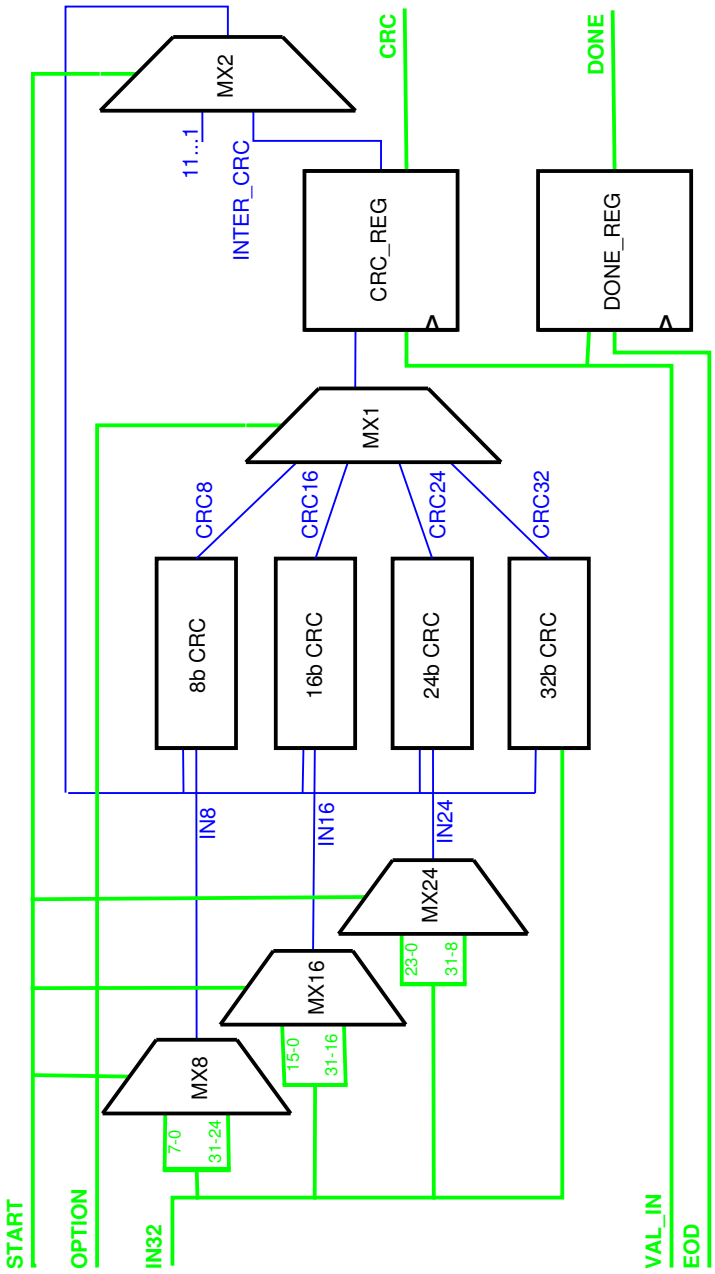
Okrem uvedených multiplexorov je schému 3.3 z 3.4 pre jej použiteľnosť v rámci tejto bakalárskej práce potreba rozšíriť o ďalšie jednotky. Medzi jednu z nich patrí aj register uchovávajúci stav výpočtu CRC. Výstupný signál z tohto registru je vyvedený na výstup celej jednotky a indikuje, či je aktuálna vypočítaná hodnota CRC platná. Túto jednotku sa bude volať DONE_REG.

Posledným pridaným komponentom obvodu je multiplexor, ktorý vyberá, čo príde na vstup do paralelne zapojených jednotiek počítajúcich CRC. Hodnota, ktorú vyberie, reprezentuje aktuálnu hodnotu CRC registra. Obvykle je vybraná hodnota posledného vypočítaného CRC, pokiaľ sa však na vstupe obvodu očakáva prvé slovo rámca, vybraná hodnota je rovná inicializačnej hodnote. Tento multiplexor budem označovať MX2.

Pridaním vymenovaných prvkov do schémy na obrázku 3.3 získavam nový obvod zobrazený na 4.3. Tento obrázok znázorňuje schému zapojenia základného riešenia mojej bakalárskej práce, ktoré bude vylepšované v nasledujúcich kapitolách.

Za pomoci metód reprezentujúcich Galoisové polia som vytvorila vnútro paralelne zapojených jednotiek. Na obrázku 4.3 sú spomínané jednotky pomenované 8 b CRC, 16 b CRC, 24 b CRC a 32 b CRC. Každá z jednotiek obsahuje rovnice počítajúce CRC pre príslušnú šírku vstupu. Pre odvodenie požadovaných rovníc, bol vytvorený program v jazyku C. Tento program pri nachádzaní požadovaných rovníc postupuje rovnako ako je popísané v podkapitole 3.3. Postupne vypočítava matice T a D, z ktorých sa dajú rovnice zistiť. Program dokáže vygenerovať rovnice pre ľubovoľnú šírku vstupného slova menšiu než 65536 b. Šírka

slova pre výpočet sa zmení pomocou prepísania $\#define$ na začiatku zdrojového súboru. Výstupom programu sú rovnice zapísané syntaxou jazyka VHDL, aby mohli byť ľahko vložené do jednotky.



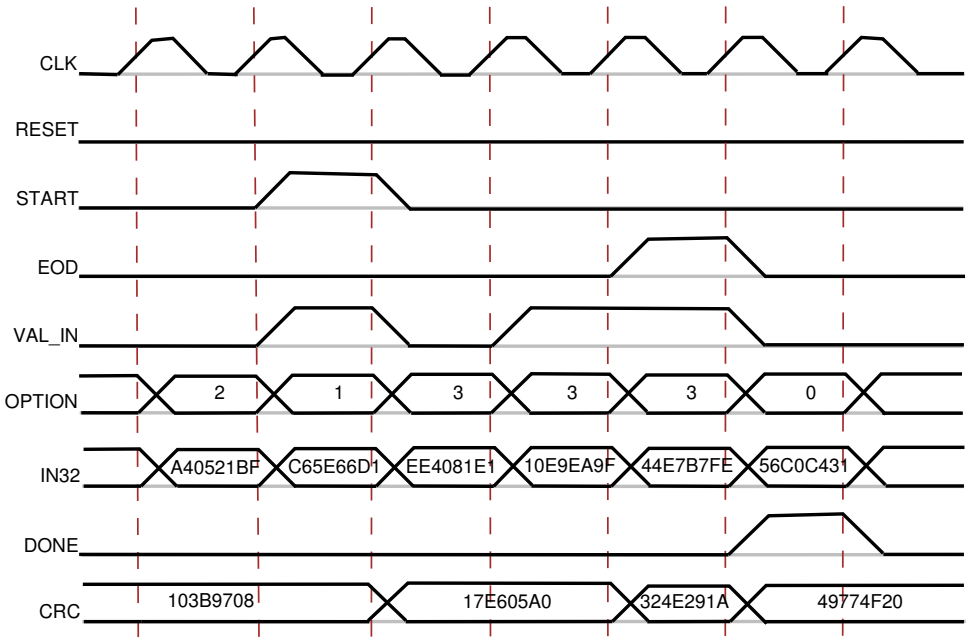
Obr. 4.3: Schéma zapojenia základného riešenia

Signály rozhrania na obrázku 4.3 sú zelené, vnútorné signály sú vyznačené modrými čiarami. Rozhranie základného riešenia je podrobnejšie popísané v tabuľke 4.1. Na obrázku 4.4 je príklad hodnôt signálov rozhrania počas niekoľkých taktov hodinového signálu zakreslený pomocou časového diagramu. Z tohto obrázku je vidieť, že pokiaľ signál VAL_IN nie je v logickej jednotke, ostatné signály (okrem RESET) nemajú žiadny efekt, hodnota CRC sa nemení. Signál DONE sa nastaví do logickej jednotky na jeden takt, takt po tom, ako

boli súčasne nastavené príznaky VAL_IN a EOD a nenastavený RESET. Výsledná hodnota CRC je na výstupe k dispozícií nasledujúci takt po prijatí vstupov, z ktorých bola počítaná.

Signál	**Smer**	**Popis**
CLK	vstup	hodinový signál
CRC	výstup	výsledná hodnota CRC
DONE	výstup	príznak určujúci, či signál CRC obsahuje platnú hodnotu
EOD	vstup	príznak indikujúci posledné slovo aktuálneho rámca
IN32	vstup	obsahuje slovo vstupného rámca, pre ktorý bude vypočítané CRC, šírka vstupného slova je 32 b
OPTION	vstup	určuje počet bajtov vstupného slova patriacich rámcu; OPTION=0 znamená 1 bajt, OPTION=1 znamená 2 bajty...
RESET	vstup	reset
START	vstup	príznak indikujúci prvé slovo aktuálneho rámca
VAL_IN	vstup	príznak určujúci platnosť aktuálnych vstupných dát

Tabuľka 4.1: Rozhranie základného riešenia



Obr. 4.4: Časový diagram rozhrania základného riešenia

Základné riešenie obvodu počítajúceho CRC, spracováva 32 bitov vstupnej správy v každom takte a dokáže si poradiť s nezarovnanými začiatkami a koncami rámca. Takéto riešenie nie je dostatočne rýchle, ale je dobrým základom pre neskoršiu optimalizáciu, ktorej sa budem venovať v nasledujúcich kapitolách.

4.3 Testovanie základného riešenia

Po fáze implementácie nasledovalo testovanie. Správnu funkčnosť vytvoreného dizajnu som najskôr overila softvérovou simuláciou v prostredí ModelSim a následne i v hardvéri na platforme FITkit.

Pre poriadne odsimulovanie bolo potrebné jednotku otestovať na veľkom množstve náhodných vstupných dát. Z dôvodu veľkého počtu testov bolo nutné vytvoriť si čo najviac automatizované testovacie prostredie.

Testovacie prostredie v ModelSime, vyvinuté pre účely tejto bakalárskej práce, načíta vstupné dáta zo súboru. Tieto dáta sú zaslané testovanej jednotke, ktorá prevedie potrebný výpočet. Testovacie prostredie okrem zaisťovania vstupu taktiež odchytáva výstup testovaného zariadenia a porovnáva s referenčným výstupným súborom. Pre rýchlu orientáciu vo výsledku testu som si do simulácie zaviedla ďalší signál, ktorý je príznakom indikujúcim zhodnosť vypočítaného CRC s referenčným výstupným súborom. Testovacie prostredie taktiež upozorňuje na prípadné nezhody CRC alebo nesprávny vstupný súbor chybovým výpisom.

Pre vstupný a referenčný výstupný súbor boli vytvorené implementácie generátorov v jazyku C. *generator-simulacnych-vstupov.c* vygeneruje niekoľko náhodných rámcov, pričom ich počet sa dá nastaviť. Dĺžka rámcov je náhodná, ale dolná a horná hranica ich dĺžky je taktiež nastaviteľná. Vytvorené rámce program uloží do dvoch súborov – jeden pre softvérový generátor výstupného referenčného súboru a druhý pre ModelSimové testovacie prostredie. V súbore pre testovacie prostredie sú na rozdiel od súboru pre generátor výstupu popridávané riadiace signály a zmenené poradie bajtov. Do tohto súboru sú navyše vkladané posunutia začiatkov rámcov a náhodné slová nepatriace žiadnemu rámcu. Pravdepodobnosť výskytu takýchto slov sa dá opäť nastaviť. *generator-referencneho-vystupu.c* je program slúžiaci na výpočet správneho CRC z dát, ktoré si načíta zo súboru. Svoje výsledky uloží do ďalšieho súboru, ktorý poslúži ako už spomínaný referenčný výstup testovanej jednotky. Výpočet sa vykoná tabuľkovou metódou po bajtoch. *generator-referencneho-vystupu.c* bol otestovaný porovnávaním výsledkov s nástrojom pycrc [13], ktorý je voľne prístupný na internete.

Pre uľahčenie vytvárania potrebných súborov slúži jednoduchý skript *test.sh*, ktorý preloží zdrojové súbory oboch generátorov, vytvorí všetky potrebné súbory spustením generátorov, uloží ich na správne miesto a vyčistí všetko nepotrebné, čo vyrobil spolu so vstupmi pre testovacie prostredie. Svojou činnosťou prispieva k automatizácií celého testovacieho prostredia.

Testovacie prostredie pre FITkit vzniklo z dôvodu vyskúšať vyvíjanú jednotku v skutočnom hardvéri. Cieľom bolo vytvoriť aplikáciu, ktorá načíta vstupné dáta, zašle ich na vstup testovanej jednotke a čaká na výsledok. Výsledok následne vypíše na displej FITkitu. Ďalšou požiadavkou na aplikáciu bolo, aby pokračovala vo výpočte a výpise ďalšieho CRC po prijatí príkazu next odoslaného prostredníctvom terminálu.

FITkit je samostatný hardvér vyvíjaný na Fakultě informačních technologií Vysokého učení technického v Brně [14]. Jedná sa o vývojovú platformu vytvorenú najmä pre výukové účely, ktorá obsahuje mikrokontrolér s nízkym príkonom, hradlové pole FPGA a mnoho periférií. Platformu FITkit som si vybrala najmä vďaka jej dostupnosti. Ďalšou veľkou výhodou je, že všetok potrebný softvér pre naprogramovanie FITkitu je k dispozícií zadarmo [15]. V tabuľke 4.2 sú uvedené dôležité parametre FITkitu.

Vytvorené testovacie prostredie, ktorého schéma je znázornená na obrázku 4.5, obsahuje zdrojový súbor *main.c* pre MCU a pridané jednotky do FPGA, vrátane ovládačov LCD

Verzia FITkitu	2.0
FPGA	Spartan 3, XC3S50 - 4PQ208C (Xilinx)
MCU	MSP430F168 (Texas Instruments)
LCD displej	Dvojriadkový, 32 znakov
Komunikácia FPGA a MCU	SPI

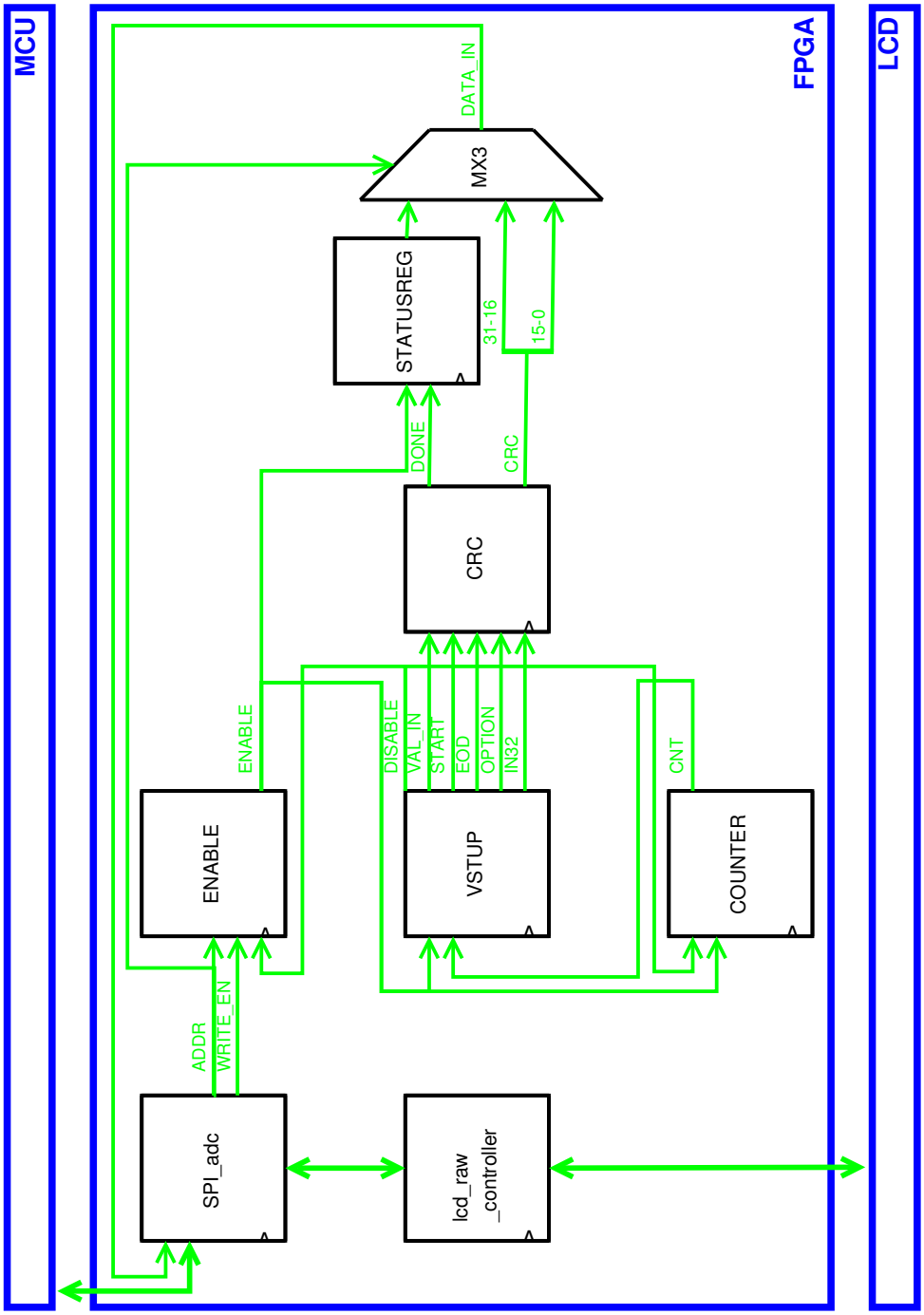
Tabuľka 4.2: Parametre testovacieho FITkitu

displeja a SPI rozhrania.

Jednou z pridaných jednotiek do FPGA je VSTUP, ktorý buď načítava vstupné dáta z Block-RAM pamäte a zasiela ich na vstup testovanej jednotke, alebo na jej vstup posiela dáta nepatriace žiadnemu rámcu. Do stavu posielania dát nepatriacich žiadnemu rámcu sa VSTUP dostane po odoslaní dát s pozitívnym príznakom posledného slova rámca. Informáciu o zmene stavu taktiež rozšíri do ostatných jednotiek. Ďalšou z pridaných jednotiek je COUNTER zvyšujúci počítadlo v prípade, že VSTUP číta z pamäte. Toto počítadlo slúži ako index do pamäte pre VSTUP. Jednotka ENABLE generuje signál informujúci okolie o tom, že VSTUP opäť môže čítať z pamäte. Dôvod takéhoto riešenia je v tom, aby si užívateľ stihol skontrolovať výsledok vypočítaného CRC skôr, než bude prepísaný ďalším. Pokým nezadá príkaz, že je pripravený na ďalší výsledok, na vstup počítajúcej jednotky sú posielané iba irelevantné neplatné dáta. Ďalšou pridanou jednotkou je register STATUSREG, ktorý slúži pre ošetrenie problému nerovnako rýchlo počítajúcich MCU a FPGA. Signál vyvíjanej jednotky indikujúci vypočítané CRC je vysielaný iba jeden takt a s tým sa objavuje riziko, že MCU ho nestihne zachytiť. STATUSREG sa nastaví do logickej jednotky po vypočítaní CRC aktuálneho rámca a do nuly sa vráti až po potvrdení od MCU. Multiplexor MX3 vyberá dáta, ktoré sú posielané na SPI rozhranie.

Komunikácia medzi MCU a FPGA pozostáva z neustáleho zisťovania stavu STATUSREG zo strany MCU. Pokiaľ MCU zistí, že STATUSREG je nastavený do jednotky, vie, že v FPGA je k dispozícií ďalšie vypočítané CRC a v dvoch krokoch si ho nechá poslať. Na zistenie hodnoty CRC je treba dva kroky z dôvodu, že posielanie CRC využíva rovnaké SPI rozhranie ako príkazy pre výpis na displej, ktoré pracujú s polovičnou šírkou dát, než je šírka CRC. Po získaní výsledku, je tento výsledok prevedený na hexadecimálne znaky v MCU a odoslaný na displej. Časť súboru *top_level.vhd*, ktorá sa venuje vypisovaniu znakov, obsahuje softvér vyvinutý na FIT VUT. Po vypísaní na displej MCU čaká na správny príkaz z terminálu a po jeho prijatí pošle po SPI rozhraní pokyn, ktorý FPGA vyhodnotí ako žiadosť k ďalšiemu výpočtu.

Pre spojazdnenie testovacieho prostredia je nutné nahrať zdrojové súbory do adresára *apps/demo/crc* v SVN pre FITkit. K ovládaniu aplikácie slúži súbor *vstup.vhd* a terminál. Do uvedeného súboru sa vpisujú dáta, na ktorých sa má vykonať testovací výpočet. *vstup.vhd* aktuálne obsahuje niekoľko testovacích rámcov a príslušné výsledky uvedené v komentároch pre rýchle odskúšanie. Zadaním *next* do terminálu sa výpočet posúva na nasledujúci rámec.



Obr. 4.5: Schéma testovacieho prostredia pre FITkit

Kapitola 5

Jednoduchá optimalizácia

5.1 Rozširovanie základného riešenia

Základné riešenie popísané v predošlej kapitole spĺňa všetky požiadavky na funkčnosť, ale spracovávaním 32 bitov v každom takte nedodržuje nároky na rýchlosť. Preto sa budem v tejto kapitole venovať rozširovaniu vstupného slova, pričom zachovám metódy využité v základnom riešení. Vstupné slovo budem rozširovať na 64, 128 a 256 b.

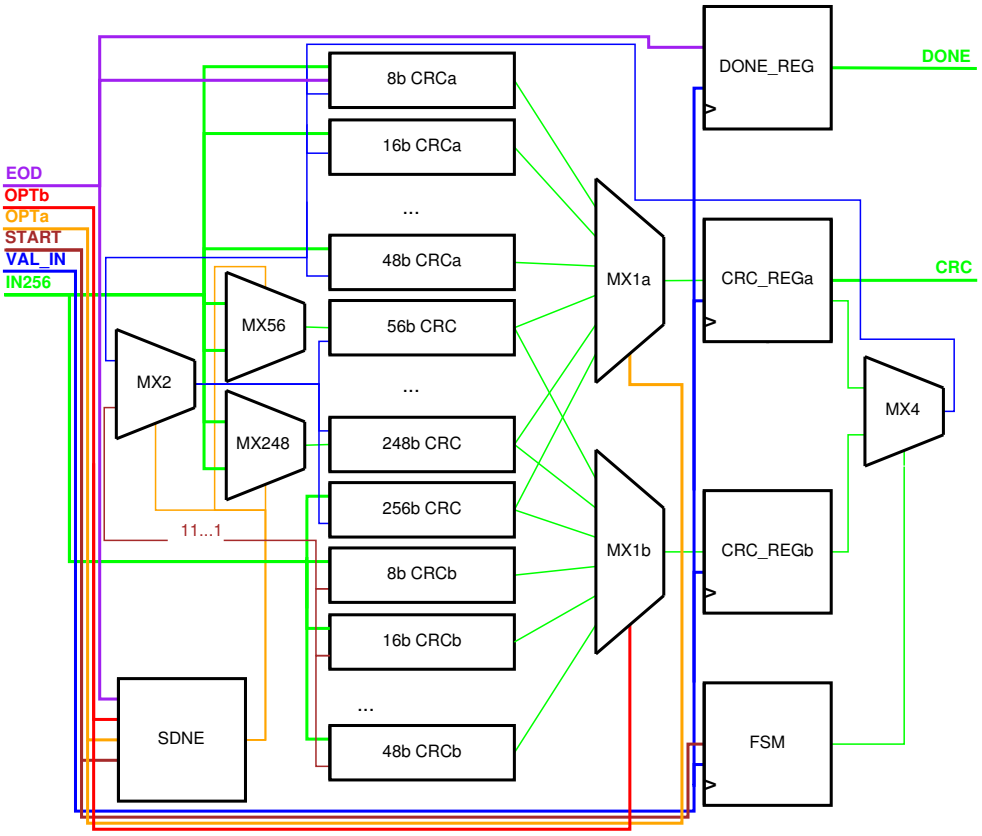
Rozšírenie na 64 a 128 b je celkom priamočiare. Základné riešenie zobrazené na obrázku 4.3 treba obohatiť o chýbajúce jednotky počítajúce CRC pre rôzne šírky vstupných dát tak, aby dáta všetkých šírok, ktoré sú násobkom 8 b až do šírky vstupného slova boli počítané práve jednou jednotkou. Všetky jednotky počítajúce CRC je opäť potrebné zapojiť paralelne a pred každou z nich, okrem tej, ktorá pracuje so vstupnými dátami s rovnakou šírkou, ako je šírka slova, treba znovu pridať multiplexor. Každý z týchto multiplexorov vyberá vstup do jednej CRC jednotky podľa toho, či sa jedná o prvé slovo v rámci, alebo nie. Ďalšou zmenou je multiplexor MX1, ktorý má toľko vstupov, koľko je jednotiek počítajúcich CRC. Posledným zmeneným je rozšírenie signálov IN na šírku vstupného slova (d b) a OPTION na šírku n bitov tak, aby platilo:

$$2^n = 8d$$

Pri rozširovaní na 256 b, treba základné riešenie doplniť ešte o ďalšie časti. Dôvodom je minimálna veľkosť medzery medzi rámcami 160 b. To znamená, že jednotka počítajúca so vstupom 256 b musí, na rozdiel od predošlých, dokázať spracovávať aj dva rámce naraz.

Na obrázku 5.1 je rozšírenie základného riešenia, ktoré prijíma na vstupe dáta o šírke 256 b a dokáže počítať v jednom takte dve hodnoty CRC, jednu pre koniec jedného rámca a druhú pre začínajúci rámec, oba prijaté v jednom slove. Jednotky v schéme, ktorých názov končí písmenom b slúžia výhradne pre začiatky rámcov, jednotky s názvom ukončeným a sú určené pre vnútro a konce rámcov.

V prípade, že sú spracovávané časti dvoch rámcov naraz, môže sa stať, že obe majú rovnakú šírku. To znamená, že by obe potrebovali tú istú jednotku pre výpočet CRC. Preto je tieto jednotky potrebné zdvojiť. O metóde pre zapojenie CRC jednotiek paralelne však platí, že zaberá veľa zdrojov a preto je vhodné zdvojiť len to čo je nevyhnutné a ostatné nechať spoločné aj na úkor malého zhoršenia kritickej cesty. Zo šírky vstupného slova 256 b a medzery minimálne 160 b vyplýva, že pokiaľ sa v slove nachádzajú časti dvoch rámcov, tak súčet veľkostí týchto častí je maximálne 96 b. To znamená, že pokiaľ tieto časti majú rovnakú veľkosť, tak sú veľké maximálne 48 b. Z uvedeného dôvodu vyplýva, že je postačujúce zdvojiť iba CRC jednotky pre šírku vstupu menšiu a rovnú 48 b. Multiplexory pred zdvojenými



Obr. 5.1: Schéma jednoduchého rozšírenia pre vstup 256 b

CRC jednotkami už nie sú potrebné, pretože každá z nich vždy počíta buď iba začiatky alebo iba konce rámca. Novými jednotkami v obvode sú SDNE a FSM. Výstupom SDNE je príznak, ktorý je nastavený do logickej jednotky iba vtedy, ak aktuálne slovo na vstupe obsahuje začiatok rámca a neobsahuje koniec predošlého, alebo ak obsahuje začiatok a tento začiatok je dlhší než koniec predošlého rámca, ktorý sa nachádza v tom istom slove. Názov je skratkou pre START Dlhší Než EOD. Jednotka SDNE slúži na vyberanie dátových vstupov a aktuálnej hodnoty CRC registra pre jednotky počítajúce CRC. FSM, ktorého výstupom je príznak udávajúci, či je aktuálne spracovávané druhé slovo rámca, taktiež slúži pre výber aktuálnej hodnoty CRC registra pre jednotky počítajúce CRC.

Okrem pridaných jednotiek treba opäť rozšíriť vstupný signál IN na 256 b. Taktiež je potrebné zdvojiť signál OPTION na OPTa a OPTb z dôvodu možnosti spracovávania dvoch rámcov naraz. Rozhranie riešenia so šírkou slova 256 b je súhrnne popísané v tabuľke 5.1.

5.2 Meranie maximálnej frekvencie a využitia zdrojov

Meranie maximálnej možnej frekvencie je potrebné pre zistenie priepustnosti navrhnutej jednotky a porovnávanie jednotlivých rozšírení medzi sebou. Použiteľná jednotka musí taktiež splňovať rozumnú mieru využitých zdrojov. Meranie pre túto bakalársku prácu bolo realizované programom Xilinx ISE, nástrojom XST. Meralo sa pre 3 rôzne typy FPGA:

- FPGA na bežnom FITkite, tzn. Spartan-3 XC3S50-4PQ208C, výsledky merania sú uvedené v tabuľke 5.2

Signál	Smer	Popis
CLK	vstup	hodinový signál
CRC	výstup	výsledná hodnota CRC
DONE	výstup	príznak určujúci, či signál CRC obsahuje platnú hodnotu
EOD	vstup	príznak indikujúci posledné slovo aktuálneho rámca
IN256	vstup	obsahuje slovo vstupného rámca, pre ktorý bude vypočítané CRC, šírka vstupného slova je 256 b
OPTa	vstup	určuje počet bajtov vstupného slova patriacich rámcu; OPTa=0 znamená 1 bajt, OPTa=1 znamená 2 bajty...
OPTb	vstup	určuje počet bajtov vstupného slova patriacich rámcu; OPTb=0 znamená 1 bajt, OPTb=1 znamená 2 bajty...
RESET	vstup	reset
START	vstup	príznak indikujúci prvé slovo aktuálneho rámca
VAL_IN	vstup	príznak určujúci platnosť aktuálnych vstupných dát

Tabuľka 5.1: Rozhranie riešenia so šírkou dátového vstupu 256 b

- FPGA na rozšírenej verzii FITkitu, tzn. Spartan-3 XC3S400-4PQ208C, výsledky v 5.3
- FPGA rady Virtex-6, konkrétne XC6VLX75T-2-FF484, výsledky v tabuľke 5.4

Pre porovnanie sú v záhlaviach všetkých troch tabuliek vypísané celkové počty dostupných zdrojov pre príslušné FPGA. Meranie bolo uskutočnené pre základné riešenie z minulej kapitoly a jeho jednoduché rozšírenia na šírku vstupného slova 64, 128 a 256 b.

Šírka slova [b]	Počet registrov (max. 1536)	Počet LUT (max. 1536)	Frekvencia [MHz]	Priepustnosť [Gb/s]
32	32	604	114,969	3,679
64	32	1836	101,688	6,508
128	39	5903	94,065	12,040
256	65	22402	66,271	16,965

Tabuľka 5.2: Maximálna frekvencia a využitie zdrojov pre XC3S50-4PQ208C

Šírka slova [b]	Počet registrov (max. 7168)	Počet LUT (max. 7168)	Frekvencia [MHz]	Priepustnosť [Gb/s]
32	32	604	114,969	3,679
64	38	1925	102,166	6,539
128	39	5664	95,969	12,284
256	65	22402	66,271	16,965

Tabuľka 5.3: Maximálna frekvencia a využitie zdrojov pre XC3S400-4PQ208C

Namerané hodnoty potvrdzujú očakávané výsledky a to konkrétne, že použitá metóda zapojenia jednotiek počítajúcich CRC je rýchla, ale zaberá príliš veľa zdrojov. Najvýraznejšie sa uvedený problém prejavuje na jednotke s najširším vstupným slovom. Táto jednotka

Šírka slova [b]	**Počet registrov (max. 93120)**	**Počet LUT (max. 46560)**	**Frekvencia [MHz]**	**Priepustnosť [Gb/s]**
32	32	548	402,941	12,894
64	61	1584	365,901	23,418
128	52	4867	322,191	41,240
256	110	18101	245,464	62,720

Tabuľka 5.4: Maximálna frekvencia a využitie zdrojov pre XC6VLX75T-2-FF484

je síce schopná dosiahnuť na rýchlom FPGA priepustnosť cez 60 Gb/s, ale zaberá desaťtisíce LUT.

Výrazné rozdiely sa vyskytujú aj pri syntéze tej istej jednotky ale na iných typoch FPGA. V prípade FPGA na bežnom FITkite, ktorý má oprosti ostatným niekoľkonásobne menej zdrojov, bolo namerané, že tieto zdroje stačia iba pre základné riešenie. Žiadne z rozšírení sa už na FPGA vo FITKite nezmestí. Virtex-6 je dostačujúci pre všetky vylepšenia. Na Virtex-6 bola taktiež nameraná takmer štvornásobne vyššia maximálna možná frekvencia než na FITkitovom Spartan-3.

Kapitola 6

Optimalizácia počtu využitých zdrojov

6.1 Návrh a implementácia optimalizácie

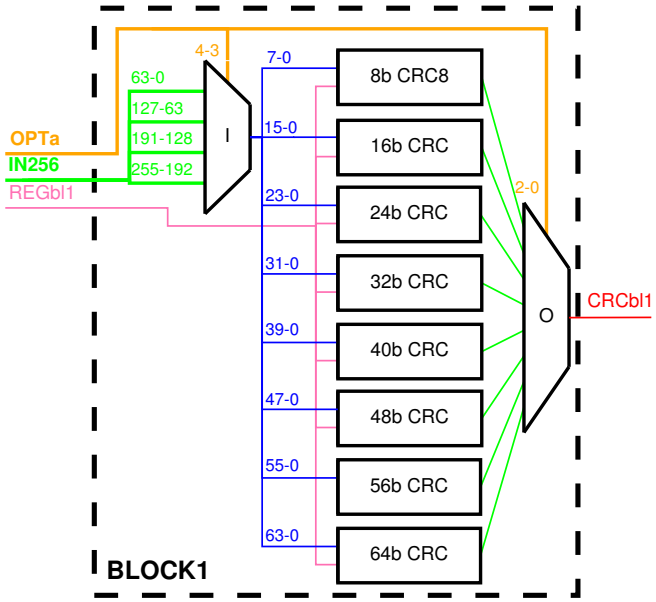
Optimalizované riešenie pre výpočet CRC z minulej kapitoly, ktoré pracuje so vstupným slovom šírky 256 b, dosahuje pre Virtex-6 priepustnosť o vyše 20 Gb/s vyššiu, než je požadované od tejto bakalárskej práce. Toto riešenie však potrebuje veľa zdrojov, zaberá až desaťtisíce LUT. Preto je potrebné znížiť počet využitých zdrojov, aj keď to prinesie pokles maximálnej možnej frekvencie. Táto kapitola sa venuje optimalizácií počtu využitých zdrojov pre jednotku, ktorá v každom takte prijíma na vstupe dáta o šírke 256 b.

Základná myšlienka riešenia problému zníženia počtu použitých zdrojov je v odstránení väčšiny paralelne zapojených jednotiek počítajúcich CRC. Pred tento výrazne zúžený paralelný blok je potrebné pripojiť ďalší blok paralelne zapojených jednotiek počítajúcich CRC. CRC pre určitú dátovú šírku bude teda väčšinou počítané kombináciou dvoch jednotiek počítajúcich CRC, ktoré sú zapojené za sebou. Každá z paralelne zapojených jednotiek prijíma dáta inej šírky tak, aby boli pokryté všetky možné kombinácie nezarovnaných začiatkov a koncov rámcov. Toto riešenie kombinuje metódy zapojenia jednotiek do série a paralelne, ktoré boli popísané v podkapitole 3.4.

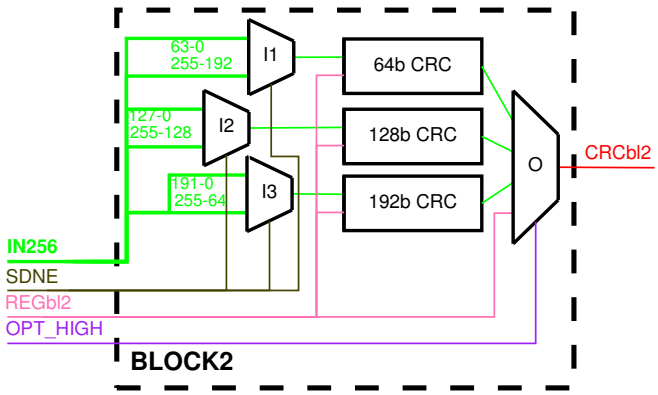
Optimalizované riešenie v tejto kapitole samozrejme zachováva rozhranie ako má riešenie s rovnakou šírkou vstupu v predošlej kapitole. Rozhranie riešenia so šírkou slova 256 b je súhrnne popísané v tabuľke 5.1.

Kritériom výberu jednotiek počítajúcich CRC s určitými šírkami vstupných slov do blokov bolo maximálne šetrenie zdrojmi. Keďže sa v jednom vstupnom slove môžu súčasne nachádzať až časti dvoch rámcov, je potrebné bloky zdvojiť. Pri vhodnom výbere paralelných CRC jednotiek však postačí duplicita len jedného z blokov. Na obrázkoch 6.1 a 6.2 je znázornené vhodné vnútro blokov. Blok BLOCK2 je zdieľaný pre obe možné časti rámcov v jednom slove, pretože počíta s tak veľkými vstupnými šírkami, že za predpokladu minimálnej medzery medzi rámcami 160 b ho môže potrebovať maximálne jedna z nich. Blok BLOCK1 musí byť v jednotke dvakrát. Jednotka BLOCK2 počíta CRC pre šírku vstupných dát, ktorá je násobkom 64 b. BLOCK1 počíta zo zvyšku.

Pre optimalizovanú jednotku, rovnako ako pre predošlé jednotky, existujú 2 druhy platných slov. Prvým druhom sú slová z vnútra rámcov, čiže také slová, ktorých celá šírka je platná. K prvému druhu taktiež patria konce rámcov. Druhým druhom sú začiatky rámcov. Rozdiel medzi uvedenými druhmi slov sa dá pozorovať na obrázku 4.2 v podkapitole 4.1.



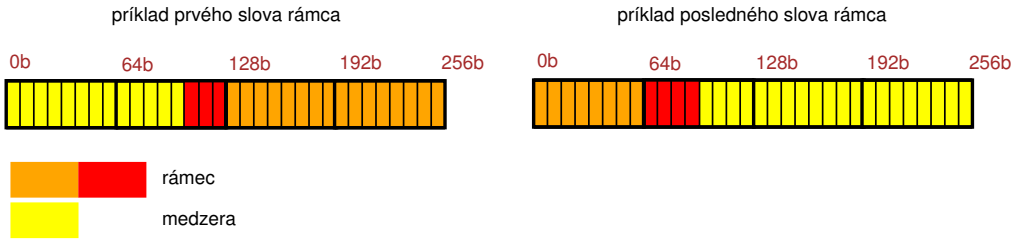
Obr. 6.1: Schéma bloku BLOCK1



Obr. 6.2: Schéma bloku BLOCK2

Obrázok ukazuje, že pre ľubovoľný počet platných bitov v slove, slová z prvej skupiny majú vždy platný prvý bit. To znamená že platné dáta v nich sú zarovnané na začiatok slova. Druhá skupina je charakteristická zarovnaním platných dát na koniec slova. Na obrázku 6.3 sú príklady 256 bitových vstupných slov. Oranžové a červené políčka patria rámcom, žlté tvoria medzery. V slovách sú zvýraznené hranice 64 b. Úseky medzi zvýraznenými hranicami, ktoré sú celé tvorené platnými dátami sú zakreslené oranžovo. Rozdeľovanie na úseky 64 b je využívané jednotkami BLOCK1 a BLOCK2. Z faktu, že BLOCK2 počíta vždy zo zarovnaných dát vyznačených oranžovou a BLOCK1 z toho, čo ostalo – čiže z červených dát, sa dá vyvodiť ich vzájomné poradie. Pre začiatky rámcov musí BLOCK1 predchádzať BLOCK2. Pre vnútro a konce rámcov je to naopak.

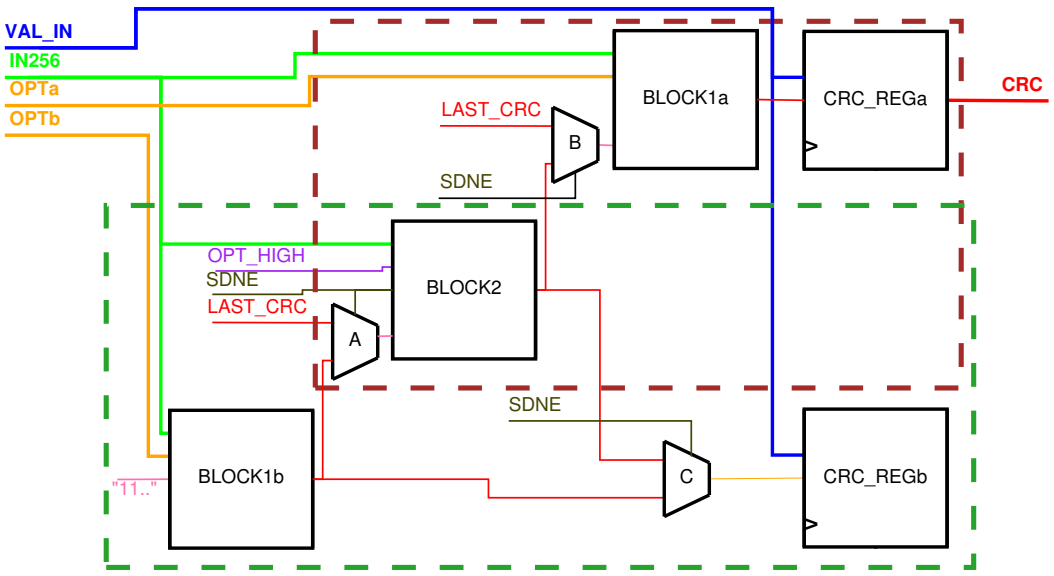
Obrázok 6.1 presne odpovedá bloku BLOCK1 využívanému pre výpočet CRC vnútra a koncov rámcov. Pre BLOCK1 počítajúci z prvých slov tento obrázok platí iba principiálne. Oba bloky obsahujú rovnaké jednotky, ale zapojenie signálov sa mierne líši. Vstupujúcim signálom nie je OPTa ale OPTb. Taktiež pripojenie IN256 na paralelné jednotky počítajúce CRC sa odlišuje a to z dôvodu vymenia poradia BLOCK1 a BLOCK2.



Obr. 6.3: Prvé a posledné vstupné slovo rámca

Počítanie jedného CRC dvomi sériovo zapojenými jednotkami je realizované pomocou správne zvolenej hodnoty, ktorá je považovaná za predošlú hodnotu CRC registra pre danú jednotku. Hodnotou, ktorá hrá úlohu predošlej hodnoty CRC registra je pre prvú jednotku v sérií buď skutočná hodnota CRC registra alebo inicializačná hodnota. Pre ďalšiu jednotku je to výsledok CRC predchádzajúcej jednotky. Priradenie zdieľaného BLOCK2 niektorému z blokov BLOCK1 je teda realizované správnym nastavením signálov reprezentujúcich minulú hodnotu CRC registra pre každý z troch blokov.

Základný princíp optimalizovaného riešenia na počet zdrojov je znázornený na obrázku 6.4. V bordovom čiarkovanom obdĺžniku sú umiestnené jednotky počítajúce CRC pre vnútro a konce rámcov. V zelenom čiarkovanom obdĺžniku sa počíta CRC pre začiatky. Na obrázku vidieť zdieľanie BLOCK2 ako aj poradie blokov vzhľadom na výpočet. Multiplexory A a B



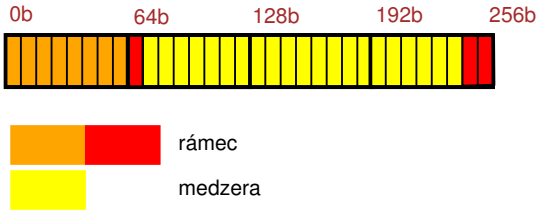
Obr. 6.4: Jednoduchá schéma optimalizácie počtu využitých zdrojov

vyberajú predošlú hodnotu CRC registra pre dané bloky. Pre BLOCK1b, čo je BLOCK1 počítajúci CRC začiatkov rámcov, je predošlá hodnota CRC registra inicializačná hodnota a teda samé jednotky. Inicializačná hodnota je zvolená z dôvodu, že BLOCK1b je prvý z blokov a vždy počíta CRC z prvého slova. LAST_CRC je minulá hodnota jedného z registrov CRC_REGa alebo CRC_REGb. CRC_REGb je zvolená ak sa jedná o druhé platné slovo rámca, pretože obsahuje výsledok začiatku rámca. Vnútorný signál SDNE je rovnako ako v predošlých kapitolách príznak toho, že aktuálne slovo obsahuje začiatok rámca a zároveň začiatok rámca v aktuálnom slove je dlhší než koniec iného rámca v tom istom slove.

Signál OPT_HIGH obsahuje potrebné vyššie bity toho zo signálov rozhrania OPTa a OPTb, ktorý nesie vyššiu hodnotu. A konečne multiplexor C vyberá novú hodnotu CRC registra v závislosti na tom, či blok BLOCK2 počíta v aktuálnom takte CRC pre začiatok rámca, alebo nie. Detailná schéma implementácie sa nachádza v prílohe B.

Fungovanie optimalizovaného riešenia priblížim uvedením popisu chovania jednotiek na troch konkrétnych prípadoch.

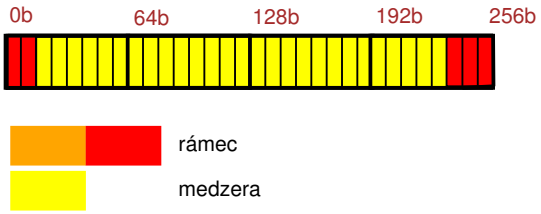
- VAL_IN='1', EOD='1', START='1', OPTa="01000", OPTb="00001".



Obr. 6.5: Vstupné slovo prvého prípadu

Na obrázku 6.5 je zakreslené vstupné slovo pre tento prípad. Žltou farbou sú zaznačené medzery medzi rámcami. Červenou a oranžovou sú rámce. Červené časti rámcov sú určené pre spracovávanie v niektorom z BLOCK1 a oranžové pre BLOCK2. V tomto prípade je na vstupe platné slovo, ktoré v sebe obsahuje 9 bajtov konca rámca a 2 bajty začiatku nasledujúceho rámca. Spoločný blok BLOCK2 je určený pre výpočet konca rámca. Blok BLOCK1b vypočíta CRC začiatku nového rámca, pričom za počiatočnú hodnotu registra ako vždy považuje samé jednotky. Multiplexor C prepustí výsledok z BLOCK1b do CRC registru CRC_REGb. Tým je spracovávanie začiatku ukončené. Počiatočnou hodnotou CRC registra pre blok BLOCK2 sa stane hodnota v niektorom z registrov CRC_REGa alebo CRC_REGb. Táto hodnota je privedená signálom LAST_CRC. Vo vnútri BLOCK2 multiplexor O pošle na svoj výstup výsledok 64 b CRC. Na výstupe B je výsledok BLOCK2. Do registru CRC_REGa je uložený výsledok vypočítaný v 8 b CRC nachádzajúceho sa v BLOCK1a. Hodnota výstupného signálu rozhrania DONE je nastavená do jednotky.

- VAL_IN='1', EOD='1', START='1', OPTa="00001", OPTb="00010".

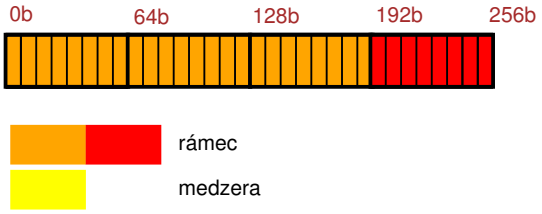


Obr. 6.6: Vstupné slovo druhého prípadu

Na obrázku 6.6 je vstupné slovo pre tento prípad. Opäť je na vstupe platné slovo obsahujúce koniec jedného rámca aj začiatok iného. Začiatku rámca patria 3 bajty, koncu 2. Nová hodnota DONE je jedna. V tomto prípade blok BLOCK2 pripadne začiatku rámca, aj keď ten ho nepotrebuje. CRC hodnota začiatku rámca je vypočítaná v BLOCK1b a prostredníctvom A je poslaná na vstup BLOCK2 ako aktuálna hodnota

CRC registra. Vo vnútri BLOCK2 multiplexor O určí ako výstup vstup, ktorý neprechádza CRC počítadlom vo vnútri tohto bloku. Novou hodnotou CRC_REGb je výsledok BLOCK2. Ako počiatočná hodnota CRC registra pre BLOCK1a, je LAST_CRC. Nová hodnota CRC_REGa je vypočítaná iba v BLOCK1a.

- VAL_IN='1', EOD='0', START='0', OPTa="11111", OPTb="00111".



Obr. 6.7: Vstupné slovo tretieho prípadu

Obrázok 6.7 ukazuje vstupné slovo pre aktuálny prípad. V tomto prípade je na vstupe slovo, ktoré obsahuje vnútro rámca. V slove sa nenachádza žiadny začiatok ani koniec rámca. Novou hodnotou DONE je nula. BLOCK2 v tomto prípade počíta vnútro rámca. Aktuálnou hodnotou CRC registra pre blok BLOCK2 je LAST_CRC. BLOCK2 na výstup posiela hodnotu vypočítanú v CRC192. Táto hodnota je aktuálnou hodnotou registra pre BLOCK1a. Výsledok BLOCK1a je nová hodnota registra CRC_REGa. Do registra CRC_REGb je taktiež zapísaná nová hodnota vypočítaná iba v BLOCK1b, táto hodnota je však nepotrebná a už nebude nikde použitá.

6.2 Testovanie optimalizácie

Optimalizované riešenie bolo otestované pomocou testovacieho prostredia v ModelSime. Bližší popis testovacieho prostredia pre ModelSim vytvoreného pre účely tejto bakalárskej práce sa nachádza v podkapitole 4.3. Pre túto bakalársku prácu bolo taktiež naimplementované testovacie prostredie na FITkit. Optimalizované riešenie však v tomto prostredí vyskúšané nebolo. Dôvodom neodskúšania na FITkite je počet dostupných zdrojov na tejto platforme, ktoré nestačia pre optimalizované riešenie.

6.3 Maximálna frekvencia a využitie zdrojov u optimalizácie

Typ FPGA	Počet registrov	Počet LUT	Frekvencia [MHz]	Priepustnosť [Gb/s]
XC3S50-4PQ208C	66	5636	47,835	12,246
XC3S400-4PQ208C	67	5396	48,318	12,369
XC6VLX75T-2-FF484	155	4716	185,630	47,521

Tabuľka 6.1: Maximálna frekvencia a využitie zdrojov u optimalizácie na počet zdrojov

Meranie maximálnej možnej frekvencie a využitia zdrojov optimalizovaného riešenia bolo uskutočnené nástrojom XST. Meralo sa pre tri rôzne typy FPGA: XC3S50-4PQ208C, ktorý sa nachádza na FITkitoch, XC3S400-4PQ208C využívaný na rozšírených FITkitoch

a pre FPGA rady Virtex-6, konkrétne XC6VLX75T-2-FF484. Výsledky syntézy sú uvedené v tabuľke 6.1.

Počet dostupných zdrojov na všetkých troch zariadeniach je vypísaný v tabuľke 6.2. Z 6.2 a 6.1 vyplýva, že počet dostupných zdrojov na XC3S50-4PQ208C je nedostačujúci pre optimalizáciu na počet zdrojov. Na ostatné dva typy FPGA sa táto optimalizácia zmestí.

Typ FPGA	**Počet registrov**	**Počet LUT**
XC3S50-4PQ208C	1536	1536
XC3S400-4PQ208C	7168	7168
XC6VLX75T-2-FF484	93120	46560

Tabuľka 6.2: Počet dostupných zdrojov pre použité typy FPGA

Z tabuľky 6.3 je vidieť, že sa v optimalizovanom riešení na počet zdrojov podarilo výrazne znížiť počet využitých zdrojov pri zachovaní priepustnosti nad 40 Gb/s pre FPGA rady Virtex-6. Oproti jednoduchému rozšíreniu klesol počet využitých zdrojov takmer na 26 % pri znížení maximálnej možnej frekvencie o necelých 25 %.

Optimalizácia	**Počet registrov (max. 93120)**	**Počet LUT (max. 46560)**	**Frekvencia [MHz]**	**Priepustnosť [Gb/s]**
Jednoduché rozšírenie	110	18101	245,464	62,720
Využitých zdrojov	155	4716	185,630	47,521

Tabuľka 6.3: Porovnanie optimalizácií pre šírku vstupu 256b na XC6VLX75T-2-FF484

Kapitola 7

Záver

V tejto bakalárskej práci som vyvíjala hardvérovú jednotku pre výpočet CRC vo vysokorýchlostných sieťach. Prvým krokom, ktorý som urobila, bolo naštudovanie teórie o CRC. Zistila som kde, prečo a ako sa používa a pochopila rôzne metódy na jeho výpočet. Okrem teórie o CRC som si naštudovala aj pravidlá o štruktúre a posielaní Ethernetových rámcov. Podľa požiadaviek zadania na túto bakalársku prácu, ktorou bola hlavne vysoká priepustnosť, som si zvolila metódy pre implementáciu. Zvolenými metódami boli metóda využívajúca Galoisové polia a metóda zapojenia jednotiek paralelne. Skombinovaním týchto dvoch metód a pridaním vlastných nápadov som vytvorila základné riešenie. Toto riešenie si dokázalo poradiť s nezarovnanými začiatkami a koncami rámcov, ale nebolo dostatočne rýchle. Správnosť fungovania základného riešenia som otestovala v dvoch testovacích prostrediach, ktoré som vytvorila pre účely tejto bakalárskej práce. Jedno z vytvorených prostredí testovalo softvérovo, generovaním náhodných rámcov, ktoré spĺňali protokoly Ethernetu a následnou simuláciou a porovnávaním výsledkov s referenčnými v ModelSime. Druhé testovacie prostredie bolo vytvorené pre platformu FITkit.

Základné riešenie som z dôvodu jeho pomalosti optimalizovala. Prvou optimalizáciou bolo rozširovanie vstupného slova ale zachovávanie princípu zapojenia jednotiek zo základného riešenia. Takáto optimalizácia sa však pri neskoršej syntéze ukázala nedostatočná, pretože síce dosahovala vysokú priepustnosť, ale zaberala príliš veľa zdrojov.

Posledná optimalizácia bola zameraná na výrazne zníženie počtu využitých zdrojov na úkor nízkeho zníženia maximálnej možnej frekvencie. Do riešenia som pripojila ďalšiu metódu a to zapojenie jednotiek do série. Široký paralelný blok jednotiek počítajúcich CRC bol vhodne rozdelený do viacerých úzkych blokov zapojených za sebou. Táto optimalizácia bola otestovaná v testovacom prostredí pre ModelSim.

Vo výsledku vznikla implementácia, ktorá v každom takte spracuje vstupné slovo o šírke 256b, pričom toto vstupné slovo môže naraz obsahovať časti až dvoch rámcov. Pre FPGA rady Virtex-6 optimalizované riešenie dosahuje priepustnosť vyše 47Gb/s a zaberá 155 registrov a 4716 LUT.

Ďalším pokračovaním tejto bakalárskej práce by mohlo byť rozšírenie vstupného slova na 512b a dosiahnutie priepustnosti 100Gb/s. Vzhľadom na požiadavky na zrýchľovanie sietí by takéto pokračovanie mohlo nájsť v budúcnosti využitie.

Literatúra

[1] Cisco Networking Academy. *CCNA Exploration Course Booklet: Network Fundamentals, Version 4.0.* Indianapolis, USA: Cisco Press, September 2009. Course Booklets. ISBN 978-1-58713-243-8.

[2] *IEEE Standard for Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications* [online]. 2008 [cit. 30. dubna 2012]. Dostupné na URL: <http://standards.ieee.org/about/get/802/802.3.html>.

[3] William, R. N. *A painless guide to CRC error detection algorithms* [online]. 3. August 1993 [cit. Január 2012]. Dostupné na URL: <http://www.epm.ornl.gov/đunigan/crc.html>.

[4] Shieh, M.-D., Sheu, M.-H., Chen, C.-H. et al. A Systematic Approach for Parallel CRC Computations. *Journal of Information Science and Engineering.* 2001, roč. 17, č. 3. S. 445–461.

[5] Pei, T.-B. a Zukowski, C. High-speed parallel CRC circuits in VLSI. *IEEE Transactions on Communications.* April 1992, roč. 40, č. 4. S. 653–657. ISSN 0090-6778.

[6] Bolton, D. *Definition of XOR* [online]. [cit. Január 2012]. Dostupné na URL: <http://cplus.about.com/od/glossar1/g/xor.htm>.

[7] Geremia, P. *Cyclic Redundancy Check Computation: An Implementation Using the TMS320C54x.* USA: Texas Instruments, April 1999. S. 35. Application Report, SPRA530.

[8] Farana, R. *Informatika: Kódování.* Ostrava: VŠB-TU Ostrava, 2008. Dostupné na URL: <http://www.352.vsb.cz/uc_texty/InformatikaSyl/02Kodovani.pdf>.

[9] Ray, J. a Koopman, P. Efficient High Hamming Distance CRCs for Embedded Networks. In *Proceedings of the International Conference on Dependable Systems and Networks.* Washington, DC, USA: IEEE Computer Society, 2006. S. 3–12. DSN '06. Dostupné na URL: <http://dx.doi.org/10.1109/DSN.2006.30>. ISBN 0-7695-2607-1.

[10] Perez, A. Byte-Wise CRC Calculations. *IEEE Micro.* June 1983, roč. 3, č. 3. S. 40–50. ISSN 0272-1732.

[11] Henriksson, T. a Liu, D. Implementation of fast CRC calculation. In *Proceedings of the ASP-DAC 2003. Asia and South Pacific Design Automation Conference.* Kitakyushu, Japan: IEEE, January 2003. S. 563–564. ISBN 0-7803-7659-5.

[12] Kennedy, C. a Reyhani Masoleh, A. High-speed parallel CRC circuits. In *Forty-Second Asilomar Conference on Signals, Systems and Computers.* California, USA: IEEE, October 2008. S. 1823–1829. ISBN 978-1-4244-2940-0.

[13] Pircher, T. *Pycrc - free CRC source code generator for C* [online]. Revised 13. 2. 2012 [cit. 1. 3. 2012]. Dostupné na URL: <`http://www.tty1.net/pycrc/`>.

[14] *Fakulta informačních technologií VUT v Brně* [online]. [cit. 10. 5. 2012]. Dostupné na URL: <`http://www.fit.vutbr.cz/`>.

[15] Vašíček, Z. *FITkit* [online]. Aktualizovano 14. 4. 2012 [cit. 14. 4. 2012]. Dostupné na URL: <`http://merlin.fit.vutbr.cz/FITkit/`>.

Dodatok A

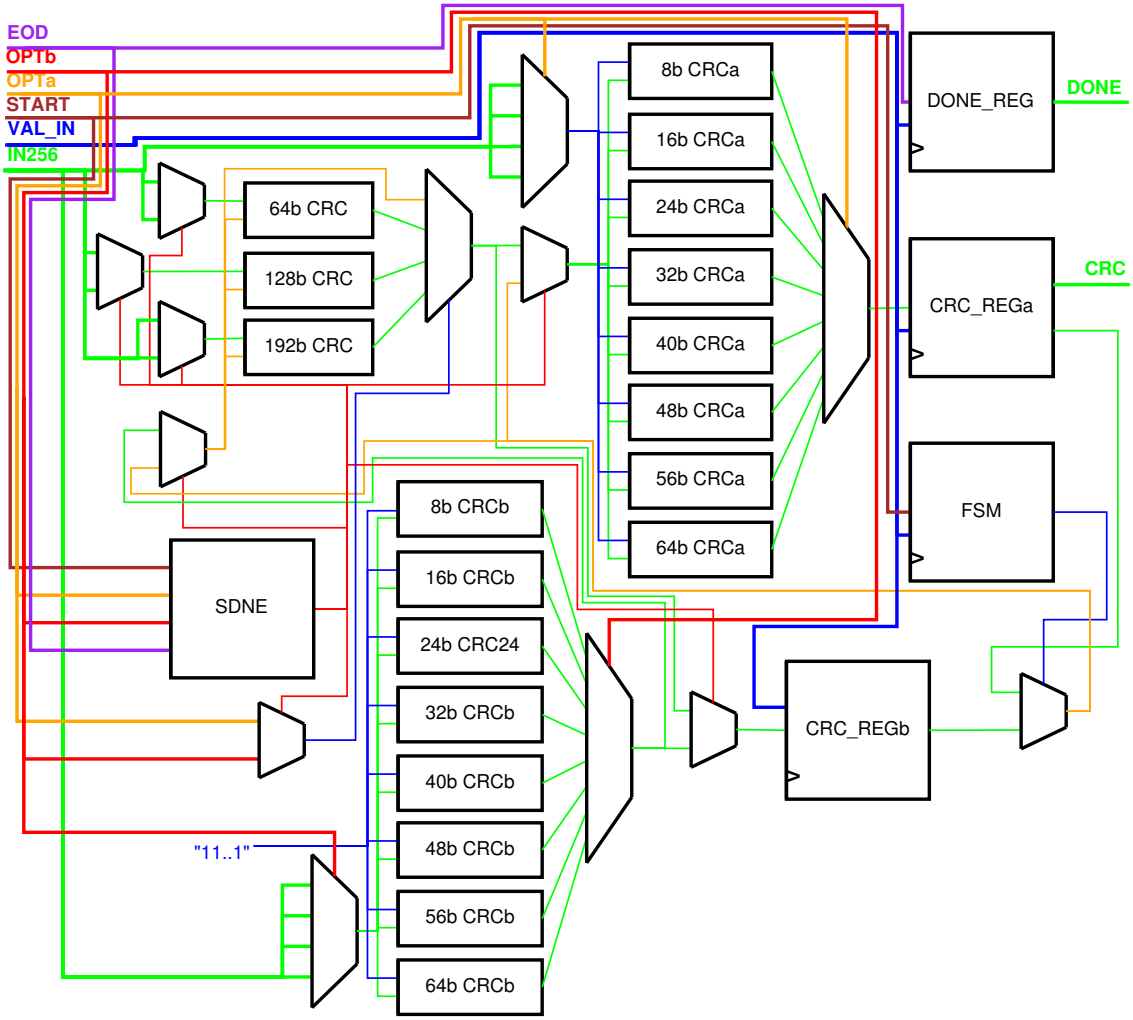
Obsah CD

Priložené CD obsahuje na najvyššej úrovni 4 zložky a 1 súbor.

- zložka *doc* obsahuje text bakalárskej práce vo formáte .pdf
- zložka *galoisfields* obsahuje zdrojový súbor v jazyku C pre generovanie rovníc pomocou metód reprezentujúcich Galoisové polia
- zložka *tex* má v sebe zdrojové súbory a obrázky pre vytvorenie textu bakalárskej práce v .pdf pomocou LaTeXu
- zložka *vhdl* obsahuje všetky zdrojové súbory vyvíjaných jednotiek vo VHDL a testovacie prostredia
  - zložka *ver32b* patrí základnému riešeniu a obsahuje generátory náhodných rámcov pre ModelSimové testovacie prostredie
  - zložka *ver64b* obsahuje jednoduché rozšírenie na šírku vstupného slova 64 b
  - zložka *ver128b* obsahuje jednoduché rozšírenie na šírku vstupného slova 128 b
  - zložka *ver256b* obsahuje jednoduché rozšírenie na šírku vstupného slova 256 b
  - zložka *ver256b-vylepsenie* obsahuje konečnú optimalizáciu na počet využitých zdrojov
  - zložka *ver32b-FITkit*je vytvorená pre testovacie prostredie na FITkite
- súbor *README.txt* popisuje obsah zložiek

Dodatok B

Schéma optimalizácie počtu využitých zdrojov



Obr. B.1: Kompletná schéma optimalizácie počtu využitých zdrojov

V tejto prílohe je znázornená detailná schéma implementácie optimalizácie počtu využitých zdrojov. Tejto implementácií sa venuje kapitola 6.